**Linux:**

* How to create a soft link for a long program language name?

Ans**: ln -s long\_application\_name short\_application\_name**

E.g.: ln -s scala\_2.10.3 Scala

* How to find name of host you are working on?

**hostname** **-f**

* How to get size of a file?

**-du -sh <path\_to\_file>** where -s is for summarize and -h is to print in readable format, use this command in local file system.

**hadoop fs -du -s -h <path\_of\_file>** on hdfs

* How to search using keyword in linux?

Syntax: **<any valid linux command> |grep <keyword>**

Eg: ls -ltr |grep scala,

history |grep consume (this will search for keyword ‘consume’ in history command)

The above example will search for keyword “scala” or “consume” in current directory

* How to go to home directory from any location?

**cd (no space, press enter key)**

* How to ssh to cloudera vm?

**ssh username@<ipaddress>**

**Eg: ssh root@<ipaddress> or cloudera@<ipaddress>**

* How to replace characters or words in linux?

Steps:

1. Navigate to file – vi example.conf
2. **:%s/<character\_or\_word>/<new-character\_or\_word>**

Eg:

:%s/a1/wh

All ‘a1’ occurences in example.conf file are replced with ‘wh’

* Shortcut for entire line delete and copy in vi editor?

Delete entire line ‘**dd**’

Copy entire line ‘**yy**’

Paste ‘**shift + p**’

* How to search for commands in history in linux?

1. Use ‘history’ command
2. Use grep, history|grep ‘keyword’
3. **Ctrl + r** then typr the keyword you want to search, it will autofill as you type (latest first). If you want that command to run just hit ‘enter’ key, else hit tab to modify that command. If you want to look at multiple matches for the keyword, keep pressing **ctrl + r**
4. First run this command in terminal to make terminal like vi editor, set -o vi

* Press escape (to change it to command mode)
* Press forward slash ‘/’
* How to search for text in vi editor?

1. Press escape
2. Forward slash ‘/’
3. ‘keyword’
4. Enter

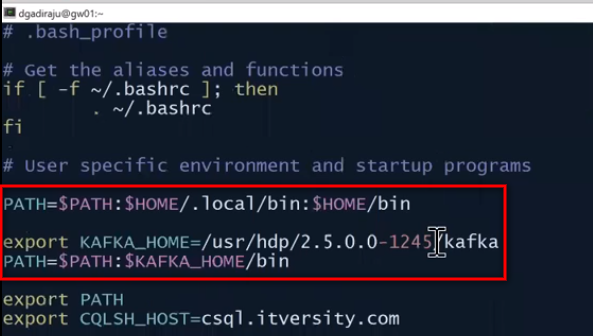
Eg: /default + enter

* How to update $PATH variable?

Steps:

Open bash\_profile file using **vi ~/.bash\_profile** where **~** represents home directory, see below example for clear instructions.

Eg: How to set kafka path to $PATH variable



**Always append new path to existing $PATH variable.**

**After updating this file, exit and re-login into terminal or run below command**

**. ~/.bash\_profile**

**HDFS:**

* Location to control environment of HDFS, YARN, etc.?

cd **/etc/hadoop/conf**

* Property in core-site.xml to give uri for namenode?

**fs.defaultFS**

* How to access namenode Web UI?

**http://fs.defaultFS\_ Property\_Value:50070**

* Where to find Information about replication factor and block size?

**hdfs-site.xml**

* Difference between copyFromLocal and put commands?

**copyFromLocal** is similar to **put** command, except that the source is restricted to a local file reference while using copyFromLocal command. Using put command you can copy from local file system and hdfs too.

* How to get size of a file?

**hadoop fs -du -s -h <path\_to\_file>** where -s is for summarize and -h is to print in readable format.

* How to get number of blocks and locations of blocks the data is stored?

**hdfs fsck <folder\_path> -files -blocks -locations**

* How to use fsck command to identify corrput files?

[**http://fibrevillage.com/storage/658-how-to-use-hdfs-fsck-command-to-identify-corrupted-files**](http://fibrevillage.com/storage/658-how-to-use-hdfs-fsck-command-to-identify-corrupted-files)

* How to get data from ITVersity Ambari to local/hdfs?

hadoop fs -get /public/crime/ /home/chaitanyapolipalli/ or

hadoop fs -cp /public/crime/csv/ /user/chaitanyapolipalli/

* What are some codecs available in core-site.xml?

1. org.apache.hadoop.io.compress.GzipCodec,
2. org.apache.hadoop.io.compress.DefaultCodec,
3. org.apache.hadoop.io.compress.SnappyCodec

* How to change read, write and execute permissions of a file in hdfs?

Read = 4

Write = 2

Execute = 1

Read + Write + execute = 4+2+1 = 7

Read + Write = 4 + 2 = 6

Read + Execute = 4 + 1 = 5

Below is the command to give read, write and execute permission to owner, read and write to group, read and execute to others for a given path

**hadoop fs -chmod 765 <hdfs\_path>**

**Eg: hadoop fs -chmod 765 /user/chaitanyapolipalli/products**

**Definitions:**

REPL – Read, Evaluate, Print, Loop (Scala)

YARN – Yet Naother Resource Negotiater

RDD - Resilient Distributed Dataset (Inmemory, distributed and resilient) (Spark)

DAG – Directed Asyclic Graph (Spark)

**Scala:**

* How to write code in paste mode? (multiple lines in scala interpreter)

After launching Scala use “**: paste** “command to enter lines of code consecutively.

* How to see metadata of a scala class?

**:javap -p Classname**

E.g. :javap -p Order

* How to create singleton object in scala?

**Syntax: Object Objectname**

E.g.: Object order { }

<https://docs.scala-lang.org/tour/singleton-objects.html>

* What is companion object in scala?

If object name and class name are same then object is called companion object.

Eg.: Class Test {}

Object Test {}

* How to search for particular text in scala interpreter?

**Syantax: :h? keyword**

E.g.: :h? Order(“

**YARN:**

* What is YARN?

Yet another resource negotiater is a distributed processing framework. It controls execution of jobs, any technology which can leverage the distributed processing framwework to run their jobs can be plugged into yarn.

* How to access YARN UI?

Open yarn-site.xml in **cd /etc/hadoop/conf**. Look for property **yarn.resourcemanager.address**

* How to access resource manager web ui?

**http://yarn.resourcemanager.webapp.address\_Property\_Value:8088**

**Sqoop:**

* Where to find sqoop user guide?

[**https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html**](https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html)

* What are Important commands for certification?

Sqoop-import

Sqoop-export

Sqoop-eval

Sqoop-list-databases

Sqoop-list-tables

Sqoop-help

* How to get help using command line?

**sqoop help <command\_name>**

**Eg. sqoop help eval**

* How to connect to mysql database on Cloudera vm using retail\_dba user?

In Cloudera:

**mysql -u username -p**

Eg.: mysql -u retail\_dba -p or mysql -u root -p

password: cloudera

In BigData labs:

**mysql -u retail\_user -h ms.itversity.com -p**

password: itversity

* How to connect to jdbc and list all databases using scoop?

sqoop list-databases \

--connect jdbc:mysql://ipaddress:portnumber(port number is optional, default 3306) \

--username retail\_dba \

--password cloudera

**Eg:** sqoop list-databases \

--connect jdbc:mysql://quickstart.cloudera:3306 \

--username=retail\_dba \

--password=cloudera \

* How to connect to jdbc and list all tables of a database using scoop?

**sqoop list-tables \**

**--connect jdbc:mysql://ipaddress:port/databasename \**

**--username retail\_dba \**

**--password cloudera**

**Eg:** sqoop list-tables \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

* How to evaluate a query without connecting to mysql? (Used in certification)

**sqoop eval \**

--connect jdbc:mysql://ipaddress:port/databasename \

--username retail\_dba \

--password cloudera \

--query “select \* from table\_name” \

**Eg:** sqoop eval \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username retail\_dba \

--password cloudera \

--query "select \* from orders"

* How to import mysql table into HDFS using sqoop?

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

--m 1

**Difference between warehouse-dir or target-dir is warehouse-dir creates a directory using table name after <hdfs\_path>, table-dir doesn’t create a directory but instead copy files in <hdfs\_path>.**

* By default sqoop checks for primary key column and performs operations specified, what if the table doesn’t contain primary key?

When there is no primary key in table, an error is thrown asking us to set mappper as 1 ( --m 1 ) or use –split-by option.

**--split-by <column\_name(numeric field)>** is used to specify explicitily on what column sqoop operations should be performed.

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

**--split-by order\_items\_order\_id**

**Or**

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

**--autoreset-to-one-mapper**

* How to save sqoop import tables in different file formats?

Types:

1. sequencefile – binary file format
2. textfile – plain text format
3. avrodatafile – binary json file format
4. parquetfile – binary columnur file format

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

**--as-avrodatafile**

* How to save sqoop import tables by using compression codecs?

By default compression is disabled, to enable use –compress or -z argument. --compress or -z is accompanied by another argument --compression-codec <c>. Where <c> can be gzip (default) if --compression-codec <c> is not specified. hadoop fs -cat or -tail command doesn’t show data when compression is used. In order to see data, copy compressed data to local file system using -get. Use gunzip <file\_name> command to unzip gzip compression.

Below code compresses in gzip format.

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

--as-textfile \

**--compress \ or -z \**

**--compression-codec <codec\_name>**

Below code compresses in Snappy format.

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

--as-textfile \

**--compress \ or -z \**

**--compression-codec** **org.apache.hadoop.io.compress.SnappyCodec**

The above fully qualified class name can be found in view /etc/hadoop/conf/core-site.xml. Search for codec and see for full list of available codecs under property **io.compression.codecs**

* What is boundary query?

By default sqoop will use query select min(<split-by>), max(<split-by>) from <table name> to find out boundaries for creating splits. In some cases this query is not the most optimal so you can specify any arbitrary query returning two numeric columns using --boundary-query argument.

If you know primary key starts from val1 and ends with val2. Then there is no point to calculate min() and max() operations. This will make sqoop command execution faster.

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

**--boundary-query “select 10000, 20000”**

Where 10000 and 20000 are min and max values of primary key column respectively.

* How to get specific columns from a table?

If you want only specific columns from a table use --columns <col,col..> along with --table argument.

If you want to perform row level transformations or write a query instead of using whole –table use --query <statement> or -e <statement>.

When using --query, --split-by should be mentioned to use more than one mappers. We cannot use –warehouse-dir attribute since we are not passing tablename, it cannot create a folder structure with that name, so insteadof –warehouse-dir we use –target-dir. Also "$CONDITIONS” placeholder should be added when using –query for convenience of sqoop developers. In linux $ is a spl symbol so you have to escape it by adding “\” before it

**Either use --table <tablename> and/or --columns <col,col,..> or just --query <statement>**

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--columns order\_item\_order\_id,order\_item\_subtotal \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

Or

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--target-dir /<hdfs\_path> \

**--query “select o.\*, sum(oi.order\_item\_subtotal) order\_revenue from orders o join order\_items oi on o.order\_id = oi.order\_item\_order\_id and \$CONDITIONS group by o.order\_id, o.order\_customer\_id, o.order\_status”**

**-m 4 \**

**--split-by order\_id**

* How to replace NULLS in string or non-string columns?

Sometimes data is null for few columns, that column can be a string or non string column. Representing string as NULL is acceptable when importing data from mysql into hdfs but representing non-string columns data as nulls is not recommended.

Use **--null-non-string <null-string>** to replace all null values.

Sometimes, fields should be terminated by some spl character instead of “comma ( , )”, use **--fields-terminated-by <character>** to achieve.

Sometimes, lines should be terminated by some spl character instead of new line, use **--lines-terminated-by <char>** to achieve

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

--input-null-non-string -1

--fields-terminated-by '\t' \

--lines-terminated-by ':' \

* How to use --where argument instead of –query to filter table data based on a condition?

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

**--where “order\_date like ‘2014-02-%’ ”** \

--append

* How to use incremental import in sqoop?

Sometimes we might not load all the data from relational databases into hdfs, as per requirement we might want to load them on a set of conditions like nightly update or weekly etc.

Instead of using --where argument for every condition we can automate the process by using incemental arguments.

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

**--check-column <colum\_name>** \

**--incremental <mode> (mode is either ‘append’ – when new rows are continously being added or ‘lastmodified’ when data is being updated)**

**--last-value <column\_value\_of\_the\_last\_import>**

Eg: sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--warehouse-dir /<hdfs\_path> **or** --target-dir <hdfs\_path> \

-m 2 \ or –num-mappers 2 \

**--check-column order\_date** \

**--incremental append**

**--last-value ‘2014-02-28’ \**

Developer’s job is to capture –last-value argument value for next increment import. It can be easily achieved using sqoop job.

* How to create a sqoop job?

sqoop job --create test \

-- import \

--connect jdbc:mysql://ms.itversity.com/retail\_export \

--username retail\_user \

--password itversity \

--table chait\_sqoop\_job\_test \

--target-dir /user/chaitanyapolipalli/sqoop\_practice/products-incremental \

--check-column ID \

--incremental append \

--last-value 0 \

--m 1

* How to import data into HIVE?

Before importing data, create database in hive using below command in hive context.

**create database databasename**

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

**--hive-import \**

**--hive-database <databasename> (not required if database name is appended to tablename)**

**--hive-table <databasename.table\_name** or **table\_name> \ (takes default table name from sqoop if not specified)**

-m 2

* How to prevent sqoop import from appending same data in the hive table if same command is run twice?

If you want to overwrite data use **--hive-overwrite**, this will drop existing table and re-create table with data.

Eg: sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--hive-import \

--hive-database <databasename> (not required if database name is appended to tablename)

--hive-table <databasename.table\_name or table\_name> \ (takes default table name from sqoop if not specified)

**--hive-overwrite \**

-m 2

If you want to **fail** sqoop command and raise **exception** for **table exists** then use **–create-hive-table** argument. Do NOT use **–hive-overwrite** and **–create-hive-table** commands in same sqoop import because they are **mutually exclusive.**

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--hive-import \

--hive-database <databasename> (not required if database name is appended to tablename)

--hive-table <databasename.table\_name or table\_name> \ (takes default table name from sqoop if not specified)

**--create-hive-table \**

-m 2

* How to handle datatypes of columns in source tables and datatypes of columns in hive tables?

Hint: Use **–map-column-hive <map>**

sqoop import \

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--table order\_items \

--hive-import \

--hive-database <databasename> (not required if database name is appended to tablename)

--hive-table <databasename.table\_name or table\_name> \ (takes default table name from sqoop if not specified)

--hive-overwrite \

**–map-column-hive order\_customer\_id=bigint,order\_id=bigint**

-m 2

* How to import all tables from mysql?

Use **sqoop import-all-tables**

**But It is not practical to know how many tables in the database have primary key if we are importing all tables, because sqoop import uses default no of mappers as 4 if a table has primary key. Sqoop import fails if primary key is not present in a table because it cannot decide how to split the data and apply --num-mappers argument. So it’s always safe to use –autoreset-to-one-mapper when using import all tables.**

sqoop **import-all-tables \**

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--warehouse-dir <hdfs\_path> \

**--autoreset-to-one-mapper**

* Points to remember during sqoop export?

**Unlike sqoop import where table is created with sqoop import, but in sqoop export a table should be created in mysql (or any traditional database) with scheme to accept the data coming from sqoop export.**

When creating table in mysql (or any traditional database) below points are important to remember:

1. **Table name can be anything (Need not be the same name created in hive)**
2. **Number of columns and their data types SHOULD match but column names need NOT match**

Hive tables are created as hdfs files, so there will be a physical path on hdfs associated with every hive table. Use describe formatted <hive\_table\_name> to get hdfs path of that table. **Default delimeter in sqoop is comma, default delimeter in hive is ctrl A ( ^A ). ASCII value of control A is 001**

Creating table in mySql:

use database\_name;

create table orderrevenue(order\_date varchar(30), revenue float);

Now run sqoop export command.

sqoop **export \**

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

**--export-dir <hdfs\_source\_path> \ (data is always exported only from hive hdfs location)**

**--table <table\_name>** \ (table name to accept exported data, mysql or any relational database table)

**--input-fields-terminated-by <char>**

--export-dir and --table are mandatory here because sqoop cannot create a new table table in mysql for writing data.

Eg: sqoop **export \**

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--export-dir /user/hive/warehouse/retail\_db.db/departments \

--table orderrevenue \

--input-fields-terminated-by ‘\001’

-m 4 or –num-mappers 4

* How to handle column mapping if target table has more columns than the table data that we are exporting?

Sometimes we must handle sqoop export scenarios where target table columns data-type can mismatch with hive table columns data type. Also, sqoop export need not have data to populate all columns in target table, in such scenarios we should use **--columns <col1,col2..> to specify in what columns data should be populated** .col1, col2 order should match source data structure and column names should match mysql target table. Eg: --columns <order\_date,order\_revenue> , hive table columns output <order\_date,order\_revenue>. Any mismatch in ordering will fail sqoop export. Again,

Source table output: **<order\_date,order\_revenue>**

Target table structure: **<total\_orders\_revenue, date ,order\_status>, here order\_status is not primary key so it need not have value but if the table have primary key then it is mandatory to pass value for primary key column. Here the order of source tables should not change in --columns attribute, if the column names match with target table then data will be populated (even if order of target table is different).**

sqoop **export \**

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--export-dir /user/hive/warehouse/retail\_db.db/departments \

--table order\_revenue \

**--columns date, total\_orders\_revenue**

--input-fields-terminated-by ‘\001’

-m 4 or –num-mappers 4

* What is stage table?

Stage table is a concept where we have a intermediate table where data is loaded temporarily, only after all the data is loaded sucessfully in this intermediate table then only it is pushed into final table.

Sometimes, when data is exported from hive table to mysql table, because of some error all the data from hive table might not get exported into mysql table. Because no of mappers are usaully more than 1 and when we use multiple mappers, one mapper might fail for some integrity constraint error then the data contained by this mapper is never updated so there will be an error in no of records in source and target tables. In order to address this issue we use stage tables.

* How to create stage table?

It is similar to creating a regular table. Some stage tables might contain constraints and some might not.

Syntax for creating staging table in mysql:

Create table daily\_revenue\_table\_staging

(Order\_date varchar(30) primary key,Revenue float);

To make sure exported data is clean, just add –staging-table <stage-table-name> to sqoop export command. If there is any error, no data is exported to final target table. –clear-staging-table argument is used to delete any data present in that table for next use because it might have primary key and it violates dulpication of data if this argument is not used. Every time we use stage table argument, the table has to be empty. If an export is succesful the staging table is cleared.

sqoop **export \**

--connect jdbc:mysql://quickstart.cloudera:3306/retail\_db \

--username=retail\_dba \

--password=cloudera \

--export-dir /user/hive/warehouse/retail\_db.db/departments \

--table order\_revenue \

--staging-table daily\_revenue\_table\_staging \

--clear-staging-table \

--input-fields-terminated-by ‘\001’

-m 4 or –num-mappers 4

**Spark:**

* Where to find spark documentation?

<https://spark.apache.org/docs/1.6.3/programming-guide.html>

* How to get details about No of Executors and Memory for each task?

Navigate to **cd /etc/spark/conf**

View **spark-env.sh** file

* How to initialize programatically?

import org.apache.spark.{SparkConf, SparkContext}

val conf = new SparkConf().setAppName(“Example\_Application”).setMaster(“yarn-client”);

val sc = new SparkContext(conf);

sc.getConf.getAll.foreach(println)

* How to get environment details?

sc.getConf.getAll.foreach(println)

* How to launch spark-shell on yarn?

By default when spark-shell command is used it runs locally but to make it run on yarn use below comamnd.

**spark-shell --master yarn**

**Note:** spark-shell = scala + spark dependencies + sc (spark context) + sqlContext (Spark SQL context)

Here sc is a web service and a port is associated with it. To have a look at web interface, copy tracking url that is genrated when above command is run and paste it in browser. You get to see spark jobs that are running in current session. If u exit from current spark-shell context in cmd, refresh web ui then it automatically redirects to history server from proxy (observe change in url).

Always try to use spark resouces wisely. Very important parameters to look out for include:

**--num-executors, --executor-cores and --executor-memory parameters**

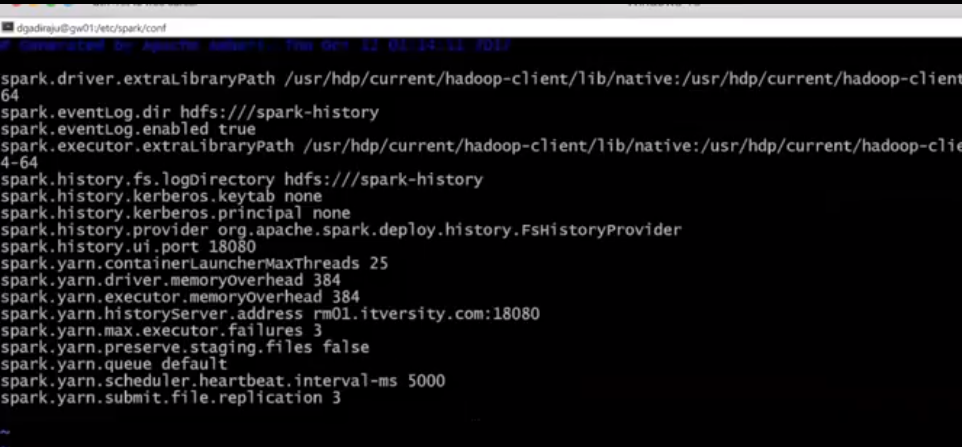
spark-shell --help gives you a insight on how to call and use these arguments.

By default when u launch spark-shell or spark-submit, it is launched with **two** executors (internally two jvm’s are created on the cluster). These two executeors are the ones which process the data. So sometimes they are too less or too high for the data we want to process.

* Where to look for default spark configuration parameters?

Go to path **vi /etc/spark/conf/spark-defaults.conf**

Below is the output

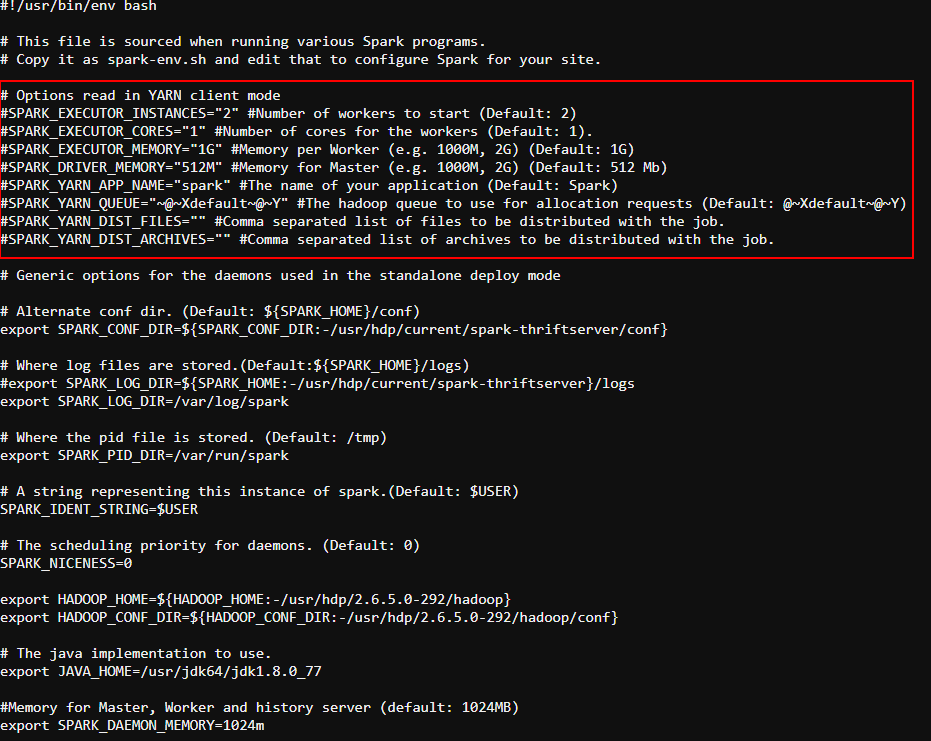


Go to path **vi /etc/spark/conf/spark-env.sh**

Below is the output:

Look for #SPARK\_EXECUTOR\_ISTANCES= ”2” and #SPARK\_EXECUTION\_MEMORY= “1G”

Defaults are 2 executors and executor memory 1 GB



* How to increase or decrease –num-executors, --executor-cores and –executor-memory when launching spark?

Eg: **spark-shell --master yarn \**

**--conf spark.ui.port=14521 \**

**--num-executors 6 \**

**--executor-memory 2G \**

**--executor-cores 2**

The arguments should be used very wisely and they are highly dependent on the amount of data being processed. The above command starts spark on yarn on port 14521 with 6 executors and each executor having 2 executor cores and 2G of memory. So we will have 6 (--num-executors) \* 2 (--executor-cores) = 12 threads running with 6 (--num-executors) \* 2G (--executor-memory) = 12G memory to finish a job. This is how we fine tune job execution.

* What is RDD?

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes. When sc is used, it returns RDD.

* What is DataFrame?

Like an RDD, a DataFrame is an immutable distributed collection of data. Unlike an RDD, data is organized into named columns, like a table in a relational database. Designed to make large data sets processing even easier. When sqlContext is used, it return DataFrame.

* How to create RDD in spark-shell?

**val orders = sc.textFile(“relative or absolute hdfs\_path until parent folder”)** // Works only for HDFS path

where sc is spark context. All files in this path will be loaded into orders variable (need not specify individual file names (part-m-00000) in path).

------------------------------------------

**import scala.io.Source**

**val ordersRaw = Source.fromFile(“fully qualified local path including file name”).getLines.toList** //since sc wont be available when loading data from local path, we have to use fully qualified scala API and this works only for local path.

* How to convert any scala collection to RDD?

**val ordersRDD = sc.parallelize(ordersRaw)**

* How to read orc, json, parquet and avro file formats?

Spark context object sc doesn’t have api’s to read data from above file formats, rather we use sqlContext object api’s to read above file formats (orc, json and parquet). To read avro file format, external dependencies needs to be added (--packages groupId: artifactId: version or --jars “jar\_path” are added when starting spark-shell. Also import com.databricks.spark.avro in program to read avro files Eg:., --packages com.databricks: spark-avro\_2.10: 2.0.1)

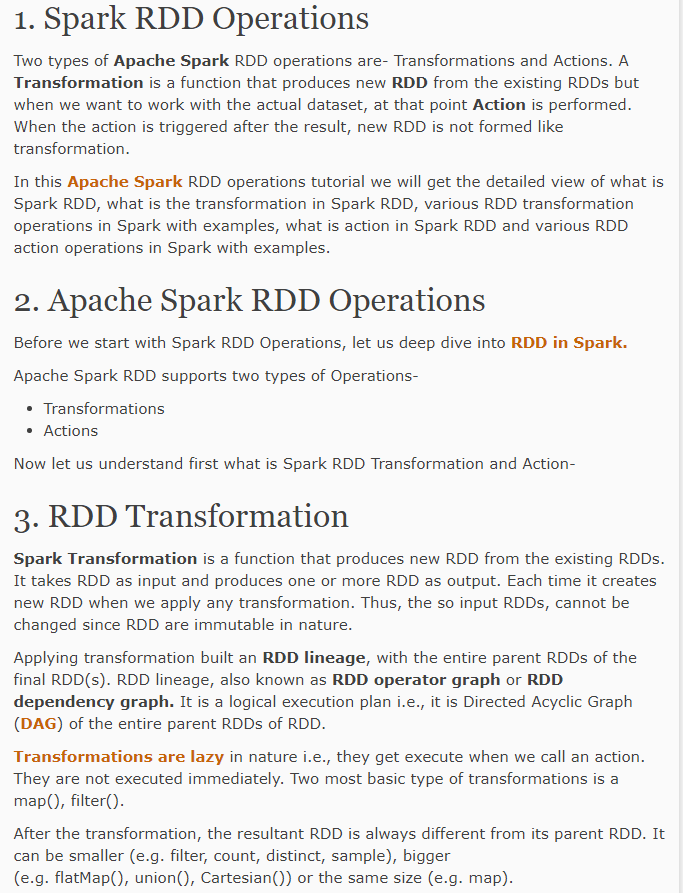
**sqlContext.load or sqlContext.read.<file\_format\_type>**

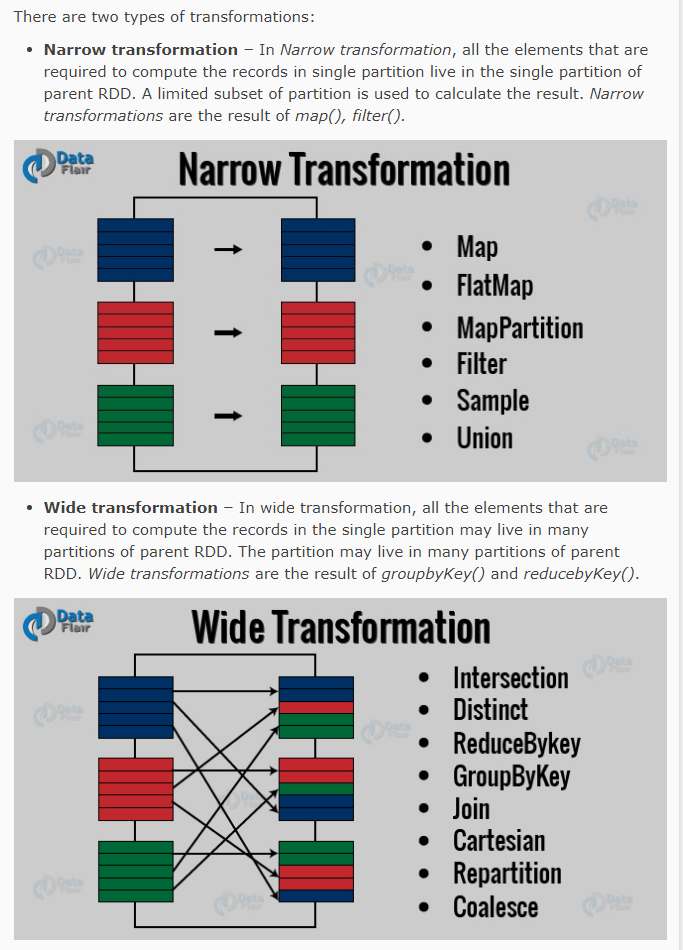
Eg: val example = sqlContext.read.json(“<hdfs\_path>”)

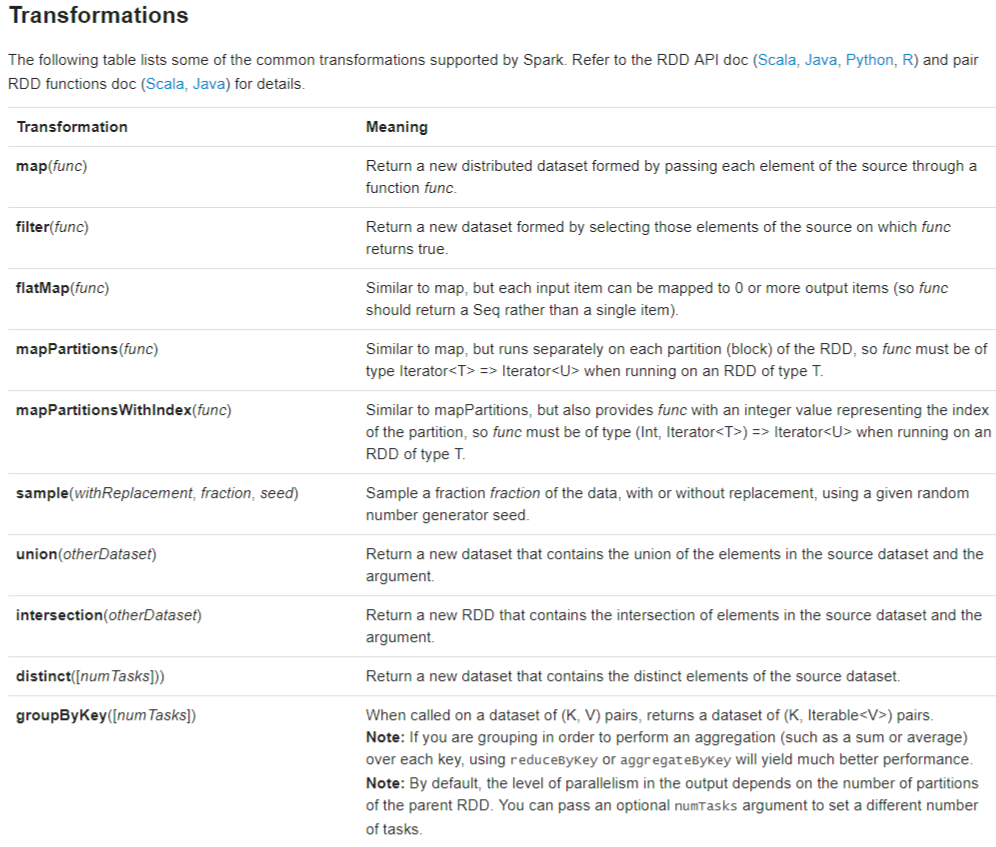
Or

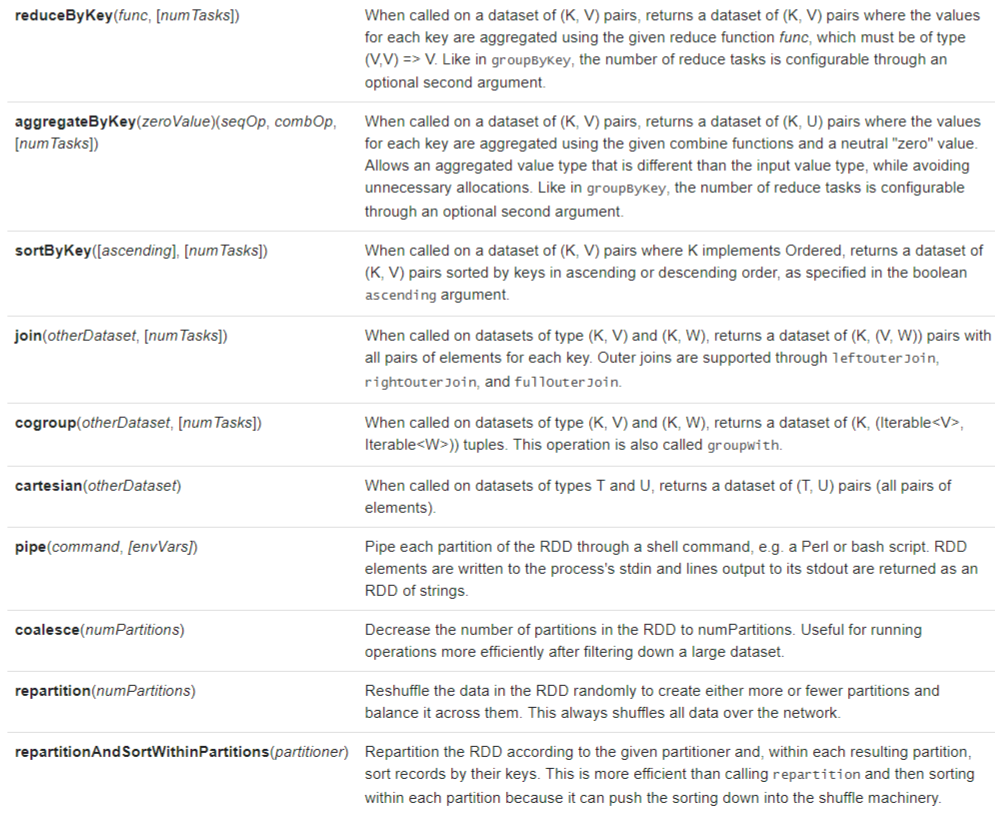
val example = sqlContext.load(“<hdfs\_path>”,”json”)

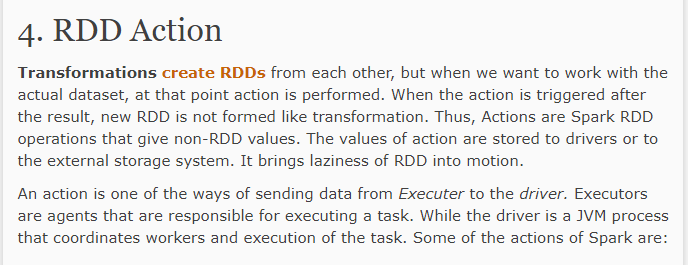
* What are Transformations and Actions in Spark?

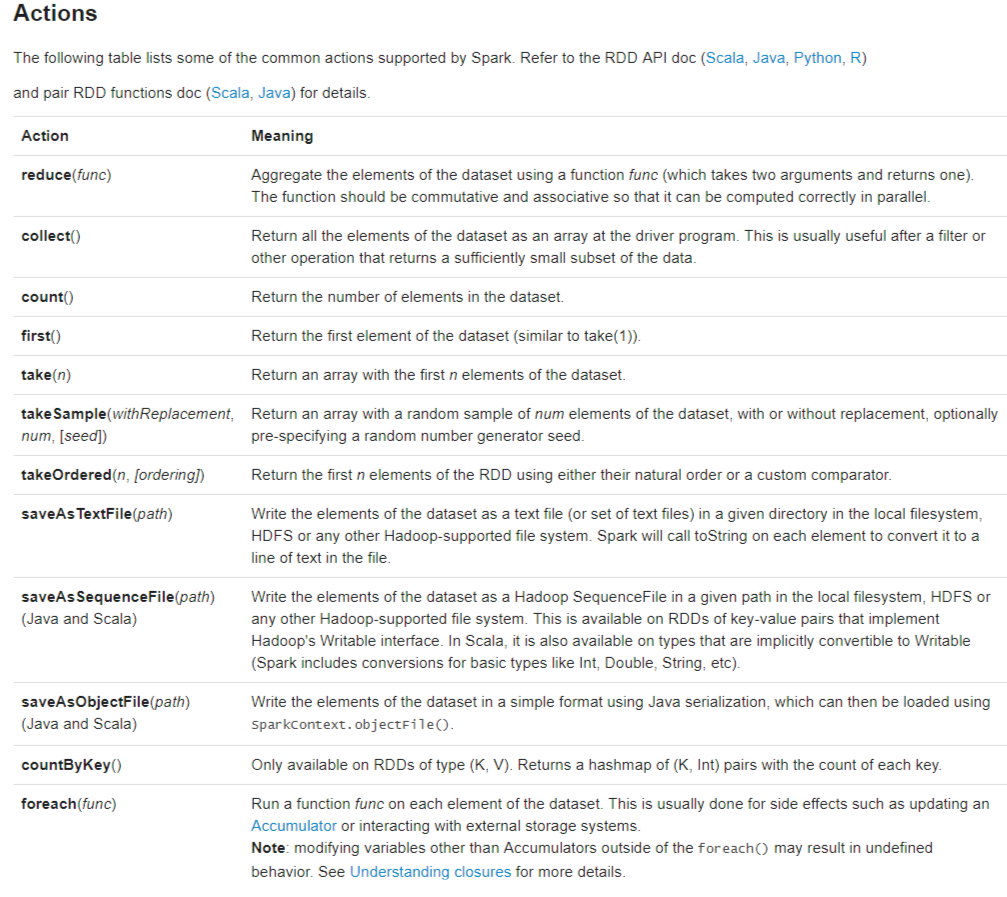












* What is difference between reduce(func) in Actions and reduceByKey(func, [numTasks]) in Transformations?

**reduce(func)** from Actions is used for global sort. Eg: If you want to compute complete revenue of order\_items\_subtotal. “reduce” in Actions cannot be used on a Key, it is used for global aggregation like sum, min, max, etc

**reduceByKey(func, [numTasks])** from Transformations is used to compute based on key like “order\_date”, ”order\_id”, etc.

* What is difference between groupByKey([numTasks]), reduceByKey(func, [numTasks]) and aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])?

**groupByKey([numTasks])** is inferior when compared to other aggregate functions because it does not implement combiner logic.

**reduceByKey(func, [numTasks])** and **aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])** both can be used only for aggregation and both use combiner. Only difference is if combiner logic is different from reducer logic we use aggregateByKey(zeroValue)(seqOp, combOp, [numTasks]) Eg: Avg of range of integers, if combiner and reducer logic are same then we use reduceByKey(func, [numTasks]) Eg: Sum of all intergers.

* When to use foreach()?

**foreach** can be used only on array, list or map (typical scala collections). It can be used on an RDD directly, it doesn’t show any results to console though RDD has data in it.

Eg: orders.map(order=> (order.split(“,”),””)).foreach(println). Here map **returns an RDD** so this foreach will not print anything.

Eg: orders.map(order=> (order.split(“,”),””).countByKey.foreach(println)). Here countByKey **return a map** so foreach can be used and it prints count of records of each key.

* How to achieve global ranking of dataset and how to use sortByKey()?

Global ranking of dataset is achieved using sortByKey().

To use sortByKey() the data has to be paired RDD <K , V>. Here K implements Ordered (all value classes or the classes related to primitive type such as int, float, string, double, etc.. implement Ordered) and only this key is used to sort the data and by default it sorts in ascending order, to change it to descending order use sortByKey(false).

* How to get top n records when records ranked globally?

Apply sortByKey() on <K, V> dataset and then take(n) gives top n records for globally sorted data. But here ranking works only when duplicates are not addressed, meaning 1 can repeat multiple times. Eg: (1,1,1,2,2,2,3,4,4,5,6…). This is one way, another way to get top n ranking records is by **not** using map and sorByKey, use takeOrdered(n) instead.

* How to achieve ranking within each key (within each product in products table)?

<https://www.youtube.com/watch?v=zmVAPY6_X-M&list=PLf0swTFhTI8rDQXfH8afWtGgpOTnhebDx&index=79>

* Points to remember while performing Set operations?

If you are trying to perform set operations between two data-sets then number of fields, attributes or columns have to be same and datatypes also have to be similar in both datasets.

* What are set operations in Spark?

**Union** and **Intersection**

* How to save RDD to HDFS?

Syntax: **<some\_rdd>.saveAsTextFile(hdfs\_path)**

With scala and java we can save rdd as Sequence or Object file but with Python we can save only as textfile.

* How to save RDD to HDFS with compression?

Syntax: **<some\_rdd>.saveAsTextFile(hdfs\_path, classOf(<some\_compressoin\_codec>))**

**<some\_compressoin\_codec>** can befound in /etc/hadoop/conf/core-site.xml, search for codec or compression and see for codec classes and use one of them.

Eg: revenue\_rdd.saveAsTextFile

(“/user/chaitanyapolipalli/snappysave”, classOf[org.apache.hadoop.io.compress.SnappyCodec])

org.apache.hadoop.io.compress.GzipCodec,

org.apache.hadoop.io.compress.DefaultCodec,

org.apache.hadoop.io.compress.SnappyCodec

Once data is compressed, you can view data using hadoop fs -ls /user/chaitanyapolipalli/snappysave command, data should be visible with no errors.

You can also view it in spark context using

sc.textFile(“user/chaitanyapolipalli/snappysave”).collect.foreach(println)

* How to support orc, json, parquet and avro (with databricks plugin) file formats?

To save data in above mentioned file formats, data needs to be in **data frames and not RDD**.

val ordersDF = sqlContext.read.json(“/user/chaitanyapolipalli/retails\_db\_json/orders”)

ordersDF.save(“/user/chaitanyapolipalli/parquetsave”,”parquet”)

Syntax: **ordersDF.save(“<hdfs\_path>”,”mode”)** or **ordersDF.write.orc(“<hdfs\_path>”)**

* How to read saved data in above step (i.e., in parquet format)?

Syntax: **sqlContext.load(“<hdfs\_path>”,”mode”).show** or **sqlCOntext.read.parquet(“<path>”).show**

sqlContext.load(“/user/chaitanyapolipalli/parquetsave”,”parquet”).show

* How to import data from local file system to create RDD in Spark?

**Syntax:**

import scala.io.Source

val productsRaw = Source.fromFile(“<file\_path\_until\_part-0000>”).getLines.toList

Here we have to convert into List because getLines returns a Iterator but in order to create RDD, parallelize takes a sequence type. Sequence is parent of List class in scala collections so converting into List will satisfy parallelize method.

Eg:

import scala.io.Source

val productsRaw =

Source.fromFile(“/home/chaitanyapolipalli/retail\_db/products/part-00000”).getLines.toList

val products = sc.parallelize(productsRaw)

* How to convert a tuple to a string?

Sometimes we might have a tuple as final output but we need a simple string with some delimiter as desired output. So in such cases use **mkString(“desired\_param”)** on tuple or any collection to make it a string with desired\_param as separator. Here all the elements in tuple has to be string.

Eg:

revenue.map(p=> p.\_2.mkString(“,”)) where p is of type (O1, O2, O3) where O1,O2,O3 are strings and the result will be O1,O2,O3

* How to import avro packages or jars when launching spark?

Either –packages or –jars should be used but not both. Either of the information is provided.

Also import below file after launching spark-shell

**import com.databricks.spark.avro.\_**

spark-shell --master yarn \

--conf spark.ui.port=14520 \

--num-executors 10 \

--executor-memory 3G \

--executor-cores 2 \

**--packages groupid:artifactid:version** //use this if packages are given in problem statement

**or**

**--jars "<jar\_file\_path>"** //use this if jar file path is given instead of packages

Eg: spark-shell --master yarn \

--conf spark.ui.port=14520 \

--num-executors 10 \

--executor-memory 3G \

--executor-cores 2 \

**--packages com.databricks:spark-avro\_2.10:2.0.1** //use this if packages are given in problem statement

**or**

**--jars "<jar\_file\_path>"** //use this if jar file path is given instead of packages

**import com.databricks.spark.avro.\_**

* How to compress and save data when using different file formats?

Below is the example for parquet but can be used for other file formats as well.

**sqlContext.setConf("spark.sql.parquet.compression.codec", "gzip")**

The value highlighted could be one of the four : uncompressed, snappy, gzip, lzo

**Hive & Spark SQL:**

* Where to find hive documentation?

Search for “**hive language manual**” in google and look for below url

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual>

* What is alias for hadoop fs in hive context?

**dfs** is alias for **hadoop fs** in hive context, dfs should be used **only** in hive context.

* How to create a database in hive?

Syntax: **create database databasename;**

Eg: create database chaitanyapolipalli;

* What is the base hdfs directory for all hive databases?

We can find this in two ways:

1. Just run **set hive.metastore.warehouse.dir;** in hive context will return you this directory **/apps/hive/warehouse**

or

1. Go to **vi /etc/hive/conf/hive-site.xml** and search for hive.metastore.warehouse.dir parameter. Here parameter value will be **/apps/hive/warehouse**

* How to see all databases on hdfs that are created using hive?

Syntax: **dfs -ls /apps/hive/warehouse;** Will list you all databases with .db extension created in hive.

Eg: dfs -ls /apps/hive/warehouse should show chaitanyapolipalli.db file that is created above.

* What information is required to create a hive table from exiting well structured data?

1. Open file and see what is the file format
2. Look for delimiters
3. Look for column names and data type

* How to create a table in hive?

Syntax:

**create table table\_name**

**(col1 datatype, col2 datatype) row format delimited fields terminated by ‘<delimiter>’**

**stored as textfile;** (this line is optional for textfile, but for any other file format you should specify its type)

Eg:

create table orders(order\_id int,order\_date string,order\_customer\_id int,order\_status string) row format delimited fields terminated by ',' stored as textfile;

* Difference between Managed and External hive table?

Managed and External Tables

By default Hive creates managed tables, where files, metadata and statistics are managed by internal Hive processes. A managed table is stored under the hive.metastore.warehouse.dir path property, by default in a folder path similar to /apps/hive/warehouse/databasename.db/tablename/. The default location can be overridden by the location property during table creation. If a managed table or partition is dropped, the data and metadata associated with that table or partition are deleted. If the PURGE option is not specified, the data is moved to a trash folder for a defined duration.

Use managed tables when Hive should manage the lifecycle of the table, or when generating temporary tables.

An external table describes the metadata / schema on external files. External table files can be accessed and managed by processes outside of Hive. External tables can access data stored in sources such as Azure Storage Volumes (ASV) or remote HDFS locations. If the structure or partitioning of an external table is changed, an MSCK REPAIR TABLE table\_name statement can be used to refresh metadata information.

Use external tables when files are already present or in remote locations, and the files should remain even if the table is dropped.

Managed or external tables can be identified using the DESCRIBE FORMATTED table\_name command, which will display either MANAGED\_TABLE or EXTERNAL\_TABLE depending on table type.

Statistics can be managed on internal and external tables and partitions for query optimization.

* How to load data into hive table?

Syntax:

Loading data from local file system:

load data **local** inpath ‘<local\_path>’ **into** table table\_name; **(use this if you want to append data to already existing table)**

Eg: load data local inpath ‘/home/chaitanyapolipalli/retail\_db/orders’ into table orders;

load data local inpath ‘<local\_path>’ **overwrite into** table table\_name; **(use this if you want to overwrite data in existing table everytime)**

Eg: load data local inpath ‘/home/chaitanyapolipalli/retail\_db/orders’ overwrite into table orders;

Loading data from hdfs path:

load data inpath ‘<hdfs\_path>’ **into** table table\_name; **(use this if you want to append data to already existing table)**

Eg: load data inpath ‘/user/chaitanyapolipalli/retail\_db/orders’ into table orders;

load data inpath ‘<hdfs\_path>’ **overwrite into** table table\_name; **(use this if you want to overwrite data in existing table everytime)**

Eg: load data inpath ‘/user/chaitanyapolipalli/retail\_db/orders’ overwrite into table orders;

* In which clause we can use columns alias names?

Only in **“order by”** clause we can use column alias, everywhere else we have to use logic.

Eg:

select o.order\_date, round(sum(order\_item\_subtotal),2) revenue

from orders o join order\_items oi on o.order\_id = oi.order\_item\_order\_id

where o.order\_status = 'COMPLETE' or o.order\_status = 'CLOSED'

group by o.order\_date

order by o.order\_date,revenue desc

limit 10;

* How to see created tables on hdfs?

As we know, all databases/tables created in hive context are available under /apps/hive/warehouse directory.

dfs -ls /apps/hive/warehouse/chaitanyapolipalli.db/orders will list the part-\* files created above.

Just run **select \* from orders limit 10;** should return data loaded into orders table.

* Why should we not specify delimiters when using any other file format other than textfile?

When using other file formats like orc,avro,parquet,etc.., they store metadata as well as data for each and every record so we need not specify delimiters explicitly.

* How to insert data into these type of file formats (orc, parquet, etc)?

It’s a two step process, first load data into regular table, then copy data from that table into file formatted table. You can not load a text file into an ORC (or any other file format other than textfile) table directly. The typical practice is to create a temporary Hive table, load the text file into the temporary table, then let Hive populate the ORC table.

**Eg:** Create a table orders with txtfile file format, copy data from this table into orc type orders table.

**Syntax:**

insert into table polipalli\_orc.orders select \* from chaitanyapolipalli\_retail\_db\_txt.orders;

insert into table polipalli\_orc.order\_items select \* from chaitanyapolipalli\_retail\_db\_txt.order\_items;

* How to see table columns, saved location, Input and Output format and Delimiter info?

Syntax: **describe formatted <table\_name>;**

**Eg:** describe formatted orders;

* How to run hive queries in spark context?

Run spark-shell –master yarn –conf spark.ui.port=14852. Here sc and sqlContext are created, sqlContext is nothing but hive context created using sc. If we want to run any sql queries just use sqlContext.

Syntax: sqlContext.sql(<any hive query>)

Eg: sqlContext.sql(“use chaitanyapolipalli\_retail\_db\_txt”)

If you are using a query to print data, use show method.

Eg: sqlContext.sql(“select \* from orders limit 10”).show

* How to see and use functions in hive?

To see all supported functions, use **show functions** command in hive context.

To know how to use a function, **describe function\_name;**

To use a function in a query, **select order\_status, length(order\_status) from orders;**

* What are important string functions in hive?

1. Substr or substring
2. Instr
3. Like
4. Rlike
5. Length
6. Lcase or lower
7. Ucase or upper
8. Initcap
9. Trim, ltrim and rtrim
10. Cast
11. Lpad and rpad
12. Split
13. concat

* What are some important date functions in hive?

1. Current\_date
2. Current\_timestamp
3. Date\_add
4. Date\_format
5. Date\_sub
6. Datediff
7. Day
8. Dayofmonth
9. To\_date
10. To\_unix\_timestamp
11. To\_utc\_timestamp
12. From\_unixtime
13. From\_utc\_timestamp
14. Minute
15. Month
16. Months\_between
17. Next\_day

* What is the replacement for “If else” in sql?

Answer: function “**case**”

To know how to use this function, **describe function case;**

* What are row level transformations?

Sometimes we apply logic to make some changes at record level. Eg: SSN’s are masked with \*\*\*, standardising phone numbers, address,etc.

* How to perform Joins in Sql?

select \* from orders limit 10;

select \* from order\_items limit 10;

select o.\*, c.\* from orders o join customers c

on o.order\_customer\_id = c.customer\_id

limit 10;

select o.\*, c.\* from customers c left outer join orders o

on o.order\_customer\_id = c.customer\_id

limit 10;

select count(1) from orders o join customers c

on o.order\_customer\_id = c.customer\_id;

select count(1) from customers c left outer join orders o

on o.order\_customer\_id = c.customer\_id;

select \* from customers c left outer join orders o

on o.order\_customer\_id = c.customer\_id

where o.order\_customer\_id is null;

* How to perform Aggregations?

select order\_status, count(1) as order\_count from orders

group by order\_status;

select o.order\_id, sum(oi.order\_item\_subtotal) as total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

group by o.order\_id limit 10;

select o.order\_id, o.order\_date, o.order\_status, sum(oi.order\_item\_subtotal) as total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

group by o.order\_id, o.order\_date, o.order\_status limit 10;

select o.order\_id, o.order\_date, o.order\_status, sum(oi.order\_item\_subtotal) as total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

group by o.order\_id, o.order\_date, o.order\_status

having sum(oi.order\_item\_subtotal) > 1000;

select o.order\_id, o.order\_date, o.order\_status, sum(oi.order\_item\_subtotal) total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

where o.order\_status in ('COMPLETE','CLOSED')

group by o.order\_id, o.order\_date, o.order\_status

having sum(oi.order\_item\_subtotal) > 1000;

select o.order\_date, round(sum(oi.order\_item\_subtotal),2) total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

where o.order\_status in ('COMPLETE','CLOSED')

group by o.order\_date;

select o.order\_date, round(sum(oi.order\_item\_subtotal)) total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

where o.order\_status in ('COMPLETE','CLOSED')

group by o.order\_date;

* How to sort data?

//global sort

select o.order\_id, o.order\_date, o.order\_status, round(sum(oi.order\_item\_subtotal),2) total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

where o.order\_status in ('COMPLETE','CLOSED')

group by o.order\_id, o.order\_date, o.order\_status

order by o.order\_date, total desc;

//sort within each date--here date may be sorted randomly but revenue is sorted within each date

select o.order\_id, o.order\_date, o.order\_status, round(sum(oi.order\_item\_subtotal),2) total from orders o join order\_items oi

on o.order\_id = oi.order\_item\_order\_id

where o.order\_status in ('COMPLETE','CLOSED')

group by o.order\_id, o.order\_date, o.order\_status

distribute by o.order\_date sort by o.order\_date, total desc;

The advantage of using “distribute by col\_name sort by col\_name” gives better performance than “order by” since “order by” sorts data globally and it cannot scale up that easily. Where as if we use “distribute by and sort by” in conjunction data can be scaled up easily and optimal performance for this kind of use cases but side effect is data is not globally sorted.

* Difference between Union and Unionall?

Union eliminates duplicates while unionall returns with duplicates.

* What are aggregate functions under Analytics Functions?

COUNT

SUM

MIN

MAX

AVG

* Is alias important in Hive nested queries?

It it **mandatory** to **alias** nested queries in hive.

* What are Ranking functions in Analytical Functions?

RANK

ROW\_NUMBER

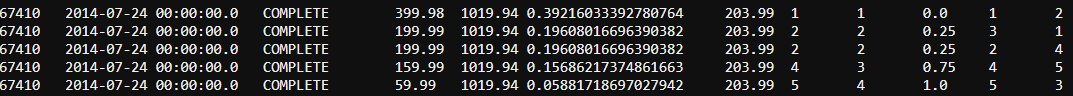
DENSE\_RANK

CUME\_DIST

PERCENT\_RANK

NTILE

* Difference between rank and dense\_rank?



Rank skips next number if same rank is given twice, dense rank doesn’t skip and continues to give rank. Observe columns 8 (Rank) and 9 (Dense Rank) .

* How to create Data Frame from RDD?

Syntax to create RDD:

val ordersRDD = sc.textFile(“/user/chaitanyapolipalli/retail\_db\_orders”)

ordersRDD.take(1).foreach(println) will return

1,2013-07-25 00:00:00.0,11599,CLOSED

**In order to convert RDD to Data frame, we have to extract each and every field seperately and type cast, then we have to convert into data frame. Converting a RDD into DF is nothing but applying a structure to RDD**. Here ordersRDD is a string, so type casting each and every column into its appropriate data type.

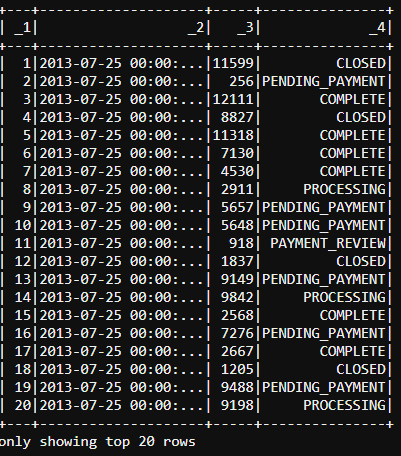
val ordersDF

= ordersRDD.map(o=>(o.split(",")(0).toInt,o.split(",")(1),o.split(",")(2).toInt,o.split(",")(3)))**.toDF**

Will create DF as **org.apache.spark.sql.DataFrame = [\_1: int, \_2: string, \_3: int, \_4: string]**

Where \_1, \_2, \_3 and \_4 are default column names.

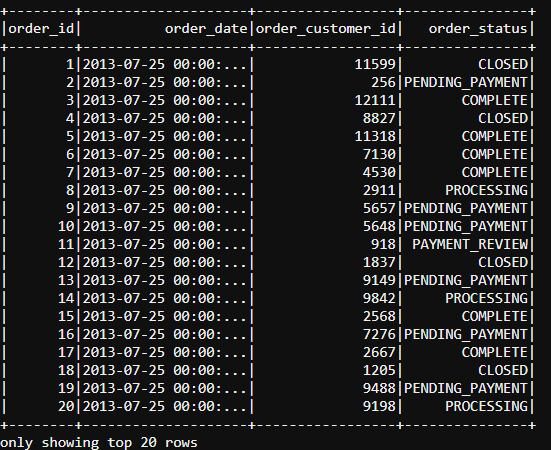
**ordersDF.show()** will return top 20 rows. Eg:



Here column names are shown as \_1, \_2, \_3 and \_4. In order to show appropriate column names use below command.

val ordersDF

= ordersRDD.map(o=>(o.split(",")(0).toInt,o.split(",")(1),o.split(",")(2).toInt,o.split(",")(3))).**toDF("order\_id","order\_date","order\_customer\_id","order\_status")**



* How to print schema of DF?

Syntax: **<RDD>.printSchema**

**Eg:** ordersDF.printSchema

* How to use sql queries on DF?

In order to run sql ques=ries against DF, first register DF to a temp table as shown below.

Syntax: **<DF>.registerTempTable(“<table\_name>”)**

Eg: ordersDF.registerTempTable(“ordersDFTable”)

* How to run Sql Queries on temp table?

Syntax: **sqlContext.sql(“<query>”).show()**

**Eg:** sqlContext.sql(“select \* from ordersDFTable limit 10”).show()

* How to read files from local file system in spark, create rdd, create df, print data from df?

**Syntax:**

val productsRaw

= scala.io.Source.fromFile(“/home/chaitanyapolipalli/products/part-00000”).getLines.toList

val productsRDD = sc.parallelize(productsRaw)

val productsDf = productsRDD.map(p=> {

(p.split(",")(0).toInt, p.split(",")(2)) }).

toDF("product\_id","product\_name") (we are interested only in product\_id & product\_name only)

productsDf.registerTempTable("productsDFTable")

sqlContext.sql("select \* from productsDFTable").show()

* How to control no of threads to run your query?

Syntax: **sqlContext.setConf(“spark.sql.shuffle.partitions”,”<no\_of\_threads>”)**

Eg: sqlContext.setConf(“spark.sql.shuffle.partitions”,”2”)

* Query for getting order\_date, product\_name, daily\_revenue\_per\_product in ascending order by order\_date and descending order by daily\_revenue\_per\_product?

sqlContext.sql(

"select o.order\_date, p.product\_name, sum(oi.order\_item\_subtotal) daily\_revenue\_per\_product " +

"from orders o join order\_items oi " +

"on o.order\_id = oi.order\_item\_order\_id " +

"join productsDFTable p on p.product\_id = oi.order\_item\_product\_id " +

"where o.order\_status in ('COMPLETE','CLOSED') " +

"group by o.order\_date, p.product\_name " +

"order by o.order\_date, daily\_revenue\_per\_product desc").show

* Steps to save above query result to hive database?

1. Create a database in hive

sqlContext.sql("create database chaitanyapolipalli\_daily\_revenue")

1. Create a table in hive, since we are already connected to different database, prefix table name with database name.

sqlContext.sql("create table chaitanyapolipalli\_daily\_revenue.daily\_revenue "+

"(order\_date string, product\_name string, daily\_revenue\_per\_product float) " +

"stored as orc")

1. Assign total output of query to a variable:

val daily\_revenue\_per\_product\_final\_result = sqlContext.sql(

"select o.order\_date, p.product\_name, sum(oi.order\_item\_subtotal) daily\_revenue\_per\_product " +

"from orders o join order\_items oi " +

"on o.order\_id = oi.order\_item\_order\_id " +

"join productsDFTable p on p.product\_id = oi.order\_item\_product\_id " +

"where o.order\_status in ('COMPLETE','CLOSED') " +

"group by o.order\_date, p.product\_name " +

"order by o.order\_date, daily\_revenue\_per\_product desc")

1. **insertInto** command lets you insert data into hive table, **write** is used to save data into file system with multiple file formats like orc, parquet, json, text, etc, or **saveAsTable** can be used to directly insert into existing table. (these commands can be found when you click tab for “daily\_revenue\_per\_product\_final\_result” dataframe variable )

**daily\_revenue\_per\_product\_final\_result.insertInto("chaitanyapolipalli\_daily\_revenue.daily\_revenue")**

1. Verify data if it is inserted into table

**sqlContext.sql("select \* from chaitanyapolipalli\_daily\_revenue.daily\_revenue").show**

**DataFrame Specific Operations:**

1. **<Df>.show** will print first 20 records, <Df>.show(100) will show 100 records.
2. **<Df>.save** will save to local file system in any file format, eg: <df>.save(“<hdfs\_path>”,”json”).
3. **<Df>.write** will give methods to write in different file formats. <df>.write.json(“<hdfs\_path>”)
4. **<df>.rdd** will convert df to rdd of sql type row. Eg: <df>.rdd.take(10).foreach(println)
5. Apart from sqlContext, data frames also has few sql type functions.

Eg: **<df>.select(“column1”,”column2”).show** will show top 20 records.

1. **<df>.filter(<df>(“filter\_condition”) === “condition”)**.count

Eg: <df>.filter(<df>(“order\_date”) === “2013-7-25 00:00:00.0”).show

<df>.filter(<df>(“order\_date”) === “2013-7-25 00:00:00.0”).count

**Flume, Kafka & Spark Streaming:**

* Why use flume, kafka or spark streaming technologies when sqoop is available for data ingestion?

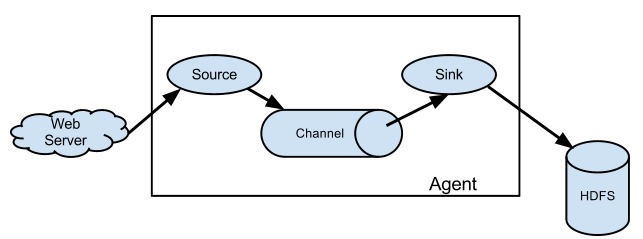
Sqoop is typically used once a day or once a week, but sometimes we might run into scenarios where data has to be ingested immediately they are availabe and analytics are run on them.

* Where to find flume documentation?

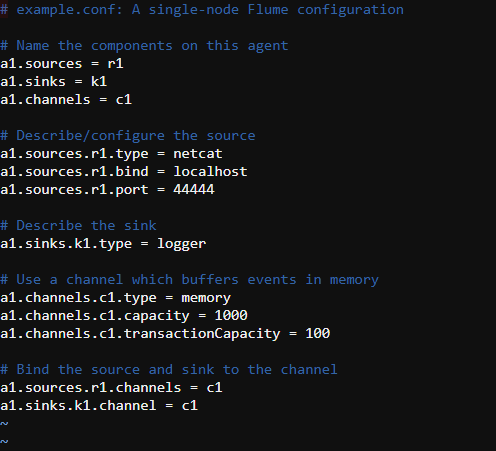
Search for “Flume user guide”, look for 1.6.0 user guide since certification uses 1.6.0 but latest available flume version is 1.8.0 and navigate to <http://flume.apache.org/FlumeUserGuide.html>

* What is a flume agent?

Flume agent is a combination of source, channel and sink. Usually source is connected to a databse or webserver for logs. Sink is usally connected to hdfs or some place where output from agent is written to, channel is the one which integrates channel and sink.



* Basic flume example?



Where ‘a1’ is name of agent, r1 is name of source, k1 is name of sink, c1 is name of channel.

* What is the command to start a flume agent?

Syntax:

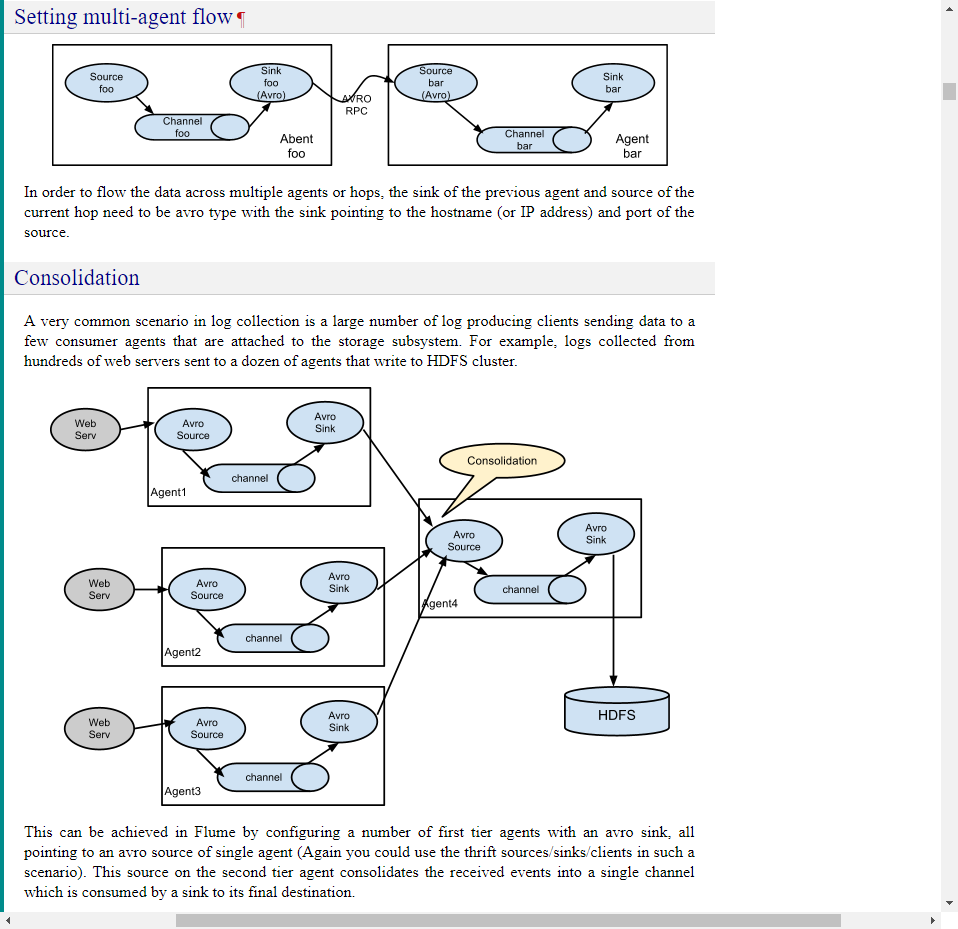
**flume-ng agent --name <agent\_name> --conf-file <configuration\_file\_path>**

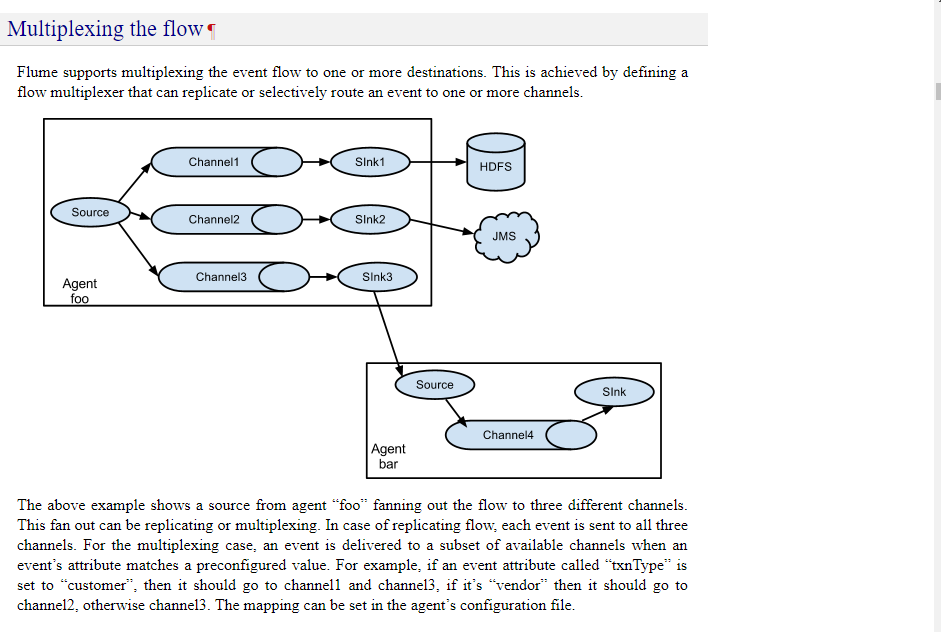
Or

**flume-ng agent -n <agent\_name> -f <configuration\_file\_path>**

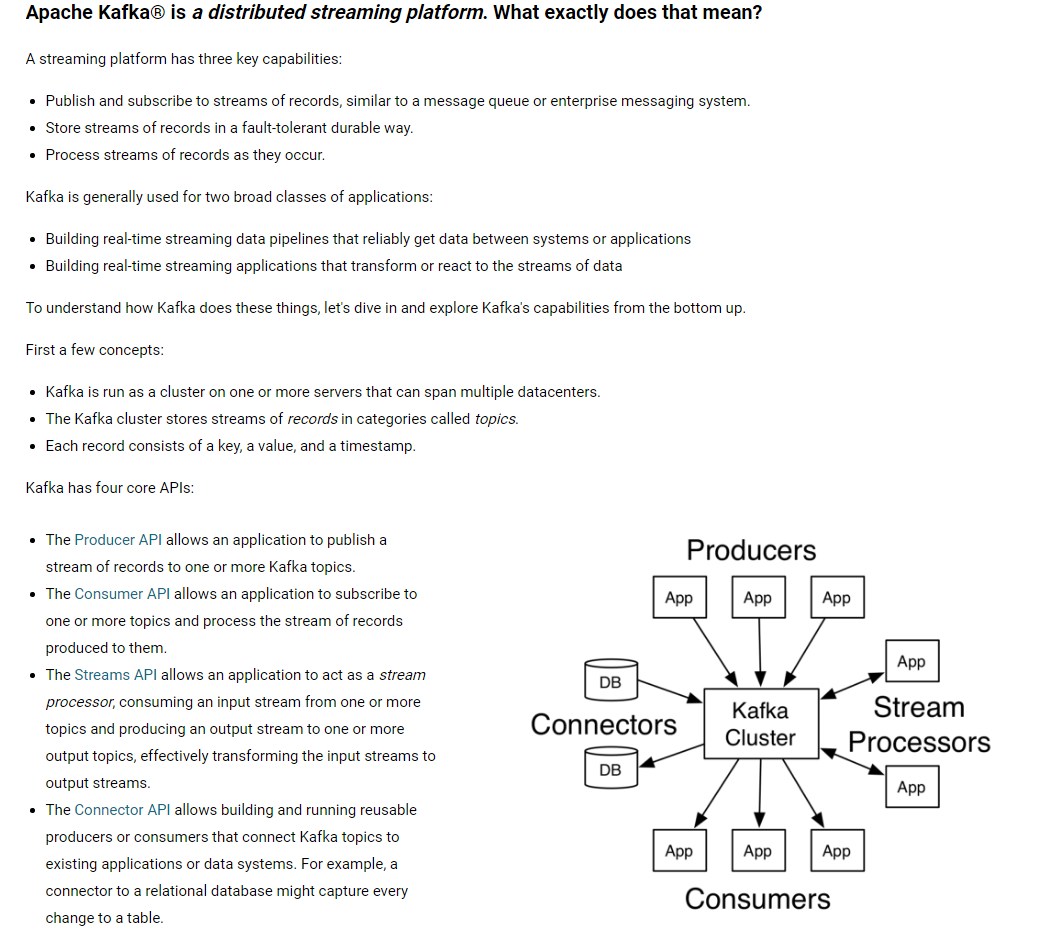
Eg: flume-ng agent --name a1 –conf-file /home/chaitanyapolipalli/flume\_demo/example.conf

* Different flume implementations?

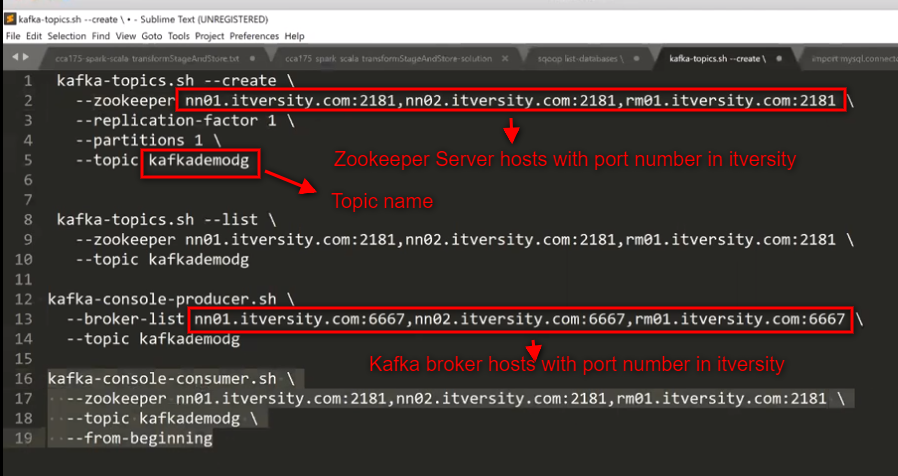




**Kafka Overview:**



* Create kafka topic, list topics, producer and consumer

****

**The messages typed in producer.sh terminal will apprear in consumer.sh terminal, consumer is consuming messages produced by producer. The connecting point between producer and consumer is “topic”**

**Spark Streaming:**

* What is difference between SparkContext and StreamingContext?

SparkContext is used to read, process and load data at a low frequency interval often atleast in hours, whereas in streamingcontext it will be in very high frequency as low as five seconds or even less. In a single session either sparkcontext or streamingcontext can exist but not both.

More differences <https://www.youtube.com/watch?v=FXqPyiutMGg&index=126&list=PLf0swTFhTI8rDQXfH8afWtGgpOTnhebDx>

* What are requirements for SparkStreaming?

StreamingContext needs to have a webservice to keep on polling for data at regular intervals

* Steps to create sparkstreaming object?

import org.apache.spark.SparkConf

import org.apache.spark.streaming.\_

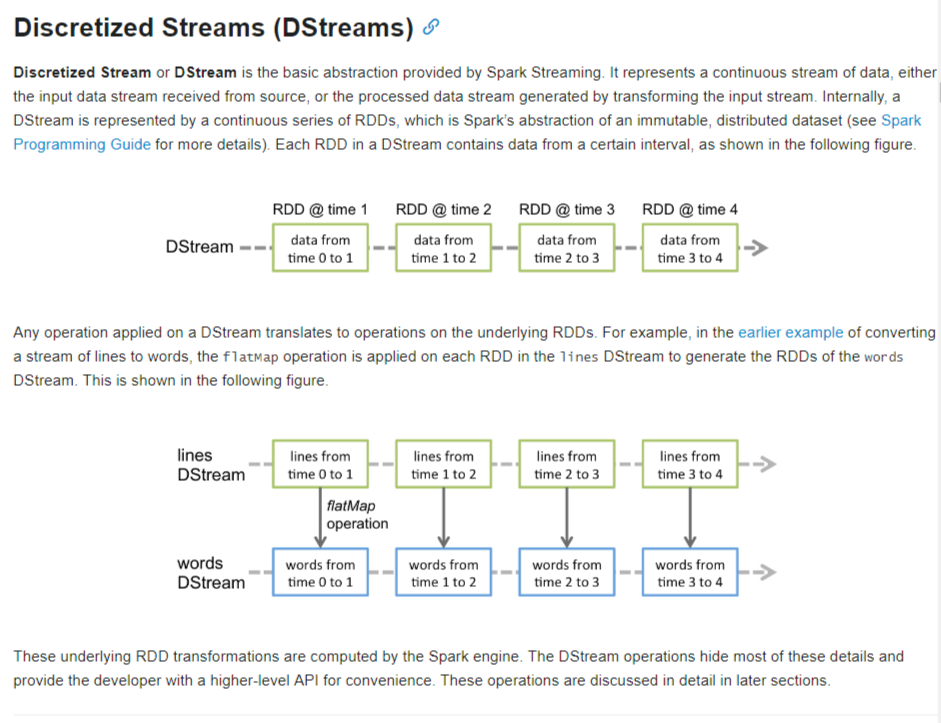
val conf = new SparkConf().setAppName(“Word Count”).setMaster(“yarn-client”)

val ssc = new StreamingContext(conf, Seconds(10)) //this object will poll every 10 seconds

* What is a DStream?

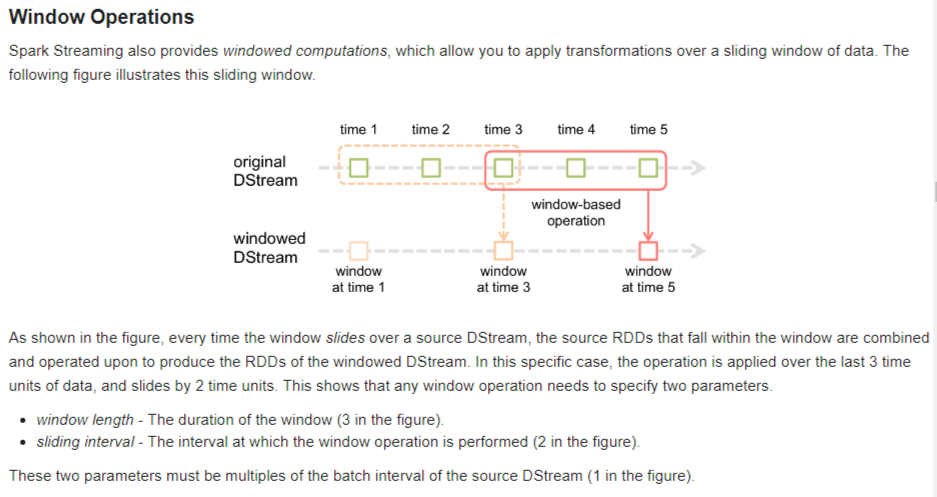
DStream is a series of RDD’s. In our above example we created ssc with time interval of 10 seconds, so a new rdd is created for every 10 seconds with new data available in that 10 seconds. So DStream is a wrapper on a series of RDD’s which are read using streaming context. More info

<https://www.youtube.com/watch?v=N2x9zmBGEuM&list=PLf0swTFhTI8rDQXfH8afWtGgpOTnhebDx&index=130>



* What are windowing operations in SparkStreaming?

<https://www.youtube.com/watch?v=N2x9zmBGEuM&list=PLf0swTFhTI8rDQXfH8afWtGgpOTnhebDx&index=130>



**GIT:**

* How to clone data in git?

**git clone** [**https://github.com/dgadiraju/data.git**](https://github.com/dgadiraju/data.git)

**Point to Remember:**

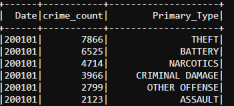
* In scala/spark sorting works only on List Iterable.
* In exam they will provide resource manager web ui address on port 8088, just visit it to find how many executors are available. If this information is not available, go to /etc/hadoop/conf/yarn-site.xml and search for “**yarn.resourcemanager.webapp.address**”. Open that url and see or search for **Memory Total** and **Vcores Total** values. Here we can never exceed Vcores total count. **Minimum allocation** and **Maximum allocation** parameters gives you the range where you can set executor memory values. Usually between 1Gb to 4Gb (in labs). If you want to play around and give more executor memory for each VCore then make sure Vcore\_Max \* Executor\_Memory should not exceed **Memory Total**.
* **vCores \* No of executors** should not exceed **VCores Total**
* Ctrl + Shift + + will increase cloudera terminal font
* Ctrl + c to copy, Ctrl + Shift + v to paste
* Gzip codec class “org.apache.hadoop.io.compress.GzipCodec”
* For a join operation to perfom on RDD’s, RDD’s have to paired RDD meaning they have to have key, value pairs.
* Where there are multiple small files in local file system, easiest way to solve these problems is to copy complete folder to hdfs and use sc.textFile("hdfs\_path"). This will create a RDD of all data in all those small files. We are using this approach because scala.io.Source doesn't have a straight forward method to read data from multiples files in a single command.
* Parquet is a columnr file type so it can't be read using below command sc.textFile("/user/chaitanyapolipalli/nyse\_parquet") instead you should use **sqlContext.read.parquet("/user/chaitanyapolipalli/nyse\_parquet").show**
* In sqoop import

1. --table and/or --columns are mutually exclusive with --query
2. If you are using --query then –split-by is mandatory if m > 1
3. If –query is used then palceholder \$CONDITIONS should be used.

* Using export SPARK\_MAJOR\_VERSION=2 will launch Spark2 instance
* In HDFS or Hive you cannot update data, you can only add data.
* In hive, default field delimeter is Control A ( ^A )
* In sqoop import and sqoop export, they have only map tasks.
* sc.parallelize is used to convert any scala collection to rdd
* reduce in Actions cannot be used on a Key, it is used for global aggregation like sum, min, max, etc
* You need not specify delimeters for other file formats except for textFileformat, it is taken care internally by other file formats.
* When ur loading data into hive table and by mistake if u give wrong path, hive doesn't validate data and the command runs successfully.
* mkString takes an array and converts int string

Eg:

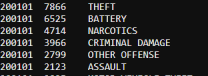
Before mkString 🡪 crim\_count\_per\_monthDF



val crimeRDD = crim\_count\_per\_monthDF.rdd

crimeRDD.map( c=> {c.mkString("\t")})

After



* To validate any gzip compressed data, you cannot view it directly. Copy zipped file from hdfs to local using -get command then use “gunzip file\_name” then data will be unzipped then you view data and validate. This way of validation only works with text file formats, with file format this might not work. So in order to validate data in other file formats save data in rdd and then using sc variable read data and validate.
* In sqoop, if ur using --query attribute then you should use –split-by (column name) if you need more than one mapper (m > 1), \$CONDITIONS should be specified in where clause in –query and cannot use –warehouse-dir attribute.