CCA175 Q&As

CCA Spark and Hadoop Developer Exam

Pass Cloudera CCA175 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.leads4pass.com/cca175.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Cloudera Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

Problem Scenario 28 : You need to implement near real time solutions for collecting information when submitted in file with below

Data

echo "IBM,100,20160104" >> /tmp/spooldir2/.bb.txt echo "IBM,103,20160105" >> /tmp/spooldir2/.bb.txt mv /tmp/spooldir2/.bb.txt /tmp/spooldir2/bb.txt After few mins echo "IBM,100.2,20160104" >> /tmp/spooldir2/.dr.txt echo "IBM,103.1,20160105" >> /tmp/spooldir2/.dr.txt mv /tmp/spooldir2/.dr.txt /tmp/spooldir2/dr.txt You have been given below directory location (if not available than create it) /tmp/spooldir2 . As soon as file committed in this directory that needs to be available in hdfs in /tmp/flume/primary as well as /tmp/flume/secondary location. However, note that/tmp/flume/secondary is optional, if transaction failed which writes in this directory need not to be rollback. Write a flume configuration file named flumeS.conf and use it to load data in hdfs with following additional properties .

1.

Spool /tmp/spooldir2 directory

2.

File prefix in hdfs sholuld be events

3.

File suffix should be .log

4.

If file is not committed and in use than it should have _ as prefix.

5.

Data should be written as text to hdfs

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution : Step 1 : Create directory mkdir /tmp/spooldir2 Step 2 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume8.conf. agent1 .sources = source1 agent1.sinks = sink1a sink1bagent1.channels = channel1a channel1b agent1.sources.source1.channels = channel1a channel1b agent1.sources.source1.selector.type = replicating agent1.sources.source1.selector.optional = channel1b agent1.sinks.sink1a.channel = channel1a agent1 .sinks.sink1b.channel = channel1b agent1.sources.source1.type = spooldir agent1 .sources.sourcel.spoolDir = /tmp/spooldir2 agent1.sinks.sink1a.type = hdfs agent1 .sinks, sink1a.hdfs. path = /tmp/flume/primary agent1 .sinks.sink1a.hdfs.tilePrefix = events agent1 .sinks.sink1a.hdfs.fileSuffix = .logagent1 .sinks.sink1a.hdfs.fileType = Data Stream agent1 .sinks.sink1b.type = hdfs agent1 .sinks.sink1b.hdfs.path = /tmp/flume/secondary agent1 .sinks.sink1b.hdfs.filePrefix = events agent1.sinks.sink1b.hdfs.fileSuffix = .log agent1 .sinks.sink1b.hdfs.fileType = Data Stream agent1.channels.channel1a.type = file agent1.channels.channel1b.type = memory step 4 : Run below command which will use this configuration file and append data in hdfs. Start flume service: flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume8.conf --name age Step 5 : Open another terminal and create a file in /tmp/spooldir2/ echo "IBM,100,20160104" » /tmp/spooldir2/.bb.txt echo "IBM,103,20160105" » /tmp/spooldir2/.bb.txt mv /tmp/spooldir2/.bb.txt /tmp/spooldir2/bb.txt After few mins echo "IBM.100.2,20160104" »/tmp/spooldir2/.dr.txt echo "IBM,103.1,20160105" » /tmp/spooldir2/.dr.txt mv /tmp/spooldir2/.dr.txt /tmp/spooldir2/dr.txt

QUESTION 2

Problem Scenario 2 :

There is a parent organization called "ABC Group Inc", which has two child companies

named Tech Inc and MPTech.

Both companies employee information is given in two separate text file as below. Please do

the following activity for employee details.

Tech Inc.txt 1,Alok,Hyderabad 2,Krish,Hongkong 3,Jyoti,Mumbai 4,Atul,Banglore 5,Ishan,Gurgaon MPTech.txt 6,John,Newyork 7,alp2004,California 8,tellme,Mumbai 9,Gagan21,Pune 10,Mukesh,Chennai

1.

Which command will you use to check all the available command line options on HDFS and How will you get the Help for individual command.

2.

Create a new Empty Directory named Employee using Command line. And also create an empty file named in it Techinc.txt

3.

Load both companies Employee data in Employee directory (How to override existing file in HDFS).

4.

Merge both the Employees data in a Single tile called MergedEmployee.txt, merged tiles should have new line character at the end of each file content.

5.

Upload merged file on HDFS and change the file permission on HDFS merged file, so that owner and group member can read and write, other user can read the file.

6.

Write a command to export the individual file as well as entire directory from HDFS to local file System.

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution :

Step 1 : Check All Available command hdfs dfs Step 2 : Get help on Individual command hdfs dfs -help get Step 3 : Create a directory in HDFS using named Employee and create a Dummy file in it called e.g. Techinc.txt hdfs dfs -mkdir Employee Now create an emplty file in Employee directory using Hue. Step 4 : Create a directory on Local file System and then Create two files, with the given data in problems. Step 5 : Now we have an existing directory with content in it, now using HDFS command line , overrid this existing Employee directory. While copying these files from local file System to HDFS. cd /home/cloudera/Desktop/ hdfs dfs -put -f Employee Step 6 : Check All files in directory copied successfully hdfs dfs -Is Employee Step 7 : Now merge all the files in Employee directory, hdfs dfs -getmerge -nl Employee MergedEmployee.txt Step 8 : Check the content of the file. cat MergedEmployee.txt Step 9 : Copy merged file in Employeed directory from local file ssytem to HDFS. hdfs dfs put MergedEmployee.txt Employee/ Step 10 : Check file copied or not. hdfs dfs -Is Employee Step 11 : Change the permission of the merged file on HDFS hdfs dfs -chmpd

664 Employee/MergedEmployee.txt Step 12 : Get the file from HDFS to local file system, hdfs dfs -get Employee Employee_hdfs

QUESTION 3

Problem Scenario 46 : You have been given belwo list in scala (name,sex,cost) for each

work done.

List(("Deeapak" , "male", 4000), ("Deepak" , "male", 2000), ("Deepika" , "female",

2000),("Deepak" , "female", 2000), ("Deepak" , "male", 1000) , ("Neeta" , "female", 2000))

Now write a Spark program to load this list as an RDD and do the sum of cost for

combination of name and sex (as key)

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution :

Step 1 : Create an RDD out of this list

val rdd = sc.parallelize(List(("Deeapak" , "male", 4000}, ("Deepak" , "male", 2000),

("Deepika" , "female", 2000),("Deepak" , "female", 2000), ("Deepak" , "male", 1000} ,

("Neeta" , "female", 2000}}}

Step 2 : Convert this RDD in pair RDD

val byKey = rdd.map({case (name,sex,cost) => (name,sex)->cost})

Step 3 : Now group by Key

val byKeyGrouped = byKey.groupByKey

Step 4 : Nowsum the cost for each group

val result = byKeyGrouped.map{case ((id1,id2),values) => (id1,id2,values.sum)}

Step 5 : Save the results result.repartition(1).saveAsTextFile("spark12/result.txt")

QUESTION 4

Problem Scenario 23 : You have been given log generating service as below. Start_logs (It will generate continuous logs) Tail_logs (You can check , what logs are being generated) Stop_logs (It will stop the log service) Path where logs are generated using above service : /opt/gen_logs/logs/access.log Now write a flume configuration file named flume3.conf , using that configuration file dumps logs in HDFS file system in a directory called flumeflume3/%Y/%m/%d/%H/%M Means every minute new directory should be created). Please us the interceptors to provide timestamp information, if message header does not have header info. And also note that you have to preserve existing timestamp, if message contains it. Flume channel should have following property as well. After every 100 message it should be committed, use non-durable/faster channel and it should be able to hold maximum 1000 events.

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution : Step 1 : Create flume configuration file, with below configuration for source, sink and channel. #Define source , sink , channel and agent, agent1 .sources = source1 agent1 .sinks = sink1 agent1.channels = channel1 # Describe/configure source1 agent1 .sources.source1.type = exec agentl.sources.source1.command = tail -F /opt/gen logs/logs/access.log #Define interceptors agent1 .sources.source1.interceptors=i1 agent1 .sources.source1.interceptors.i1.type=timestamp agent1 .sources.source1.interceptors.i1.preserveExisting=true ## Describe sink1 agent1 .sinks.sink1.channel = memory-channel agent1 .sinks.sink1.type = hdfs agent1 .sinks.sink1.hdfs.path = flume3/%Y/%m/%d/%H/%M agent1 .sinks.sjnkl.hdfs.fileType = Data Stream # Now we need to define channel1 property. agent1.channels.channel1.type = memory agent1.channels.channel1.capacity = 1000 agent1.channels.channel1.transactionCapacity = 100 # Bind the source and sink to the channel Agent1.sources.source1.channels = channel1 agent1.sinks.sink1.channel = channel1 Step 2 : Run below command which will use this configuration file and append data in hdfs. Start log service using : start_logs Start flume service: flume-ng agent -conf /home/cloudera/flumeconf -conf-file /home/cloudera/flumeconf/flume3.conf -Dflume.root.logger=DEBUG,INFO,console –name agent1 Wait for few mins and than stop log service. stop logs

QUESTION 5

Problem Scenario 55 : You have been given below code snippet.

val pairRDDI = sc.parallelize(List(("cat",2), ("cat", 5), ("book", 4),("cat", 12))) val

pairRDD2 = sc.parallelize(List(("cat",2), ("cup", 5), ("mouse", 4),("cat", 12)))

operation1

Write a correct code snippet for operationl which will produce desired output, shown below.

Array[(String, (Option[lnt], Option[lnt]))] = Array((book,(Some(4},None)),

(mouse,(None,Some(4))), (cup,(None,Some(5))), (cat,(Some(2),Some(2)),

(cat,(Some(2),Some(12))), (cat,(Some(5),Some(2))), (cat,(Some(5),Some(12))),

(cat,(Some(12),Some(2))), (cat,(Some(12),Some(12)))J

Correct Answer: See the explanation for Step by Step Solution and configuration.

Solution : pairRDD1.fullOuterJoin(pairRDD2).collect

fullOuterJoin [Pair]

Performs the full outer join between two paired RDDs.

Listing Variants

def fullOuterJoin[W](other: RDD[(K, W)], numPartitions: Int): RDD[(K, (Option[V],

OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W}]}: RDD[(K, (Option[V], OptionfW]))]

def fullOuterJoin[W](other: RDD[(K, W)], partitioner: Partitioner): RDD[(K, (Option[V],

Option[W]))]