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.pass4itsure.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 62 : You have been given below code snippet.

val a = sc.parallelize(List("dogM, "tiger", "lion", "cat", "panther", "eagle"), 2)

val b = a.map(x => (x.length, x))

operation1

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

Array[(lnt, String)] = Array((3,xdogx), (5,xtigerx), (4,xlionx), (3,xcatx), (7,xpantherx),

(5,xeaglex))

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

Solution : b.mapValuesf\\'x" + _ + "x").collect mapValues [Pair] : Takes the values of a RDD that consists of two-component tuples, and applies the provided function to transform each value. Tlien,.it.forms newtwo-componend tuples using the key and the transformed value and stores them in a new RDD.

QUESTION 2

Problem Scenario 10 : You have been given following mysql database details as well as other info. user=retail_dba password=cloudera database=retail_db jdbc URL = jdbc:mysql://quickstart:3306/retail_db Please accomplish following.

1. Create a database named hadoopexam and then create a table named departments in it, with following fields. department_id int, department_name string

e.g. location should be hdfs://quickstart.cloudera:8020/user/hive/warehouse/hadoopexam.db/departments

2.

Please import data in existing table created above from retaidb.departments into hive table hadoopexam.departments.

3.

Please import data in a non-existing table, means while importing create hive table named hadoopexam.departments_new

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

Solution :

Step 1 : Go to hive interface and create database.

hive

create database hadoopexam;

Step 2. Use the database created in above step and then create table in it. use

hadoopexam; show tables;

Step 3 : Create table in it.

create table departments (department_id int, department_name string);

show tables;

desc departments;

desc formatted departments;

Step 4 : Please check following directory must not exist else it will give error, hdfs dfs -Is

/user/cloudera/departments

If directory already exists, make sure it is not useful and than delete the same.

This is the staging directory where Sqoop store the intermediate data before pushing in

hive table.

hadoop fs -rm -R departments

Step 5 : Now import data in existing table

sqoop import \

-connect jdbc:mysql://quickstart:3306/retail_db \

~username=retail_dba \

-password=cloudera \

--table departments \

-hive-home /user/hive/warehouse \

-hive-import \

-hive-overwrite \

-hive-table hadoopexam.departments

Step 6 : Check whether data has been loaded or not.

hive;

use hadoopexam;

show tables;

select" from departments;

desc formatted departments;

Step 7 : Import data in non-existing tables in hive and create table while importing.

sqoop import \

-connect jdbc:mysql://quickstart:3306/retail_db \

--username=retail_dba \

~password=cloudera \

-table departments \

-hive-home /user/hive/warehouse \

-hive-import \

-hive-overwrite \

-hive-table hadoopexam.departments_new \

-create-hive-table

Step 8 : Check-whether data has been loaded or not.

hive;

use hadoopexam;

show tables;

select" from departments_new;

desc formatted departments_new;

QUESTION 3

Problem Scenario 32 : You have given three files as below. spark3/sparkdir1/file1.txt spark3/sparkd ir2ffile2.txt spark3/sparkd ir3Zfile3.txt Each file contain some text.

spark3/sparkdir1/file1.txt

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework

spark3/sparkdir2/file2.txt

The core of Apache Hadoop consists of a storage part known as Hadoop Distributed File System (HDFS) and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed.

spark3/sparkdir3/file3.txt

his approach takes advantage of data locality nodes manipulating the data they have access to to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies

on a parallel file system where computation and data are distributed via high-speed networking Now write a Spark code in scala which will load all these three files from hdfs and do the word count by filtering following words. And result should be sorted by word count in reverse order. Filter words ("a","the","an", "as", "a","with","this","these","is","are","in", "for", "to","and","The","of") Also please make sure you load all three files as a Single RDD (All three files must be loaded using single API call). You have also been given following codec import org.apache.hadoop.io.compress.GzipCodec Please use above codec to compress file, while saving in hdfs.

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

Solution :

Step 1 : Create all three files in hdfs (We will do using Hue). However, you can first

create in local filesystem and then upload it to hdfs.

Step 2 : Load content from all files.

val content =

sc.textFile("spark3/sparkdir1/file1.txt,spark3/sparkdir2/file2.txt,spark3/sparkdir3/file3.

txt") //Load the text file

Step 3 : Now create split each line and create RDD of words.

val flatContent = content.flatMap(word=>word.split(" "))

step 4 : Remove space after each word (trim it)

val trimmedContent = f1atContent.map(word=>word.trim)

Step 5 : Create an RDD from remove, all the words that needs to be removed.

val removeRDD = sc.parallelize(List("a","theM,ManM, "as",

"a","with","this","these","is","are\\'\"in\\'\ "for", "to","and","The","of"))

Step 6 : Filter the RDD, so it can have only content which are not present in removeRDD.

val filtered = trimmedContent.subtract(removeRDD}

Step 7 : Create a PairRDD, so we can have (word,1) tuple or PairRDD. val pairRDD =

filtered.map(word => (word,1))

Step 8 : Now do the word count on PairRDD. val wordCount = pairRDD.reduceByKey(_ +

_)

Step 9 : Now swap PairRDD.

val swapped = wordCount.map(item => item.swap)

Step 10 : Now revers order the content. val sortedOutput = swapped.sortByKey(false)

Step 11 : Save the output as a Text file. sortedOutput.saveAsTextFile("spark3/result")

Step 12 : Save compressed output.

import org.apache.hadoop.io.compress.GzipCodec

sortedOutput.saveAsTextFile("spark3/compressedresult", classOf[GzipCodec])

QUESTION 4

Problem Scenario 25 : You have been given below comma separated employee information. That needs to be added in /home/cloudera/flumetest/in.txt file (to do tail source) sex,name,city 1,alok,mumbai 1,jatin,chennai 1,yogesh,kolkata 2,ragini,delhi 2,jyotsana,pune 1,valmiki,banglore Create a flume conf file using fastest non-durable channel, which write data in hive warehouse directory, in two separate tables called flumemaleemployee1 and flumefemaleemployee1 (Create hive table as well for given data}. Please use tail source with /home/cloudera/flumetest/in.txt file. Flumemaleemployee1 : will contain only male employees data flumefemaleemployee1 : Will contain only woman employees data

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

Solution :

Step 1 : Create hive table for flumemaleemployeel and .\\'

CREATE TABLE flumemaleemployeel

(

sex_type int, name string, city string) ROW FORMAT DELIMITED FIELDS TERMINATED BY \\',\\'; CREATE TABLE flumefemaleemployeel (sex_type int, name string, city string) ROW FORMAT DELIMITED FIELDS TERMINATED BY \\',\\'; Step 2 : Create below directory and file mkdir /home/cloudera/flumetest/ cd /home/cloudera/flumetest/ Step 3 : Create flume configuration file, with below configuration for source, sink and channel and save it in flume5.conf. agent.sources = tailsrc agent.channels = mem1 mem2 agent.sinks = stdl std2 agent.sources.tailsrc.type = exec agent.sources.tailsrc.command = tail -F /home/cloudera/flumetest/in.txt agent.sources.tailsrc.batchSize = 1 agent.sources.tailsrc.interceptors = i1 agent.sources.tailsrc.interceptors.i1.type = regex_extractor agent.sources.tailsrc.interceptors.il.regex = A(\\d} agent.sources.tailsrc. interceptors. M.serializers = t1 agent.sources.tailsrc. interceptors, i1.serializers.t1. name = type agent.sources.tailsrc.selector.type = multiplexing agent.sources.tailsrc.selector.header = type agent.sources.tailsrc.selector.mapping.1 = memi agent.sources.tailsrc.selector.mapping.2 = mem2 agent.sinks.std1.type = hdfs agent.sinks.stdl.channel = mem1 agent.sinks.stdl.batchSize = 1 agent.sinks.std1.hdfs.path = /user/hive/warehouse/flumemaleemployeei agent.sinks.stdl.rolllnterval = 0 agent.sinks.stdl.hdfs.tileType = Data Stream agent.sinks.std2.type = hdfs agent.sinks.std2.channel = mem2 agent.sinks.std2.batchSize = 1 agent.sinks.std2.hdfs.path = /user/hi ve/warehouse/flumefemaleemployee1 agent.sinks.std2.rolllnterval = 0 agent.sinks.std2.hdfs.tileType = Data Stream agent.channels.mem1.type = memory agent.channels.meml.capacity = 100 agent.channels.mem2.type = memory agent.channels.mem2.capacity = 100 agent.sources.tailsrc.channels = mem1 mem2 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/flume5.conf --name agent Step 5 : Open another terminal create a file at /home/cloudera/flumetest/in.txt. Step 6 : Enter below data in file and save it. l.alok.mumbai 1 jatin.chennai 1,yogesh,kolkata 2,ragini,delhi 2,jyotsana,pune 1,valmiki,banglore Step 7 : Open hue and check the data is available in hive table or not. Step 8 : Stop flume service by pressing ctrl+c

QUESTION 5

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