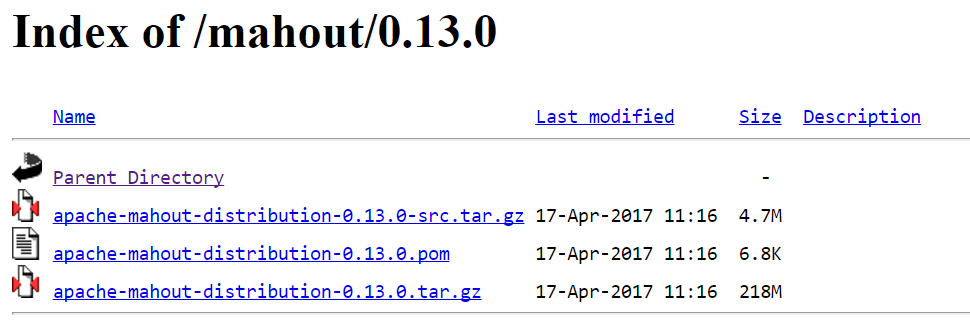
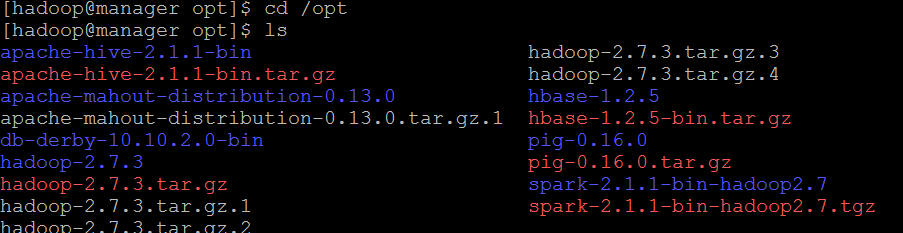
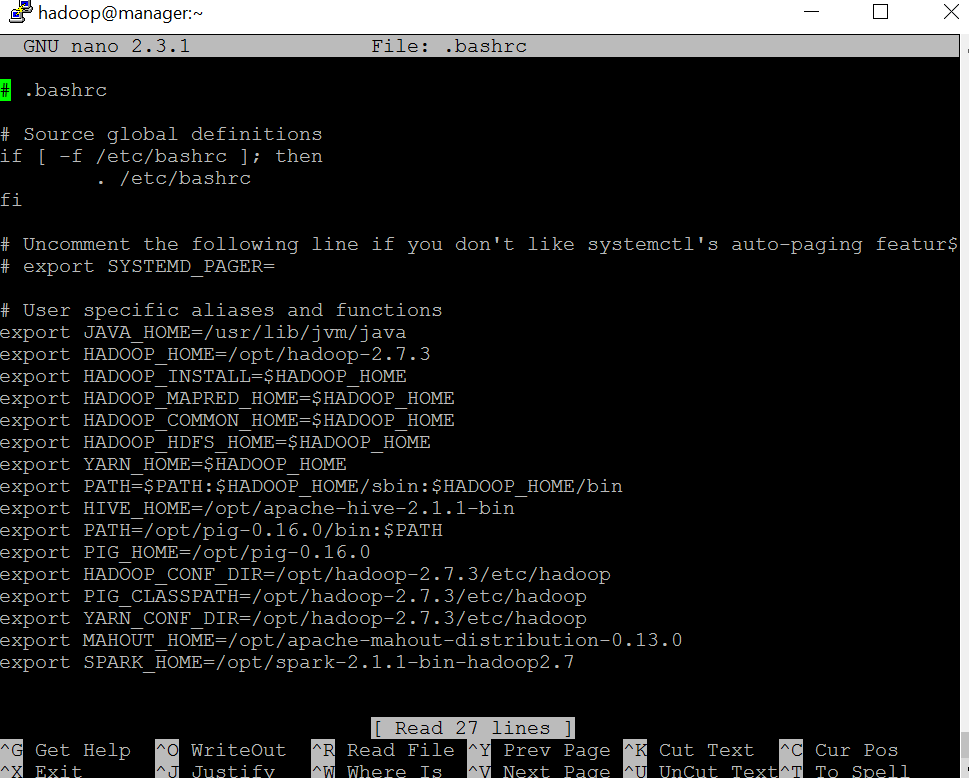
**Apache Mahout Lab:**



Downloaded and untarred mahout distribution



Set properties in .bashrc

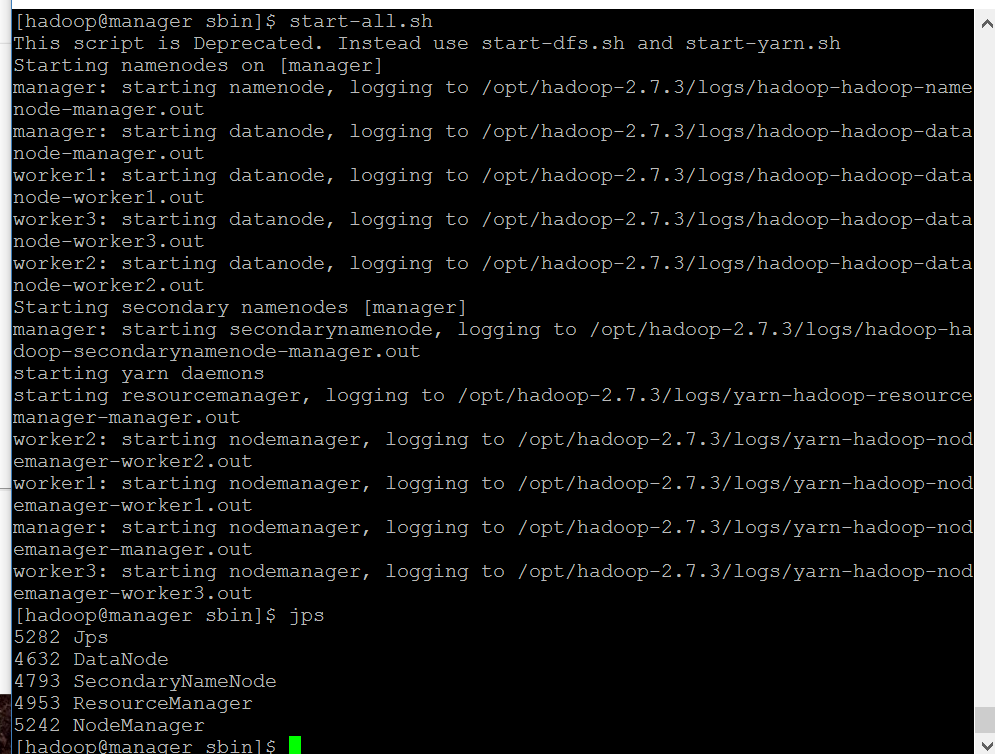


Secure Copied and Changed Owner for mahout base distribution to worker nodes.

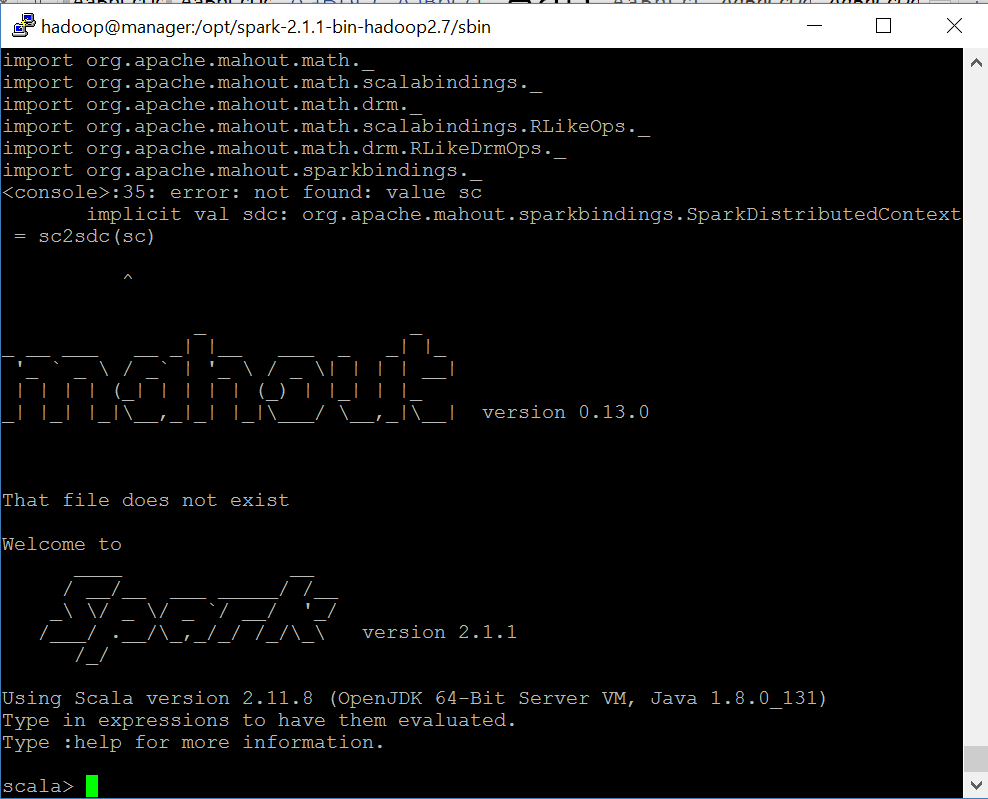
Started dfs.sh , started yarn.sh

Started Spark across cluster via start.all sh

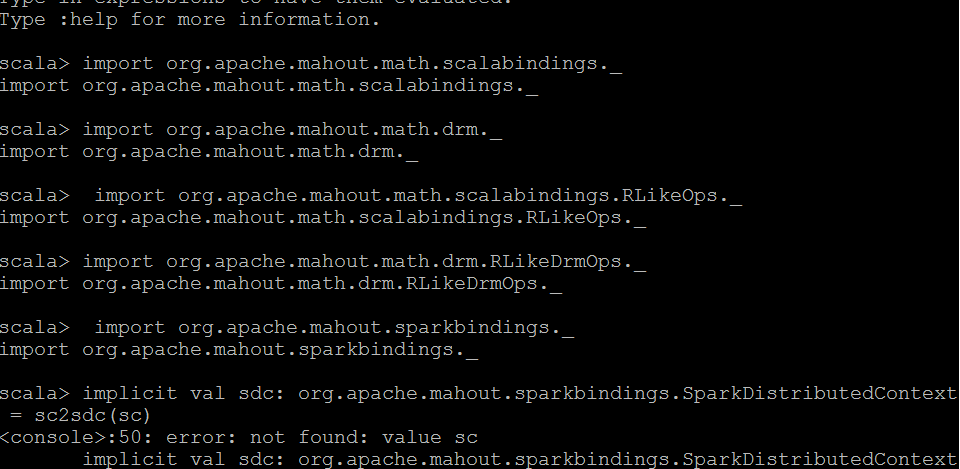
Checked jps



Started Mahout Spark Shell



Ran imports in scala prompt



For the next steps I had issues with my spark context not being found.

I tried the following workaround but it didn’t work

The original error is because you haven't created a SparkContext, which can be done:

implicit val mc = mahoutSparkContext()

Thereafter I think the implicit conversion of the mc (a SparkDistributedContext) to sc (a SparkContext) will be handled by the package helper functions. If the sc is still missing try:

implicit val sc = sdc2sc(mc)

I therefore read about the exercise steps. And made sure I understood how we setup matrices in memory and then carry out operations on them using mahout.

1. First we setup a matrix of values:

val drmData = drmParallelize(dense(

(2, 2, 10.5, 10, 29.509541), // Apple Cinnamon Cheerios

(1, 2, 12, 12, 18.042851), // Cap'n'Crunch

(1, 1, 12, 13, 22.736446), // Cocoa Puffs

(2, 1, 11, 13, 32.207582), // Froot Loops

(1, 2, 12, 11, 21.871292), // Honey Graham Ohs

(2, 1, 16, 8, 36.187559), // Wheaties Honey Gold

(6, 2, 17, 1, 50.764999), // Cheerios

(3, 2, 13, 7, 40.400208), // Clusters

(3, 3, 13, 4, 45.811716)), // Great Grains Pecan

numPartitions = 2);

1. Second we slice our original matrix into a x and y vector

val drmX = drmData(::, 0 until 4)

val y = drmData.collect(::, 4)

3.Third we calculate x transposed times X and x transposed times Y

val drmXtX = drmX.t %\*% drmX

val drmXty = drmX.t %\*% y

4.Finally, we collect the matrices and solve the system as shown by commands below

val XtX = drmXtX.collect

val Xty = drmXty.collect(::, 0)

val beta = solve(XtX, Xty