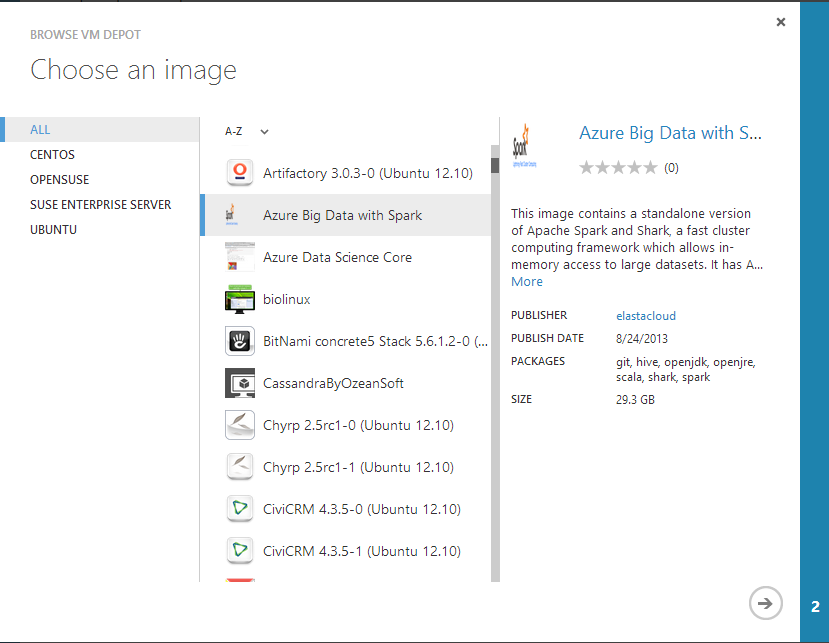
# Running Spark examples in Standalone mode

## Deployment

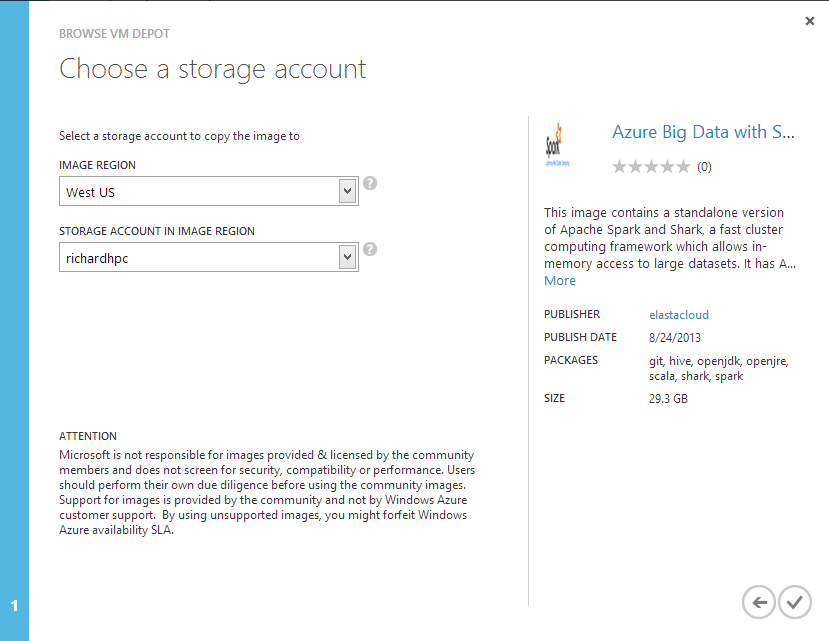
This should take about 7 mins to deploy so begin with the deployment initially and then continue with slides describing Spark and continue with the examples afterwards.

Deploy Spark using the Elastacloud vmdepot image:

Browse the vmdepot per the earlier tutorial and choose **Azure Big Data with Spark**.



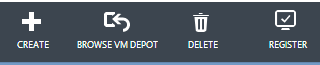
Follow the next step and select a storage account:



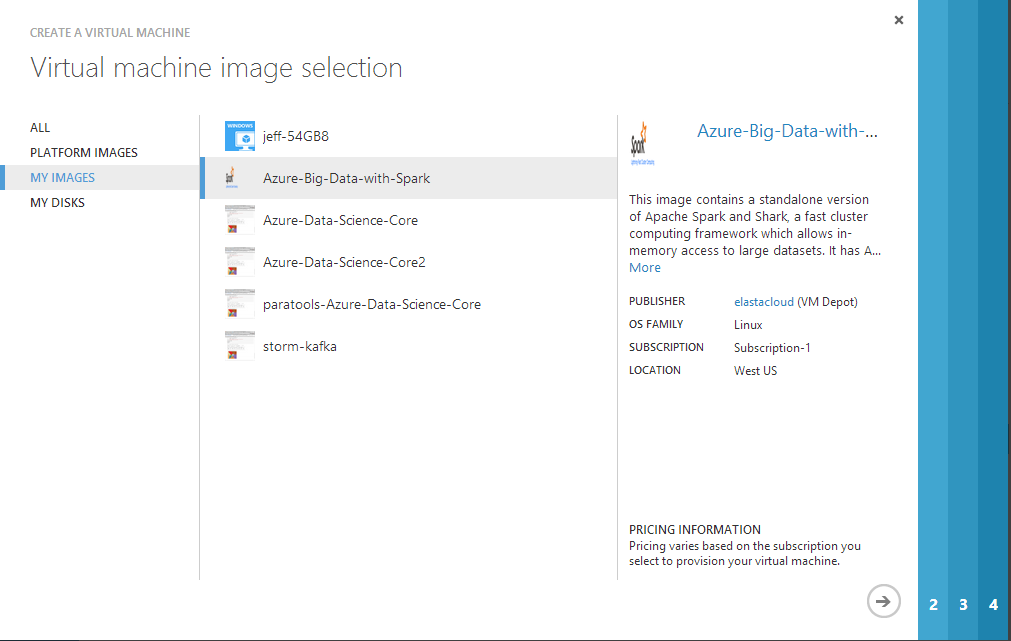
Wait for the copy process to complete.



Click on the register button when complete.



When this is registered we should create a new virtual machine with the registered image.



Follow the wizard steps and then connect to the new virtual machine over SSH with your username and password.

## Spark

Spark is installed in the /usr directory, navigate to this and run all commands:

* cd /usr/spark/spark-0.7.3

Ensure that you’re running in admin context so enter:

* sudo su

Followed by the password that was entered when the VM was created.

In order to test Spark we’ll use one of the examples which will calculate Pi using approximation:

* ./run spark.examples.SparkPi local[2]

You should see the following:

* Pi is roughly 3.1434

To start up the spark REPL (after navigating to the above directory):

* ./spark-shell

Run through the examples:

#load the readme.md file

* **val textFile = sc.textFile("README.md")**

#count the lines with Spark in them from the readme

* ***val linesWithSpark = textFile.filter(line => line.contains("Spark"))***
* ***linesWithSpark.count()***

#count the lines directly from the file loaded into the RDD

* ***textFile.filter(line => line.contains("Spark")).count()***

#this will do a map-reduce by counting each word and finding the line with the largest number of words

* ***textFile.map(line => line.split(" ").size).reduce((a, b) => if (a > b) a else b)***

#this allows us to do a map-reduce like hadoop/HDInsight

* **val wordCounts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey((a, b) => a + b)**

#to show the count of each word

* **wordCounts.collect()**

Do a similar kind of map reduce operation but persist the results to the file system:

* **val olympicWins = List(("Athens",3),("Athens",1),("London", 4),("London", 3),("London",2),("London",3))**
* **val hosts = sc.parallelize(olympicWins)**
* **hosts.reduceByKey(\_ + \_).saveAsTextFile("output")**

We’ll leave the spark shell now:

* exit;

Navigate to directory output:

* **cd output**
* **more part-00000**
* **cd ..**

## More than Map-Reduce (Logistic Regression with Spark)

In this section we’ll create a Scala file which can be used to perform Logistic Regression. The dataset in this case is a random dataset created by the Spark.

Spark uses a SparkContext and slices the data across the cluster and parallelizes like so. This is not code that should be copied into the REPL – it’s just for reporting.

* val sc = new SparkContext(args(0), "SparkLR",System.getenv("SPARK\_HOME"), Seq(System.getenv("SPARK\_EXAMPLES\_JAR")))
* val numSlices = if (args.length > 1) args(1).toInt else 2
* val points = sc.parallelize(generateData, numSlices).cache()

This is the example command line which can be run which shows the Logistic Regression across 10 dimensions and a handful of data points.

* ./run spark.examples.SparkLR local[2]

The results should be the best fit line in a hyperplane. The results should show the iterations of the logistic regression calculation.

The logistic regression calculation is:

* val gradient = points.map { p => (1 / (1 + exp(-p.y \* (w dot p.x))) - 1) \* p.y \* p.x}.reduce(\_ + \_)

## Shark

Navigate to:

* cd /usr/spark/shark/shark-0.7.0/bin

To get up the shark CLI:

* ./shark

When the CLI starts up issue the following:

DROP TABLE src;

CREATE TABLE src(key INT, value STRING);

LOAD DATA LOCAL INPATH '${env:HIVE\_HOME}/examples/files/kv1.txt' INTO TABLE src;

SELECT \* FROM src;

SELECT COUNT(1) FROM src;

To exit the SHARK REPL:

* exit;

If you receive an error with any of the above:

* fallocate -l 10g /mnt/resource/swap1
* chmod 600 /mnt/resource/swap1
* mkswap /mnt/resource/swap1
* swapon /mnt/resource/swap1
* nano /etc/fstab

Then add:

* /mnt/resource/swap1 swap swap defaults 0 0

Each of these commands should be run individually. You ***should*** navigate to the file on disk which can be found at:

* cat /usr/spark/shark/hive-0.9.0-bin/examples/files/kv1.txt