

Instructions for preparing Premium Linux HDI Spark clusters running MRS (preview) and performing analytical tasks described in the KDD 2016 tutorial: “Scalable R on Spark”

John-Mark Agosta, Debraj GuhaThakurta, Robert Horton, Mario Inchiosa, Srinu Kumar,

Vanja Paunic, Hang Zhang, Mengyue Zhao

Data Group, Microsoft Corporation

Date: October 06, 2016

KDD 2016 tutorial: Scalable R on Spark

In August 2016, we presented a tutorial titled: (links to [KDD tutorial page](#); [ACM digital library](#), and [public GitHub repository for tutorial materials, video of presentation](#)). This document provides instructions on how to create HDInsight Premium clusters (preview) running Microsoft R Server (MRS), and execute the R code, which was a part of the hands-on exercise at the tutorial.

Overview of what is provided in this document

The objective of this document is to provide a description to the scripts that can be used to quickly create, customize and delete HDInsight clusters running MRS. These clusters have the same configuration as the ones provided to the attendees of the KDD tutorial in Aug 2016.

For preparing the clusters we provide [Windows PowerShell](#) scripts with [Azure commandlets](#), a set of Windows PowerShell modules that help you to automate your Microsoft Azure resource and service management tasks. For the hands-on exercises running R code, we have provided R files, which can be easily executed to generate the output results.

Here we provide details of the following steps:

1. [Pre-requisites:](#)
 - a) [Creating an Azure account and login into Azure portal](#)
 - b) [Setting up Windows PowerShell and Azure management modules](#)
 - c) [Removing or renaming folders on local drive](#)
 - d) [Installing software for logging into HDInsight cluster using SSH](#)
2. [Download and unzip script files from GitHub repo](#)
3. [Provisioning HDInsight Premium clusters with MRS \(preview\) and the necessary blob storage accounts](#)
4. [Customizing cluster with script actions: Installing software and R packages, and copying code files from GitHub by running script actions on cluster edge node](#)
5. [Copying files from a public blob storage to cluster's HDFS using AzCopy](#)
6. [Verifying clusters for files and Spark compute environment](#)
7. [Hands-on exercises: Execution of the R code](#)
8. [Deleting clusters and associated storage accounts \(when cluster is not needed further\)](#)
9. [Summary](#)

For steps 3, 4, 6, and 8, we provide scripts that will run in parallel, thereby reducing the total cluster preparation and deletion times.

All the scripts and code are available for download from a [public GitHub repository](#). Scripts for creating, preparing, and deleting clusters are in the folder "Scripts". R code for exercises are in the folder "Code". Slides presented during the tutorial are in the folder "Slides". Slides provide details about the tutorial, including Microsoft R server architecture on HDInsight Spark clusters (preview).

Pre-requisites:

Create an Azure account

Before you begin, you need to create an Azure account and subscription. You can follow instructions provided [here](#). Once you create an account, you can login to your Azure [portal](#).

You will first need to [create an Azure subscription](#), and a [resource group](#) under that subscription. Resources such as blobs and HDInsight clusters can then be created within that resource group. You will need to specify a subscription id and a resource group name when creating the HDInsight clusters with Microsoft R server using PowerShell scripts described below. Once you create a resource group, you will see the subscription id, once you click on it, towards the top right-hand corner.

Setting up Windows PowerShell and Azure management modules

First, download or update your Azure PowerShell commandlets by following the instructions given [here](#).

Run Windows PowerShell as administrator and install the Azure Resource Manager and Service Management modules from PowerShell Gallery as described [here](#).

```
# Install the Azure Resource Manager modules from the PowerShell Gallery
Install-Module AzureRM
```

```
# Install the Azure Service Management module from the PowerShell Gallery
Install-Module Azure
```

The above commands may ask you to install NuGet provider. Select 'Yes' if prompted. It may also ask if you want to install the modules from 'PSGallery'. Select 'Yes'.

You will also need to download and install [AzCopy](#) on your Windows machine. Instructions are provided [here](#).

Renaming or removing existing folder

If you already have an existing folder named C:\RSpark\RSpark_KDD2016, rename or remove it, since script files for this exercises will be downloaded there by default.

Installing client software for logging into cluster using SSH

If client SSH software is not present (e.g. putty), download and install [putty](#). In addition, [MobaXterm](#) may also be very helpful to install for using SSH to login to the cluster nodes.

Check for availability of unzip.exe

Check to see if you have unzip.exe installed. If not, you can install it from [here](#). We had the following unzip.exe installed:

UnZip 6.00 of 20 April 2009, by Info-ZIP

Download and unzip script files from GitHub repo

To begin using scripts, first download and unzip the script files from GitHub by executing the following commands in the Windows PowerShell. Right click and run Windows PowerShell as administrator.

```
# Initiate WebClient
$web_client = new-object System.Net.WebClient

# Specify download directory and filepath
$destination_dir = "C:\RSpark";
if (!(Test-Path $destination_dir)) {mkdir $destination_dir}
$destination_file = Join-Path $destination_dir "RSpark_KDD2016.zip";
cd $destination_dir
```

```
# Specify which file to download
$url = "http://cdsparksamples.blob.core.windows.net/hdiscripts/KDD2016R/RSpark_KDD2016.zip";

# Download and inflate file using unzip.exe
$web_client.DownloadFile($url, $destination_file)
unzip.exe .\RSpark_KDD2016.zip -d .

# End of Functions / Start of Script
Write-Output "Completed getting zip files and inflating."
cd $destination_dir\RSpark_KDD2016
```

NOTE: Depending on where your unzip.exe is installed and if that path is in your profile, you may need to add the full path to the unzip.exe when unzipping the `RSpark_KDD2016.zip` file.

After the code finishes running, you will see the following folders being created.

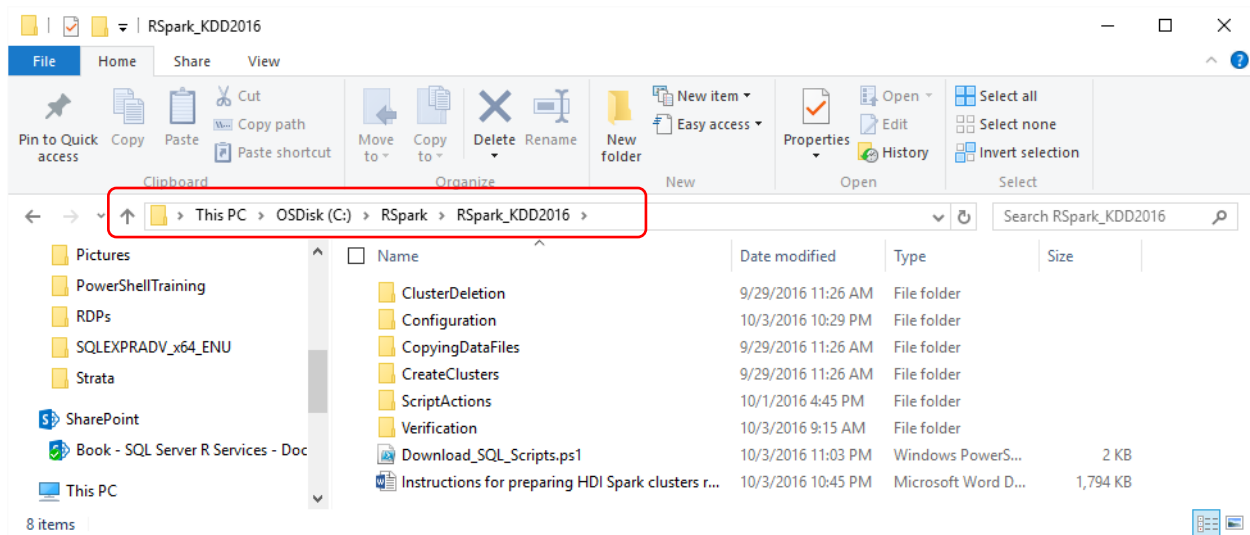


Figure 1: Downloaded files in your local C:\ directory.

Provisioning HDInsight Premium clusters with MRS (preview) and the necessary blob storage accounts

An overview of Microsoft R Server on HDInsight (preview) is provided [here](#). For provisioning HDInsight Premium clusters using Windows PowerShell, you will need the following configuration files and scripts:

1. A configuration parameter file in csv format to specify cluster names, logins, cluster-size (how many worker nodes are needed), Azure subscription id which is to be used for creating the clusters, resource group within the subscription in which the cluster/s are to be created etc. Windows PowerShell will use this as input for getting all the relevant parameters about the clusters to be created. File name: `ClusterParameters.csv`. Location to be specified in the script (see below).

2. A Windows PowerShell script which submits the job for creating individual clusters. File name: [SubmitHDICreation.ps1](#)
3. A driver Windows PowerShell file which runs the jobs in parallel using the submission script (#2). File name: [RunHDICreation.ps1](#)

The [ClusterParameters.csv](#) file is located in: C:\RSpark\RSpark_KDD2016\Configuration, and the other two files will be located in: C:\RSpark\RSpark_KDD2016>CreateClusters.

1	Parameter	Value	Comment
2	SubscriptionID		Specify your Azure subscription ID
3	ResourceGroup		Specify a resource group within your subscription
4	Profilepath	AzureSubscriptionInfo.json	This is the name of the file where your Azure profile will be saved. It will reside in the Configurations folder
5	ClusterStartIndex	1	Specify cluster start index. You will have to specify a start and end index in order to create multiple clusters. The index
6	ClusterEndIndex	2	Specify cluster end index. You will have to specify a start and end index in order to create multiple clusters
7	ClusterPrefix	hdimrs	Cluster prefix. Usually a short text phrase (no gaps, numbers, or special characters). Make sure it is at least 6 characters
8	SourceDataStorageName	cdsparksamples	This is the Azure Blob data source from where you can copy files. Currently set to a public blob.
9	SourceDataContainerName	data	Container in the blob storage where source data is.
10	ClusterInfoOutputFileName	clusterOutFile.csv	File name where cluster names and passwords will be saved
11	NumWorkerNodes	2	Number of worker or data nodes of the clusters
12	AdminUsername	admin	Administrator user name
13	SshUsername	remoteuser	Remote login ssh user name

Figure 2: [ClusterParameters.csv](#) file, specifying parameters for cluster creation. Fields are described in the comments column.

The example settings in Figure 2 indicate that:

1. Value of **subscription ID** and **resource group** should be provided. You must have a subscription and a resource group within a subscription before you proceed.
2. **Login profile** is saved in AzureSubscriptionInfo.json file in the C:\RSpark\RSpark_KDD2016\Configuration folder.
3. **Cluster names** will have prefix hdimrs.
4. **Cluster start** and end indices are 1 and 1 respectively, so cluster names will be hdimrs1.azurehdinsight.net and hdimrs2.azurehdinsight.net. If you want to create just one cluster, set cluster start and end indices to 1 (or the same integer).
5. **SourceDataStorageName** & **SourceDataContainerName**: Provides location of the KDD data-sets. The data will be copied over from this location the blobs attached to the clusters (serving as HDFS).
6. **Number of worker-nodes** for clusters: 2
7. **Admin and SSH usernames** as admin and remoteuser respectively. Passwords are generated at the time of cluster creation and output into the clusterOutFile.csv file.

NOTE: The downloaded [ClusterParameters.csv](#) file may be read-only. In that case, change the name of the original downloaded file to something else, and after you edit the file, save it with file name [ClusterParameters.csv](#).

Navigate to C:\RSpark\RSpark_KDD2016\CreateClusters, right click on [RunHDICreation.ps1](#), and select “Run with PowerShell”.

NOTE: If “Run with PowerShell” available on your client machine, you can open Windows PowerShell and run the entire code there. Alternatively, you can right-click and open the code in Windows PowerShell ISE, and run the code there. This also applies to running other scripts (.ps1) files described below.

Once you start running [RunHDICreation.ps1](#), this will first prompt to login to your Azure subscription. After your login, it may prompt you again to save the login information in a profile file (json).

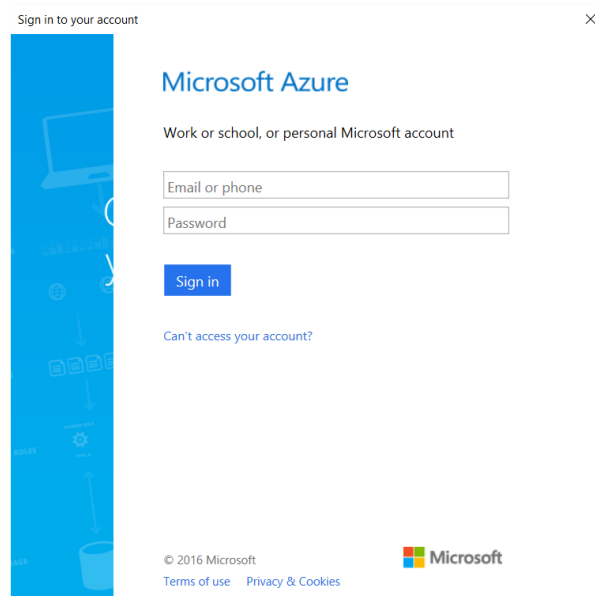


Figure 3: Pop-up window for logging into to Azure subscription and getting profile information.

NOTES:

- If you want to create just one cluster, set cluster start and end indices to 1 (or the same integer).
- Note here that clusters will be created for login with SSH authentication with a [username/](#)
[password](#) – please see [link](#) for further details. This is why you need to specify the SSH Username

- Clusters are currently set to be created using the VM size of “[Standard_D12_v2](#)”. Currently, the cost of running an HDInsight cluster with MRS is with 1 head node, 1 edge node (for MRS) and 2 worker nodes with Standard D12 V2 VM size is \$4.20/hr (5 nodes, 20 cores). This was the setup for the KDD tutorial.
- **Advanced users:** If you want to change the size of VM, then download [azuredeploy.json](#) from GitHub, modify it according to your specifications, save it under C:\RSpark\RSpark_KDD2016\Configuration, and then replace the existing pointer to azuredeploy.json with C:\RSpark\RSpark_KDD2016\Configuration\azuredeploy.json in the file C:\RSpark\RSpark_KDD2016\CreateClusters\SubmitHDICreation.ps1.

Once the script is submitted, the clusters take about 20-25 mins to provision and be active for use. A blob storage account is created for every cluster. The name of the blob storage account associated with each cluster will be: name_of_cluter + “storage”. For example, for a cluster hdimrs1, the corresponding blob storage will be hdimrs1storage. This will serve as the HDFS for the cluster.

During cluster provision, you can click on “Deployments” under your resource group. If clusters are being deployed, a blue circle will indicate that (see below). After the clusters are finish deploying, these circles will turn green, and you will see the clusters under your resources.

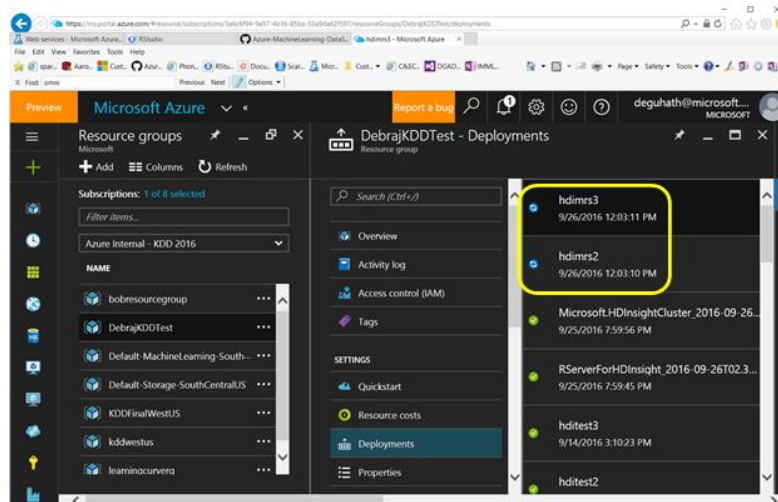


Figure 4: Figure showing clusters are being deployed under your resources.

Once the cluster is ready to be used, you can navigate and examine it through the GUI on your Azure portal.

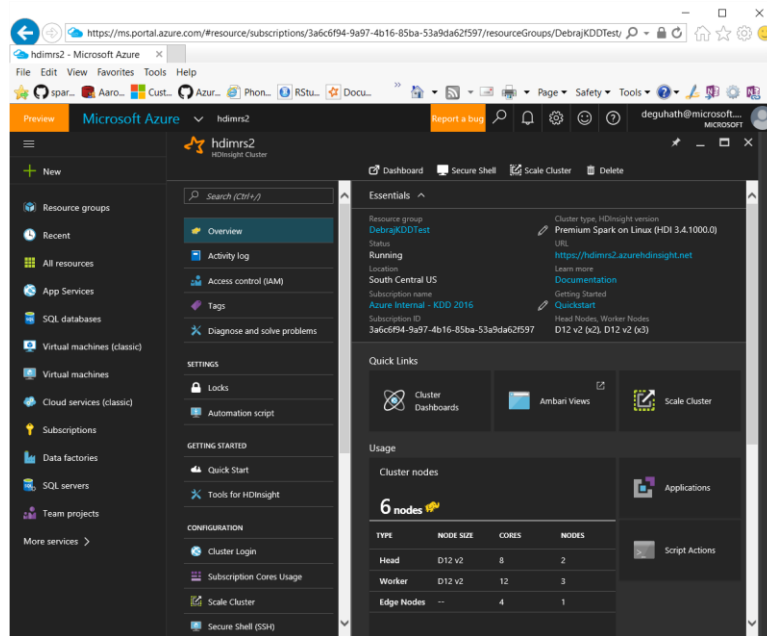


Figure 5: Dashboard for managing HDI Spark cluster on Azure portal. Once the cluster is created, you can access this dashboard for your cluster. You can change the size of the cluster and scale it up or down using the “scale cluster” tile.

Customizing cluster with script actions: Installing software and R packages, and copying code files from GitHub by running script actions on cluster edge node

After the cluster is created, you can login to the r-server edge node using a client such as [putty](#) or [MobaXterm](#). For example, for a cluster with name hdimrs1, the edge node will have address: [r-server.hdimrs1-ssh.azurehdinsight.net](#). The ssh login and username are needed to login. Ssh passwords for each of the clusters you create is output in the `C:\RSpark\RSpark_KDD2016\clusterOutFile.csv`.

Script actions need be run on the edge nodes of the clusters to install necessary software (e.g. RStudio) and R packages (e.g. SparkR, sparklyr and others), and copy code and script files from the GitHub repository, so that the hands-on exercises can run. Data files are copied from a public blob storage location using AzCopy that needs to be installed on your client machine. For more on how to customize HDInsight clusters using script actions, read [this](#).

For customizing your clusters to the specifications of the KDD tutorial, you will need the following scripts, which are located in `C:\RSpark\RSpark_KDD2016\ScriptActions`.

1. A Windows PowerShell script which submits the job for creating individual clusters. File name: [SubmitScriptActionsOnHDI.ps1](#)

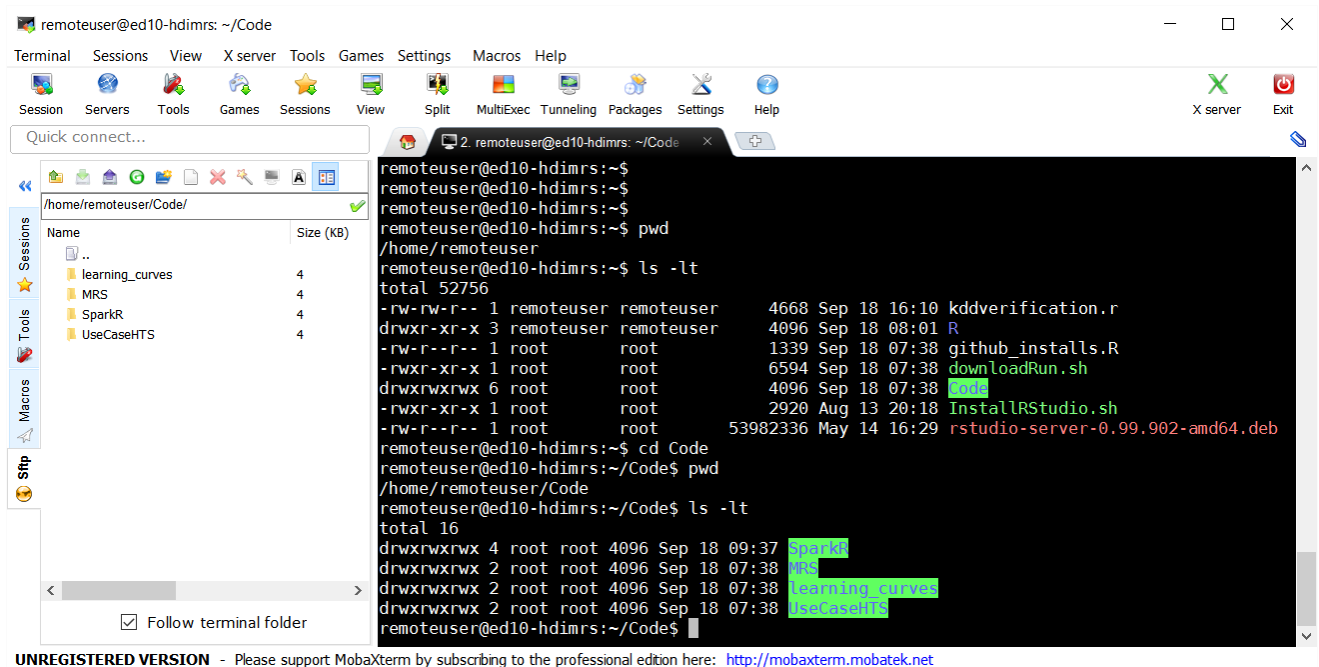
2. A driver Windows PowerShell file which runs the jobs in parallel using the submit script (#2). File name: [RunScriptActionsOnHDI.ps1](#)

For execution, navigate to C:\RSpark\RSpark_KDD2016\ScriptActions, right click on [RunScriptActionsOnHDI.ps1](#) select "Run with PowerShell". You will be prompted for the Azure login again.

This will download two scripts to your cluster edge node from a GitHub [folder](#):

1. An R file for necessary package installations from CRAN and GitHub. File name: [github_installs.R](#)
2. A shell script file which calls the R-script file for installing R packages, downloading necessary files and installing RStudio. File name: [downloadRun.sh](#)

Once the scripts finish execution on your edge node (which can take upto 15 mins), you will observe several files/folders in your edge node /remoteuser/home folder:



```
remoteuser@ed10-hdimrs: ~/Code
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. remoteuser@ed10-hdimrs: ~/Code
Name Size (KB)
.. 4
learning_curves 4
MRS 4
SparkR 4
UseCaseHTS 4
remoteuser@ed10-hdimrs:~$
remoteuser@ed10-hdimrs:~$
remoteuser@ed10-hdimrs:~$ pwd
/home/remoteuser
remoteuser@ed10-hdimrs:~$ ls -lt
total 52756
-rw-rw-r-- 1 remoteuser remoteuser 4668 Sep 18 16:10 kddverification.r
drwxr-xr-x 3 remoteuser remoteuser 4096 Sep 18 08:01 R
-rw-r--r-- 1 root root 1339 Sep 18 07:38 github_installs.R
-rwxr-xr-x 1 root root 6594 Sep 18 07:38 downloadRun.sh
drwxrwxrwx 6 root root 4096 Sep 18 07:38 .code
-rwxr-xr-x 1 root root 2920 Aug 13 20:18 InstallRStudio.sh
-rw-r--r-- 1 root root 53982336 May 14 16:29 rstudio-server-0.99.902-amd64.deb
remoteuser@ed10-hdimrs:~$ cd Code
remoteuser@ed10-hdimrs:~/Code$ pwd
/home/remoteuser/Code
remoteuser@ed10-hdimrs:~/Code$ ls -lt
total 16
drwxrwxrwx 4 root root 4096 Sep 18 09:37 sparkR
drwxrwxrwx 2 root root 4096 Sep 18 07:38 MRS
drwxrwxrwx 2 root root 4096 Sep 18 07:38 learning_curves
drwxrwxrwx 2 root root 4096 Sep 18 07:38 UseCaseHTS
remoteuser@ed10-hdimrs:~/Code$
```

Figure 6: Files and folders that are created on the edge node of a Spark HDI cluster running MRS after script actions are run on that node.

NOTE: If there are issues running script actions from Windows PowerShell, you can download the script file from GitHub to your edge node /home/remoteuser directory, make it executable, and run it:

```
cd /home/remoteuser
```

```
wget https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/KDDCup2016/Scripts/RunningScriptActions/downloadRun.sh

chmod +x downloadRun.sh

./downloadRun.sh
```

R code for running the hands-on exercises are in the “Code” folder. Brief descriptions of the code files in various folders are provided here:

SparkR: Code files in this directory show how to perform data manipulations using [SparkR](#) ([SparkR_sparklyr_NYCTaxi.Rmd](#)), and perform featurization and ML model training using [sparklyr](#) ([sparklyr_NYCTaxi.Rmd](#)). These files are copied from [this GitHub location](#).

MRS: R code files showing how to use SparkR for pre-processing data using SparkR (1-Clean-Join-Subset.r), build ML models using MRS functions (2-Train-Test-Subset.r), deploy these models on [Azure ML](#) (3-Deploy-Score-Subset.r). These files are copied to your cluster edge node from [this GitHub location](#).

learning_curves: R code files showing how to create learning curves on big data using MRS. Description of the files are given [here](#), which is also the location in GitHub from where files are copied to your cluster edge node.

UseCastHTS: Contains a sample R script for the tutorial use case: Training parallel models for hierarchical time series optimization. This script uses Australian tourism data set from 'fpp' package to forecast quarterly visitor nights spent by international tourists to Australia. The quarterly historic data is available for years 1999-2010. The file used for this exercise is copied to your cluster edge node from [this](#) GitHub location.

Copying data files from a public blob storage to cluster's HDFS using

Data-sets for the exercises are copied from a public Azure blob storage [<http://cdsparksamples.blob.core.windows.net/data>] using Windows PowerShell script (C:\RSpark\RSpark_KDD2016\CopyingDataFiles\CopyDataWithAzCopy.ps1) which uses [AzCopy](#) to copy files from the public blob storage to the HdiSamples folder in the cluster's blob storage (HDFS).

To execute, right click on C:\RSpark\RSpark_KDD2016\CopyingDataFiles\CopyDataWithAzCopy.ps1 and select “Run with PowerShell”. It will ask you for Azure login. This script uses AzCopy to copy files from the public blob location specified in the ClusterParameters.csv to your cluster's HDFS.

The figure below shows the files that are copied from the public blob storage. The public blob storage and the container for the files to be copied are specified in the configuration parameters file. You will need to specify the location of AzCopy in the [CopyDataWithAzCopy.ps1](#). It is currently set to: 'C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\AzCopy.exe'.

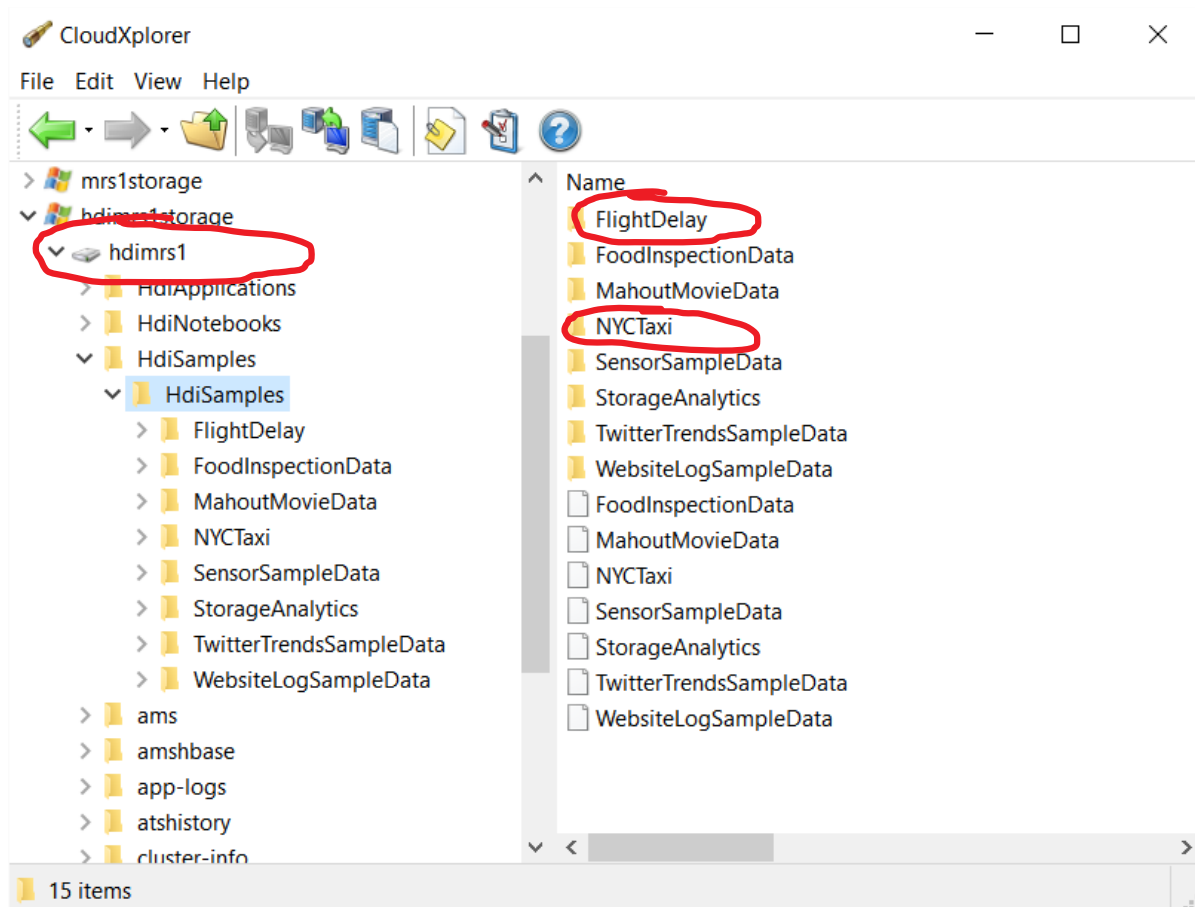


Figure 7: Folders that are copied to the cluster blob storage are shown for the example cluster hdimsr1. CloudXplorer is used to view the files in the HDFS of the cluster. CloudXplorer can be downloaded from [here](#).

Description of data-sets downloaded to HDFS.

- **NYCTaxi:** Files in the NYCTaxi directory in HDFS are used for SparkR/sparklyr exercises, with code files that are located in Code/SparkR on your cluster edge node. The original NYC Taxi data contains taxi trip and fare data from 170 mil trips in NY City in 2013 (<http://www.andresmh.com/nyctaxitrips>). There are 12 pairs of trip and fare files in the original data-set, one for each of the months in 2013. We have only used data from December for the hands-on exercises here. For description of the prediction task, please see below.
- **US domestic commercial flight arrival/departure & weather data:** This data-set contains information about US domestic commercial flight arrival and departure (<http://www.transtats.bts.gov>), and related weather data at these airports in 2011 and 2012

[<http://www.ncdc.noaa.gov/orders/qclcd/>]. For description of the prediction task, please see below.

Verifying clusters for files and Spark compute environment

Navigate to C:\RSpark\RSpark_KDD2016\VerifyClusters, right click on RunVerification.ps1, and select “Run with PowerShell”. You will be prompted for the Azure login.

Verification should complete in a couple of minutes, and the file /home/remoteuser/verification.csv will be created on the edge node. If verification succeeded, verification.csv should have the following contents:

```
"hdfsFiles","edgeMRSFiles","edgeSparklyRFiles","sparkHPCJob","sparkHPAJob","azureMLExists","rstudioRunning"
```

```
TRUE,TRUE,TRUE,TRUE,TRUE,TRUE,TRUE
```

To troubleshoot verification, a directory called /tmp/verif is created on the edge node. This directory contains a log of the verification script’s execution. By navigating to the cluster in question in the Azure Portal and clicking the Script Actions button, you can view the Script Action History. The Verification script action will be listed there, if it was successfully submitted.

NOTE: If there are issues running the verification script from Windows PowerShell, you can download the script file from GitHub to your edge node /home/remoteuser directory, make it executable, and run it:

```
cd /home/remoteuser

wget https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/KDDCup2016/Scripts/VerifyClusters/verification.sh

chmod +x verification.sh

./verification.sh
```

Hands-on exercises

Brief description of the exercises

SparkR and sparklyr

This tutorial describes how to use [SparkR](#) for data manipulation and use [sparklyr](#) for building ML models. We used the NY City taxi trip and fare data for this exercise. The [NYC Taxi data](#) is about 20GB of compressed comma-separated values (CSV) files (~48GB uncompressed), comprising more than 173 million individual trips and the fares paid for each trip. Each trip record includes the pickup and drop-off location and time, anonymized hack (driver's) license number and medallion (taxi's unique id) number. The data covers all trips in the year 2013 and is provided in the following two datasets for each month

(Jan through Dec). Each fare record includes information about the fare amount, tip paid, taxes and tolls etc. We use the trip and fare files from December 2013. We join the data-sets and use the joined data-set to model the amount of tip paid (regression) using three different modeling approaches from [sparklyr](#) (viz. regularized regression, random forest, boosted regression tree). Two R markdown files are provided in Code/SparkR/:

[SparkR_sparklyr_NYCTaxi.Rmd](#): Shows how to use SparkR for data manipulations and joining data-sets.

[sparklyr_NYCTaxi.Rmd](#): Shows how to use sparklyr for training ML models.

MRS

The MRS tutorial demonstrates a typical end-to-end data science workflow:

1. Data Wrangling - transformation and joining data sets
2. Model estimation and evaluation
3. Operationalization – deploying a predictive model as a web service

Two data sets are used: an airline data set of passenger flight on-time performance data from the US Department of Transportation’s TranStats data collection at <http://www.transtats.bts.gov>, and weather data derived from NOAA’s hourly land-based weather observations at <http://www.ncdc.noaa.gov/orders/qclcd/>.

The Apache SparkR package, “rx” functions provided with Microsoft R Server in the RevoScaleR package, and the AzureML CRAN package are used in this workflow. Start by running 1-Clean-Join-Subset.r, followed by 2-Train-Test-Subset.r and 3-Deploy-Score-Subset.r.

The Operationalization script, 3-Deploy-Score-Subset.r, creates and deploys a web service in an Azure Machine Learning workspace. Edit azureml-settings.json to include the “id” and “authorization_token” of your workspace. Instructions for creating an Azure Machine Learning workspace are available [here](#). Workspace ID and Authorization Tokens can be found in the “Name” and “Authorization Tokens” listings under the Settings tab in Azure Machine Learning Studio.

SparkSQL

The Structured Query Language, or SQL, is a very powerful paradigm to organize and query data. This has been traditionally implemented in relational databases. Now, the same basic language constructs are available in big data technologies such as Hive and Spark. We demonstrate here how to make SQL calls from within R to Spark. We use a subset of the New York Taxicab data, which is order of magnitude of hundreds of millions of records. We analyze the data for interesting features, and all our attempts are using the SQL language. We demonstrate the recognition of the schema, some preliminary exploratory SQL and then analyze passenger behavior, including influx and efflux of passengers from locations in the city. We chart the results using the popular ggplot2 package. The reader can retrace our path by running the script available at <https://raw.githubusercontent.com/Azure/Azure-MachineLearning-DataScience/master/Misc/KDDCup2016/Code/SparkSQL/SparkSQL.R>.

Use Case HTS

In this demo, we demonstrate how to use Microsoft R Server and Spark on HDInsight to efficiently explore a vast search space of parameters for hierarchical time series forecasting. Hierarchical time series are obtained when time series data are disaggregated by attributes of interest to form groups of time series or a hierarchy. For example, hierarchical time series forecasting might be applied to forecasting company sales of all products in total, by product category, or by a specific product. Here, we train many parallel models to forecast a hierarchical time series, and distribute the parallel tasks using HDInsight cluster on Microsoft R Server. To evaluate our approach, we used mean absolute percent error (MAPE) and measured its improvement from the baseline value. The validation is done using training/testing split on time series data of length L , where we train the model on $L-h$ months and test it on the last h months. We used a customer data set to run the evaluation on real-world demand forecasting example. We provide a sample R script ([sample demo.R](#)) to demonstrate the process on a smaller problem of forecasting quarterly visitor nights spent by international tourists to Australia using Australian tourism data set from 'fpp' package.

Learning curves

Learning curves are plots of predictive error across a range of training set sizes. They provide a fundamental kind of diagnostic plot to visualize how well a machine learning algorithm is improving as a function of experience. The general approach, including the how we simulate this data, are described in two earlier blog posts: [Why Big Data?](#) and [Learning from Learning Curves](#).

In this exercise we use the scalable functions of Microsoft R Server to train models on large datasets, and we use the Spark compute context to parallelize the training and evaluation of the large numbers of models required to cover all the data points. A learning curve is like a parameter scan of training set sizes, and this framework makes it easy to compare learning curves from a family of models, or across a range of values of other model parameters. The code uses dynamic sampling so that all of the data for training and validation comes from a single data frame or eXternal Data Frame (XDF) file. It is specific to Microsoft R Server in that it makes extensive use of transform functions that are applied to chunks of data (which is how MRS is able to work on datasets too large to fit into memory.) The main script is [run_learning_curve_demo.R](#), which runs a simple example on a dataframe. Comments in the code show how to run larger datasets in XDF files on HDFS. This script sources two other code files: [sim_data.R](#) contains functions for generating the simulated dataset, and [learning_curve_lib.R](#) defines the function that generates the points of the learning curves (plus helper functions).

How to execute code using RStudio server on cluster edge node

Instructions for logging into the RStudio server on the edge-node of the Spark cluster are given [here](#). RStudio is already installed on the edge node using script actions. So, you will have to follow instructions

on how to login to the RStudio server and run R code (steps 7-11). Code files are in GitHub in the “Code” folder.

Once you login to the RStudio server, you will see the code files in different directories, as shown below:

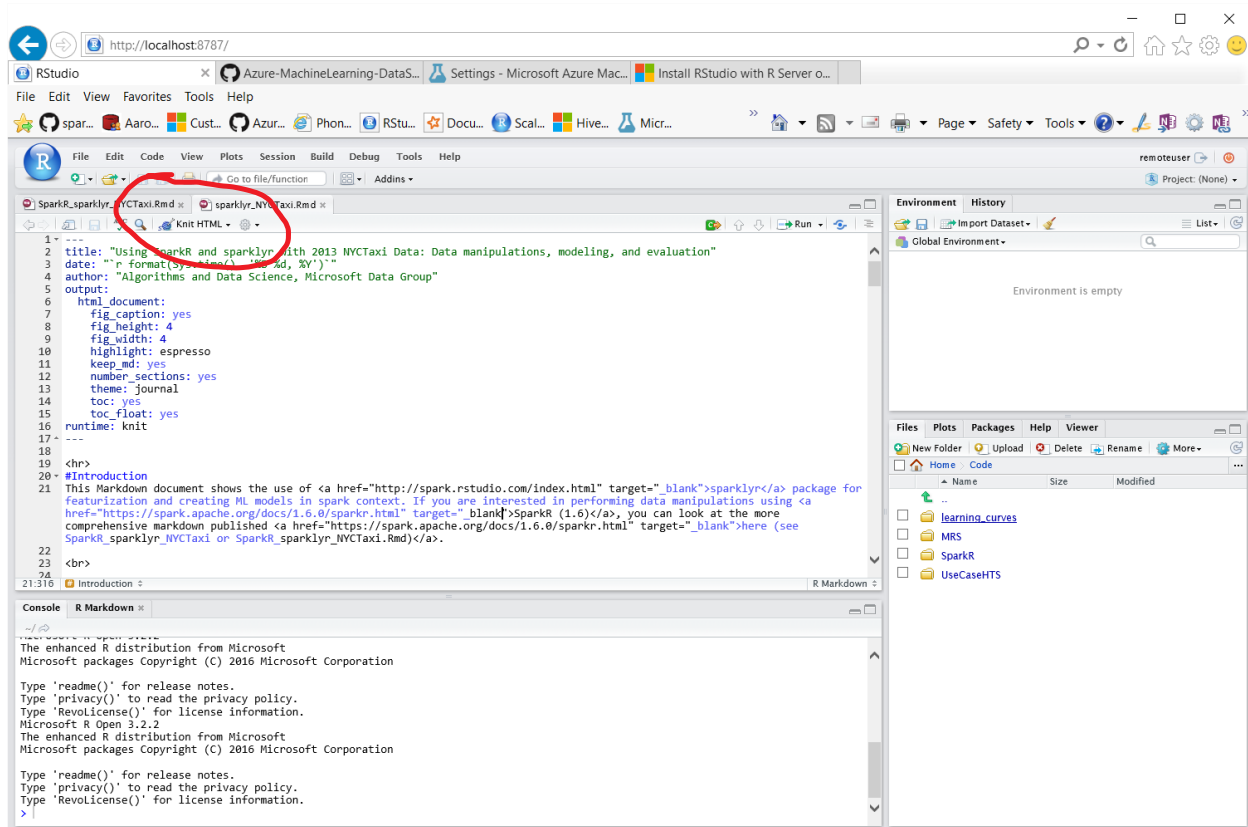


Figure 8: R markdown file in RStudio. To execute the code in R markdown files (.Rmd), just click on “knit HTML”, and it will execute and produce an HTML output.

For R markdown files (.Rmd), you can click on ‘knit HTML’, and see the code run. After completion of the execution, and HTML output file will be produced. For example, see below.

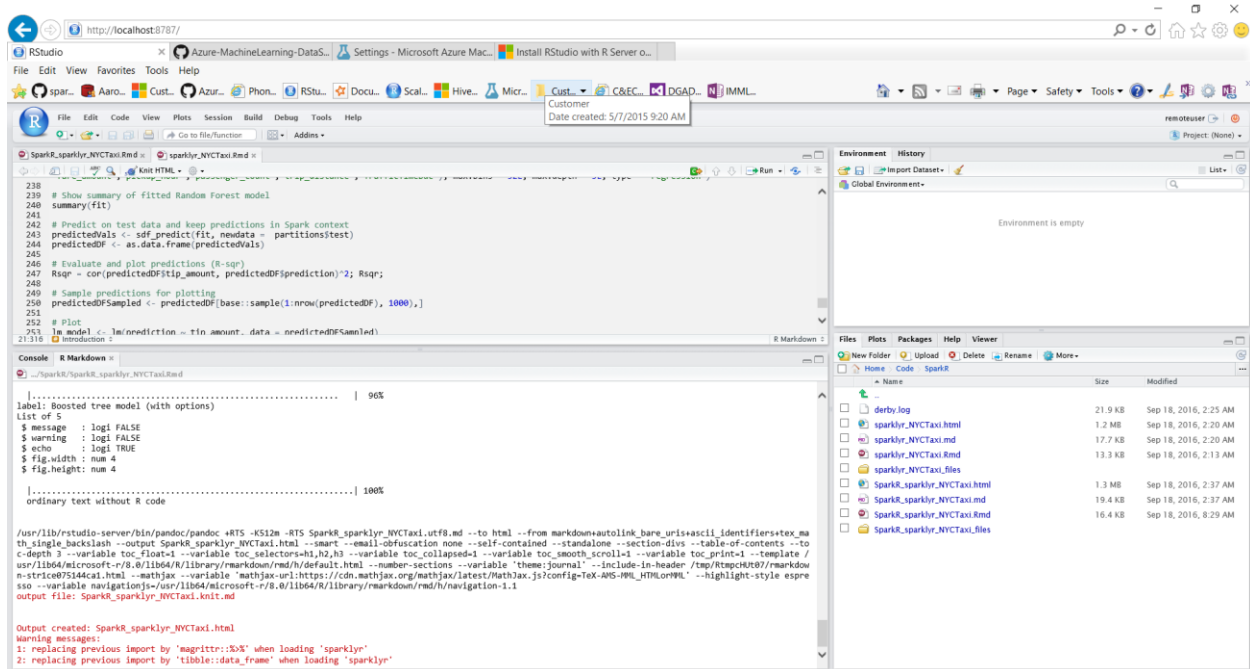


Figure 9: RStudio server snapshot after a R markdown file is successfully executed. Note in the bottom right panel the HTML files which appeared after the code in the R markdown file fully executed.

Depending on which file you are running, code can take 3-15 mins to finish. Code that are not in Rmd file, can be loaded in RStudio and executed in the Console.

Deleting clusters and associated storage accounts

After you are done using the clusters, you can easily delete them using Windows PowerShell scripts and Azure commandlets. For cluster deletion, you will need the following scripts, which are located in “C:\RSpark\RSpark_KDD2016\ClusterDeletion\DeleteClusters”.

Folder has 2 files:

1. A Windows PowerShell script which submits the job for deleting individual clusters. File name: [SubmitHDIClusterDeletion.ps1](#)
2. A driver Windows PowerShell script, which reads in the cluster parameters from [ClusterParameters.csv](#), runs the deletion jobs in parallel using the submit script (#2). File name: [RunHDIClusterDeletion.ps1](#)

Navigate to this directory, right click on [RunHDIClusterDeletion.ps1](#), and select “Run using PowerShell”. You will get a prompt for Azure login. After which the clusters and the associated blobs will be deleted from your subscription.

After the script starts running you should see that the cluster is “Deleting” if you click on the cluster name in your subscription (shown below for cluster hdimrs1).

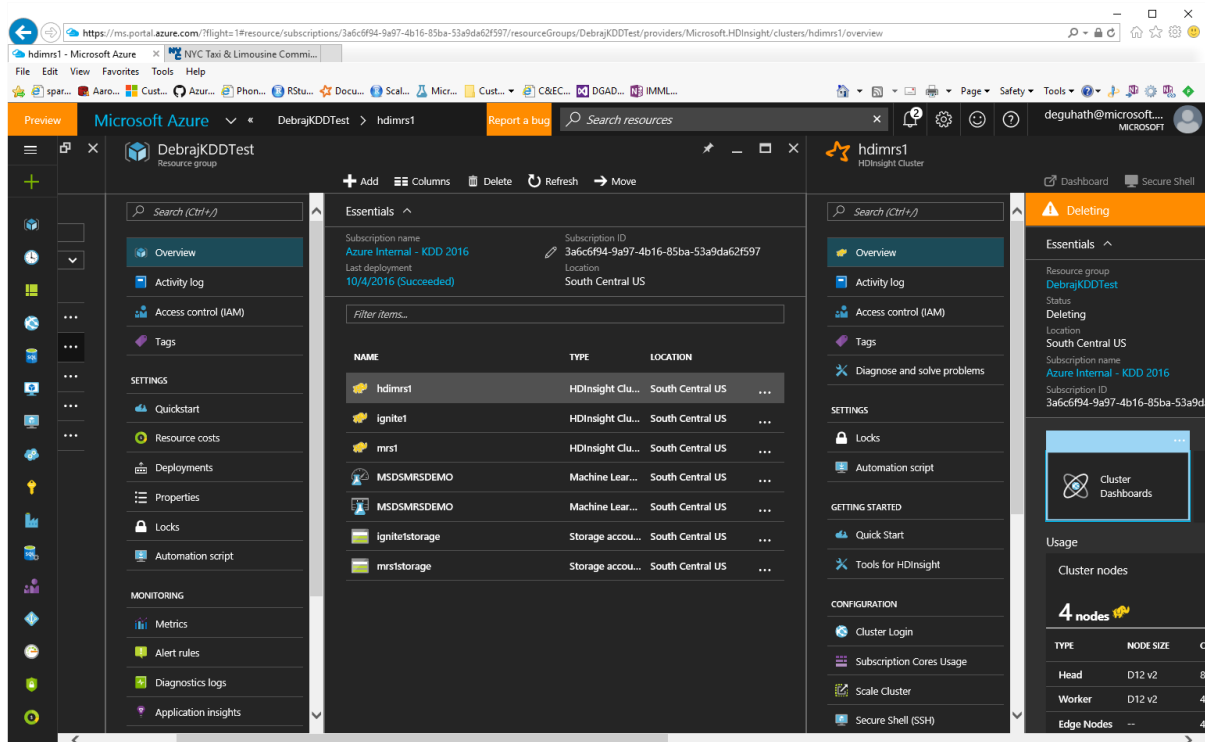


Figure 10: Figure showing cluster being deleted using PowerShell script.

Clusters can take 15-20 mins to get deleted. Check Azure portal to make sure clusters are deleted. After the clusters and blobs are deleted, you should see them disappear from your subscription.

Summary

In this document we have provided the instructions for provisioning, configuring and deleting Spark HDI Premium clusters running MRS, as well as instructions for running the R codes that were provided for the hands-on exercises in the KDD 2016 tutorial on Scalable R on Spark.

We sincerely appreciate feedback. We would find it very helpful if users try our scripts and provide feedback, especially if parts of the documents are not clearly explained, or if some scripts are not working as expected. That will help us improve our documentation and scripts for future users.