Getting Started with Microsoft R Server on HDInsight (R Server)

Contents

Introduction

Introduction

In this lab we are going demonstrate:

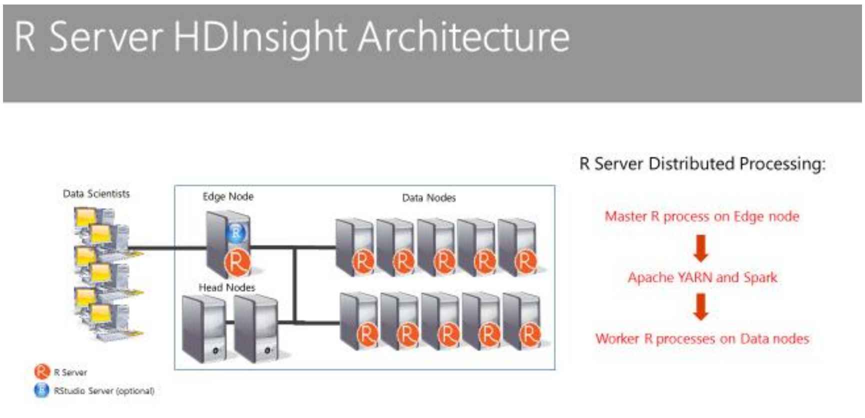
- **Checking the Yarn Memory configuration for the cluster using the cluster Dashboards.**
- **Connecting to the R Server Edge node and calling the console.**

Using R Studio (Community Edition) to run the following:

- **Running a simple test script, which articulates how MRS interacts with Hadoop i.e.**
  - **Compute on the edge node with data streamed from HDFS**
  - **Compute in the Hadoop cluster using MapReduce**
  - **Compute in the Hadoop cluster using Spark engine**
- **Getting data from CSVs into Spark DataFrames**
- **Cleaning and manipulating Spark DataFrames with SparkR**
- **Exporting Spark DataFrames to XDF**
- **Training models (logistic regression and decision forests) using ScaleR functions in Spark compute context**
- **Deploying models trained on ‘big’ data and operationalizing them in the cloud with Azure ML**

Prerequisites

Before starting this lab, you should have completed the accompanying documentation: *R Server on Spark - Creating a Cluster*. Having provisioned the cluster, you have the following architecture set-up in Azure:



R Server HDInsight Architecture

R Server Distributed Processing:

Master R process on Edge node

Apache YARN and Spark

Worker R processes on Data nodes

Data

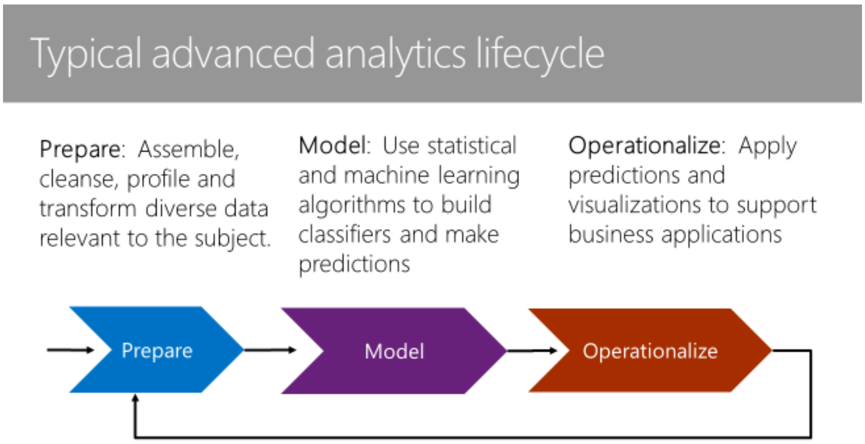
We will be using

- **Airline dataset (http://www.transtats.bts.gov): contains flight information for every US commercial flight**

- **Weather dataset (http://www.ncdc.noaa.gov/orders/qclcd/): contains hourly land-based weather observations from NOAA.**

Process

We will follow a typical analytics lifecycle i.e.



Typical advanced analytics lifecycle

Prepare: Assemble, cleanse, profile and transform diverse data relevant to the subject.

Model: Use statistical and machine learning algorithms to build classifiers and make predictions

Operationalize: Apply predictions and visualizations to support business applications

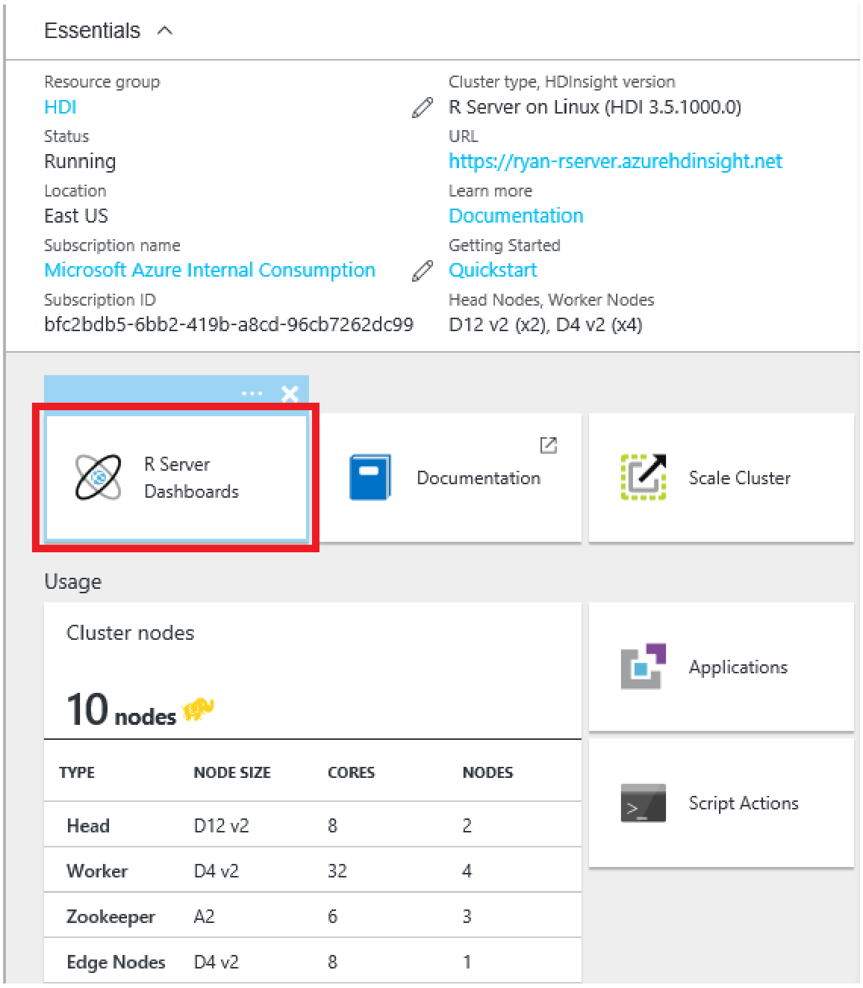
Where each step is done in a *scalable* way. For the data preparation (i.e. joining and aggregating) we use the data manipulation functions contained in the SparkR package. For the modelling we use Microsoft R Server’s ScaleR package to train models using the Spark Engine. Lastly, for operationalizing we publish the model to Azure ML as a web service.

Adjust the Yarn Memory limits

Updating the Yarn Memory configuration

In order to run the first sample script, we'll first have to ensure the memory settings for yarn.scheduler.maximum-allocation-mb' and 'yarn.nodemanager.resource.memory-mb' are at least 8192MB.

From the cluster summary page in the Azure Portal, we're going to adjust the memory settings in Ambari. You can get to ambary either from the 'Dashboard' button, or from some of the other dashboards available via 'Cluster Dashboards' -> R Server Dashboard:



Essentials

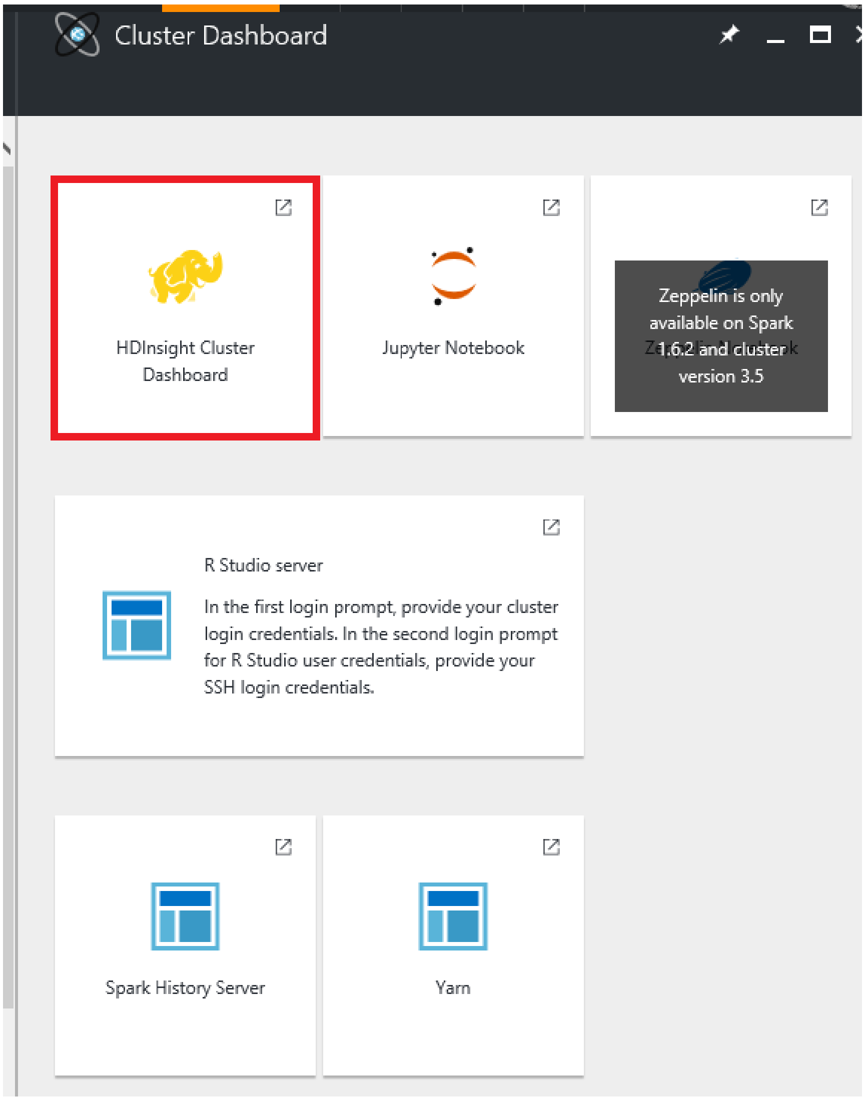
Resource group: HDI
Status: Running
Location: East US
Subscription name: Microsoft Azure Internal Consumption
Subscription ID: bfc2bdb5-6bb2-419b-a8cd-96cb7262dc99
Cluster type, HDInsight version: R Server on Linux (HDI 3.5.1000.0)
URL: https://ryan-rserver.azurehdinsight.net
Learn more: Documentation
Getting Started: Quickstart
Head Nodes, Worker Nodes: D12 v2 (x2), D4 v2 (x4)

Cluster nodes

10 nodes

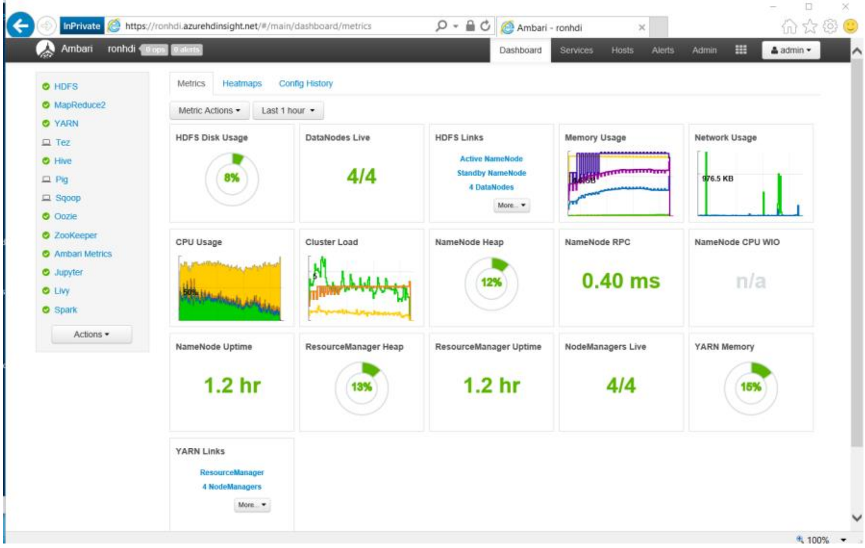
TYPE	NODE SIZE	CORES	NODES
Head	D12 v2	8	2
Worker	D4 v2	32	4
Zookeeper	A2	6	3
Edge Nodes	D4 v2	8	1

Then Select 'HDInsight Cluster Dashboard' to be taken to Ambari:


Cluster Dashboard
HDInsight Cluster Dashboard
Jupyter Notebook
Zeppelin is only available on Spark 1.6.2 and cluster version 3.5
R Studio server
Spark History Server
Yarn

In the first login prompt, provide your cluster login credentials. In the second login prompt for R Studio user credentials, provide your SSH login credentials.

You'll be logged into Ambari as the cluster admin (not the SSH Username):

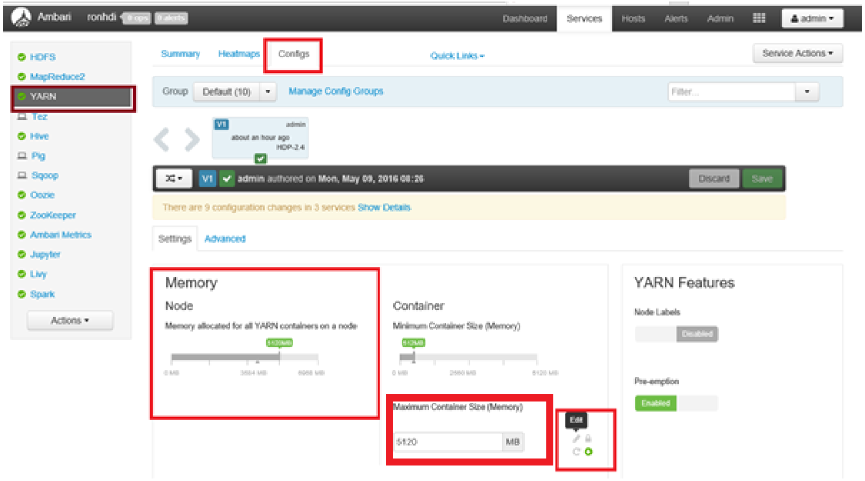


If you receive 'access denied' open a new 'in private browsing' and log back in using the Cluster Admin details.

Under 'Yarn' (left hand pane) 'Configs' tab, look at the values for Memory and Maximum Container Size.

<u>If these settings are already greater than 9024MB, then you do not need to do the following step and you can move on to the *Connecting to the R Server Edge Node* section.</u>

If your settings are **<u>less than</u>** 9024MB, then click on edit and adjust to 9024 > select **save** – proceed anyway, and select ok, restart affected, restart all and restart the services on the cluster. This may take 5 minutes or so:



V1 admin about an hour ago HDP-2.4

V1 admin authored on Mon, May 09, 2016 08:26

There are 9 configuration changes in 3 services Show Details

Node

Memory allocated for all YARN containers on a node: 5120MB

Minimum Container Size (Memory): 512MB

Maximum Container Size (Memory): 5120 MB

Node Labels: Disabled

Pre-emption: Enabled

Do not change any other settings using this version of Ambari, including anything regards Users, this will create unexpected behavior and may need you to recreate the cluster.

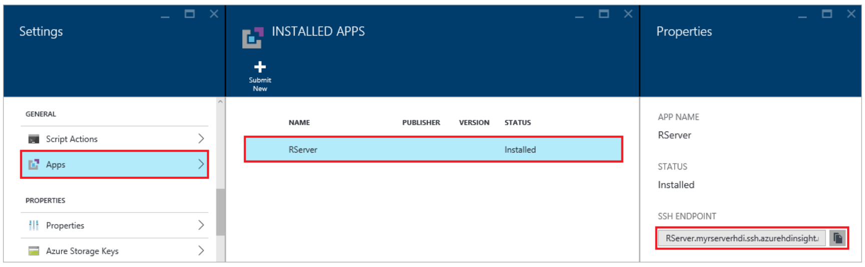
Connecting to the R Server Edge Node console

Connect to the R Server edge node

Whilst you'll run this lab using R Studio Community Edition, sometimes its useful to be able to get a Secure Shell onto the Edge node for example to create additional users, or run the R Command Line. You do not need to complete this step to use the rest of the lab, so feel free to move onto the next section if the console is not something you're interested in.

Connect to R Server edge node of the HDInsight cluster using SSH: or PuTTy (http://www.putty.org/)

You can find the R-Server.CLUSTERNAME.ssh.azurehdinsight.net address in the Azure portal by selecting your cluster, then All Settings, Apps, and RServer. This will display the SSH Endpoint information for the edge node:



NAME	PUBLISHER	VERSION	STATUS
RServer			Installed

APP NAME: RServer

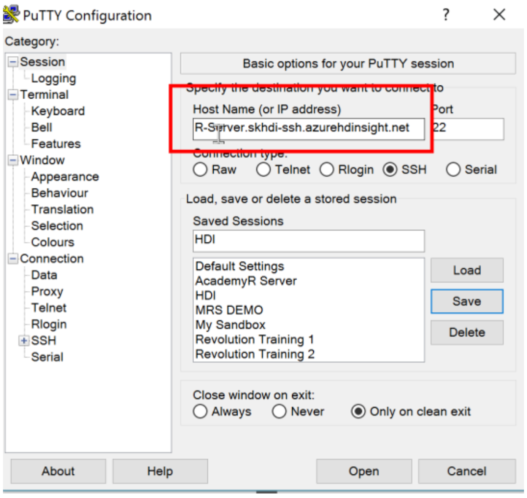
STATUS: Installed

SSH ENDPOINT: RServer.myrserverhdi.ssh.azurehdinsight.

If you used a password to secure your SSH user account (not the cluster admin), you will be prompted to enter it. If you used a public key, you may have to use the -i parameter to specify the matching private key. For example,

```
ssh -i ~/.ssh/id_rsa USERNAME@RServer.CLUSTERNAME-
ssh.azurehdinsight.net.
```

or in PuTTY you would enter the SSH endpoint information in the hostname text field:



Host Name (or IP address): R-Server.skhdi-ssh.azurehdinsight.net
Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Saved Sessions: HDI

Default Settings
AcademyR Server
HDI
MRS DEMO
My Sandbox
Revolution Training 1
Revolution Training 2

Close window on exit: ○ Always ○ Never ● Only on clean exit

For more information on using SSH with Linux-based HDInsight, see the following articles:

•Use SSH with Linux-based Hadoop on HDInsight from Linux, Unix, or OS X

•Use SSH with Linux-based Hadoop on HDInsight from Windows

Test the R Console

For this lab we'll be following R Studio, however, testing the R console is a quick way to ensure your edge node is up and running. From the SSH session, use the following command to start the R console:

```
R
```

You will see output similar to the following:

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical
Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO
WARRANTY.
You are welcome to redistribute it under certain
conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English
locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in
publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Server version 8.0: an enhanced
distribution of R
Microsoft packages Copyright (C) 2016 Microsoft
Corporation

Type 'readme()' for release notes.

>
```

From the > prompt, you can enter R code. R server includes packages that allow you to easily interact with Hadoop and run distributed computations. For example, use the following command to view the root of the default file system for the HDInsight cluster.

```
rxHadoopListFiles("/")
```

Note that in this instance, the HDFS store is actually Azure Blob store, so you can upload files to the cluster via blob store and process them with R at scale!

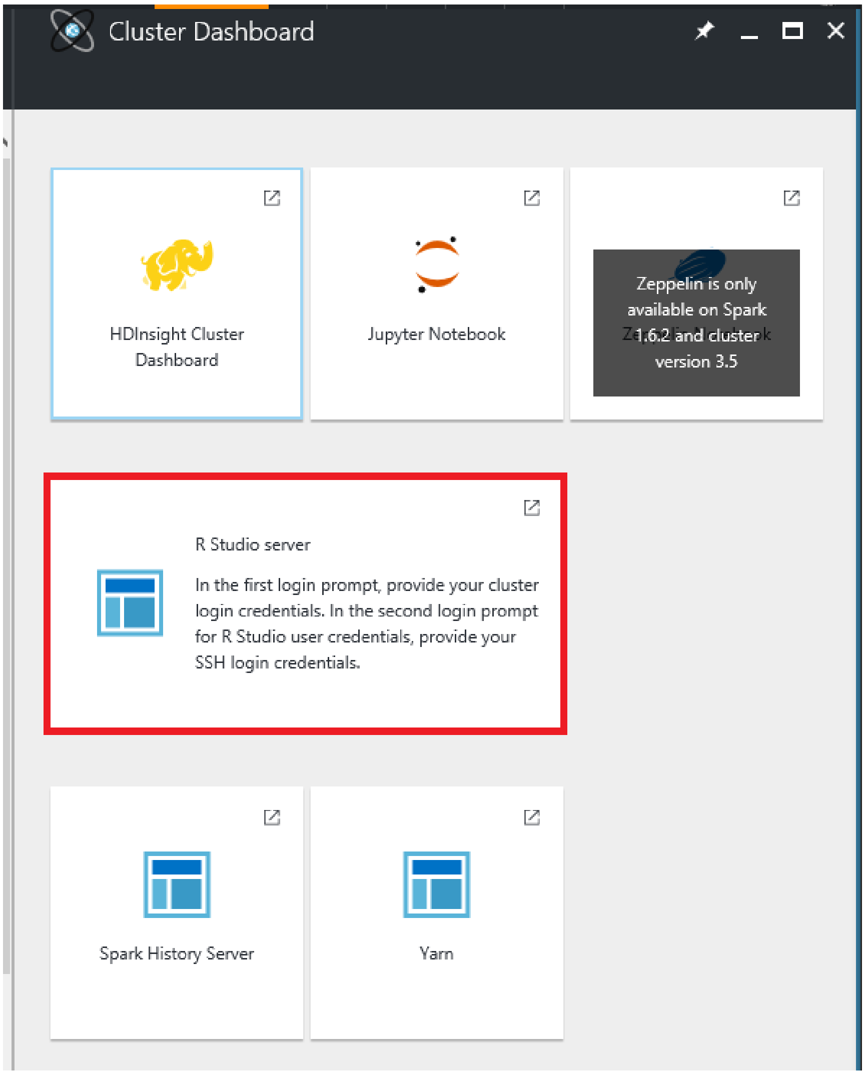
Working with R Studio Community Edition

Overview

There are multiple integrated development environments (IDE) available for R today, including Microsoft's recently announced R Tools for Visual Studio (RTVS), a family of desktop and server tools from RStudio, or Walware's Eclipse-based StatET. Among the most popular on Linux is the use of RStudio Server that provides a browser-based IDE for use by remote clients. HDInsight installs RStudio Server community edition on the edge node of an HDInsight Standard R Server cluster which provides a full IDE experience for the development and execution of R scripts with R Server on the cluster, and can be considerably more productive than default use of the R Console. Note that the community edition allows only 1 user at a time to be interactive with the console, if a second user tries to connect then the previous logged in will be logged off.

Opening RStudio

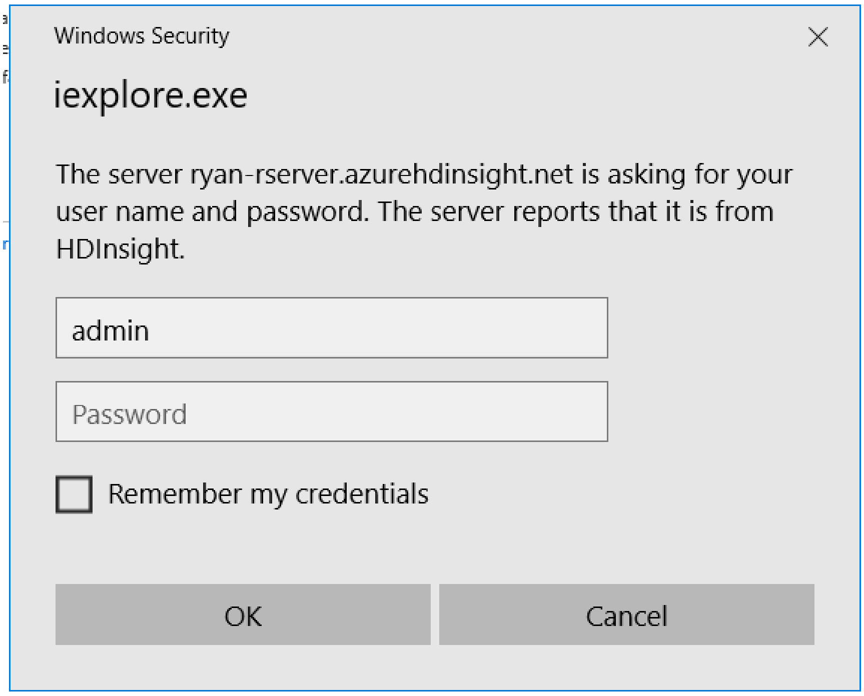
1. You can browse to https://<your-cluster-name>.azurehdinsight.net/rstudio or from the cluster dashboard pane you can select 'R Studio Server'



Cluster Dashboard
HDInsight Cluster Dashboard
Jupyter Notebook
Zeppelin is only available on Spark 1.6.2 and cluster version 3.5
R Studio server
Spark History Server
Yarn

In the first login prompt, provide your cluster login credentials. In the second login prompt for R Studio user credentials, provide your SSH login credentials.

1. You'll be prompted twice to log in – first for plain text authentication to authenticate with the cluster (use the local **admin cluster login**), then second to actually sign into the studio, use the **SSH user**.



Windows Security

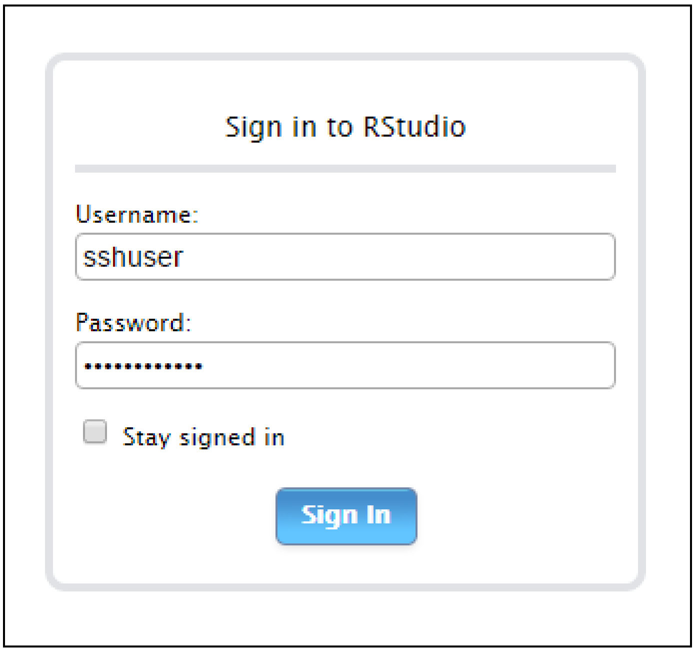
iexplore.exe

The server ryan-rserver.azurehdinsight.net is asking for your user name and password. The server reports that it is from HDInsight.

admin

Password

□ Remember my credentials



Sign in to RStudio

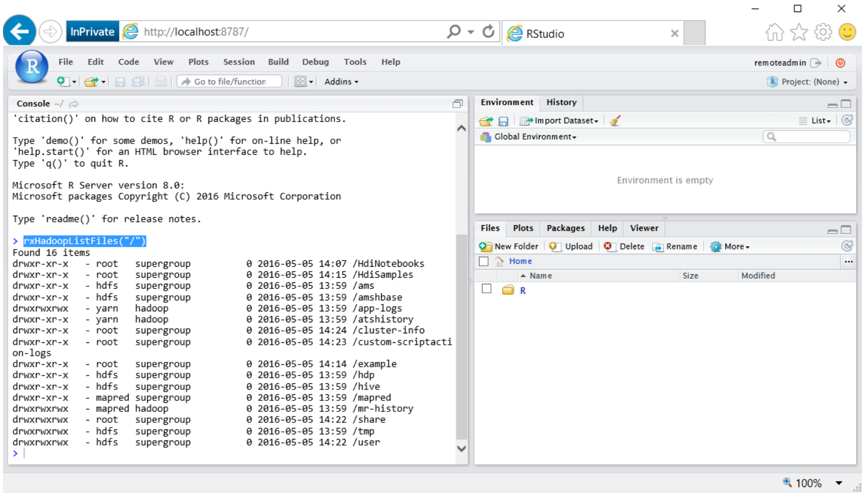
Username: sshuser

Password: ••••••••••••

□ Stay signed in

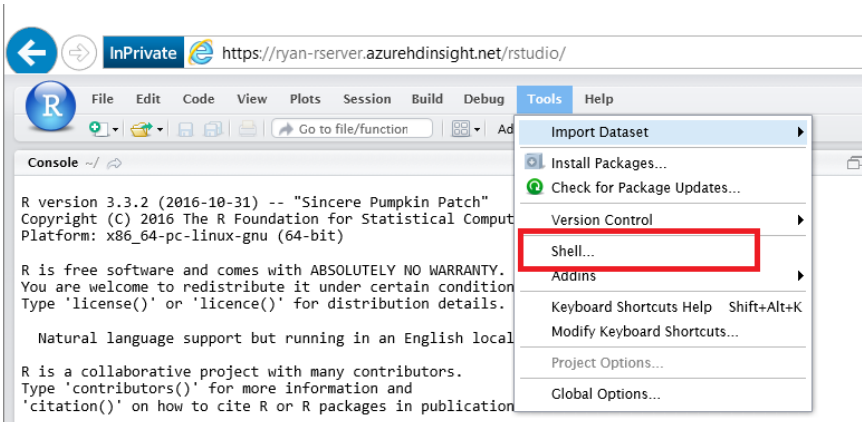
2. As with the command prompt, You can browse the HDFS file system using

```
rxHadoopListFiles("/"):
```



```
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Server version 8.0:
Microsoft packages Copyright (C) 2016 Microsoft Corporation

Type 'readme()' for release notes.

> rxHadoopListFiles("/")
Found 16 items
drwxr-xr-x   - root   supergroup          0 2016-05-05 14:07 /HdiNotebooks
drwxr-xr-x   - root   supergroup          0 2016-05-05 14:15 /HdiSamples
drwxr-xr-x   - hdfs   supergroup          0 2016-05-05 13:59 /ams
drwxr-xr-x   - hdfs   supergroup          0 2016-05-05 13:59 /amshbase
drwxrwxrwx   - yarn   hadoop              0 2016-05-05 13:59 /app-logs
drwxr-xr-x   - yarn   hadoop              0 2016-05-05 13:59 /atshistory
drwxr-xr-x   - root   supergroup          0 2016-05-05 14:24 /cluster-info
drwxr-xr-x   - root   supergroup          0 2016-05-05 14:23 /custom-scriptacti
on-logs
drwxr-xr-x   - root   supergroup          0 2016-05-05 14:14 /example
drwxr-xr-x   - hdfs   supergroup          0 2016-05-05 13:59 /hdp
drwxr-xr-x   - hdfs   supergroup          0 2016-05-05 13:59 /hive
drwxr-xr-x   - mapred supergroup          0 2016-05-05 13:59 /mapred
drwxrwxrwx   - mapred hadoop              0 2016-05-05 13:59 /mr-history
drwxrwxrwx   - root   supergroup          0 2016-05-05 14:22 /share
drwxrwxrwx   - hdfs   supergroup          0 2016-05-05 13:59 /tmp
drwxrwxrwx   - hdfs   supergroup          0 2016-05-05 14:22 /user
> 
```

To test whether the RStudio installation was successful, you can run a test script that executes R based MapReduce and Spark jobs on the cluster. You can run the following from a remote shell session, or use the shell tool in R Studio by selecting ‘Tools.. shell’



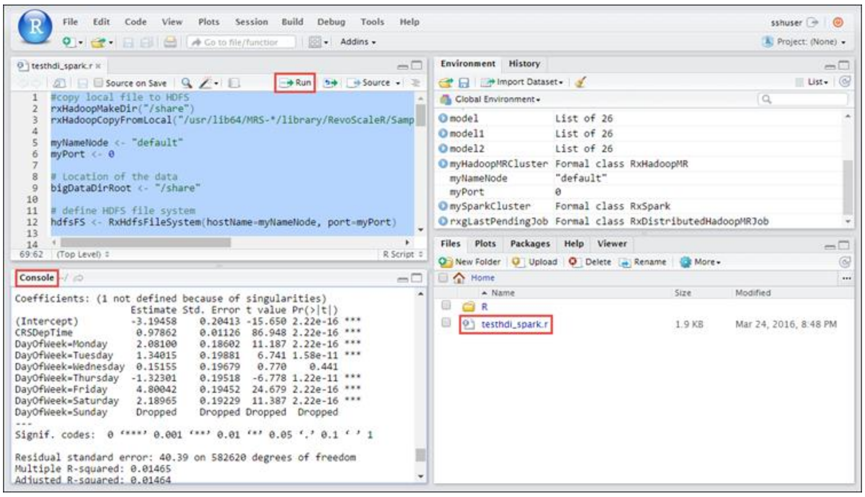
```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Comput
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain condition
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English local

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publication
```

https://ryan-rserver.azurehdinsight.net/rstudio/
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Ad
Console ~/
Import Dataset
Install Packages...
Check for Package Updates...
Version Control
Shell...
Addins
Keyboard Shortcuts Help Shift+Alt+K
Modify Keyboard Shortcuts...
Project Options...
Global Options...

3. Run the following from your shell to download the test script:

```
wget
http://mrsactionscripts.blob.core.windows.net/rstudio
-server-community-v01/testhdi_spark.r
```

4. In RStudio, you will see the test script **(testhdi_spark.r)** you downloaded. Double click the file to open it, select the contents of the file using ‘ctrl-a’, and then click Run. You should see the output in the Console pane.



```
#copy local file to HDFS
rxHadoopMakeDir("/share")
rxHadoopCopyFromLocal("/usr/lib64/MRS-*/library/RevoScaleR/Samp

myNameNode <- "default"
myPort <- 0

# Location of the data
bigDataDirRoot <- "/share"

# define HDFS file system
hdfsFS <- RxHdfsFileSystem(hostName=myNameNode, port=myPort)
```

```
Coefficients: (1 not defined because of singularities)
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           -3.19458    0.20413 -15.650 2.22e-16 ***
CRSDepTime             0.97862    0.01126  86.948 2.22e-16 ***
DayOfWeek=Monday       2.08100    0.18602  11.187 2.22e-16 ***
DayOfWeek=Tuesday      1.34015    0.19881   6.741 1.58e-11 ***
DayOfWeek=Wednesday    0.15155    0.19679   0.770    0.441
DayOfWeek=Thursday    -1.32301    0.19518  -6.778 1.22e-11 ***
DayOfWeek=Friday       4.80042    0.19452  24.679 2.22e-16 ***
DayOfWeek=Saturday     2.18965    0.19229  11.387 2.22e-16 ***
DayOfWeek=Sunday       Dropped    Dropped Dropped  Dropped
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 40.39 on 582620 degrees of freedom
Multiple R-squared: 0.01465
Adjusted R-squared: 0.01464
```

model	List of 26
model1	List of 26
model2	List of 26
myHadoopMRCluster	Formal class RxHadoopMR
myNameNode	"default"
myPort	0
mySparkCluster	Formal class RxSpark
rxgLastPendingJob	Formal class RxDistributedHadoopMRJob

Name	Size	Modified
R		
testhdi_spark.r	1.9 KB	Mar 24, 2016, 8:48 PM

This test script shows running in **"local" mode and streaming data from HDFS** i.e.

```
# define the data source
airDS <- RxTextData(file = inputFile, missingValueString = "M",
                    colInfo = colInfo, fileSystem = hdfsFS)

# First test the "local" compute context
rxSetComputeContext("local")

# Run a linear regression
system.time(
  model <- rxLinMod(ArrDelay~CRSDepTime+DayOfWeek, data = airDS)
)
```

The test script shows running the same linear model as a **MapReduce** job, by just switching the compute context i.e.

```
# define MapReduce compute context
myHadoopMRCluster <- RxHadoopMR(consoleOutput=TRUE,
                                nameNode=myNameNode,
                                port=myPort,
                                hadoopSwitches="-libjars /etc/hadoop/conf")

# set compute context
rxSetComputeContext(myHadoopMRCluster)

# Run a linear regression
system.time(
  model1 <- rxLinMod(ArrDelay~CRSDepTime+DayOfWeek, data = airDS)
)
```

The test script also demonstrates running the same linear model as a **Spark job** i.e.

```
# define Spark compute context
mySparkCluster <- RxSpark(consoleOutput=TRUE)

# set compute context
rxSetComputeContext(mySparkCluster)

# Run a linear regression
system.time(
  model2 <- rxLinMod(ArrDelay~CRSDepTime+DayOfWeek, data = airDS)
)

# display a summary of model
```

Once you have finished with this script, switch back to local compute context by issuing the following command:

```
> rxSetComputeContext("localpar")
```

Downloading the Sample Data and loading into HDFS

Overview

There is some sample airline data for you to clean, join and then predict on using R Studio, along with the required script files.

Before continuing you'll need to load them onto the R Server Edge Node.

Loading the files

From the Shell Tool (either Putty or Tools.. shell)

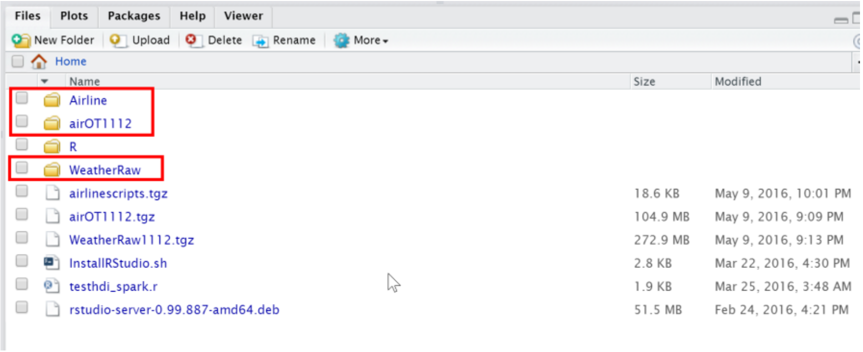
Run the following:

```
wget
https://msrlabdemos.blob.core.windows.net/airlinesubset/airOT1112.tgz

wget
https://msrlabdemos.blob.core.windows.net/airlinesubset/WeatherRaw1112.tgz

wget
https://msrlabdemos.blob.core.windows.net/airlinesubset/airlinescripts.tgz
```

Once downloaded, unpack on the R node to the home directory (default) using the following commands:

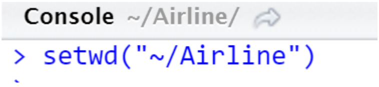
```
tar -xvf WeatherRaw1112.tgz
tar -xvf airOT1112.tgz
tar -xvf airlinescripts.tgz
```

If you haven't already done so, log into R Studio. You should see the unzipped folders in your home directory:



Name	Size	Modified
Airline		
airOT1112		
R		
WeatherRaw		
airlinescripts.tgz	18.6 KB	May 9, 2016, 10:01 PM
airOT1112.tgz	104.9 MB	May 9, 2016, 9:09 PM
WeatherRaw1112.tgz	272.9 MB	May 9, 2016, 9:13 PM
InstallRStudio.sh	2.8 KB	Mar 22, 2016, 4:30 PM
testhdi_spark.r	1.9 KB	Mar 25, 2016, 3:48 AM
rstudio-server-0.99.887-amd64.deb	51.5 MB	Feb 24, 2016, 4:21 PM

At the R Console set your working directory to the Airline directory, i.e.



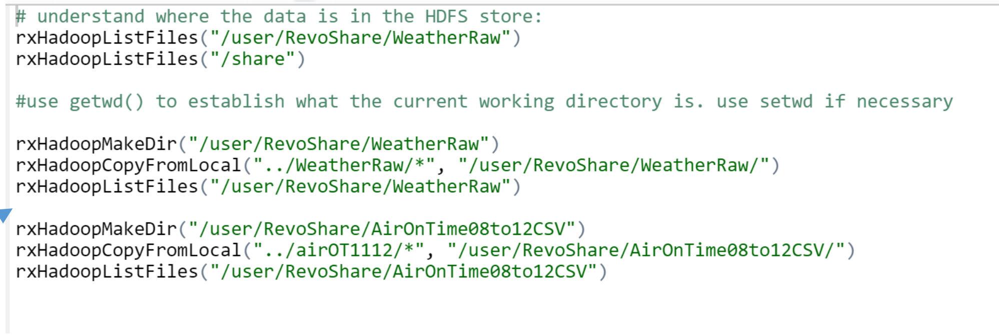
```
> setwd("~/Airline")
```

In the **Airline** directory you will see an R script called **LoadData.r** > Open and run this script line-by-line in R. Below is a description of what the LoadData.r script is doing:

List the files in HDFS – do they exist?

Make a directory called WeatherRaw in HDFS and copy the local (linux) files to that location

Do the same for the airlines data



```
# understand where the data is in the HDFS store:
rxHadoopListFiles("/user/RevoShare/WeatherRaw")
rxHadoopListFiles("/share")

#use getwd() to establish what the current working directory is. use setwd if necessary

rxHadoopMakeDir("/user/RevoShare/WeatherRaw")
rxHadoopCopyFromLocal("../WeatherRaw/*", "/user/RevoShare/WeatherRaw/")
rxHadoopListFiles("/user/RevoShare/WeatherRaw")

rxHadoopMakeDir("/user/RevoShare/AirOnTime08to12CSV")
rxHadoopCopyFromLocal("../airOT1112/*", "/user/RevoShare/AirOnTime08to12CSV/")
rxHadoopListFiles("/user/RevoShare/AirOnTime08to12CSV")
```

The data files now exist in HDFS and this means that we can use them in Spark and/or MapReduce.

Running the scripts

The Scripts can be found in the ~/Airline Directory:

1. Run script **0-Weather-Preprocess.r** (do this ‘command-by-command’ to get an understanding of what is running). This script uses an rxDataStep (in a spark Compute context) to adjust time in the airline dataset to be the same as the weather dataset.

2. Run script **1-Clean-Join.r** (do this ‘command-by-command’ to get an understanding of what is running). In this script we are using SparkR is aggregate and join datasets. Once we are happy with the dataset, we export it to XDF for training a predictive model.

   Exercises:

   a. For the SparkR DataFrame airDF, can you create a new DataFrame containing only observations from the year 2012

      Hint: the SparkR package has a filter function. More details can be found on the help page here.

   b. Can you compute (using SparkR) the average Arrival Delay (ARR_DELAY column) by year (YEAR column)?

      Hint: Look up the groupBy function in the documentation here for an example. You should have something like:

      mydf <- agg(groupBy(),…)

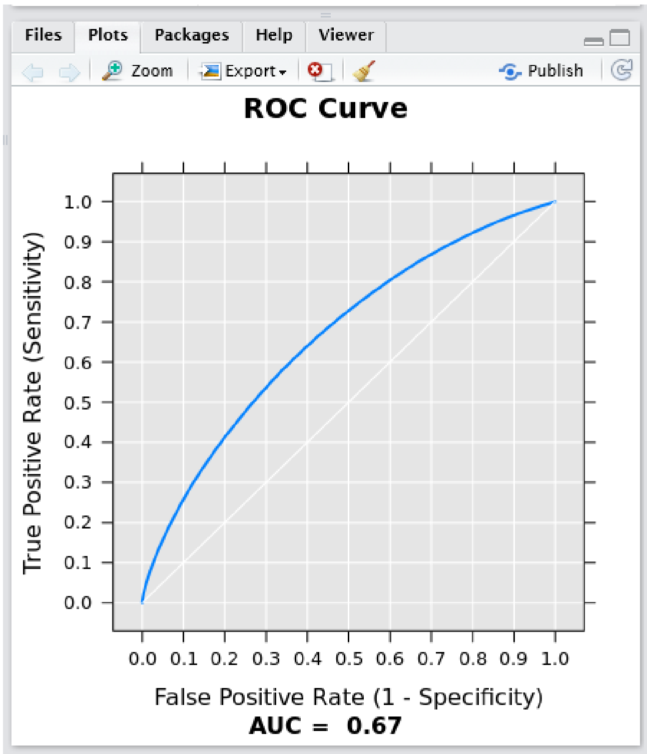
      When you do your aggregation does anything run in Spark? What does mydf look like?

      How we get Spark to run the aggregation and pull back to a native data.frame?

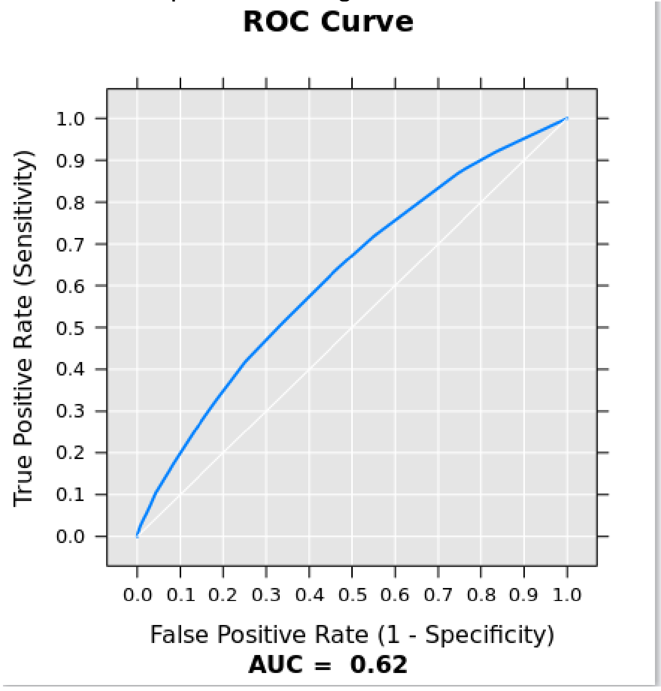
      Hint: Look up “Under the Hood: Intelligent Optimization and Code Generation” here. A DataFrame can be coerced to a data.frame in R using as.data.frame(). In this case the aggregated data is small, but we must be careful with other cases where the data could be huge and we risk blowing RAM on the edge node.

3. Run script **2-Train-Test.r** (do this ‘command-by-command’ to get an understanding of what is running). In this script we split our XDF data into training and test sets. We build a logistic regression and decision tree (in Spark) on the training set. We use rxPredict and rxRoc to test and validate the model with Receiver Operator Curves.

   i.e. you will see the following plots when the ROCs are plotted



Output for the regression model



The plot for the decision trees – note the slightly lower area under the curve

Exercises:

a. Can you create a wrapper function to predict to send to Azure ML?

   Hint: you will need to coerce the dTreeModel object to an open source equivalent because Azure ML does not yet use ScaleR functions. To coerce, use:

```
rpartModel <- as.rpart( dTreeModel )
```

b. Publish the wrapper function to AzureML as a web service.

Terms of Use

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a "sandbox" testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of the features and enhancements in Microsoft Azure. Some of the features might change in future releases of the product.