



BAIL 302

Getting Started with Advanced Analytics using Azure HDInsight R Server

Introduction

In this lab, you will start with creating a R Server on HDInsight Cluster. Once the server is running, you connect to the server using the R Studio Server, and work with some basic R scripts which articulates how Microsoft R Server 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.

Next, you will upload sample data from CSVs and sample scripts into the server and then clean, transform the data using Spark DataFrames with SparkR.

You will then export Spark DataFrames to XDF, build and train predictive models using (logistic regression and decision forests) using ScaleR functions in Spark compute context.

Lastly, to operationalize the model, you will deploy the trained model as a web service to Azure Machine Learning and consume it using R scripts.

As an optional step, you will also learn how to use Yarn to change the memory settings of the HDInsight Cluster when you do analysis for large datasets.

Estimated Time

75 minutes

Objectives

At the end of this lab, you will be able to:

- Learn how to create a R Server on Azure HDInsight
- Learn how to connect to R Studio Server on the Spark Edge Node
- Work with Microsoft R Server
- Ingest data from CSV to Spark DataFrames, perform pre-processing, cleaning, and manipulating data with Spark R
- Create/train models using SparkR functions
- Learn how to publish the predictive model to Azure ML as web service from R Studio
- Learn how to consume the web service R Studio
- How to use Yarn to modify HDInsight Cluster memory settings

Logon Information

Use the following credentials to login into virtual environment

- Username: Administrator
- Password: Password1

Note: The above logon information is different from the logon information we will use for creating the HDInsight Cluster and subsequent connections to it.

Table of Contents

BAIL 302.....	1
Getting Started with Advanced Analytics using Azure HDInsight R Server	1
Lab Title: Getting Started with Advanced Analytics using Azure HDInsight R Server	4
Pre-work: Redeeming an Azure Pass.....	4
Exercise 1: Create R Spark HDInsight cluster	6
Exercise 2: Connecting to the R Studio Server	16
Exercise 3: Uploading the Sample Data and loading into HDFS	23
Exercise 4: Create Predictive Models	29
Exercise 5: Publish to Azure Machine Learning as a Web service.....	38
Exercise 6: (If time permits) Adjust the Yarn Memory limits.....	43

Lab Title: Getting Started with Advanced Analytics using Azure HDInsight R Server

The steps in this lab document covers how to create an R Server on HDInsight, use for advanced analytics workload.

Pre-work: Redeeming an Azure Pass

In this pre-work, if required, you will redeem an Azure Pass by using the provided promo code and account.

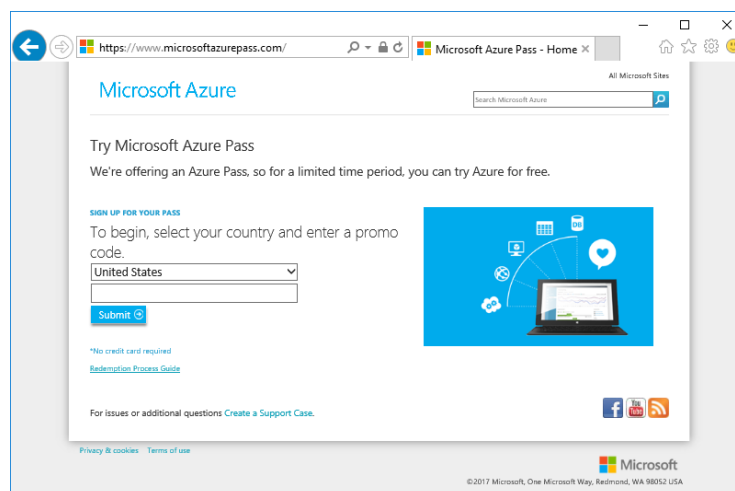
In order to complete this lab, you will need an Azure subscription. It is recommended that you do not use your Microsoft Azure subscription. An Azure Pass has been made available for you to complete this lab, and it should be redeemed by the provided trial account. Note that it takes about 5 minutes to complete the registration process.

Redeeming an Azure Pass

In this task, you will redeem the Azure Pass. If you have previously redeemed an Azure Pass, and you have available credit, you can continue to use it in this lab—just sign in to the Azure Portal.

1. To open Internet Explorer, on the taskbar, click the Internet Explorer program shortcut.
2. To redeem your Azure Pass, navigate to <https://www.microsoftazurepass.com> , and then complete the registration process:
 - Select **your country**, and enter the **promo code** provided (available in the lab portal)

The lab instructions will use the English version of the Azure Portal, so consider selecting US as the country.



- Sign in with the account provided (available in the lab portal)
 - When signing in, you will be required to change the password—**be sure to remember the password**, as it will be required later in this lab.
 - Enter all additional requested details, and then click **Activate**
 - When directed to the Microsoft Azure site, enter your first and last names, email address, and phone number (the phone number will not be used during the registration process)
 - If you agree to the terms, click **Sign Up**.
 - The registration process will take a couple of minutes to complete. When it has completed click **Get Started with Your Azure Subscription**.
3. Verify that you are signed in to the Azure Portal.

Wait for the instructor-led session to commence.

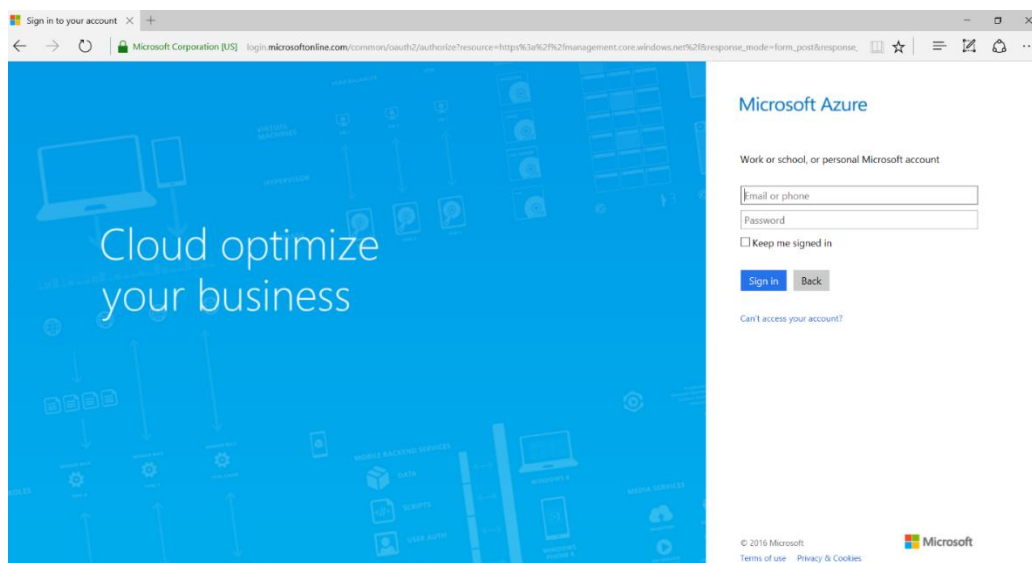
Exercise 1: Create R Spark HDInsight cluster

The steps in this exercise covers how to create an R Server on HDInsight using basic configuration steps.

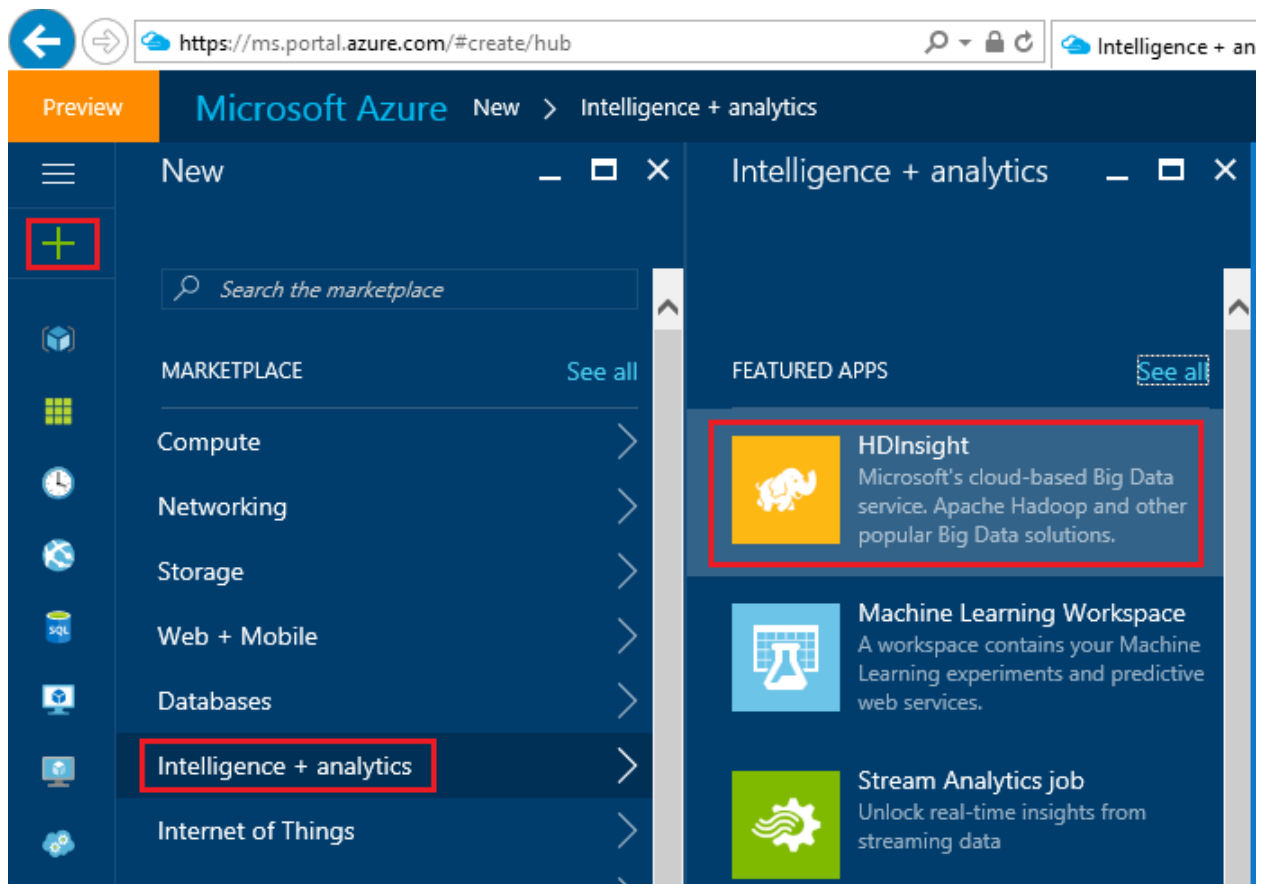
To perform the cluster creation steps should take no more than 5-7 minutes; however, it may currently take up to 30 minutes to provision a cluster. You will create this cluster so that it can be used later to connect to the R server and carry out the remaining exercises

First step is to access Microsoft Azure Portal.

1. Login to Azure portal (<https://portal.azure.com>) with Internet Explorer and sign in using your Azure Pass account given by your instructor.



2. From the Dashboard, select the **+**New in the upper-left corner. From the blade, select Intelligence + Analytics, and then HDInsight.



3. Enter a name for the cluster in the Cluster Name field.

Important Note: Ensure you use lowercase characters only for the cluster name. Using upper case or a mix of upper and lower case will result in errors in some of the R scripts later

Resources created in Azure are required to be unique. Enter <your initials><yyyymmdd>clus as the cluster name. For example: johng20161210clus

Basics

*

 Cluster name

tr24bail302sparkclus

✓

.azurehdinsight.net

*


 Subscription

Microsoft Azure Internal Consumption

▼

Under the Resource Group section, check “Create new” and provide a unique Resource Group name. *i.e. tr24bail302sparkrg*

*

 Resource group 

☒ Create new ☐ Use existing

tr24bail302sparkrg

×

*

 Location

East US

▼

 [Click here](#) to check your cores usage in each region.

- Click Cluster Configuration and select the following on the Cluster configuration blade:

Cluster Type: R server

Version: R server 9.0 (HDI 3.5)

Cluster Tier: Standard

R Studio Community Edition on R server: Checked

Click the Select button to save the selections made.

Cluster configuration

Learn about HDInsight and cluster versions. [Learn more](#)

Cluster configuration

* Cluster Type ⓘ

R Server

* Operating System

Linux

* Version

R Server 9.0 (HDI 3.5)

* Cluster Tier ⓘ

STANDARD

PREMIUM

R Server : Terabyte-scale, enterprise grade R analytics with transparent parallelization on top of Spark and Hadoop.

Configuration Options:

- R Server 9.0.0 on Spark 2.0.0 with Java 8
- R Server 8.0.5 on Spark 1.6.2 with Java 7

Adds \$0.244 per Core-Hour.

Features

* denotes preview feature

Available

☒ R Studio community edition for R Server

+ Secure shell (SSH) access

+ HDInsight applications

+ Custom virtual network

+ Custom Hive metastore

+ Custom Oozie metastore

+ Data Lake Store access

+ ADLS as primary FS (storage)

Not available

+ Apache Ranger* (PREMIUM) ⓘ

+ Domain joining* (PREMIUM) ⓘ

+ Remote Desktop access ⓘ

Select

9

5. For the cluster login username and password, enter the below:

Cluster Login Username: myadmin

Cluster Login Password: Password12345abc+

SSH is used to connect to the cluster using a RStudio Server later. Click the Select button to save the credentials information entered.

* Cluster login username ⓘ

✓

Secure Shell (SSH) username ⓘ

* Cluster login and SSH password ⓘ

🔍 ✓

6. Click Next after ensuring all the above parameters are filled correctly.

Basics

* Cluster name

tr24bail302sparkclus



.azurehdinsight.net

* Subscription

Microsoft Azure Internal Consumption



* Cluster type 

R Server 9.0 on Linux (HDI 3.5)



* Cluster login username 

myadmin



Secure Shell (SSH) username 

sshuser

* Cluster login and SSH password 

.....



* Resource group 

☒ Create new ☐ Use existing

tr24bail302sparkrg



* Location

East US



Click here to check your cores usage in each region.

Next

7. Under **Storage** blade, click on the “Create New” hyperlink to create a new storage account, fill below parameters and click next.

Create a new storage account: <your initials><yyyymmdd>sa eg: johng20161210sa

Choose Default Container: <your initials><yyyymmdd>c eg: johng20161210c

Storage

The cluster will use this data source as the primary location for most data access, such as job input and log output.

Storage Account Settings

* Primary storage type

☒ Azure Storage ☐ Data Lake Store

* Selection method ⓘ

☒ My subscriptions ☐ Access key

* Create a new Storage account

tr24bail302sparksa ✓

Select existing

* Default container ⓘ

tr24bail302sparkd × ✓

Additional storage accounts

Optional >

Metastore Settings (optional)

Filtered to location and subscription of cluster.

i

To preserve your metadata outside this cluster, link a SQL database to this account.

Select a SQL database for Hive

Select a database, type to filter.

Select a SQL database for Oozie

Select a database, type to filter.

Next

12

8. On the **Applications** blade, click next with defaults.
9. Select **Pricing** blade to display information about the nodes that will be created for this cluster. Leave the number of worker nodes at the default of 4. The estimated cost of the cluster will be shown within the blade. Click Next.

Cluster size

To learn more, visit our pricing page.
Learn more

Number of Worker nodes

4

* Worker node size

D4 v2 (4 nodes, 32 cores)

* Head node size

D12 v2 (2 nodes, 8 cores)

* R Server edge node size


D4 v2 (1 node, 8 cores)

WORKER NODES	$1.243 \times 4 = 4.972$
HEAD NODES	$0.760 \times 2 = 1.520$
EDGE NODE	$1.243 \times 1 = 1.243$
R SERVER SURCHARGE	$0.080 \times 48 = 3.840$
TOTAL COST	11.58
	USD/HOUR (ESTIMATED)

48 of 60 cores would be used in East US.

This price estimate does not include storage costs, network egress costs, or subscription discounts.

Questions? [Contact billing support.](#)



Note: Clusters with more than 32 Worker nodes require a Head node size with at least 8 cores and 14 GB RAM.

Next

10. In the **Advanced Settings** blade, accept defaults and click next.

Advanced settings

Remote Access

* Secure Shell (SSH) username ⓘ

sshuser

✕

☒ Use same password as cluster login

Script Actions (optional)

Script actions

Optional

>

Virtual Network Settings (optional)

Filtered to location and subscription of cluster.

Virtual network

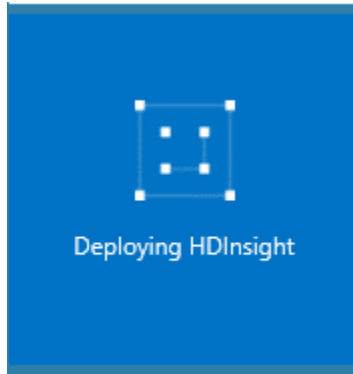
Select or type to filter virtual networks

Subnet

Please select a valid virtual network

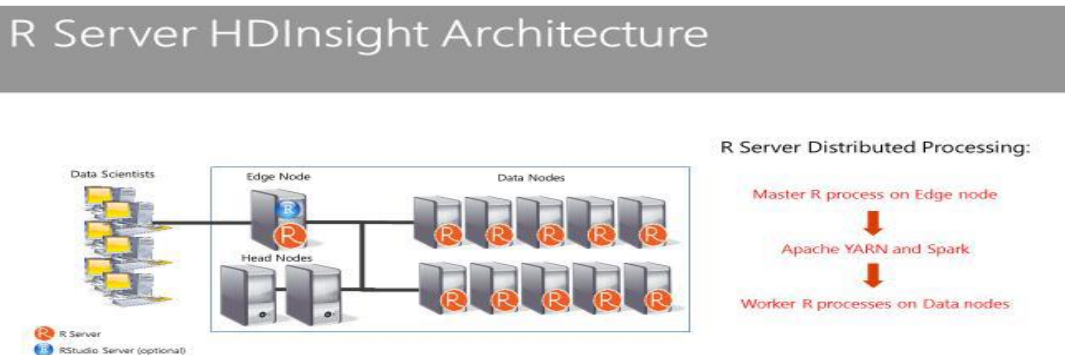
Next

11. On the summary blade, review the parameters and click Create.
12. The icon on the Azure dashboard will indicate that the cluster is creating, and will change to running and display the HDInsight icon once creation has completed.



It will take some time for the cluster to be created, usually around 15 minutes, however allow up to 30. Use the tile on the Dashboard, or the Notifications entry on the left of the page to check on the creation process.

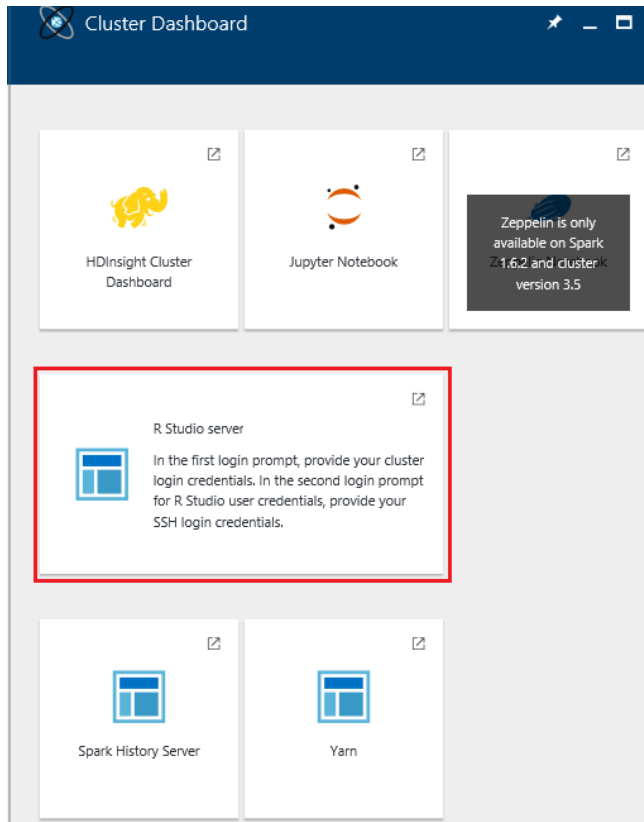
Summary: In this exercise, you have learnt how to create a R Server on HDInsight Cluster. Once the cluster is running, you have the following architecture set-up in Azure.



Exercise 2: Connecting to the R Studio Server

In this exercise, you will connect to R Studio Server from the Azure portal, open R Studio and run a test R script.

1. From the Quick Links click on, click on R Server dashboards, and on the Cluster dashboard click on R Studio server.



2. When prompted to enter the username and password, enter the myadmin as user name and password as Password12345abc+
3. On the connect to R Studio dialog, enter the SSH username and password (user name as sshuser and password as Password12345abc+)

Sign in to RStudio

Username:

sshuser

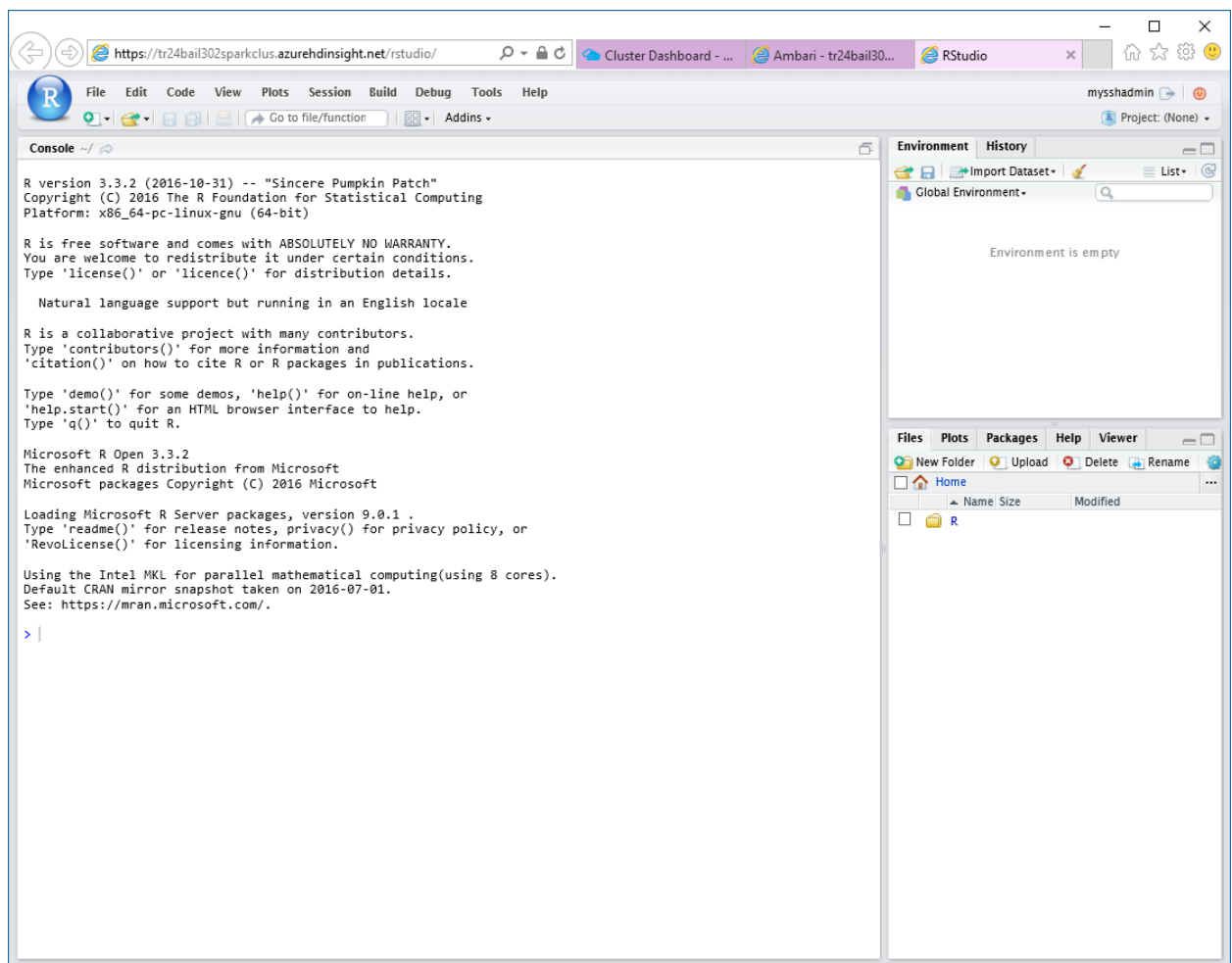
Password:

.....

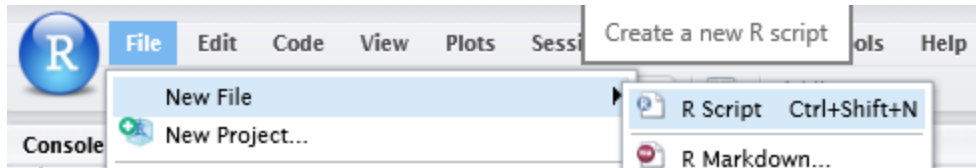
☐ Stay signed in

Sign In

- You will be logged on to the R Studio.

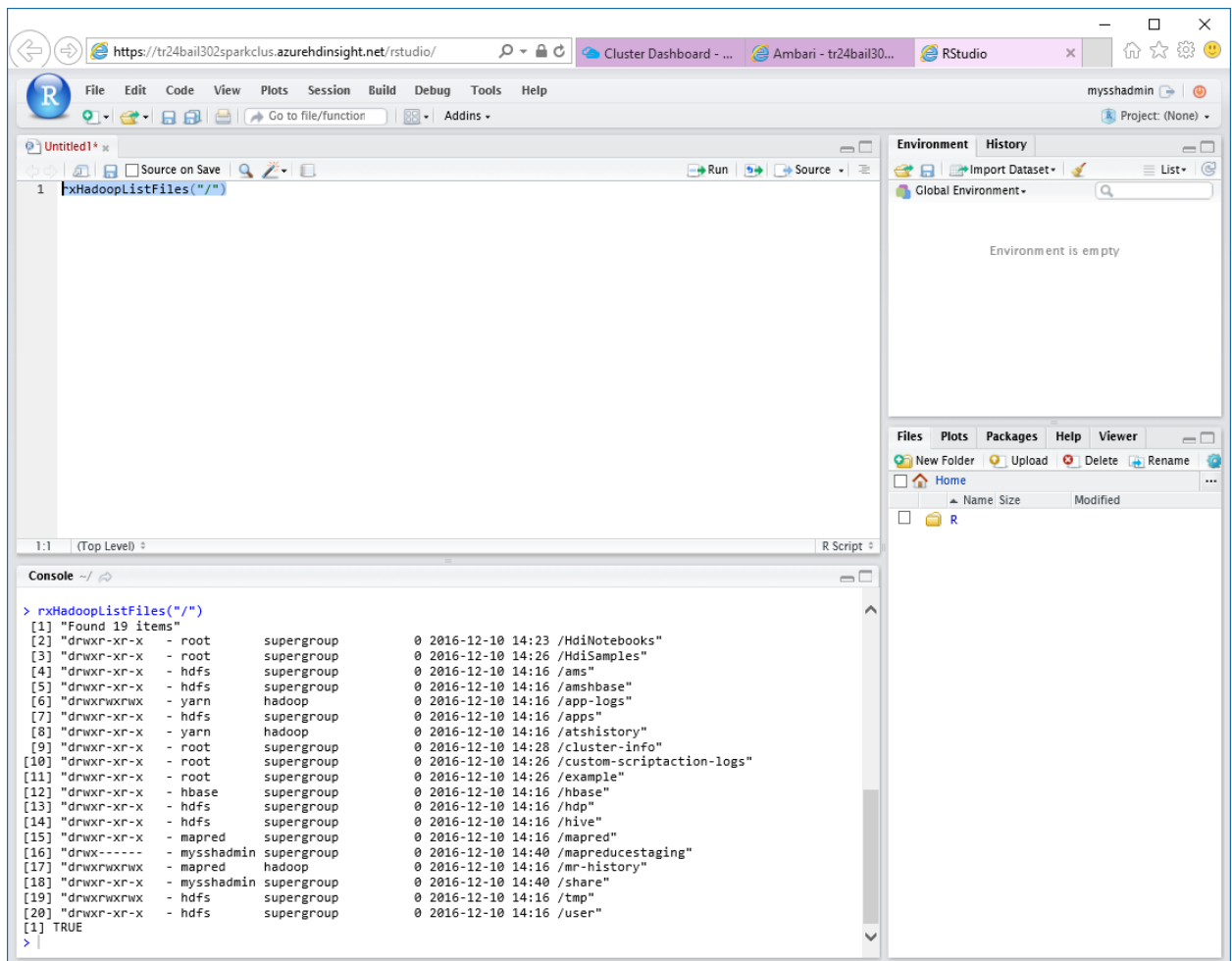


- From File menu, select New File then select R script.

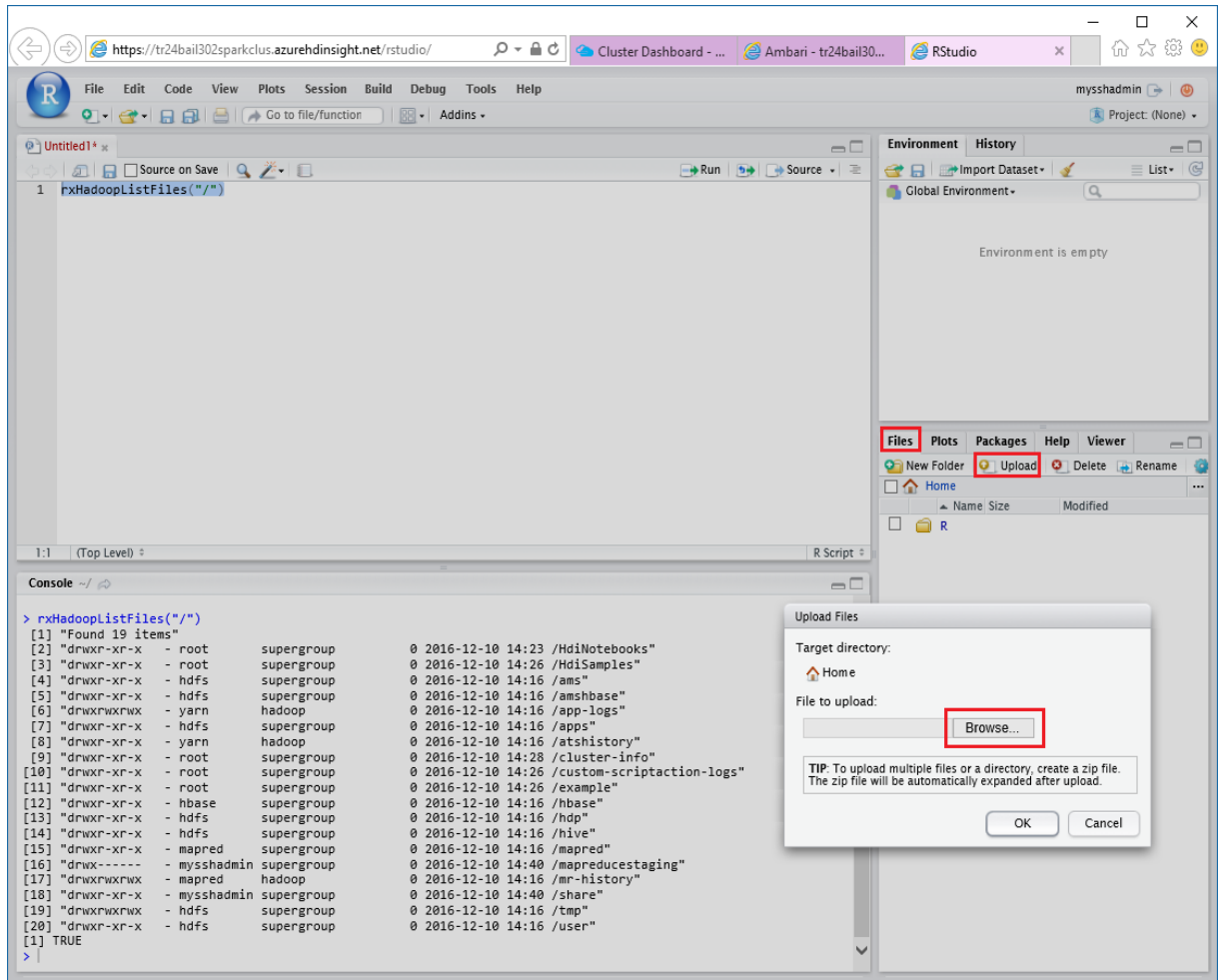


Step 6 and 7 shows how to run test R script from RStudio. These steps are optional. If you wish, you can skip to Exercise 3 directly.

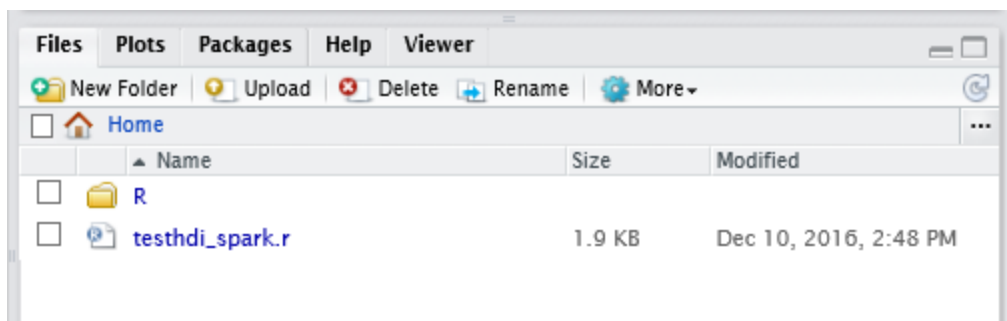
6. Type `rxHadoopListFiles("/")` to browse the filesystem, then press `ctrl+enter` to run the command.



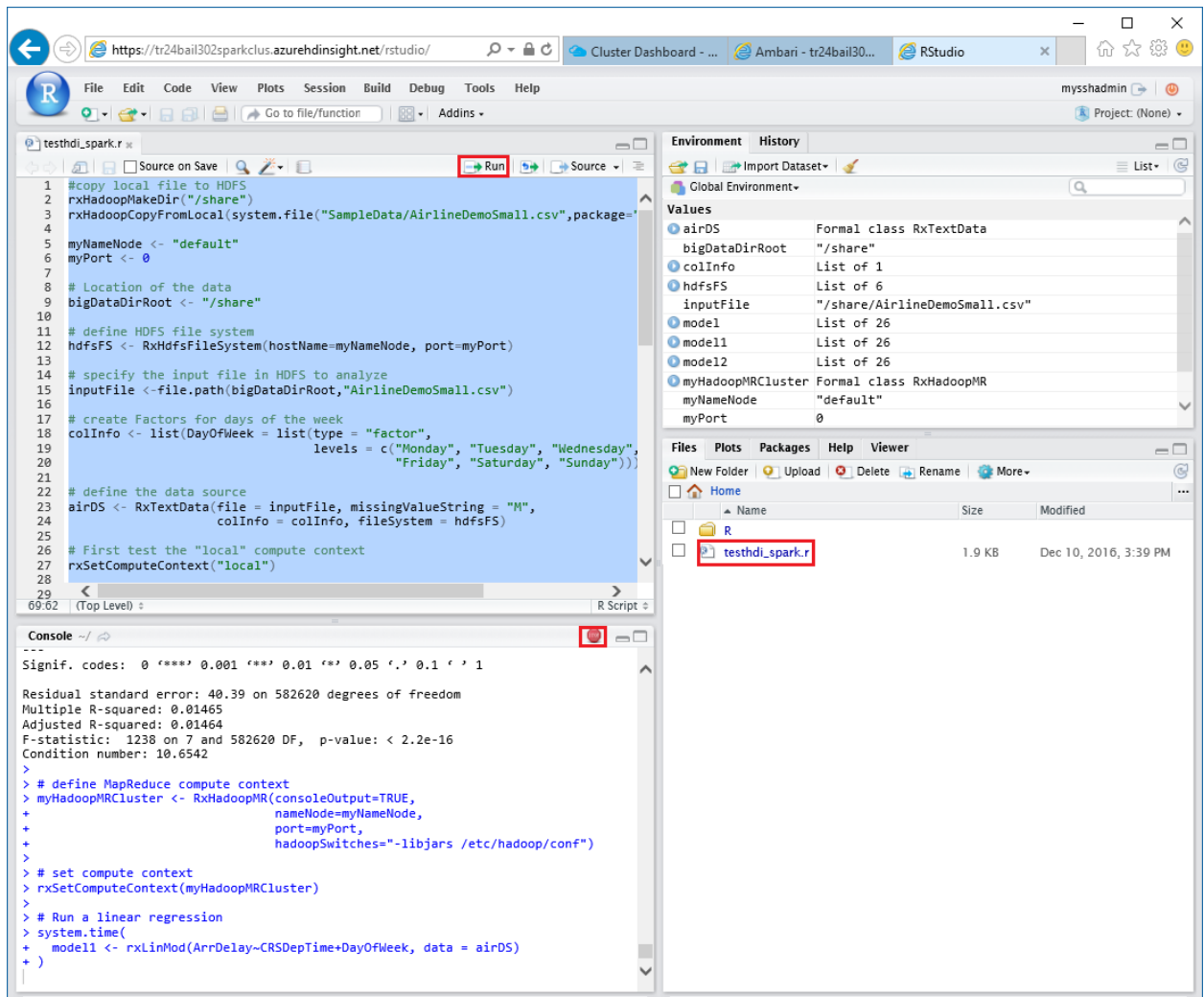
7. Now, you will upload and run a test script that executes R based MapReduce and Spark jobs on the cluster. To do that, from the Files Pane, select upload file.



Upload file testhdi_spark.r from C:\TR24BAIL302Labs folder.



- In RStudio, you will see the test script (testhdi_spark.r) you uploaded. Open the file, select the contents of the file using 'ctrl+r+a', and then click Run.



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.

```
# 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.

```
# 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)
)
```

The screenshot shows the RStudio IDE interface. The main editor window displays a script named 'testthdi_spark.r'. The script defines a MapReduce cluster, sets the compute context, and runs a linear regression model. The console window at the bottom shows the output of the script, including the linear regression results for the model $ArrDelay \sim CRSDepTime + DayOfWeek$.

Linear Regression Results for: $ArrDelay \sim CRSDepTime + DayOfWeek$
 Data: airDS (R_TextData Data Source)
 File name: /share/AirlineDemoSmall.csv
 Dependent variable(s): ArrDelay
 Total independent variables: 9 (Including number dropped: 1)
 Number of valid observations: 582628
 Number of missing observations: 17372

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

The Environment pane on the right shows the loaded packages and data objects, including 'myHadoopMRCluster', 'mySparkCluster', and 'airDS'. The Files pane at the bottom shows the file structure, including 'testthdi_spark.r'.

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

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

Summary: In this exercise, you have connected to R Studio using the Azure portal and have learnt how to execute R scripts from R Studio. Please proceed with next exercise.

Exercise 3: Uploading the Sample Data and loading into HDFS

In this exercise, you will upload the sample data and scripts into the R Server Edge Node first.

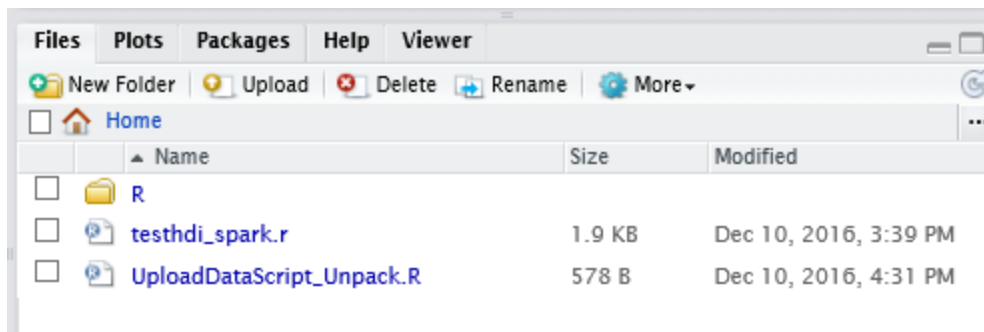
Data: You will use following public data:

- 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

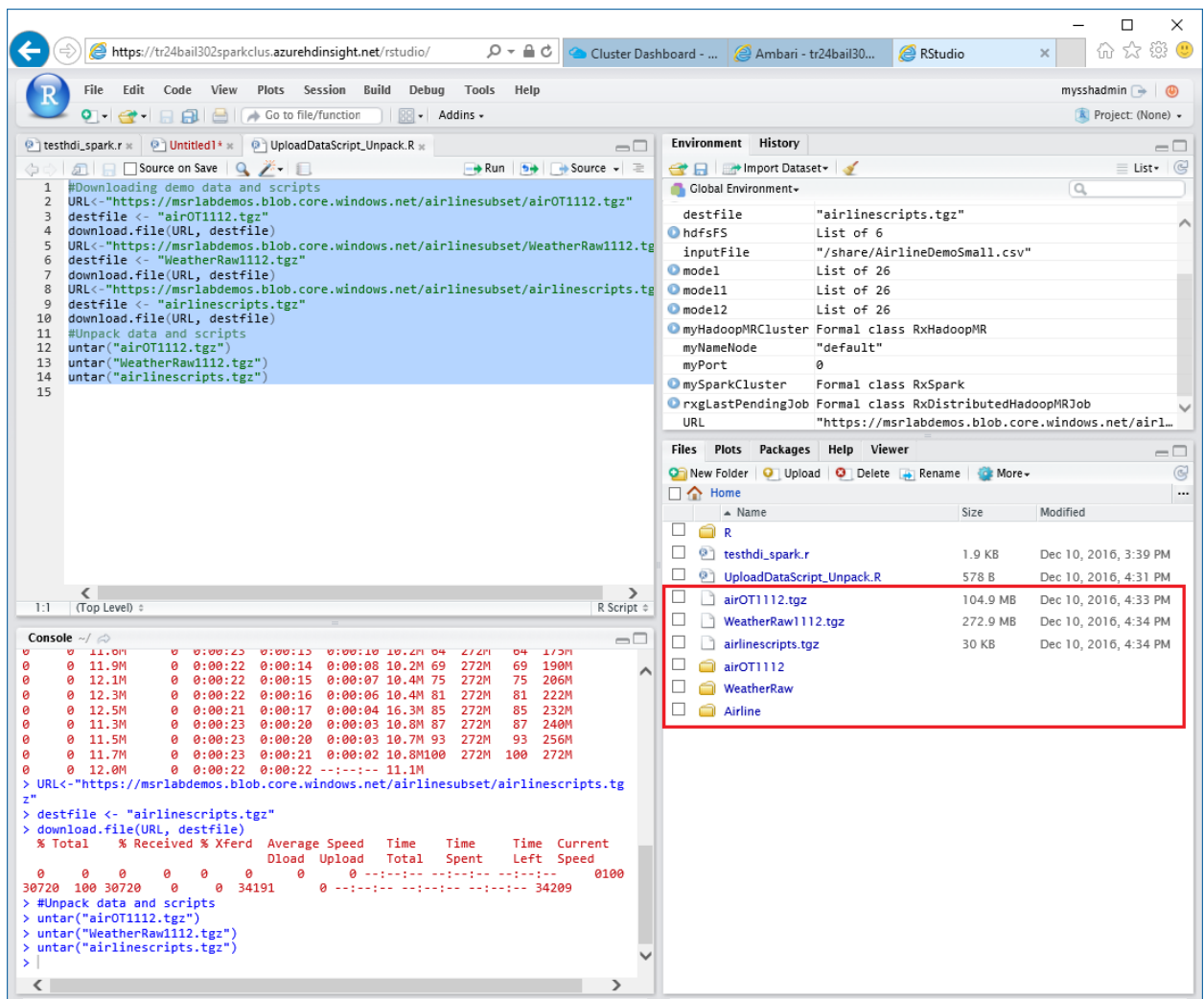
Loading the files

First the data and lab scripts should be uploaded.

1. Upload the UploadDataScript_Unpack.R script from C:\TR24BAIL302Labs folder.

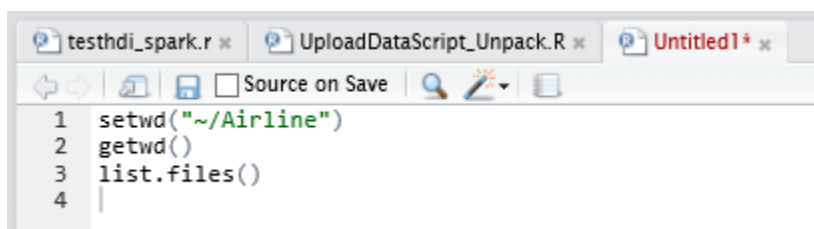


2. Open the file, then press ctrl+a and click on Run.



If you are unable to successfully run the above script use **C:\TR24BAIL302Labs\R\UploadDataScript_Unpack_from_alternative.R** to download the files from an alternative location.

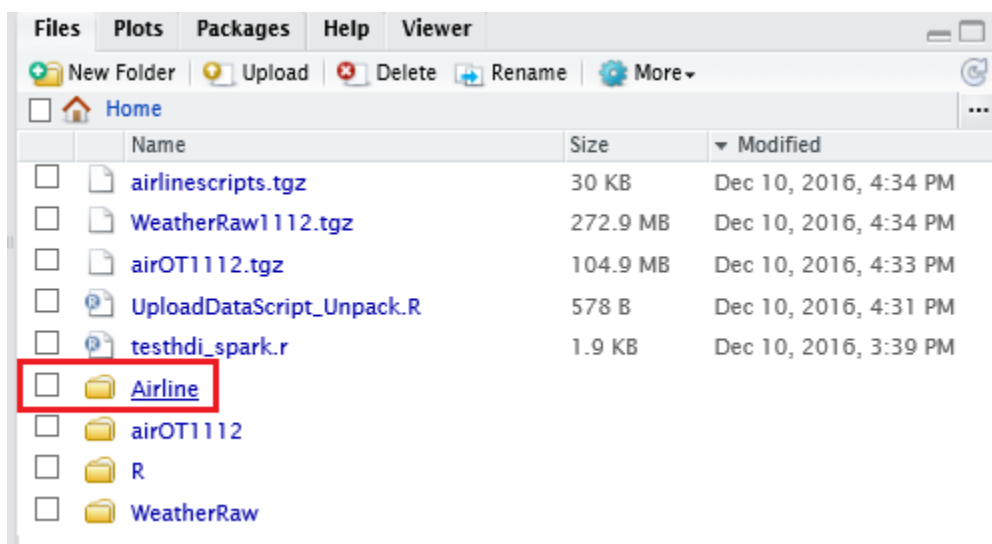
- Open a new R script and type below commands to set your working directory to the Airline directory and cross check.



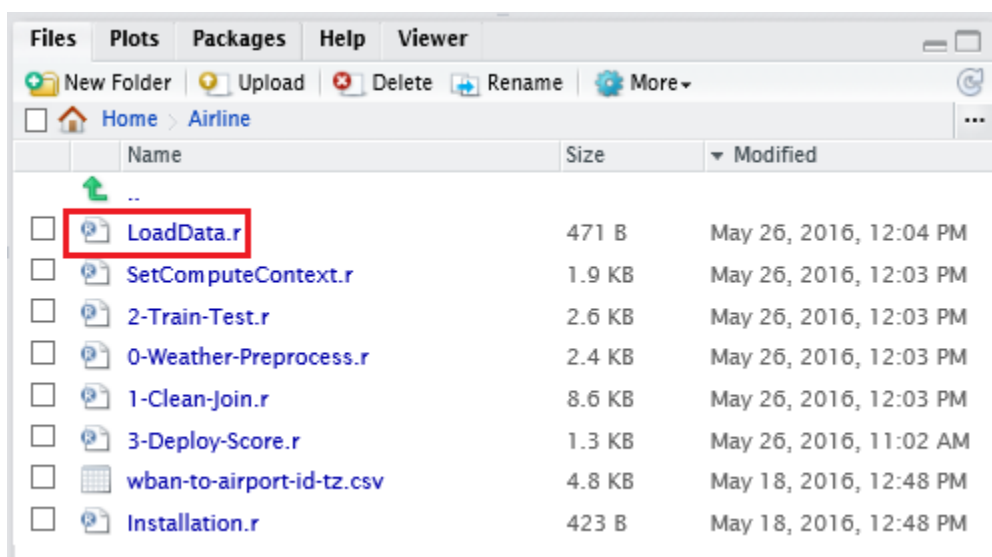
In the Airline directory, you will see an R script called LoadData.r.

```
4:1 (Top Level) R Script
Console ~/Airline/
> setwd("~/Airline")
> getwd()
[1] "/home/mysshadmin/Airline"
> list.files()
[1] "0-Weather-Preprocess.r" "1-Clean-Join.r"
[3] "2-Train-Test.r"        "3-Deploy-Score.r"
[5] "Installation.r"        "LoadData.r"
[7] "SetComputeContext.r"   "wban-to-airport-id-tz.csv"
> |
```

- Click on the Airline folder from the file pane.



- Open LoadData.r script file.



The script copies data to HDFS to be accessed by the worker nodes.

```
testhdi_spark.r UploadDataScript_Unpack.R Untitled1* LoadData.r
Source on Save Run Source
1 # #####
2 # Updated 2016-05-26 Ryan Simpson
3 # #####
4
5 rxHadoopMakeDir("/user/RevoShare/WeatherRaw")
6 rxHadoopCopyFromLocal("../WeatherRaw/*", "/user/RevoShare/WeatherRaw/")
7 rxHadoopListFiles("/user/RevoShare/WeatherRaw")
8
9 rxHadoopMakeDir("/user/RevoShare/AirOnTime08to12CSV")
10 rxHadoopCopyFromLocal("../airOT1112/*", "/user/RevoShare/AirOnTime08to12CSV/")
11 rxHadoopListFiles("/user/RevoShare/AirOnTime08to12CSV")
12 |
```

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

The screenshot displays the RStudio IDE interface. The top toolbar includes buttons for File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. The main editor window shows the script 'testhdi_spark.r' with the following content:

```
1 # #####
2 # Updated 2016-05-26 Ryan Simpson
3 # #####
4
5 rxHadoopMakeDir("/user/RevoShare/WeatherRaw")
6 rxHadoopCopyFromLocal("../WeatherRaw/*", "/user/RevoShare/WeatherRaw/")
7 rxHadoopListFiles("/user/RevoShare/WeatherRaw")
8
9 rxHadoopMakeDir("/user/RevoShare/AirOnTime08to12CSV")
10 rxHadoopCopyFromLocal("../airOT1112/*", "/user/RevoShare/AirOnTime08to12CSV/")
11 rxHadoopListFiles("/user/RevoShare/AirOnTime08to12CSV")
12
```

The Console window at the bottom shows the output of the script execution:

```
> rxHadoopListFiles("/user/RevoShare/WeatherRaw")
[1] "Found 6 item
s"
[2] "-rw-r--r-- 1 mysshadmin supergroup 518564650 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1110.csv"
[3] "-rw-r--r-- 1 mysshadmin supergroup 502259511 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1111.csv"
[4] "-rw-r--r-- 1 mysshadmin supergroup 533886007 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1112.csv"
[5] "-rw-r--r-- 1 mysshadmin supergroup 528632733 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1113.csv"
[6] "-rw-r--r-- 1 mysshadmin supergroup 503555982 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1114.csv"
[7] "-rw-r--r-- 1 mysshadmin supergroup 551540131 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1115.csv"
[1] TRUE
> rxHadoopMakeDir("/user/RevoShare/AirOnTime08to12CSV")
[1] TRUE
> rxHadoopCopyFromLocal("../airOT1112/*", "/user/RevoShare/AirOnTime08to12CSV/")
[1] TRUE
> rxHadoopListFiles("/user/RevoShare/AirOnTime08to12CSV")
[1] "Found 6 item
s"
[2] "-rw-r--r-- 1 mysshadmin supergroup 113083814 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1110.csv"
[3] "-rw-r--r-- 1 mysshadmin supergroup 107249903 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1111.csv"
[4] "-rw-r--r-- 1 mysshadmin supergroup 110908206 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1112.csv"
[5] "-rw-r--r-- 1 mysshadmin supergroup 115129701 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1113.csv"
[6] "-rw-r--r-- 1 mysshadmin supergroup 109298312 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1114.csv"
[7] "-rw-r--r-- 1 mysshadmin supergroup 111452061 2016-12-10 16:00 /user/RevoShare/WeatherRaw/airOT1115.csv"
[1] TRUE
>
```

The Environment pane on the right shows the following variables:

Variable	Value
destfile	"airlinescripts.tgz"
hdFsFS	List of 6
inputFile	"/share/AirlineDemoSmall.csv"
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
URL	"https://msrlabdemos.blob.core.windows.net/air..."

The Files pane at the bottom shows a list of files in the 'Airline' directory:

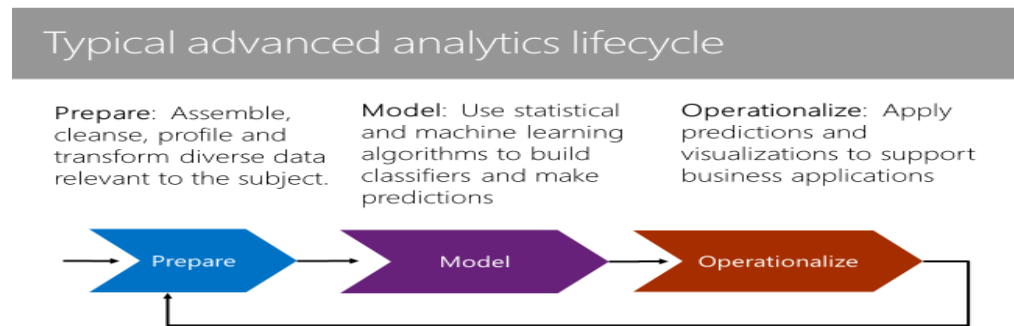
Name	Size	Modified
LoadData.r	471 B	May 26, 2016, 12:04 PM
SetComputeContext.r	1.9 KB	May 26, 2016, 12:03 PM
2-Train-Test.r	2.6 KB	May 26, 2016, 12:03 PM
0-Weather-Preprocess.r	2.4 KB	May 26, 2016, 12:03 PM
1-Clean-Join.r	8.6 KB	May 26, 2016, 12:03 PM
3-Deploy-Score.r	1.3 KB	May 26, 2016, 11:02 AM
wban-to-airport-id-tz.csv	4.8 KB	May 18, 2016, 12:48 PM
Installation.r	423 B	May 18, 2016, 12:48 PM

Summary: In this exercise, you have loaded the sample data and scripts into HDFS, set the working directory correctly and executed a R script to load the data and scripts into HDFS to be used next. Please proceed with next exercise.

Exercise 4: Create Predictive Models

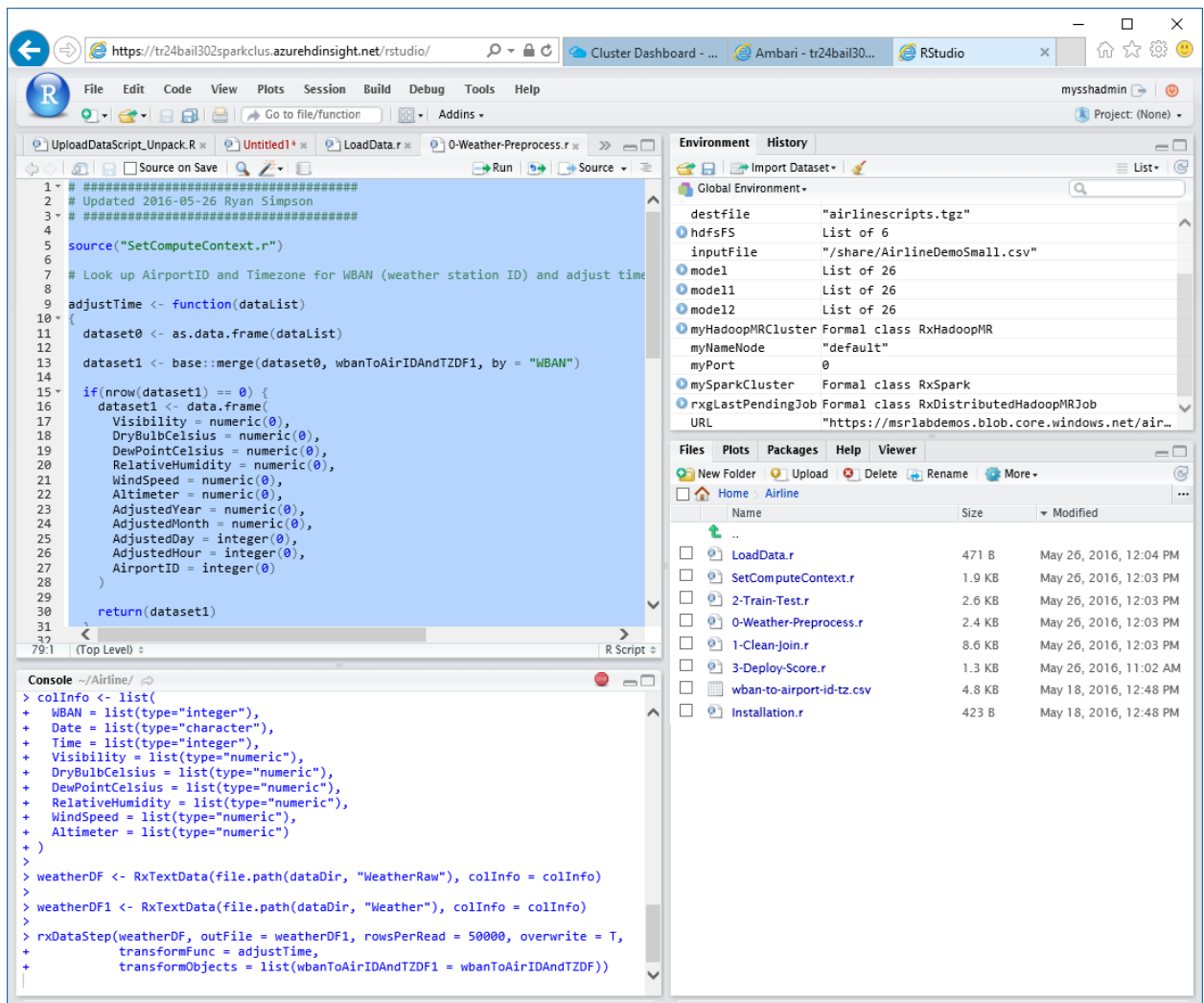
In this exercise, you will perform pre-processing, data cleaning and transformation and create a binary XDF file. You will then train, test and save a Logistic Regression Model and a Decision Tree Model. The Scripts can be found in the ~/Airline Directory:

Process You will use the typical analytical life cycle as shown below.

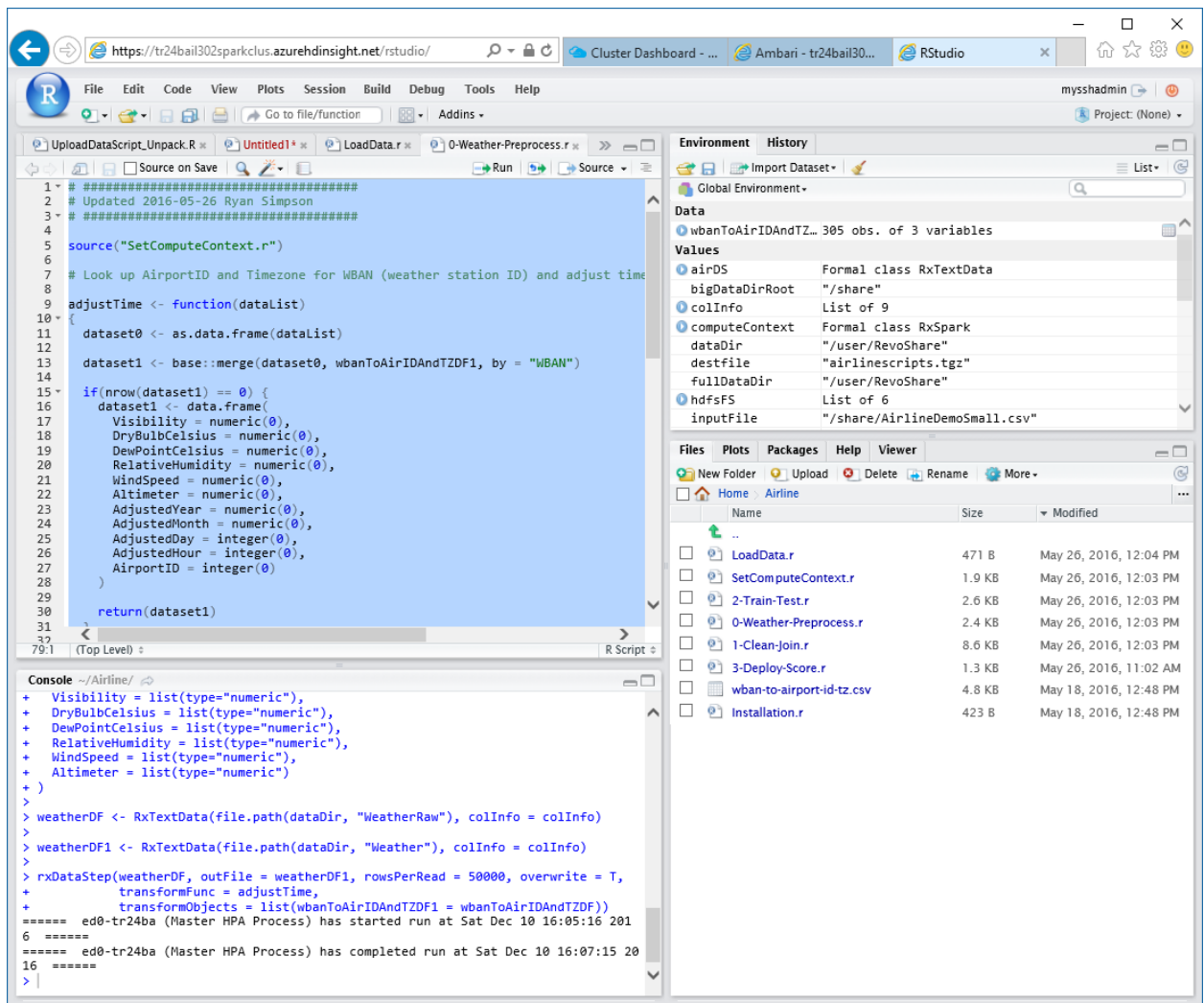


For the data preparation (i.e. joining and aggregating), data manipulation functions contained in the SparkR package will be used. For the modelling, Microsoft R Server's ScaleR package to train models using the Spark Engine will be used.

1. Open the script **0-Weather-Preprocess.r** and run. This script uses rxDataStep (in a spark Compute context) to adjust time in the airline dataset to be the same as the weather dataset.



The output should resemble the below.



2. Open the script **1-Clean-Join.r** and run. In this script, you will use SparkR to aggregate and join datasets.

The screenshot shows the RStudio interface with the following components:

- Top Panel:** Browser tabs for the RStudio URL, Cluster Dashboard, Ambari, and RStudio. The RStudio menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help.
- Source Panel:** Displays the R script being executed. The script includes comments, library loading, Spark environment setup, and data loading commands.
- Console Panel:** Shows the execution output, including dependency resolution and Spark context setup.
- Environment Panel:** Lists the loaded data objects and their types.
- Files Panel:** Shows the file structure of the project, including the R script files.

```

1 # #####
2 # Updated 2016-05-26 Ryan Simpson
3 # #####
4
5 source("SetComputeContext.r")
6
7 .libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
8
9 library(SparkR)
10
11 sparkEnvir <- list(spark.executor.instances = '10',
12                   spark.yarn.executor.memoryOverhead = '8000')
13
14 sc <- sparkR.init(
15   sparkEnvir = sparkEnvir,
16   sparkPackages = "com.databricks:spark-csv_2.10:1.3.0"
17 )
18
19 sqlContext <- sparkRSQL.init(sc)
20
21 airPath <- file.path(fullDataDir, "AirOnTime08to12CSV")
22 weatherPath <- file.path(fullDataDir, "Weather") # pre-processed weather data
23
24 # create a SparkR DataFrame for the airline data
25
26 airDF <- read.df(sqlContext, airPath, source = "com.databricks.spark.csv",
27                 header = "true", inferSchema = "true")
28
29 # Create a SparkR DataFrame for the weather data
30
31 weatherDF <- read.df(sqlContext, weatherPath, source = "com.databricks.spark.csv",
32                     header = "true", inferSchema = "true")
33
34 # Save the DataFrames to XDF format
35 airDF$writeXDF("airline.xdf")
36 weatherDF$writeXDF("weather.xdf")
37
38 # Clean up
39 sc$stop()
40 
```

Console Output:

```

downloading https://repo1.maven.org/maven2/com/univocity/univocity-parsers/1.5.1/univocit
y-parsers-1.5.1.jar ...
[SUCCESSFUL ] com.univocity#univocity-parsers;1.5.1#univocity-parsers.jar (12ms)
:: resolution report :: resolve 1127ms :: artifacts dl 40ms
:: modules in use:
com.databricks#spark-csv_2.10;1.3.0 from central in [default]
com.univocity#univocity-parsers;1.5.1 from central in [default]
org.apache.commons#commons-csv;1.1 from central in [default]
-----
| conf | number | modules | search | dwnlded | evicted | artifacts |
|-----|-----|-----|-----|-----|-----|-----|
| default | 3 | 3 | 3 | 0 | 3 | 3 |
-----
:: retrieving :: org.apache.spark#spark-submit-parent
confs: [default]
3 artifacts copied, 0 already retrieved (317kB/8ms)
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).

```

Environment Panel:

Variable	Type	Value
airDS	Formal class RxTextData	
bigDataDirRoot	String	"/share"
collInfo	List of 9	
computeContext	Formal class RxSpark	
dataDir	String	"/user/RevoShare"
destfile	String	"airlinescripts.tgz"
fullDataDir	String	"/user/RevoShare"
hdfsFS	List of 6	
inputFile	String	"/share/AirlineDemoSmall.csv"

Files Panel:

Name	Size	Modified
LoadData.r	471 B	May 26, 2016, 12:04 PM
SetComputeContext.r	1.9 KB	May 26, 2016, 12:03 PM
2-Train-Test.r	2.6 KB	May 26, 2016, 12:03 PM
0-Weather-Preprocess.r	2.4 KB	May 26, 2016, 12:03 PM
1-Clean-Join.r	8.6 KB	May 26, 2016, 12:03 PM
3-Deploy-Score.r	1.3 KB	May 26, 2016, 11:02 AM
wban-to-airport-id-tz.csv	4.8 KB	May 18, 2016, 12:48 PM
Installation.r	423 B	May 18, 2016, 12:48 PM

After this, you will export it to XDF format to be used in the next step.

The screenshot displays the RStudio web interface. The main script editor contains the following R code:

```

1 # #####
2 # Updated 2016-05-26 Ryan Simpson
3 # #####
4
5 source("SetComputeContext.r")
6
7 .libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
8
9 library(SparkR)
10
11 sparkEnvir <- list(spark.executor.instances = '10',
12                   spark.yarn.executor.memoryOverhead = '8000')
13
14 sc <- sparkR.init(
15   sparkEnvir = sparkEnvir,
16   sparkPackages = "com.databricks:spark-csv_2.10:1.3.0"
17 )
18
19 sqlContext <- sparkRSQL.init(sc)
20
21 airPath <- file.path(fullDataDir, "AirOnTime08to12CSV")
22 weatherPath <- file.path(fullDataDir, "Weather") # pre-processed weather data
23
24 # create a SparkR DataFrame for the airline data
25
26 airDF <- read.df(sqlContext, airPath, source = "com.databricks.spark.csv",
27                 header = "true", inferSchema = "true")
28
29 # Create a SparkR DataFrame for the weather data
30
31 weatherDF <- read.df(sqlContext, weatherPath, source = "com.databricks.spark.csv",
32                     header = "true", inferSchema = "true")
33
34 # (Top Level)

```

The Environment pane on the right shows the following global variables:

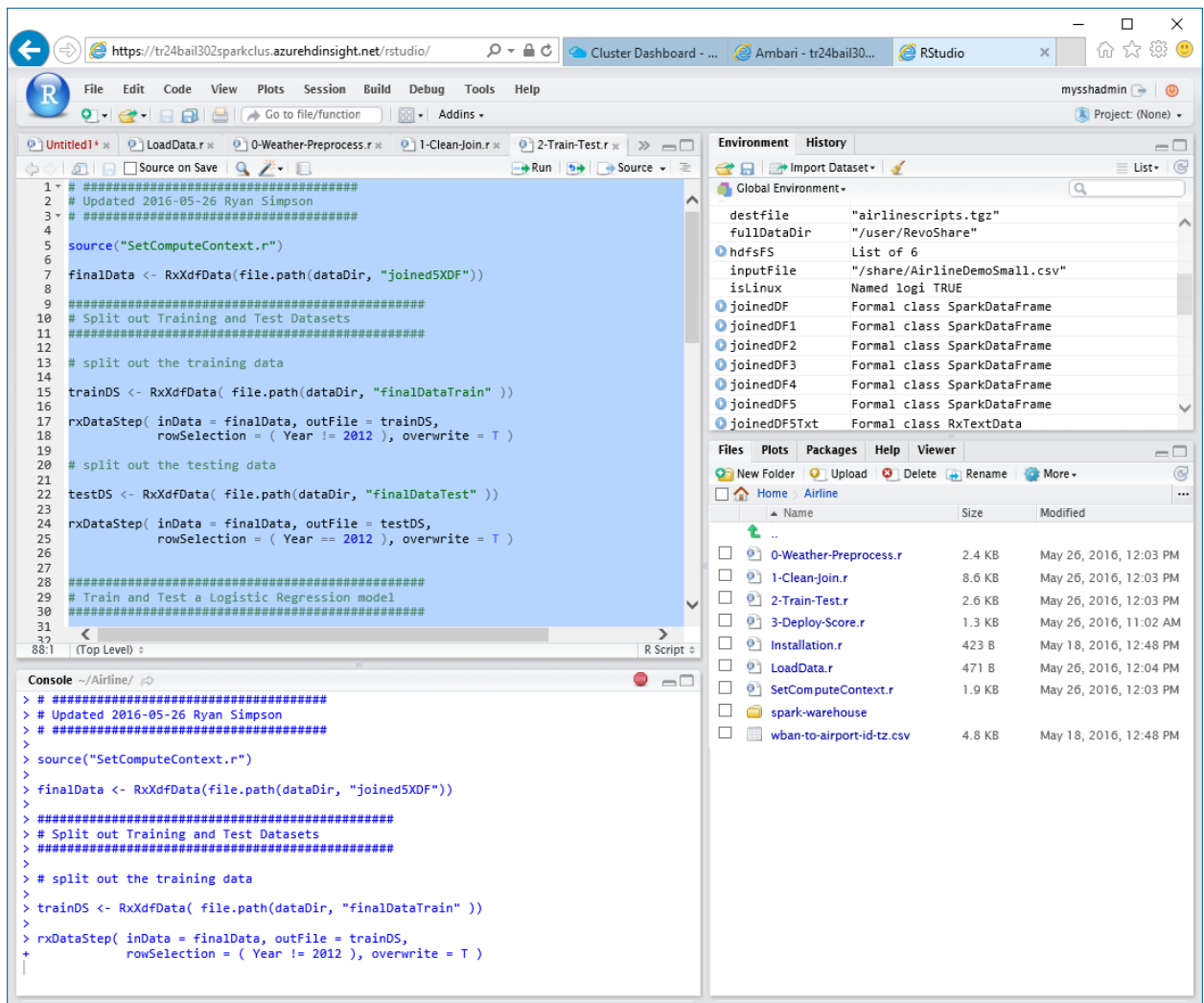
- destfile: "airlinescripts.tgz"
- fullDataDir: "/user/RevoShare"
- hdfsFS: List of 6
- inputFile: "/share/AirlineDemoSmall.csv"
- isLinux: Named logi TRUE
- joinedDF: Formal class SparkDataFrame
- joinedDF1: Formal class SparkDataFrame
- joinedDF2: Formal class SparkDataFrame
- joinedDF3: Formal class SparkDataFrame
- joinedDF4: Formal class SparkDataFrame
- joinedDF5: Formal class SparkDataFrame
- joinedDF5Txt: Formal class RxTextData

The Files pane shows a directory structure with the following files:

- LoadData.r (471 B, May 26, 2016, 12:04 PM)
- SetComputeContext.r (1.9 KB, May 26, 2016, 12:03 PM)
- 2-Train-Test.r (2.6 KB, May 26, 2016, 12:03 PM)
- 0-Weather-Preprocess.r (2.4 KB, May 26, 2016, 12:03 PM)
- 1-Clean-Join.r (8.6 KB, May 26, 2016, 12:03 PM)
- 3-Deploy-Score.r (1.3 KB, May 26, 2016, 11:02 AM)
- wban-to-airport-id-tz.csv (4.8 KB, May 18, 2016, 12:48 PM)
- Installation.r (423 B, May 18, 2016, 12:48 PM)
- spark-warehouse

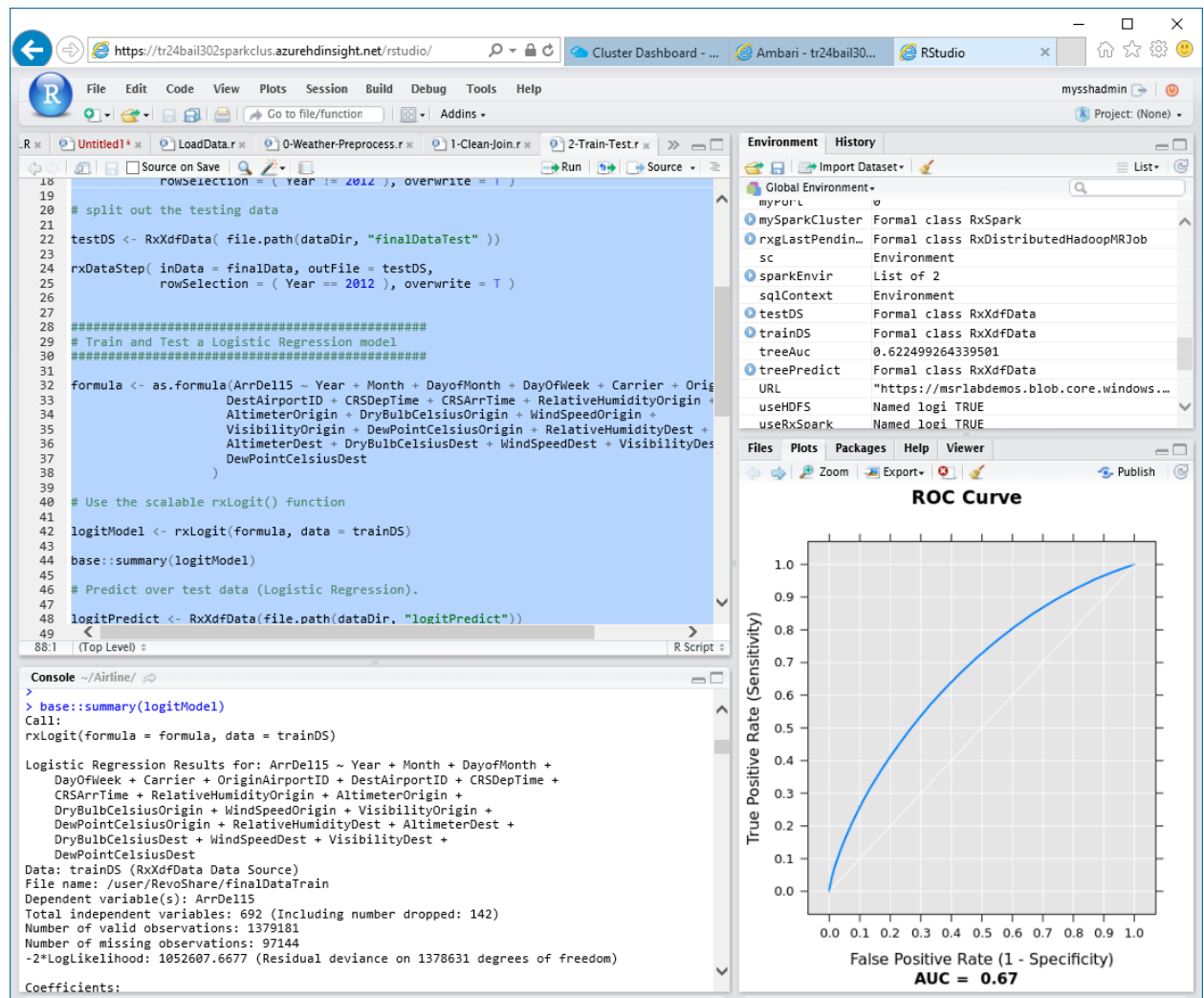
The Console pane at the bottom shows the output of the script, including factor levels and variable types for the loaded data.

- Open the script **2-Train-Test.r** and run. In this script, you split the XDF data into training and test sets. You build logistic regression and decision tree models (in Spark).

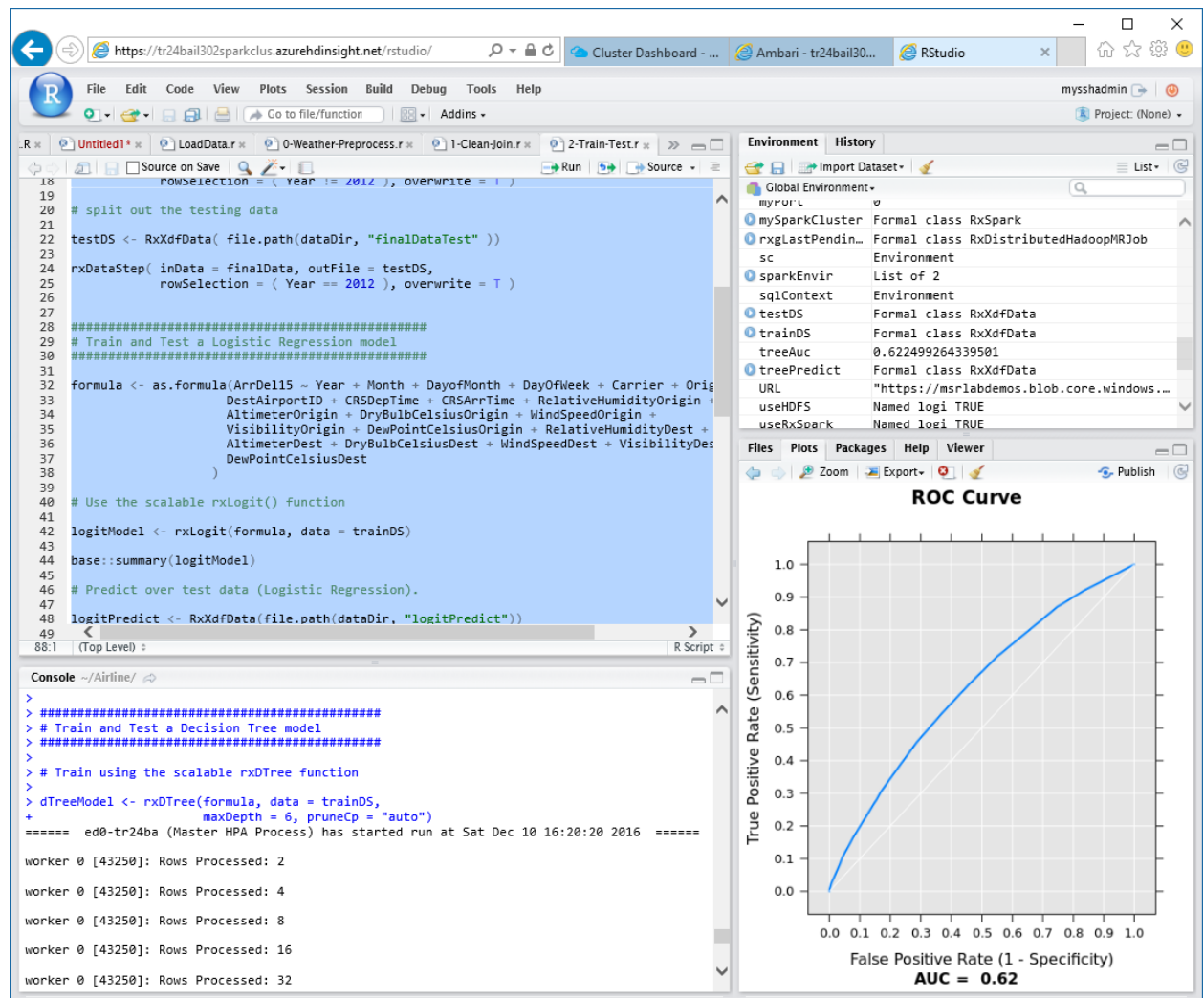


You will see the following plots when the ROCs (Receiver Operating Characteristic) are plotted. ROC is useful for visualizing a model performance. This will help us in selecting the classifiers based on their performance

ROC Curve for logistic regression:



ROC Curve for Decision Tree:



To compare classifiers, a common method is to calculate the area under the ROC curve also known as Area Under the Curve(AUC). You will notice that the decision tree has a greater area in AUC and therefore has better average performance - 0.67 Vs 0.62

Note: This script will save the Decision Tree model as **dTreeModel.RData**.

Summary: In this exercise, you have performed pre-processing, data cleaning and transformation tasks and created a binary XDF file. You also have trained, tested and saved a Decision Tree Model. You have also compared the performance between both the models and have could conclude which one performs better.

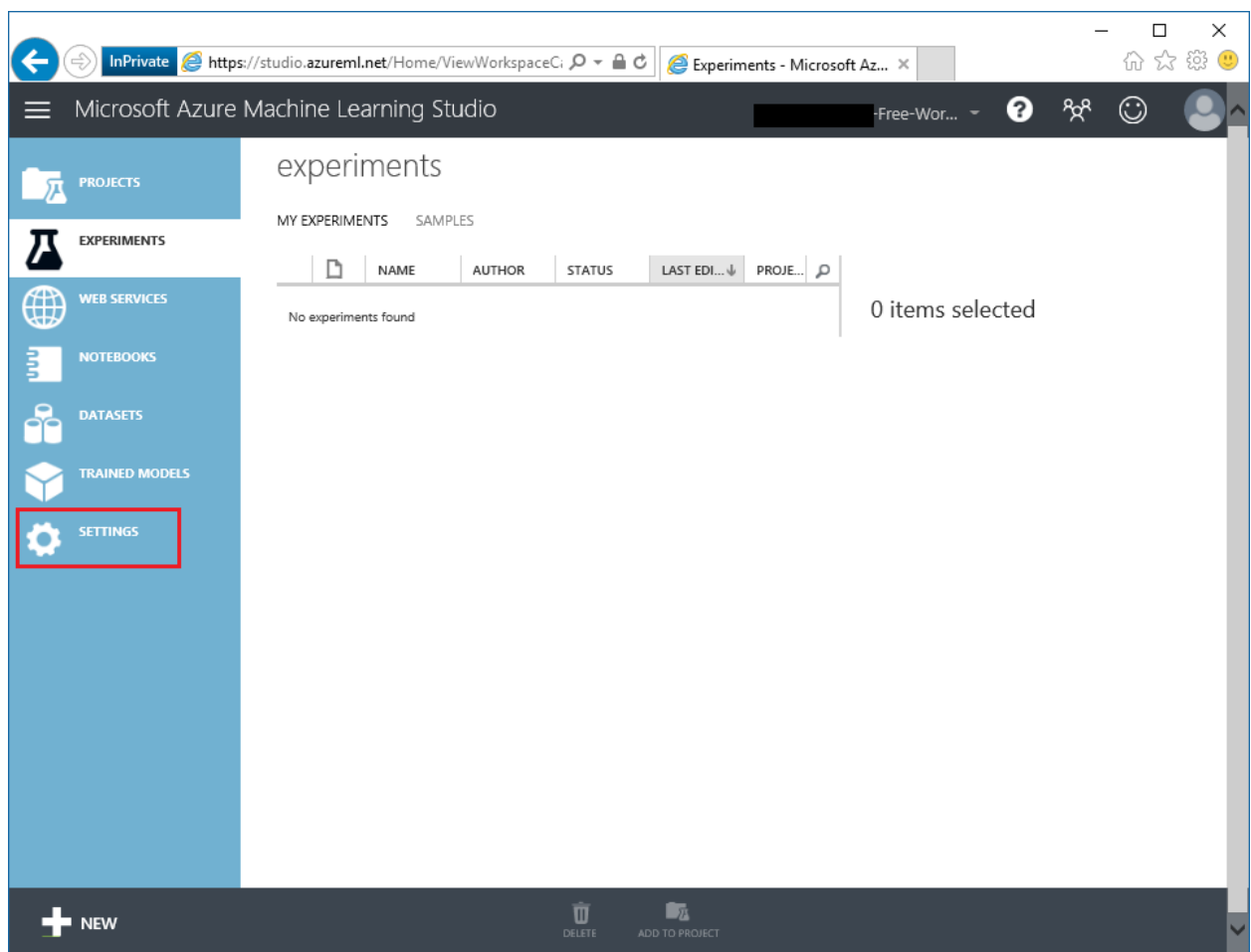
Please proceed with next exercise.

Exercise 5: Publish to Azure Machine Learning as a Web service

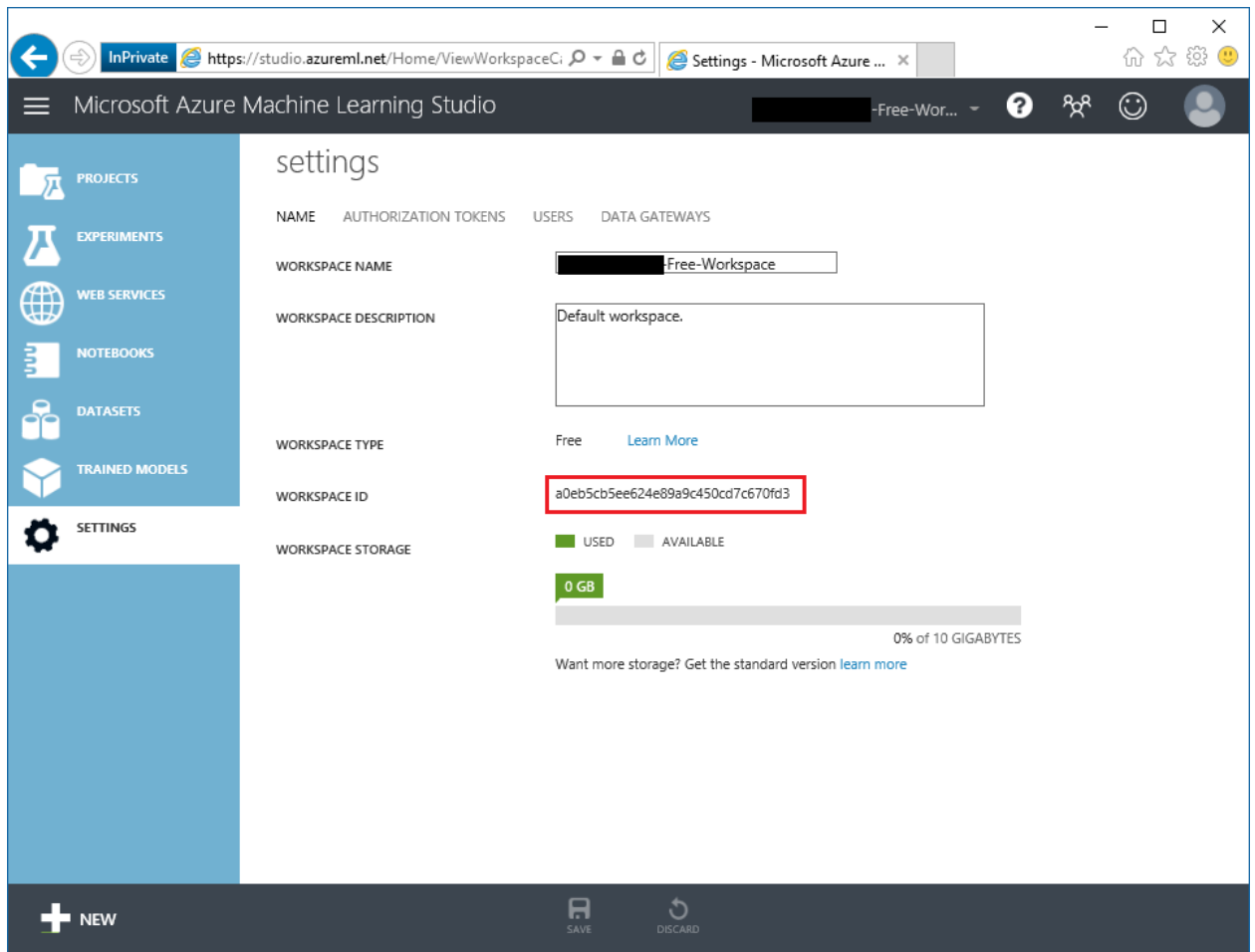
In this exercise, you will use the AzureML CRAN package to deploy the tree-based model as a scalable web service to Azure ML.

As a pre-requisite, you need to have access to an Azure ML workspace

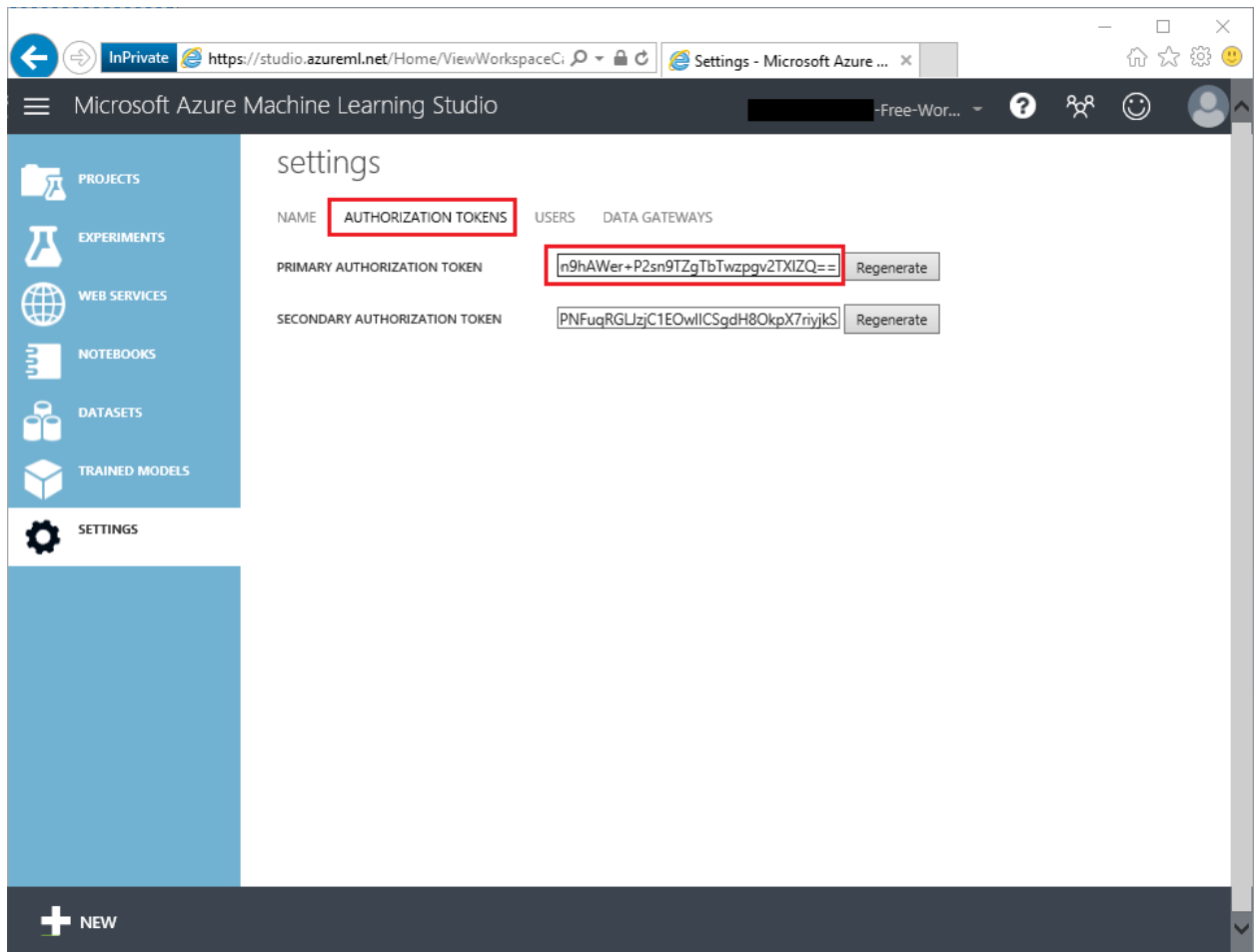
1. Go to <https://studio.azureml.net/> link and sign-up for a free Azure ML workspace or use an existing Azure ML workspace if you have one. After sign-in, click on settings tab.



2. Copy the workspace ID to a Notepad file. You will use this workspace ID in the R Script later. Do not close this Notepad file yet.



3. Go to AUTHORIZATION TOKENS tab and copy the PRIMARY AUTHORIZATION TOKEN as below.



4. Copy the PRIMARY AUTHORIZATION TOKEN to the same Notepad file. You will use this workspace ID in the R Script later. Do not close this Notepad file.
5. Open the script 3-Deploy-Score.r in the editor. You need to change the workspace variable.

```

31 - #####
32 - # Publish the scoring function as a web service
33 - #####
34 -
35 - library(AzureML)
36 -
37 - workspace <- workspace(config = "azureml-settings.json")
38 -
39 - endpoint <- publishWebService(workspace, scoringFn,
40 -                               name="Delay Prediction Service",
41 -                               inputSchema = exampleDF)
42 -

```

Instead of pointing to azureml-settings.json as shown above, change the script to replace with Workspace ID and Authentication token you have copied earlier as shown below.


```

31 #####
32 # Publish the scoring function as a web service
33 #####
34
35 library(AzureML)
36
37 workspace <- workspace(
38   id = "a0eb5cb5ee624e89a9c450cd7c670fd3",
39   auth = "jv6kzkUuhkn7iRmsBlujd9ZT7i6Ey1LjdpUQv7nOPSkdfkWAZWSAvoq6n9hAWer+P2sn9TZgTbTwzpgv2TXIZQ=="
40 )
41
42 endpoint <- publishWebService(workspace, scoringFn,
43                               name="Delay Prediction Service",
44                               inputSchema = exampleDF)
45

```

6. Select all the script by Ctrl + a, then run it.

The screenshot shows the RStudio interface with the following components:

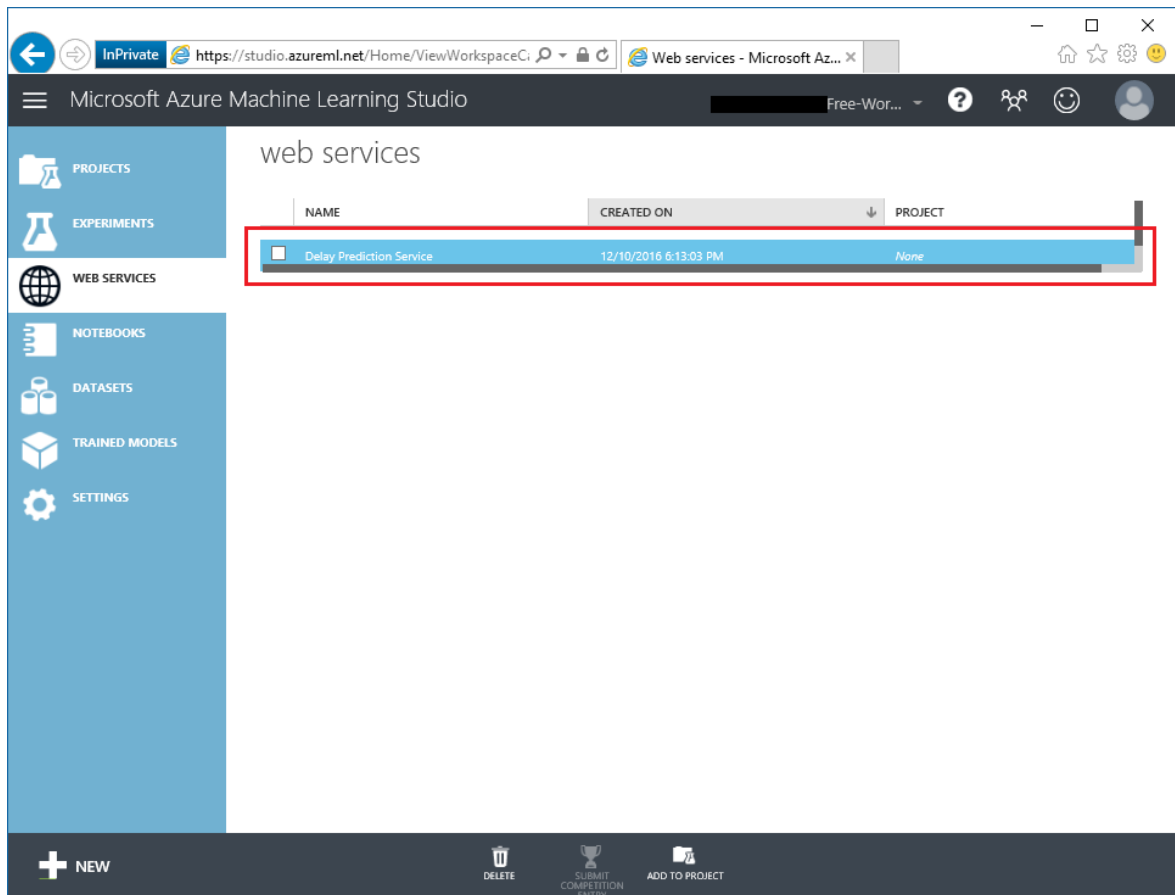
- Editor:** Contains the R script for publishing the scoring function and testing it. The script defines a workspace, publishes a web service named "Delay Prediction Service", and then attempts to score new data.
- Environment:** Shows the global environment with variables like `testDS`, `trainDS`, `treeAuc`, `treePredi...`, `useHDFS`, `useRSpark`, `vars`, `varsToDrop`, `varsToKeep`, `weatherDF`, `weatherDF1`, and `weatherPa...`.
- Files:** Shows the file explorer with files like `0-Weather-Preprocess.r`, `1-Clean-Join.r`, `2-Train-Test.r`, `3-Deploy-Score.r`, `dLogitModel.RData`, `dTreeModel.RData`, `Installation.r`, `LoadData.r`, `SetComputeContext.r`, `spark-warehouse`, and `wban-to-airport-id-tz.csv`.
- Console:** Shows the output of the scoring function, which is a vector of probabilities for the flight being on time. The output is:


```

> head(scores)
ans
1 0.21297532
2 0.24896333
3 0.20017774
4 0.10099716
5 0.10099716
6 0.06071949
      
```

In the output, you will see the probability of the flight being on time.

- Go back to Azure ML Studio, click on Web services tab. You will notice that Delay Prediction Service has been published successfully.



Summary: In this exercise, you used the AzureML package to deploy the tree-based model as a scalable web service to Azure ML and called it to predict flight delays

Please proceed with next exercise.

Exercise 6: (If time permits) Adjust the Yarn Memory limits

In this exercise, you will adjust the Yarn memory to 9024MB using Ambari to ensure acceptable performance for bigger datasets.

From the Cluster Dashboard in the Azure Portal, you will open Ambari and adjust the memory settings in Ambari. To launch Ambari, click on R Server Dashboards and then click on HDInsight Cluster Dashboard

The screenshot shows the Azure Portal interface for an HDInsight cluster. The breadcrumb navigation at the top indicates the path: **tr24bail302sparkclus** > Cluster Dashboard. In the 'Quick Links' section, the 'R Server Dashboards' link is highlighted with a red box. In the main content area, the 'HDInsight Cluster Dashboard' tile is also highlighted with a red box. The dashboard provides a comprehensive overview of the cluster, including its status, configuration, and node details.

Cluster Details:

- Resource group: tr24bail302sparkg
- Status: Running
- Location: East US
- Subscription name: Microsoft Azure Internal Consumption
- Subscription ID: [REDACTED]
- Cluster type, HDInsight version: R Server on Linux (HDI 3.5.1000.0)
- URL: <https://tr24bail302sparkclus.azurehdinsight...>
- Head Nodes, Worker Nodes: D12 v2 (x2), D4 v2 (x4)

Quick Links:

- R Server Dashboards (highlighted)
- Documentation
- Scale Cluster

Usage:

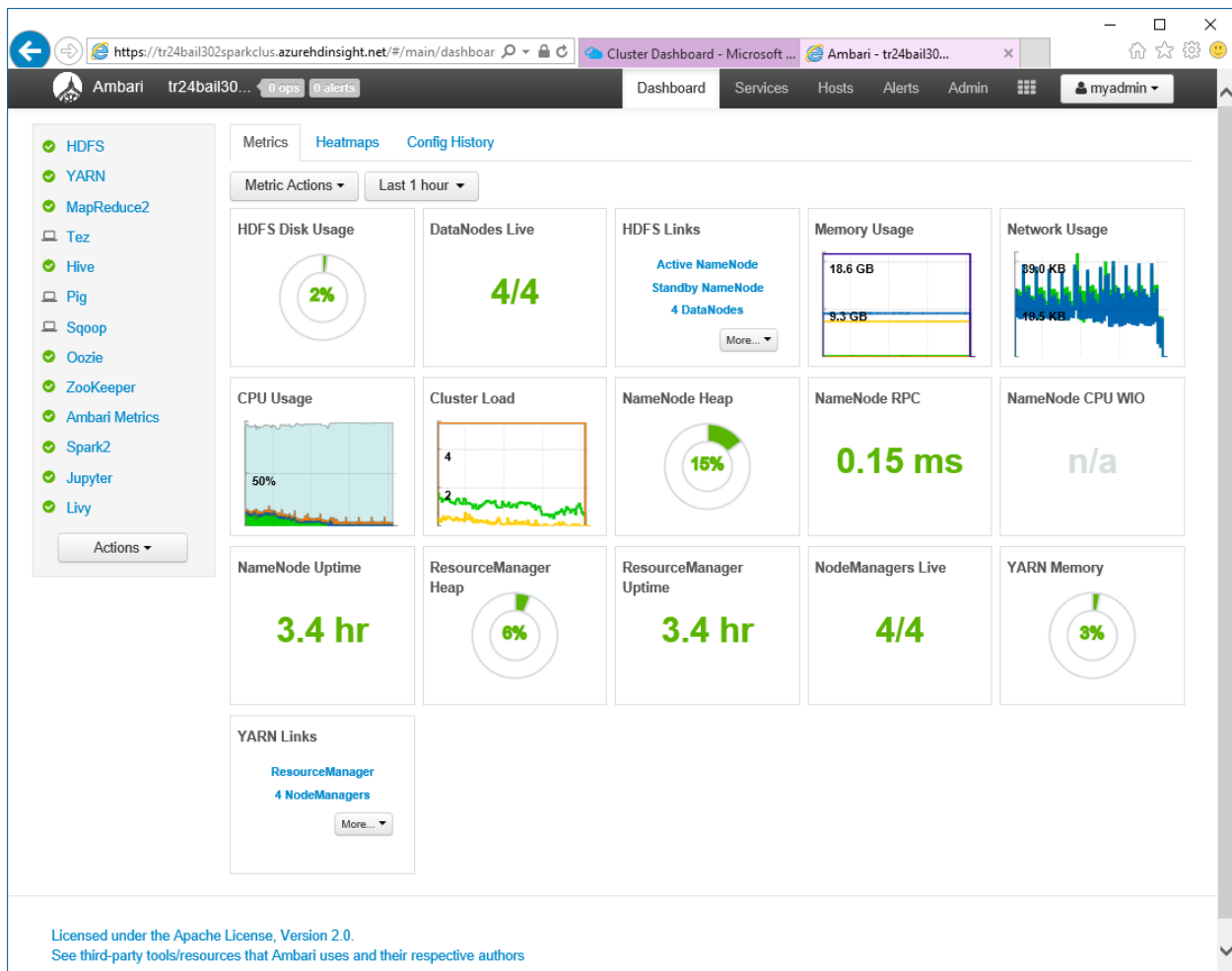
Cluster nodes: 10 nodes

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

Cores in East US for subscription:

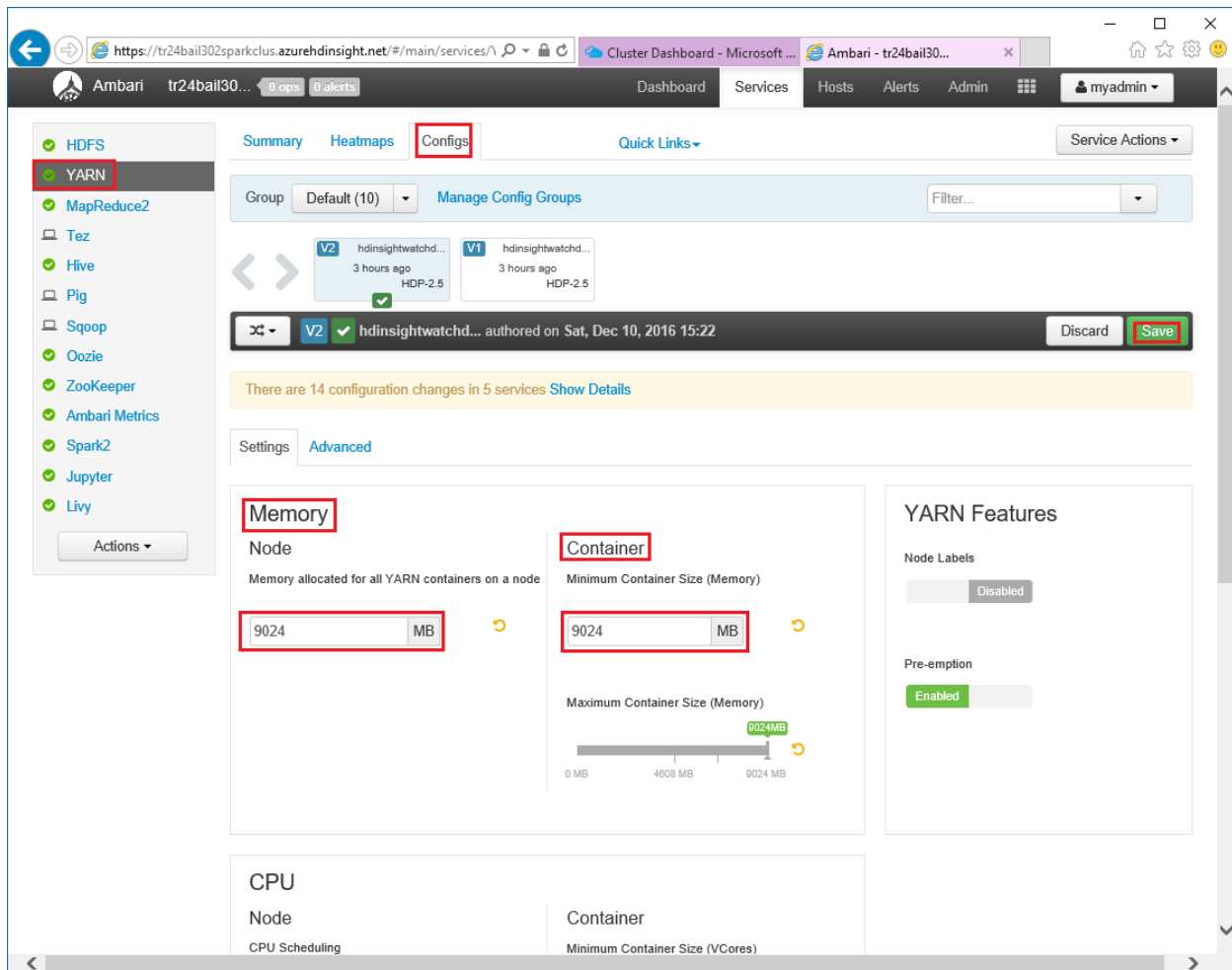
THIS CLUSTER: 48
OTHER CLUSTERS: 0
AVAILABLE: 0

Login into Ambari as the cluster admin as myadmin and password as Password12345abc+

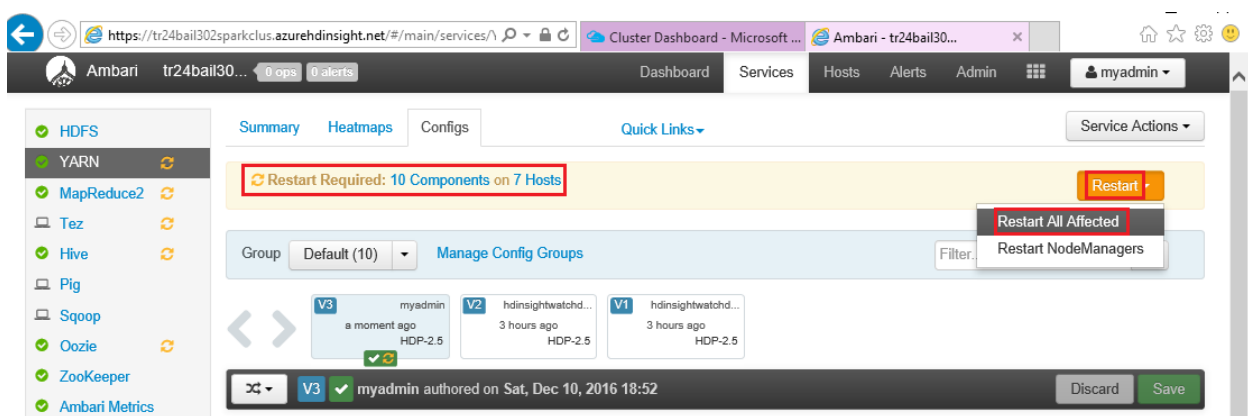


Under 'Yarn' (in left hand pane), click '**Configs**' tab, review the values for Memory and Maximum Container Size. If these settings are already greater than 9024MB, then you do not need to perform the following step and you can move on to the Exercise 3.

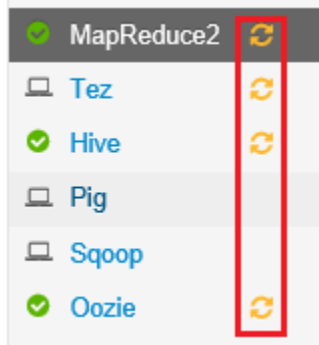
If your settings are less than 9024MB, then click on edit and adjust to 9024



Click Save. Click ok for all the other prompts. Click Restart and select Restart All Affected, then Confirm Restart All.



For all the other components that requires restart, navigate into the required component and perform the restart action as above by clicking Restart and select Restart All Affected.



Summary: In this exercise, you have logged into Ambari and changed the Yarn memory settings to ensure optimal performance