

Microsoft Cloud Workshop

Vending machines hackathon

Leader guide

February 2017

© 2017 Microsoft Corporation. All rights reserved. This document is confidential and proprietary to Microsoft. Internal use only. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

Contents

Vending machines hackathon leader guide.....	1
Overview.....	1
Requirements	1
Exercise 0: Before the hackathon.....	1
Task 1: Provision Power BI.....	1
Task 2: Provision Power BI Desktop.....	2
Task 3: Provision an R Server on HDInsight with Spark Cluster.....	2
Task 4: Prepare an SSH client.....	6
Exercise 1: Environment setup	8
Task 1: Download and open the vending machines starter project.....	8
Task 2: IoT Hub.....	9
Task 3: Provision Azure Machine Learning Workspace	13
Task 4: Storage.....	14
Task 5: Cognitive Services Face API	17
Task 6: SQL Database	19
Exercise 2: Create Dynamic Pricing Model.....	20
Task 1: Create a model locally	20
Task 2: Try a prediction locally.....	23
Task 3: Create the model in R Server on HDInsight.....	24
Task 4: Create predictive service in Azure Machine Learning	26
Exercise 3: Implement dynamic pricing.....	34
Task 1: Implement photo uploads to Azure Storage	34
Task 2: Invoke Face API.....	35
Task 3: Invoke pricing model	36
Task 4: Configure the Simulator.....	36
Task 5: Test dynamic pricing in Simulator	36

Exercise 4: Implement purchasing.....	40
Task 1: Create the transactions table	40
Task 2: Configure the Simulator.....	41
Task 3: Test purchasing	42
Exercise 5: Implement device command and control	43
Task 1: Listen for control messages	43
Task 2: Send control messages	44
Task 3: Configure the DeviceControlConsole and the Simulator	44
Exercise 6: Analytics with Power BI Desktop	47
Task 1: Build the query and create the visualization	47
Exercise 7: Cleanup	53

Vending machines hackathon leader guide

Overview

Trey Research Inc. looks at the old way of doing things in retail and introduces innovative experiences that delight customers and drive sales. Their latest initiative focuses on intelligent vending machines that support commerce, engagement analytics, and intelligent promotions.

Requirements

- Microsoft Azure subscription must be pay-as-you-go or MSDN.
 - Trial subscriptions will *not* work.
- Local machine or a virtual machine configured with:
 - Visual Studio 2015 Community Edition with Update 1 or later
 - Azure SDK 2.9 or later for Visual Studio
 - Azure PowerShell 1.0.0 or later
 - [Microsoft R Client](#)
 - [R Tools for Visual Studio](#) 0.3.2 or later
 - [Power BI Desktop](#) (June 2016 build or later)
- A running R Server on HD Insight Spark cluster (see Exercise 0).

Exercise 0: Before the hackathon

Duration: 60 minutes

Synopsis: You should follow all of the steps provided in Exercise 0 of the Proctor Guide (included in this document) *before* attending the hackathon.

Task 1: Provision Power BI

1. If you do not already have a Power BI account, go to <https://www.powerbi.com>.
2. On the page, enter your work email address (which should be the same account as the one you use for your Azure subscription) and select **Use it free**.



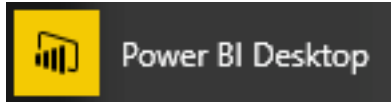
Enter your work email address

Use it free

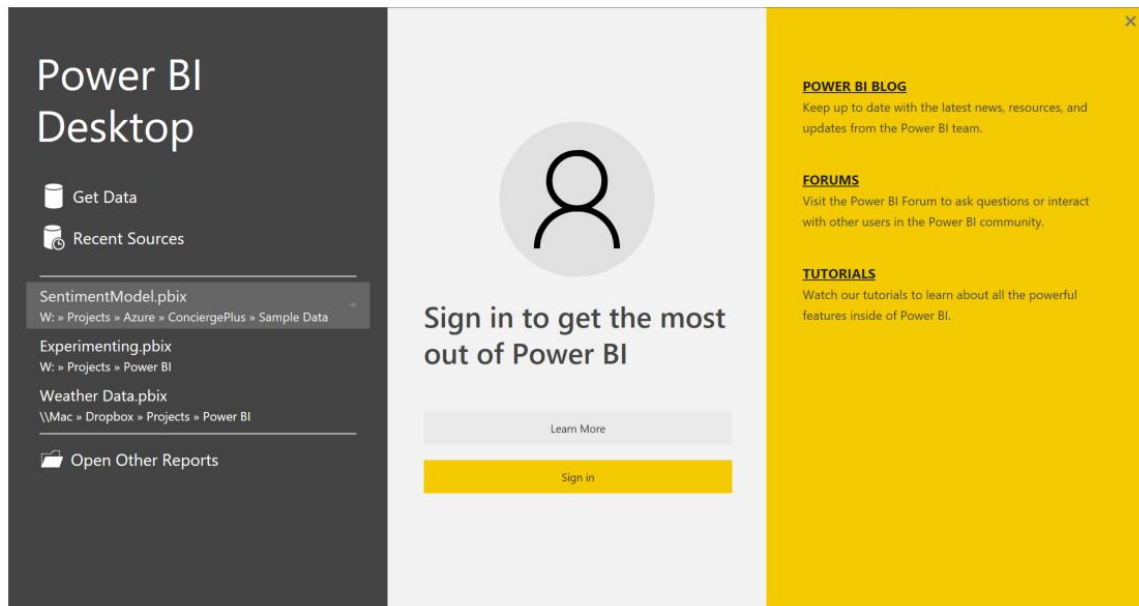
3. Follow the on-screen prompts and your Power BI environment should be ready within minutes. You can always return to it via <https://app.powerbi.com>.

Task 2: Provision Power BI Desktop

1. Download Power BI Desktop from <https://powerbi.microsoft.com/desktop>.
2. Step through the installation wizard.
3. Run Power BI Desktop from your Start Menu.



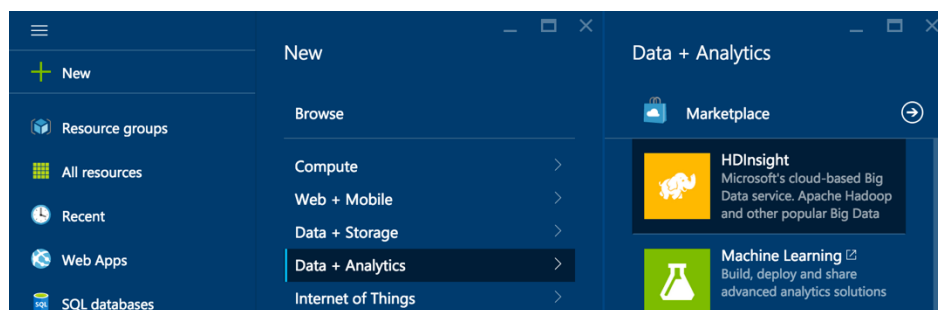
4. At the initial dialog, select Sign In and confirm that you can sign in to Power BI using your work or school credentials.



Task 3: Provision an R Server on HDInsight with Spark Cluster

Using the Azure Portal, provision a new HDInsight cluster.

1. Select **+New**, select **Data + Analytics**, **HDInsight**.



2. Provide a unique Cluster Name.
3. Choose the Azure Subscription into which you want to deploy the cluster.
4. Select **Cluster configuration**.

- Set the **Cluster type** to **R Server** and the **Version** to **R Server 9.0 (HDI 3.5)**. Note that the Operating System option for the Spark cluster is fixed to Linux.

Cluster configuration

* Cluster type ⓘ	* Operating system	* Version
<div>R Server ▼</div>	<div>Linux</div>	<div>R Server 9.0 (HDI 3.5) ▼</div>

* Cluster tier ⓘ

<div>STANDARD</div>	<div>PREMIUM</div>
---------------------	--------------------

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

Configuration Options:

- R Server 9.0 on Spark 2.0 with Java 8
- R Server 8.0 on Spark 1.6 with Java 7

Adds \$0.08 per Core-Hour.

- Select **Select**.
- Select **Credentials**.
- Leave Cluster Login Username as admin, set a Cluster Login Password and Confirmation. Set the SSH username to remoteuser (this is required) and the password of your choice. Leave SSH Authentication Type set to Password.

Cluster Login Username ⓘ

admin

Cluster Login Password

.....



This is used for job submission, for admin cluster access, and to access cluster dashboard, notebook, and application HTTP/web endpoints.

SSH Username ⓘ

remoteuser



SSH Authentication Type

PASSWORD

PUBLIC KEY

* SSH Password

.....



This is used to remotely connect to your cluster's nodes, including application edge nodes.

9. Select **Select**.

10. Select **Data Source**.

11. Leave the Primary storage type set to Azure storage, leave Selection Method at From all subscriptions, then choose or create a new storage account.

* Primary storage type

☒ Azure Storage ☐ Data Lake Store

Selection method ⓘ

From all subscriptions ▼

* Create a new Storage account

solvendinghack ✓

[Select existing](#)

12. Set the Default Container to the name of your cluster, choose a location for the storage account and leave Data Lake Store access as not configured.

* Default container ⓘ

sol-vending-hack ✓

* Location >

West US

Data Lake Store access ⓘ >

Not configured

13. Select **Select**.
14. Select **Cluster Size**.
15. Set the number of Worker Nodes to **2**.
16. Select **Worker node size**, select **D12 v2**, and select **Select**.
17. Select **Head node size**, select **D12 v2**, and select **Select**.
18. Select **R-Server node size**, select **D12 v2**, and select **Select**.
19. Select **Select** on the Pricing blade.
20. Select **Resource Group** and create a new Resource Group called **Hackathon**.

* Create a new resource group

Hackathon ✓

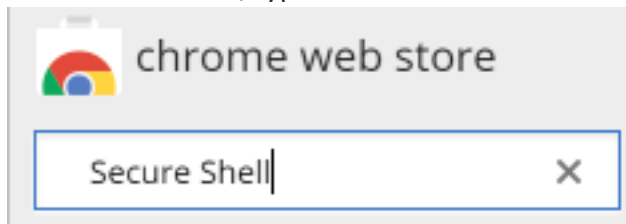
[Select existing](#)

21. On the New HDInsight Cluster blade, select **Create**. Your cluster should be ready within 20–30 minutes.

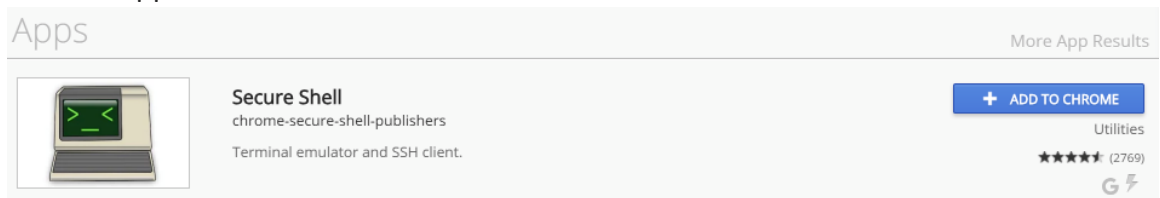
Task 4: Prepare an SSH client

In this task you will prepare an SSH client that you will use to access your HDInsight cluster.

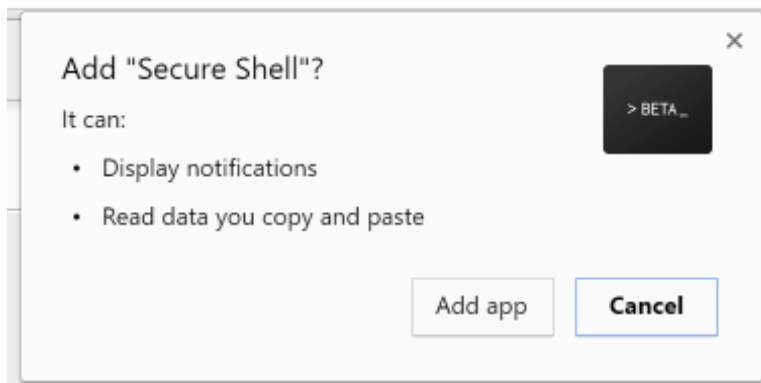
1. Download and install the Chrome browser from:
<https://www.google.com/intl/en/chrome/browser/desktop/index.html>.
2. Launch Chrome.
3. Navigate to the Chrome Web Store at:
<https://chrome.google.com/webstore/category/apps>.
4. In the search field, type Secure Shell.



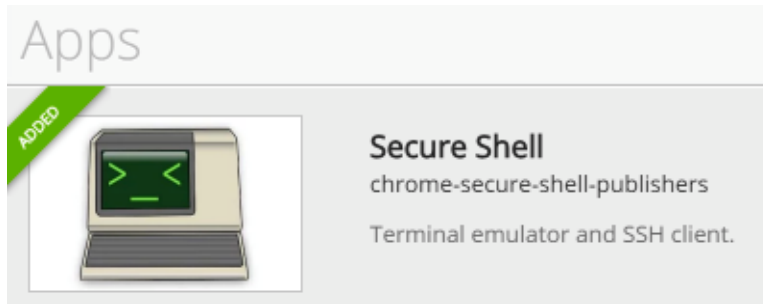
5. Find the app titled Secure Shell and select Add to Chrome.



6. Select Add app when prompted.



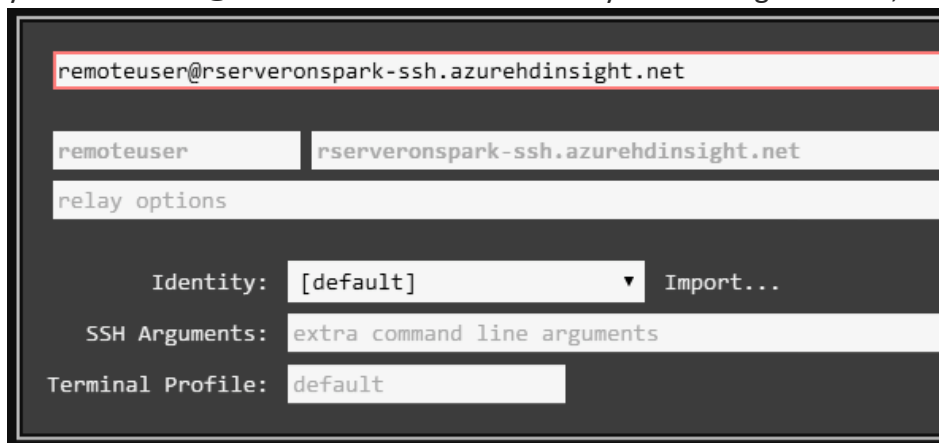
7. Return the Chrome Web Store, search for Secure Shell, and select the listing.



8. In the dialog that appears, select Launch App.



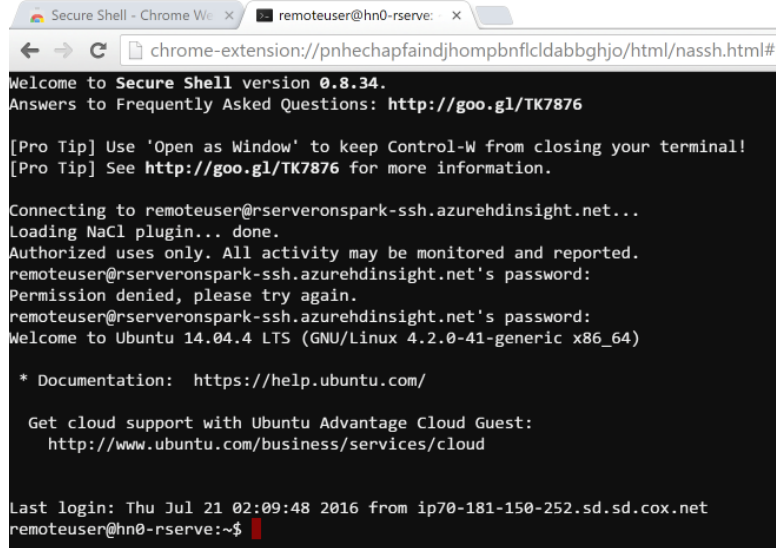
9. The SSH client should appear in a new browser tab. In the box at the center, enter your username@clustername-ssh to SSH into your HDInsight cluster, for example:



10. Select Connect.



11. Use this App for SSH during the hackathon for any instructions requiring an SSH connection. You can repeat these steps any time to re-connect.



```
Secure Shell - Chrome W... x remoteuser@hn0-rserve: x
chrome-extension://pnhechapfaindjhompbncfldabbghjo/html/nassh.html#
Welcome to Secure Shell version 0.8.34.
Answers to Frequently Asked Questions: http://goo.gl/TK7876

[Pro Tip] Use 'Open as Window' to keep Control-W from closing your terminal!
[Pro Tip] See http://goo.gl/TK7876 for more information.

Connecting to remoteuser@rserveronspark-ssh.azurehdinsight.net...
Loading NaCl plugin... done.
Authorized uses only. All activity may be monitored and reported.
remoteuser@rserveronspark-ssh.azurehdinsight.net's password:
Permission denied, please try again.
remoteuser@rserveronspark-ssh.azurehdinsight.net's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

Last login: Thu Jul 21 02:09:48 2016 from ip70-181-150-252.sd.sd.cox.net
remoteuser@hn0-rserve:~$
```

Exercise 1: Environment setup

Duration: 45 minutes

Trey Research has provided a starter solution for you. They have asked you to use this as the starting point for creating the Vending Machines solution in Azure.

Task 1: Download and open the vending machines starter project

1. Download the starter project from the following URL: <http://bit.ly/2j96OxF>
2. Unzip the contents.
3. Open VendingMachines.sln with Visual Studio.

Note: If you attempt to build the solution at this point, you will see many build errors. This is intentional. You will correct these in the exercises that follow.

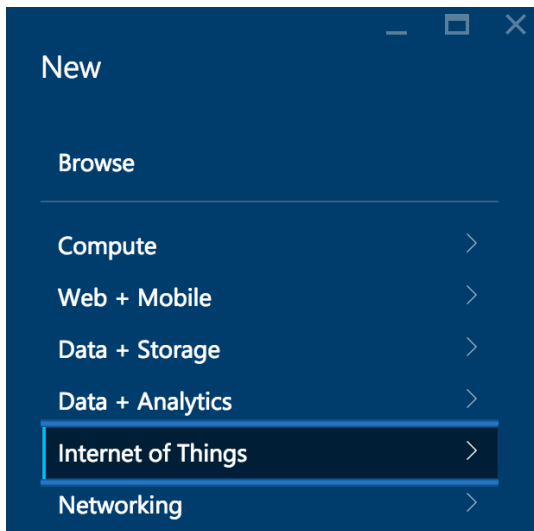
Task 2: IoT Hub

In these steps, you will provision an instance of IoT Hub.

1. In your browser, navigate to the **Azure Portal** (<https://portal.azure.com>).
2. Select **+New** in the navigation bar at the left.



3. Within the **New** blade, select **Internet of Things**.



4. On the **Internet of Things** blade, select **IoT Hub**.
5. In the **IoT Hub** blade, provide a name for your new IoT Hub, choose a **Pricing and scale tier** (**S1 Standard**), set the **IoT Hub units** to **1**, choose the **Resource group**, and choose the **Location**.

* Name
vendinghackhub ✓

* Pricing and scale tier >
S1 - Standard

* IoT Hub units ⓘ
1

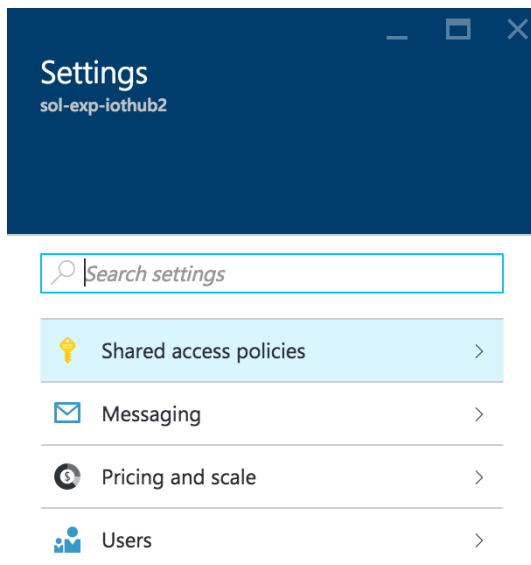
* Device-to-cloud partitions ⓘ
4 partitions ▼

* Subscription
Microsoft Azure Enterprise ▼

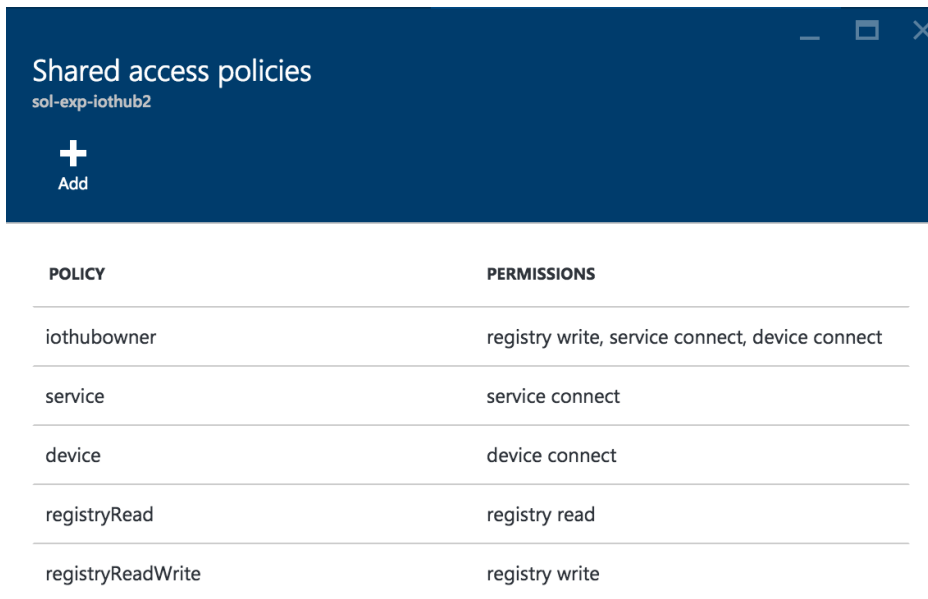
* Resource group ⓘ
☐ Create new ☒ Use existing
 Hackathon ▼

* Location
West US ▼

6. Select **Create**.
7. In the **Settings** blade that appears for your new IoT Hub, select **Shared access policies**.

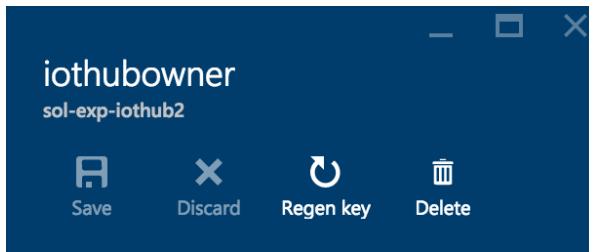


8. Select **iothubowner** policy.



POLICY	PERMISSIONS
iothubowner	registry write, service connect, device connect
service	service connect
device	device connect
registryRead	registry read
registryReadWrite	registry write

9. In the blade that appears, select the **Copy** button to the right of the Connection string—primary key located under the **Shared access keys** section. Take note of this connection string for later in the hackathon.



Access policy name

iothubowner

Permissions

- ☒ Registry read ⓘ
- ☒ Registry write ⓘ
- ☒ Service connect ⓘ
- ☒ Device connect ⓘ

Shared access keys

Primary key ⓘ

Gk/p0O07zWSJ4/dv3nagQxDg27hRhWs=



Secondary key ⓘ

6GBwE+nZPHspugxpWOPR11LkfwXbPvuI



Connection string—primary key ⓘ

HostName=sol-exp-iothub2.azure-device



Connection string—secondary key ⓘ

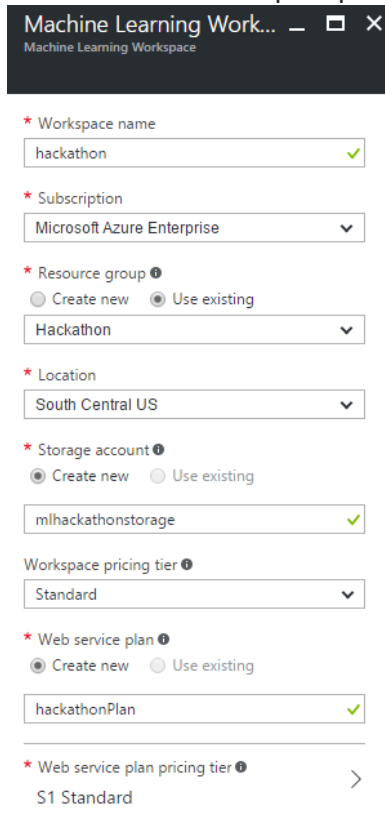
HostName=sol-exp-iothub2.azure-device



Task 3: Provision Azure Machine Learning Workspace

In these steps, you will provision an Azure Machine Learning Workspace

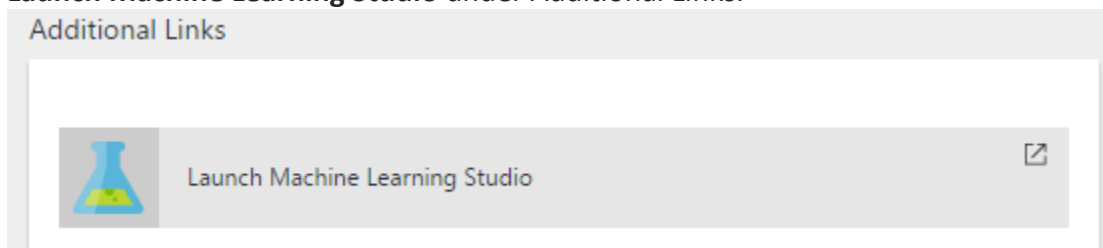
1. Within the Azure Portal, select **+ New, Intelligence + Analytics, Machine Learning Workspace**.
2. Provide a name for the workspace.
3. Select your **Subscription** and **Resource Group**.
4. Choose the **Location** nearest you.
5. Create a new **Storage account**.
6. Set the Workspace pricing tier to **Standard**.
7. For Web service plan, select **Create new and give the new plan a name**.
8. For the Web service plan pricing tier, select **S1 Standard**.



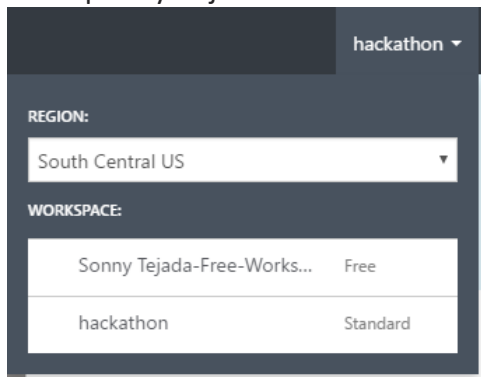
The screenshot shows the 'Machine Learning Workspace' creation form in the Azure Portal. The form includes the following fields and options:

- Workspace name:** 'hackathon' (with a green checkmark).
- Subscription:** 'Microsoft Azure Enterprise' (dropdown menu).
- Resource group:** 'Hackathon' (dropdown menu, with radio buttons for 'Create new' and 'Use existing').
- Location:** 'South Central US' (dropdown menu).
- Storage account:** 'mlhackathonstorage' (with a green checkmark, and radio buttons for 'Create new' and 'Use existing').
- Workspace pricing tier:** 'Standard' (dropdown menu).
- Web service plan:** 'hackathonPlan' (with a green checkmark, and radio buttons for 'Create new' and 'Use existing').
- Web service plan pricing tier:** 'S1 Standard' (dropdown menu).

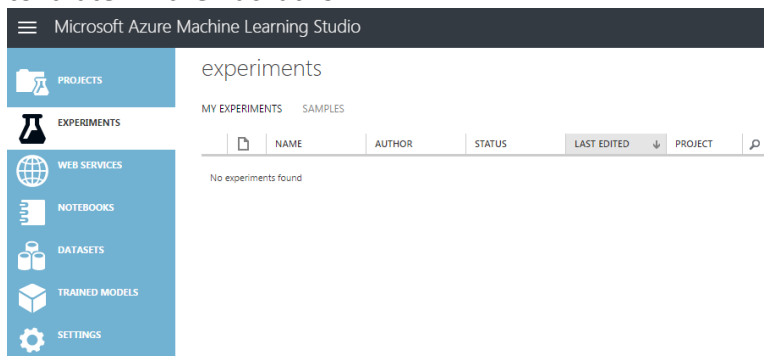
9. Select **Create**.
10. Navigate to your new Machine Learning Workspace blade in the Azure Portal, and select **Launch Machine Learning Studio** under Additional Links.



11. From the navigation menu at the top, expand your workspace name and click on the workspace you just created.



- 12.
13. This will take you Azure Machine Learning Studio where you will later construct an Azure ML model and expose it as a web service. Leave this tab open, as you will return to it later in the hackathon.



Task 4: Storage

In these steps, you will provision a storage account that will be used for storing photos sent from the vending machine simulator and for the storage of the promotional package resources.

1. Using the Azure Portal, select **+ New, Storage, Storage account**.
2. Provide a unique name for the storage account.
3. Leave Resource Manager selected for the deployment model.
4. Leave the Account kind set at General purpose.
5. Leave Performance set to Standard.
6. Set the **Replication to LRS**.
7. Leave Storage service encryption disabled.

8. Choose your Subscription, Resource Group, and Location to be consistent with the other resources you have created.

The cost of your storage account depends on the usage and the options you choose below.
[Learn more](#)

* Name ⓘ
vendinghackphotos ✓
core.windows.net

Deployment model ⓘ
Resource manager Classic

Account kind ⓘ
General purpose ▼

Performance ⓘ
Standard Premium

Replication ⓘ
Locally-redundant storage (LRS) ▼

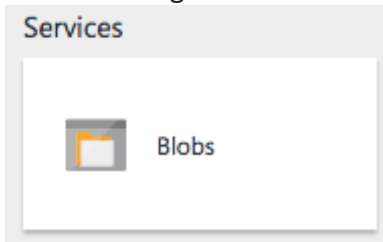
* Storage service encryption ⓘ
Disabled Enabled

* Subscription
Microsoft Azure Enterprise ▼

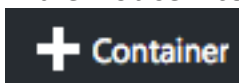
* Resource group ⓘ
☐ Create new ☒ Use existing
Hackathon ▼

* Location
West US ▼

9. Select **Create**.
10. On the Storage account blade, select **Blobs**.



11. In the Blob service blade, select **+ Container** from the command bar.



12. On the New container blade, set the name to photos and Access type of Private.

* Name
photos

Access type ⓘ
Private

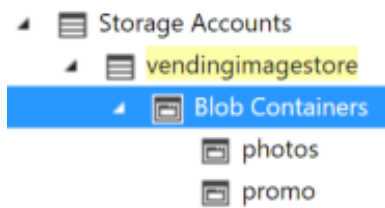
13. Select **Create**.

14. Repeat steps 11-13 to create another container named “promos”.

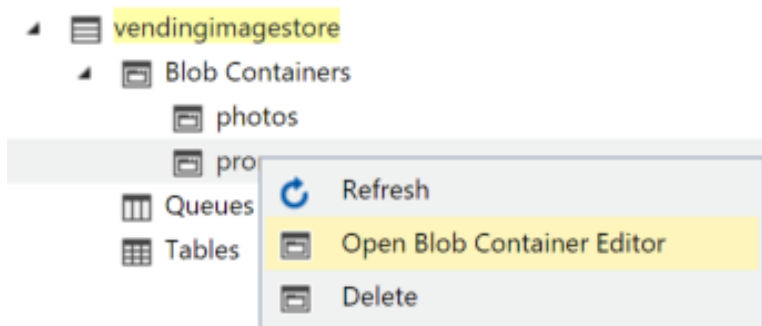
15. Open Visual Studio and from the View Menu select Cloud Explorer.



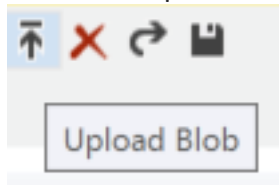
16. Expand the Storage account that you just created, and the Blob Containers item underneath it.



17. Right-click the promo container and select Open Blob Container Editor.



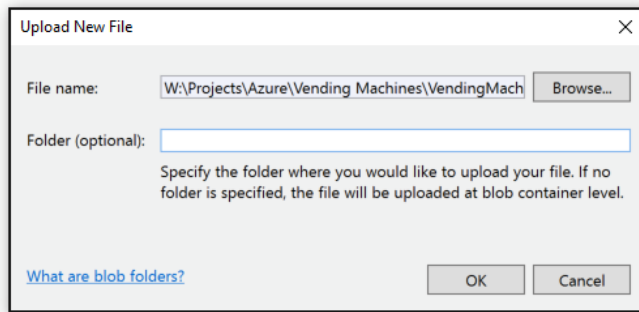
18. Select the Upload blob button.



19. Select Browse.

20. In the dialog, select the three images CoconutWater.png, Water.png, and Soda.png from the starter solution Simulator\Images folder and select Open.

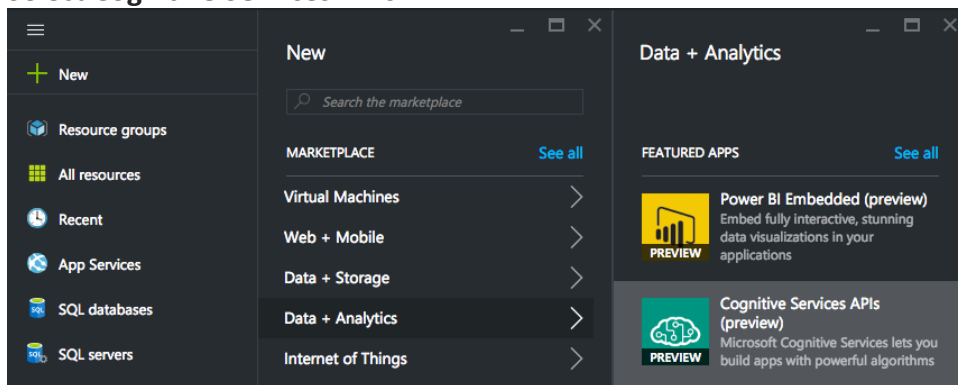
21. Select OK on the Upload New File Dialog to upload the images to the container.



Task 5: Cognitive Services Face API

To provision access to the Face API (which provides demographic information about photos of human subjects), you will need to provision a Cognitive Services account.

1. In the Azure Portal, select **+ NEW**.
2. Select **Intelligence + Analytics**.
3. Select **Cognitive Services APIs**.



4. Provide an Account Name.
5. Choose your Subscription.
6. Select API type and choose Face API.
7. Choose a Location near you.
8. For the Pricing tier, choose the Free tier (**F0**).
9. Choose your existing Resource Group.
10. Agree to the Legal terms.

* Account name
vendhackcog ✓

* Subscription
Microsoft Azure Enterprise ▼

* API type
Face API (preview) ▼

* Location
West US ▼

* Pricing tier ([View full pricing details](#))
F0 (20 Calls per minute, 30K Calls per mon... ▼

* Resource group ⓘ
☐ Create new ☒ Use existing
 Hackathon ▼

☒ * I have read and agree to the legal terms.

By clicking "I Agree", you acknowledge that the Microsoft Cognitive Services are in preview. You may use the Academic Knowledge API, Computer Vision API, Emotion API, Face API, Language Understanding Intelligent Service

11. Select Create to provision the Cognitive Services account.
12. When it finishes provisioning, acquire the key for the API by going to Settings, then Keys and copying the value for Key 1.

The screenshot displays the Azure portal interface for a 'face-detect' Cognitive Services account in preview mode. The main 'Essentials' panel shows account details: Resource group 'sol-zst-demo', Status 'Active', Location 'West US', and Endpoint 'https://api.projectoxford.ai/face/v1.0'. The 'Settings' panel on the right includes a 'Filter settings' search bar and a list of settings categories: 'SUPPORT + TROUBLESHOOTING' (Audit logs), 'MANAGE' (Properties, Keys, Pricing tier). The 'Manage keys' panel on the far right shows the 'ACCOUNT NAME' as 'face-detect' and two keys: 'KEY 1' with value 'c166...' and 'KEY 2' with value '162...'. Both key values are partially obscured by a grey bar.

Task 6: SQL Database

In these steps, you will provision a SQL database to support the transactions and real-time analytics.

1. In the Azure Portal, select **+ NEW**.
2. Select **Databases**.
3. Select **SQL Database**.
4. For the database name, supply “vending”. Choose your Subscription and Resource Group as appropriate. Leave Select source at Blank database. For the Server, choose an existing Server or create a new one.

* Database name
vending ✓

* Subscription
[Dropdown menu]

* Resource group ⓘ
☐ Create new ☒ Use existing
[Dropdown menu]

* Select source ⓘ
Blank database ✓

* Server
[Dropdown menu] >

* Pricing tier ⓘ
P1 Premium >

* Collation ⓘ
SQL_Latin1_General_CP1_CI_AS

5. Change the **Pricing tier to P1** and select **Select**.
6. Select **Create**.
7. Once the SQL Database finishes provisioning, select the Show database connection strings.

Connection strings
[Show database connection strings](#)

8. Take note of the connection string under ADO.NET.

ADO.NET(SQL authentication)

```
Server=tcp:sol-demo-sql.database.windows.net,1433;Initial Catalog=vending;Persist Security
```



Exercise 2: Create Dynamic Pricing Model

Duration: 45 minutes

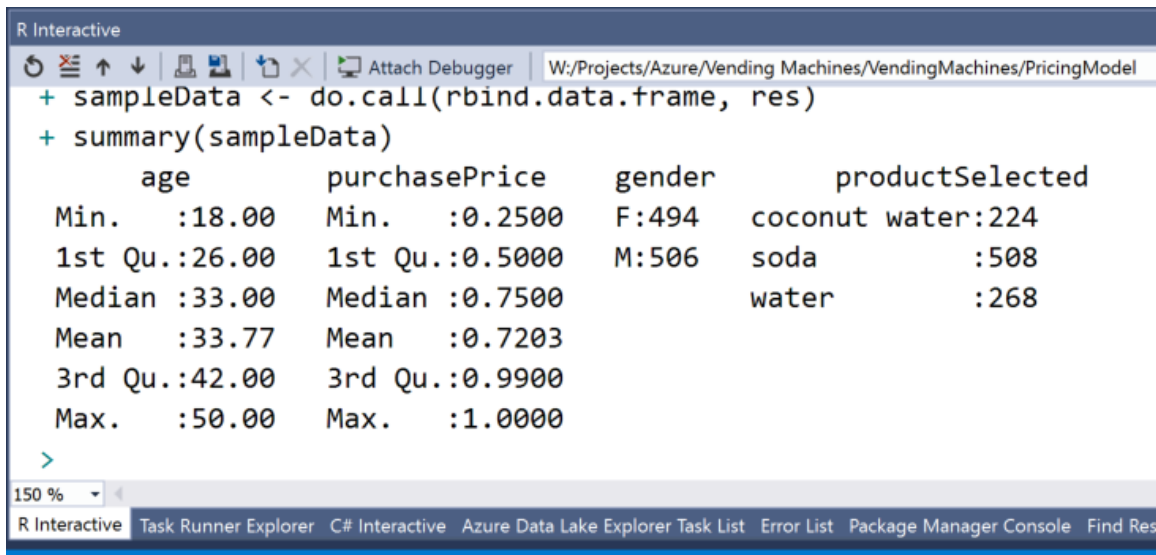
In this exercise, you will create a machine learning model that predicts the purchase price for an item sold by the vending machine provided the demographics of the customer and the item. You will then operationalize this model by exposing it as a web service hosted in Azure Machine Learning and test it out.

Task 1: Create a model locally

1. Within Visual Studio Solution Explorer, expand the PricingModel project and open the file TrainModel.R.
2. Read the script. The top portion, entitled Create Sample Data, has been provided for you and you will generate the sample data you will use to train your model.
3. Highlight all the text between the “Create Sample Data” and “END Create Sample Data” comments.
4. Right-click the selected text and select Execute In Interactive.

```
ages <- c(18:50)
prices <- c(0.25, 0.25, 0.50, 0.75, 0.95)
genders <- c("M", "F", "M", "F", "M")
products <- c("Coke", "Pepsi", "Sprite", "Fanta", "Orange Juice")
distAges <- rep(1, 33)
if (x > 0) {
  rep(1, 33)
} else {
  x
}
generateExampleData(
  age <- sample(18:50, 1)
```


5. You should see it execute in the R Interactive Window, ending with a summary of the created data.



```
R Interactive
W:/Projects/Azure/Vending Machines/VendingMachines/PricingModel
+ sampleData <- do.call(rbind.data.frame, res)
+ summary(sampleData)
```

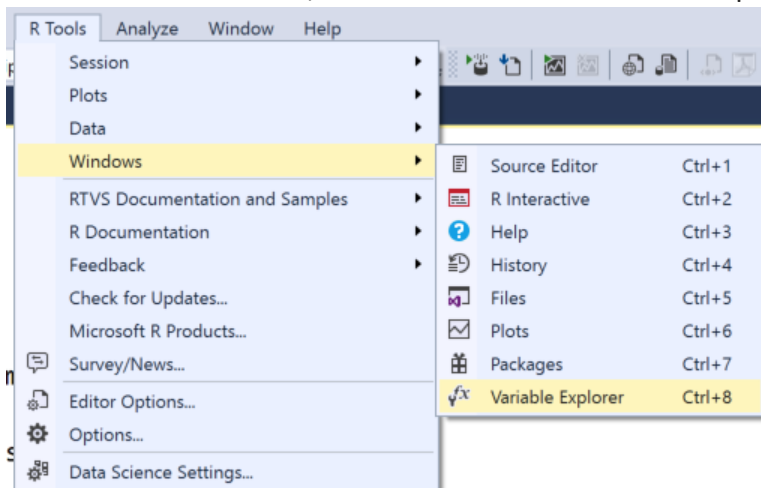
	age	purchasePrice	gender	productSelected
Min.	:18.00	Min. :0.2500	F:494	coconut water:224
1st Qu.	:26.00	1st Qu.:0.5000	M:506	soda :508
Median	:33.00	Median :0.7500		water :268
Mean	:33.77	Mean :0.7203		
3rd Qu.	:42.00	3rd Qu.:0.9900		
Max.	:50.00	Max. :1.0000		

>

150 %

R Interactive Task Runner Explorer C# Interactive Azure Data Lake Explorer Task List Error List Package Manager Console Find Res

6. From the R Tools menu, select Windows and Variable Explorer.



- Expand the variable `sampleData` and explore the structure of the created data.

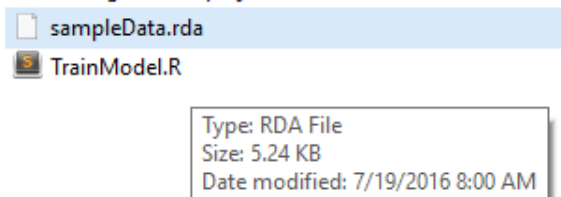
The screenshot shows the RStudio Variable Explorer window. The 'sampleData' variable is selected and expanded, showing its internal structure. The table below represents the data shown in the explorer.

Name	Value	Class	Type
ages	int [1:33] 18 19 20 21 22 23 24 ;	integer	integer
distAges	function (x)	function	closure
genders	Factor w/ 2 levels "F","M": 2 1	factor	integer
generateExample	function (index)	function	closure
prices	num [1:7] 0.25 0.35 0.5 0.75 0.95	numeric	double
products	Factor w/ 3 levels "coconut water"	factor	integer
res	List of 1000	list	list
sampleData	1000 obs. of 4 variables	data.frame	list
@.Data	List of 4	list	list
@.names	chr [1:4] "age" "purchasePrice" "	character	character
@.row.names	chr [1:1000] "2" "2100" "3" "4" "	character	character
@.S3Class	"data.frame"	character	character
age	int [1:1000] 35 19 28 20 48 37 2	integer	integer
purchasePrice	num [1:1000] 1 0.25 0.35 1 0.5 1	numeric	double
gender	Factor w/ 2 levels "F","M": 2 1 1	factor	integer
productSelected	Factor w/ 3 levels "coconut water"	factor	integer

- Now save this `sampleData` to a file by replacing TODO 1 with the following code:

```
# TODO: 1. Export the sample data to a file
save(sampleData, file = "sampleData.rda")
```

- Highlight the save line and select Execute in Interactive.
- Open File Explorer and navigate to the location of the PricingModel project on disk. You should see the file `sampleData.rda` on disk.



- Back in the `TrainModel.R` file in Visual Studio, replace TODO 2 with the following code that builds the model using a Linear Regression.

```
# TODO: 2. Build a linear regression model to predict purchase price given age, gender
and productSelect
pricingModel <- rxLinMod(purchasePrice ~ age + gender + productSelected, data =
sampleData)
```

- Save that trained model to disk by replacing TODO 3 with:

```
# TODO: 3. Export the trained model to a file named pricingModel.rda
save(pricingModel, file = "pricingModel.rda")
```

13. Finally, save the first row of the sample data to a file so you can re-use the structure later when operationalizing the model. Replace TODO 4 with:

```
# TODO: 4. Save one example of the sample data to serve as an input template, to a
file called inputExample.rda
inputExample <- sampleData[1,]
save(inputExample, file = "inputExample.rda")
```

14. Save your changes to Train Model.R.
15. Highlight TODO items 2 through 4 and execute them in interactive.
16. In the same folder as your script, you should now have the files sampleData.rda, pricingModel.rda, and inputExample.rda.

Task 2: Try a prediction locally

1. Within Visual Studio, open **PredictUsingModel.r**.
2. Replace TODO 1 with the following:

```
# TODO: 1. Prepare the input to use for prediction
inputExample[1,]$age <- 30
inputExample[1,]$gender <- "F"
inputExample[1,]$productSelected <- "coconut water"
```

3. Replace TODO 2 with the following:

```
# TODO: 2. Execute the prediction
prediction <- rxPredict(pricingModel, data = inputExample)
```

4. Highlight all the script in the file and execute it in interactive.
5. Using Variable Explorer, expand the prediction variable and observe the price the model suggested to use for purchasing the coconut water for input of a 30-year-old female.

prediction	1 obs. of 1 variable		data.frame	list
▷ @.Data	List of 1		list	list
@names	"purchasePrice_Pred"		character	character
@row.names	1		integer	integer
@.S3Class	"data.frame"		character	character
purchasePrice_Prec	0.949		numeric	double

Task 3: Create the model in R Server on HDInsight

1. SSH into your deployed R Server in HDInsight cluster. (You can get the syntax for your cluster from the HDInsight Blade in the Azure Portal, Settings, and then Secure Shell.) For example:

```
ssh <user>@rserveronspark-ssh.azurehdinsight.net.
```

2. When prompted if you want to continue connecting, enter yes.
3. Enter your password.
4. At the command prompt, type R to load the R shell (be sure to use a capital letter "R").
5. Run the following command to create a spark context for R:

```
sparkCluster <- RxSpark()
```

```
rxSetComputeContext(sparkCluster)
```

6. In Visual Studio, open TrainModel.r and copy the entire script.
7. Paste the script in the R shell and press ENTER. (You may need to press ENTER a few times until you get to the last line of the script.)
8. When the script has finished executing, type the following:
dir().
9. You should see it list the three files created by the script, as follows:

```
> dir()  
[1] "inputExample.rda" "pricingModel.rda" "sampleData.rda"
```

10. Now, copy those files from local storage to Blob storage by using the Hadoop File System. First, create a folder in which to store your output.

```
modelExportDir <- "/models"
```

```
rxHadoopMakeDir(modelExportDir)
```

11. List the contents of the /models directory and confirm your folder has been created. Notice that the list you are looking at is folders directly underneath the container in Azure Storage that was created with your cluster.

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

```
[> rxHadoopListFiles("/")
Found 18 items
drwxrwxrwx - 0 1970-01-01 00:00 /HdiApplications
drwxr-xr-x - root supergroup 0 2016-07-20 19:09 /HdiNotebooks
drwxr-xr-x - root supergroup 0 2016-07-20 19:15 /HdiSamples
drwxr-xr-x - hdfs supergroup 0 2016-07-20 18:56 /ams
drwxr-xr-x - hdfs supergroup 0 2016-07-20 18:56 /amshbase
drwxrwxrwx - yarn hadoop 0 2016-07-20 18:56 /app-logs
drwxr-xr-x - yarn hadoop 0 2016-07-20 18:56 /atshistory
drwxr-xr-x - root supergroup 0 2016-07-20 19:28 /cluster-info
drwxr-xr-x - root supergroup 0 2016-07-20 19:29 /custom-scriptaction-logs
drwxr-xr-x - root supergroup 0 2016-07-20 19:15 /example
drwxr-xr-x - hdfs supergroup 0 2016-07-20 18:56 /hdp
drwxr-xr-x - hdfs supergroup 0 2016-07-20 18:56 /hive
drwxr-xr-x - mapred supergroup 0 2016-07-20 18:56 /mapred
drwxr-xr-x - remoteuser supergroup 0 2016-07-20 20:03 /models
drwxrwxrwx - mapred hadoop 0 2016-07-20 18:56 /mr-history
drwxrwxrwx - root supergroup 0 2016-07-20 19:28 /share
drwxrwxrwx - hdfs supergroup 0 2016-07-20 18:56 /tmp
drwxrwxrwx - hdfs supergroup 0 2016-07-20 19:28 /user
```

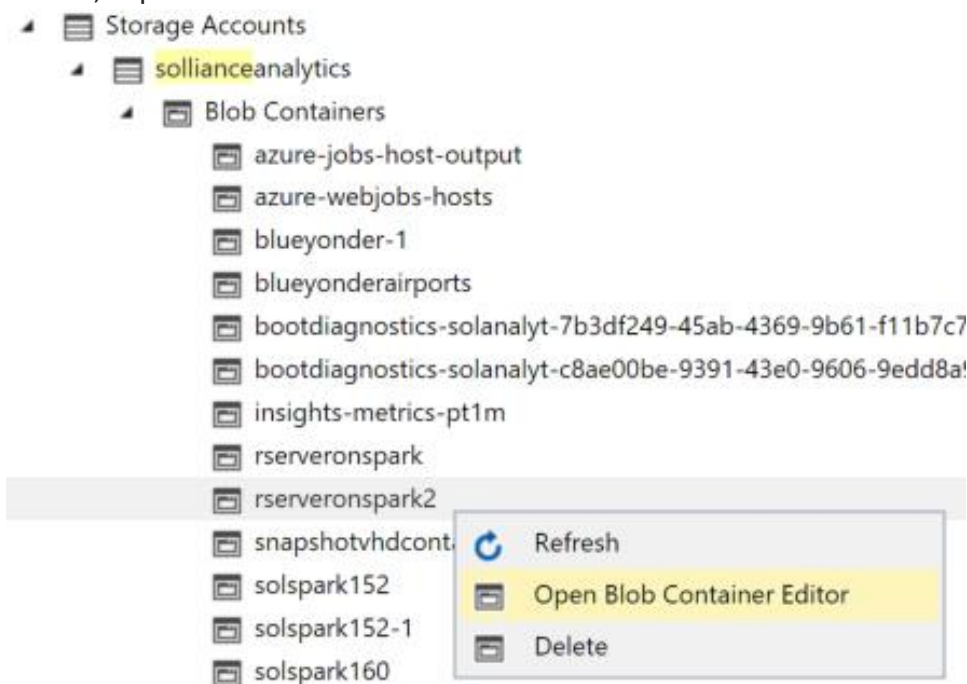
12. Copy the pricingModel.rda from the local directory to HDFS by running the following command:

```
rxHadoopCopyFromLocal("pricingModel.rda", modelExportDir)
```
13. Repeat the previous step for inputExample.rda and sampleData.rda.
14. Run the following command to verify the three files now exist in HDFS (and Blob storage), under /Models.

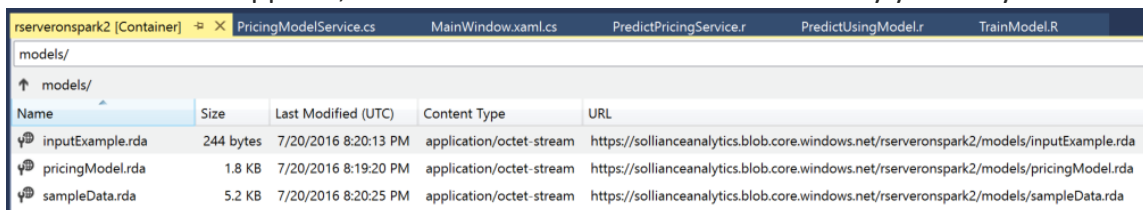
```
rxHadoopListFiles("/models")
```
15. The output should look similar to the following:

```
[> rxHadoopListFiles("/models")
Found 3 items
-rw-r--r-- 1 remoteuser supergroup 244 2016-07-20 20:20 /models/inputExample.rda
-rw-r--r-- 1 remoteuser supergroup 1852 2016-07-20 20:19 /models/pricingModel.rda
-rw-r--r-- 1 remoteuser supergroup 5369 2016-07-20 20:20 /models/sampleData.rda
```

16. Using Visual Studio, Cloud Explorer, navigate to the storage account for your HDInsight cluster, expand:



17. In the editor that appears, double-click the models folder and verify you see your files.

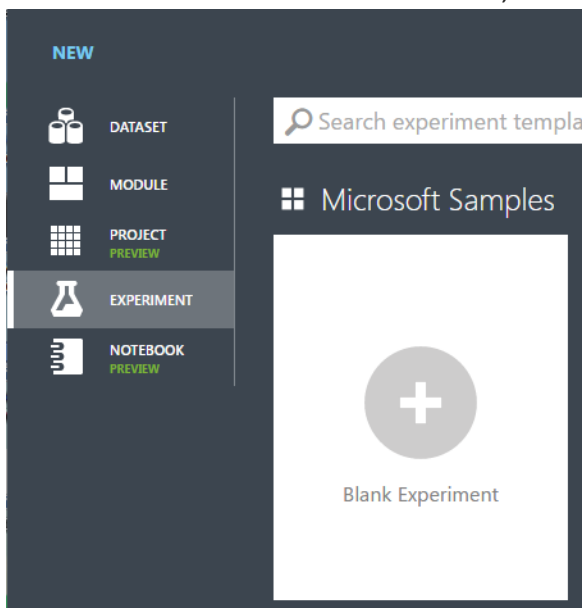


18. Right-click inputExample.rda and select Save As... and choose a directory on your local computer in which to save the file.
19. Repeat the previous step for pricingModel.rda and sampleData.rda.
20. You have now used R Server on HDInsight to train a model that you can then upload to DeployR to expose it as a web service.

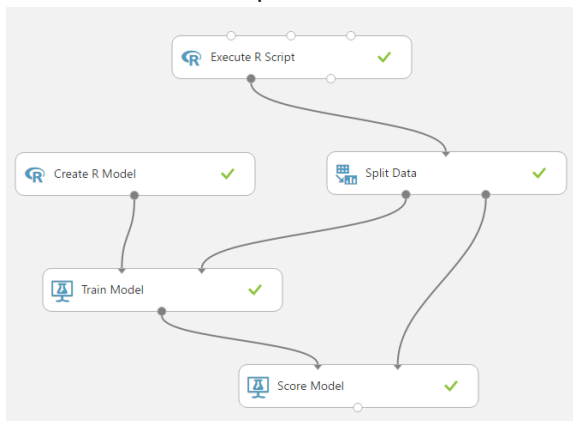
Task 4: Create predictive service in Azure Machine Learning

After training a model, you want to operationalize the model so that it becomes available for integration by developers. One way to operationalize and trained model is to re-build the on-premises script base training as an Azure Machine Learning experiment and then to expose that as a predictive web service. In this task, you take a version of the scripts you have been running locally and in HDInsight and migrate them to run in Azure Machine Learning.

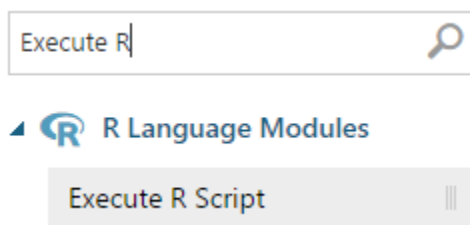
1. Using a browser, return to the Azure Machine Learning Studio tab you had previously opened.
2. From the command bar at the bottom, select **+ New, Blank Experiment**.



3. You will build an experiment that looks like the following:



4. Using the Search experiment items in the upper left, type **Execute R**, then select the Execute R Script item that appears, and drag and drop it on the design surface.



5. In the Properties panel on the right, click the double windows icon to view the R script code.

Properties Project

Execute R Script

R Script

```
1 # Create Sample Data
2 # 1k rows
3 # ProductName | eAge
4 # CoconutWater | 24 |
5 # CoconutWater | 24 |
6
```

6. Return to Visual Studio.
7. In Solution Explorer, expand the PricingModel project, and then open TrainAndPredictAzureML.R.
8. Copy all of the script between the comments
#Create Sample Data -----
and
END Create Sample Data -----
9. Return to your browser where you had the R Script open in Azure ML Studio.
10. Paste the script and click the checkmark. This script will generate some data that simulates historical purchases from the vending machine.
11. From the modules, select a Create R Model module and drop it beneath the Execute R Script.
12. In the properties panel for the Create R Model module, click the double windows icon below the Trainer R Script to view the script.

Create R Model

Trainer R script

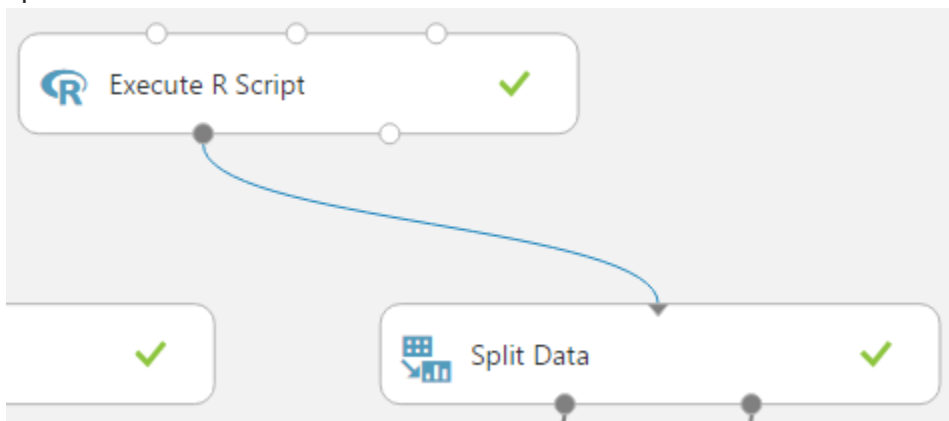
```
1 # Input: dataset
2 # Output: model
3
4 # The code below is a
5 # See the help page c
6
```

13. Replace the contents with script from TrainAndPredictAzureML.R, this time copying the script between
Build the Model
and
END Build the Model

14. Select the checkmark in the Trainer R Script dialog to save the script.
15. Next, select the double window underneath Scorer R script in the properties panel of the Create R Model module.

```
Scorer R script
1 # Input: model, data
2 # Output: scores
3
4 # Score an example using the model
5 prediction <- predict(model, data)
6 summary(prediction)
```

16. Replace the contents of the Scorer R script with the script from TrainAndPredictAzureML.R for the script between
Score an example using the model
and
END score an example
17. From the modules, select Split Data and drop it on the design surface.
18. Connect the left output of the Execute R Script (labeled Result Dataset) to the input of the Split Data module.



19. With the Split Data Module selected, in the Properties panel, set the Fraction of the rows in the first output dataset to 0.7 (so that 70% will flow out the left output, and 30% of the rows will go to the right output).

Properties Project

Split Data

Splitting mode

Split Rows

Fraction of rows in the first...

0.7

☒ Randomized split

Random seed

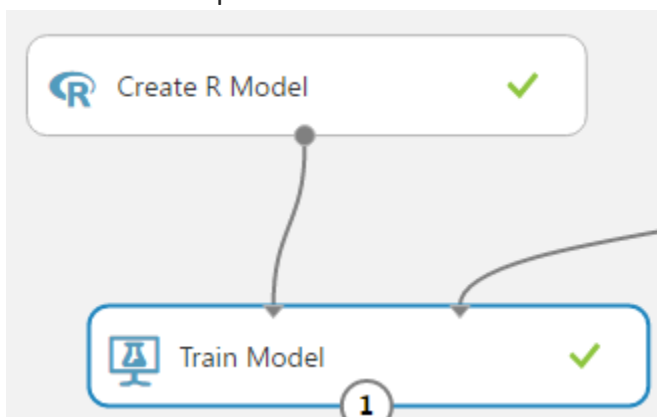
0

Stratified split

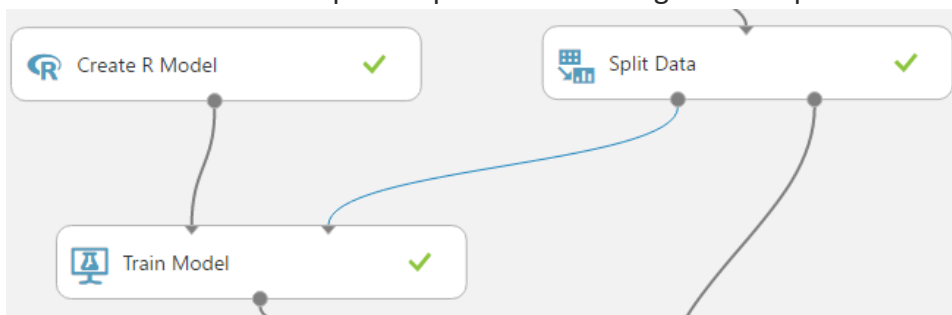
False

20. From the modules, select Train Model and drop it on the design surface, below the Train Model module.

21. Connect the output of the Create R Model module to the leftmost input of Train Model.



22. Connect the leftmost output of Split Data to the rightmost input of Train Model.



23. Select the Train Model module and in the Properties panel, select Launch column selector.

▲ Train Model

Label column

Selected columns:

Column names:

purchasePrice

Launch column selector

24. In the **Select a single column** dialog, set the options to **Include**, **column names** and type **purchasePrice**, and the select the check mark.

Select a single column ×

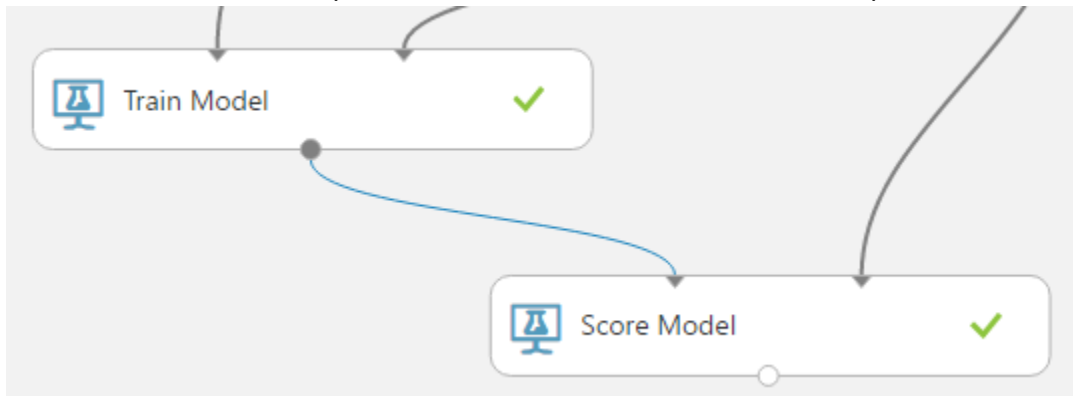
BY NAME
WITH RULES

Include ▼ column names ▼ purchasePrice ×

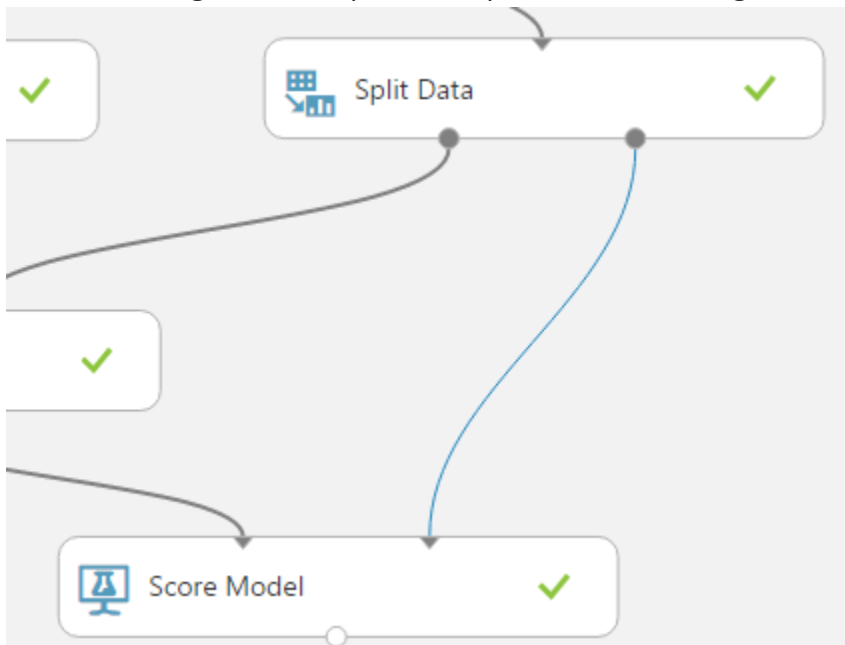
✓

25. From the module list, select a Score Model module and drop it on the design surface below Train Model.

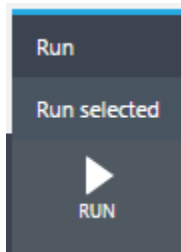
26. Connect the leftmost output from Train Model into the leftmost input of Score Model.



27. Connect the rightmost output from Split Data into the rightmost input of Score Model.



28. At the bottom of the screen, select Run, Run and let your experiment train your model.

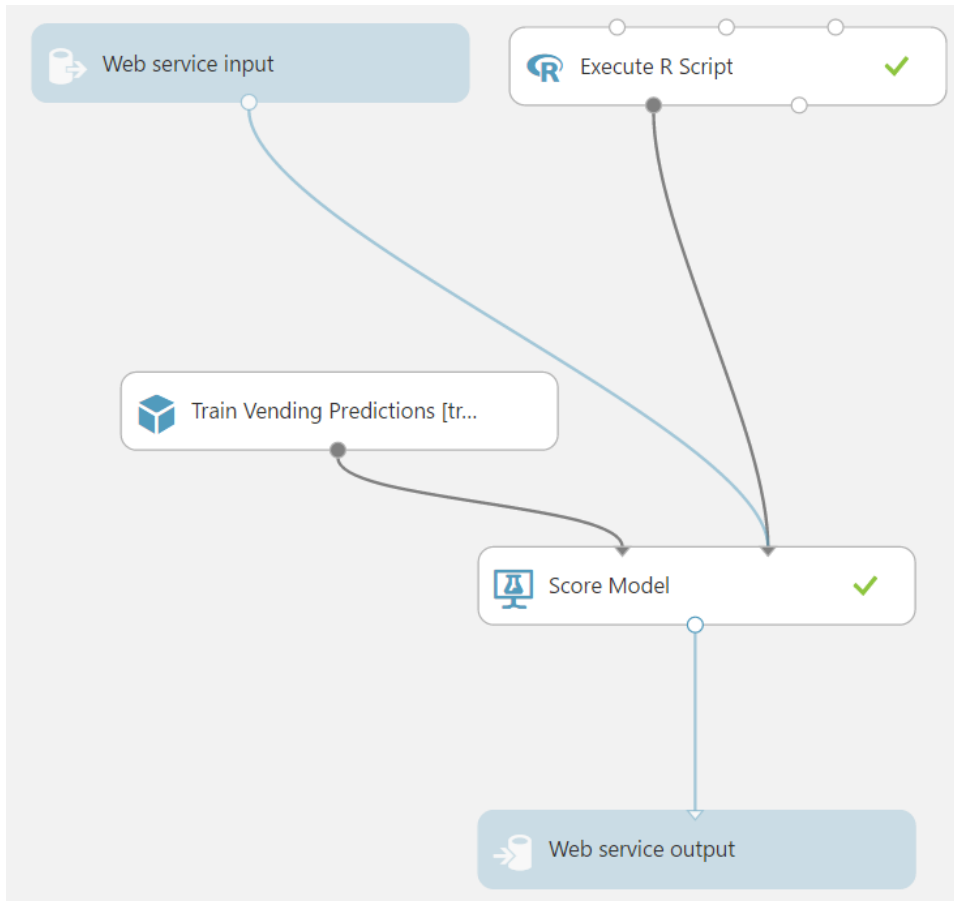


29. After running successfully, all of the modules should have a green checkmark.

30. Select Setup Web Service and choose Create Predictive Experiment.

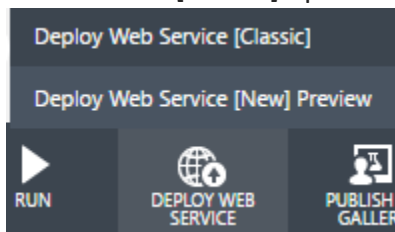
31. In the new Predictive experiment that is created, add a new Web service input module and connect it directly to the right most input of the Score Model module.

32. Ensure the Execute R Script is also connected to the rightmost input of the Score Model module.



33. Select Run.

34. When the predictive experiment finishes, select Deploy Web Service and choose the Deploy Web Service [Classic] option.



35. On the Dashboard that appears, copy the API key.

API key

6tbLGoDobfRXXV0glUvnXGABn81NwpJ5mbVekF

36. In Visual Studio, open the App.config for the Simulator project.

37. Locate the appSetting azureMLServiceApiKey and paste this for its value.

38. Return to the browser.

39. Below the API Key, select the link REQUEST/RESPONSE

Default Endpoint

API HELP PAGE

REQUEST/RESPONSE

40. Copy the Request URI value displayed (just the URI).

Request

Method	Request URI
POST	https://ussouthcentral.services.azureml.net/workspaces/f5e2fba8111243b193eca577328e75de/services/de331398617a4a4485782f9b6e81cc06/execute?api-version=2.0&details=true

41. Back in Visual Studio, in the App.config for the Simulator, paste this URI as the value for the azureMLServiceBaseAddress setting. Be sure to replace the & in the url with &, similar to the following:

<https://ussouthcentral.services.azureml.net/workspaces/f5e2fba8111243b193eca577328e75de/services/de331398617a4a4485782f9b6e81cc06/execute?api-version=2.0&details=true>

42. Save the app.config.

Exercise 3: Implement dynamic pricing

Duration: 45 minutes

In this exercise, you will implement the code that performs dynamic pricing, capitalizing on the Face API to acquire demographics and your deployed pricing model to suggest the price based on those demographics. You will then run the vending machine simulator and see the dynamic pricing in action.

Task 1: Implement photo uploads to Azure Storage

1. In Visual Studio Solution Explorer, expand the Simulator project and then MainWindow.xaml and then open MainWindow.xaml.cs.
2. Scroll down to the method UpdateDynamicPricing.
3. Replace TODO 1 with the following:

```
// TODO 1. Retrieve storage account from connection string.  
CloudStorageAccount storageAccount =  
CloudStorageAccount.Parse(_storageConnectionString);  
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();  
CloudBlobContainer container = blobClient.GetContainerReference("photos");
```

4. Replace TODO 2 with the following:

```
// TODO 2. Retrieve reference to a blob named with the value of fileName.  
string blobName = Guid.NewGuid().ToString() + System.IO.Path.GetExtension(filename);  
CloudBlockBlob blockBlob = container.GetBlockBlobReference(blobName);
```

5. Replace TODO 3 with the following:

```
// TODO 3. Create or overwrite the blob with contents from a local file.
using (var fileStream = System.IO.File.OpenRead(filename))
{
    blockBlob.UploadFromStream(fileStream);
}
```

6. Save MainWindow.xaml.cs.

Task 2: Invoke Face API

1. Continuing with MainWindow.xaml.cs, scroll down to GetBlobSasUri. This method will create a Shared Access Signature URI that the Face API can use to securely access the image in blob storage.
2. Replace TODO 4 with the following:

```
//TODO: 4. Create a Read blob and Write blob Shared Access Policy that is effective 5
minutes ago and for 2 hours into the future
SharedAccessBlobPolicy sasConstraints = new SharedAccessBlobPolicy();
sasConstraints.SharedAccessStartTime = DateTime.UtcNow.AddMinutes(-5);
sasConstraints.SharedAccessExpiryTime = DateTime.UtcNow.AddHours(2);
sasConstraints.Permissions = SharedAccessBlobPermissions.Read |
SharedAccessBlobPermissions.Write;
```

3. Replace TODO 5 with the following:

```
//TODO: 5. construct the full URI with SAS
string sasBlobToken = blob.GetSharedAccessSignature(sasConstraints);
return blob.Uri + sasBlobToken;
```

4. With the SAS URI to the upload photo in hand, scroll to GetPhotoDemographics to implement the call to the Face API.
5. Replace TODO 6 with the following:

```
//TODO 6. Invoke Face API with URI to photo
IFaceServiceClient faceServiceClient = new FaceServiceClient(_faceApiKey);
```

6. Replace TODO 7 with the following:

```
//TODO 7. Configure the desired attributes Age and Gender
IEnumerable<FaceAttributeType> desiredAttributes = new FaceAttributeType[] {
    FaceAttributeType.Age, FaceAttributeType.Gender };;
```

7. Replace TODO 8 with the following:

```
//TODO 8. Invoke the Face API Detect operation
Face[] faces = await faceServiceClient.DetectAsync(sasUri, false, true,
    desiredAttributes);
```

8. Replace TODO 9 with the following:

```
//TODO 9. Extract the age and gender from the Face API response
double computedAge = faces[0].FaceAttributes.Age;
string computedGender = faces[0].FaceAttributes.Gender;
```

9. Save the file.

Task 3: Invoke pricing model

1. Within MainWindow.xaml.cs, scroll to the end of Update Dynamic Price and replace TODO 10 with the following:

```
//TODO 10. Invoke the actual ML Model
PricingModelService pricingModel = new PricingModelService();
string gender = d.gender == "Female" ? "F" : "M";
suggestedPrice = await pricingModel.GetSuggestedPrice((int)d.age, gender, _itemName);
```

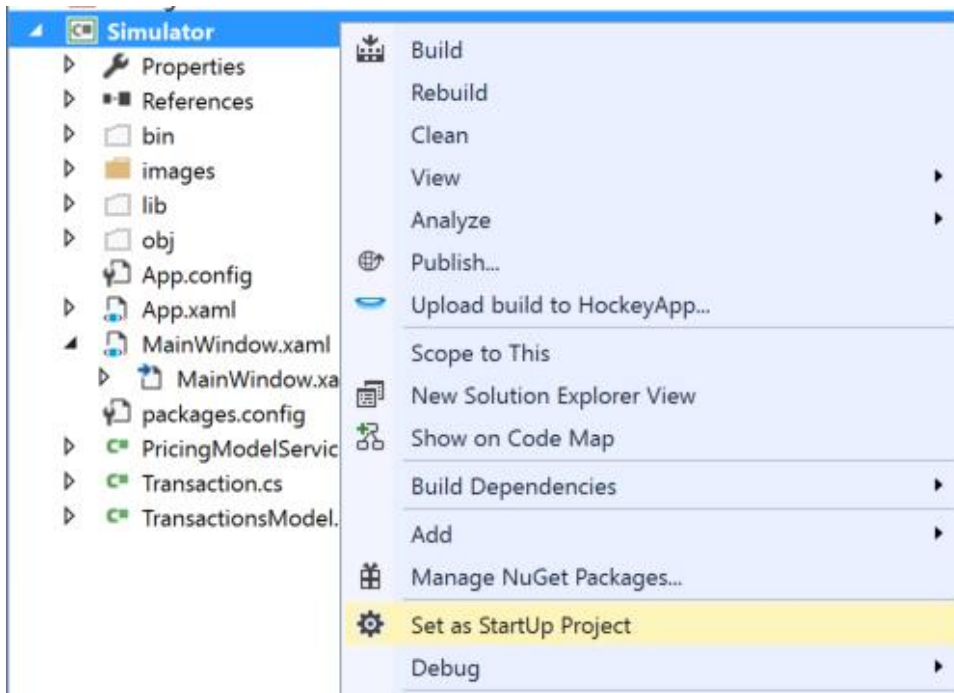
2. Save the file.

Task 4: Configure the Simulator

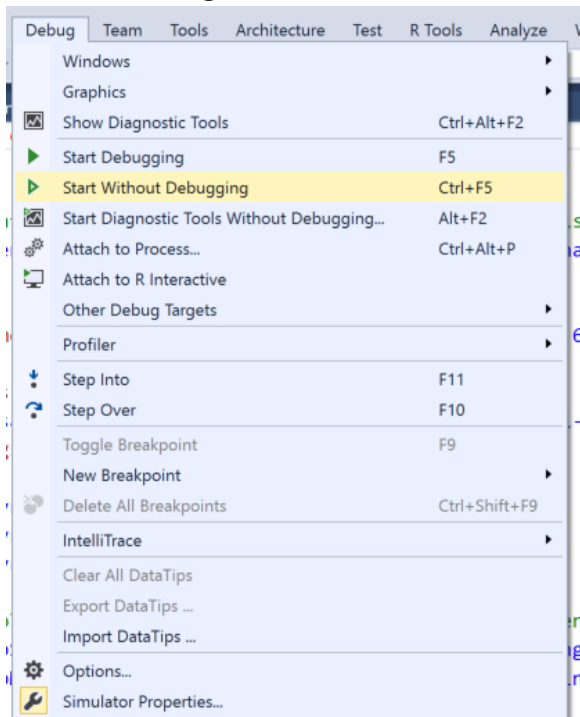
1. In the Simulator project, open App.config.
2. Within the appSettings section, set the following settings:
 - a. faceAPIKey: set this to the KEY 1 value for your Face API as acquired from the Azure Portal.
 - b. storageConnectionString: set this to connection string the Storage Account you created with the photos container.
3. Save the App.config.

Task 5: Test dynamic pricing in Simulator

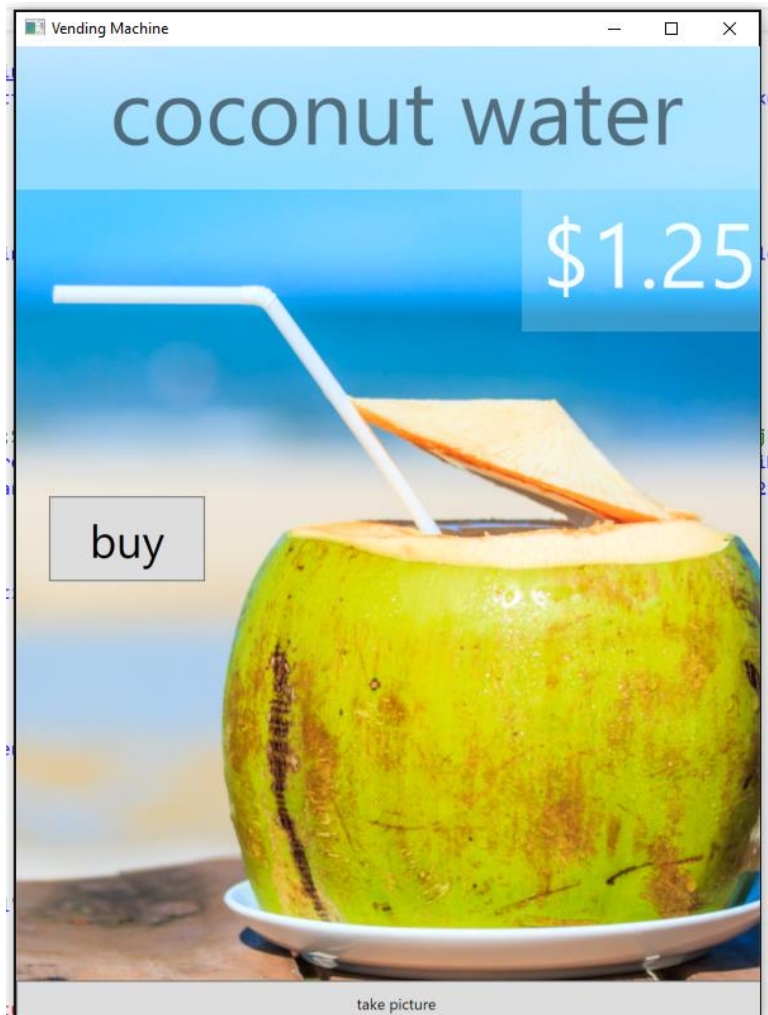
1. In solution explorer, right-click the Simulator project and select Build.
2. Ensure that your build generates no errors.
3. Again, in solution explorer, right-click the Simulator project and select Set as Startup Project.



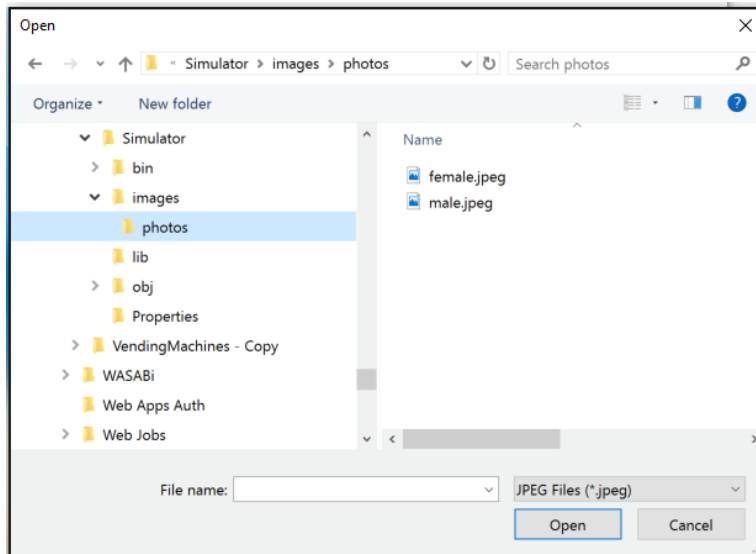
4. From the Debug menu, select Start Without Debugging.



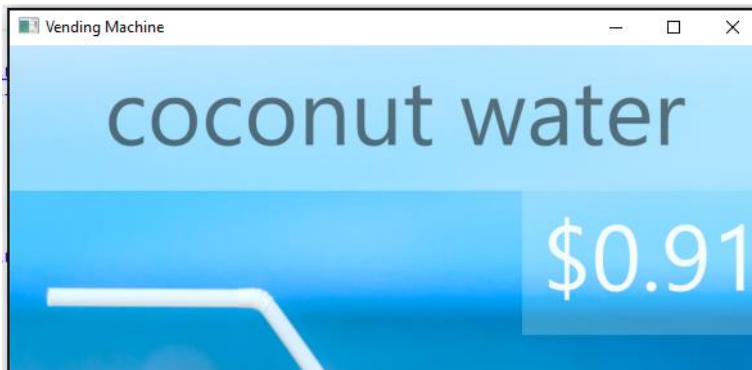
5. When the vending machine simulator appears, select take picture at the bottom.



6. In the dialog that appears, navigate to the images folder under Simulator\bin\debug and pick the photo of either the man or woman to upload and select Open.



7. In a few moments, you should see the price change from \$1.25 to whatever value the predictive model suggested.



8. Try using the other photo or your own photo to see what prices are suggested.

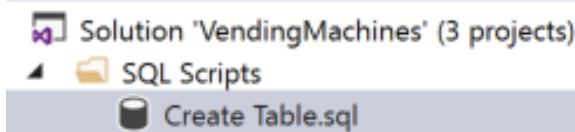
Exercise 4: Implement purchasing

Duration: 15 minutes

In this exercise, you will create an in-memory, columnar index table in SQL DB that will be used to support purchase transactions in a real-time analytics fashion, and then implement the purchasing process in the vending machine simulator. Finally, you will run the simulator and purchase items.

Task 1: Create the transactions table

1. Within Visual Studio Solution Explorer, expand the SQL Scripts folder and open the file Create Table.sql.



2. Replace TODO 1 with the following:

```
-- TODO: 1. Transaction ID should be a Primary Key, fields with a b-tree index  
TransactionId int IDENTITY NOT NULL PRIMARY KEY NONCLUSTERED,
```

3. Replace TODO 2 with the following:

```
-- TODO: 2. This table should have a columnar index  
INDEX Transactions_CCI CLUSTERED COLUMNSTORE
```

4. Replace TODO 3 with the following:

```
-- TODO: 3. This should be an in-memory table  
MEMORY_OPTIMIZED = ON
```

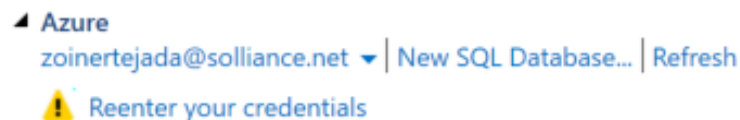
5. Replace TODO 4 with the following:

```
-- TODO: 4. In-memory tables should auto-elevate their transaction level to Snapshot  
ALTER DATABASE CURRENT SET MEMORY_OPTIMIZED_ELEVATE_TO_SNAPSHOT=ON ;
```

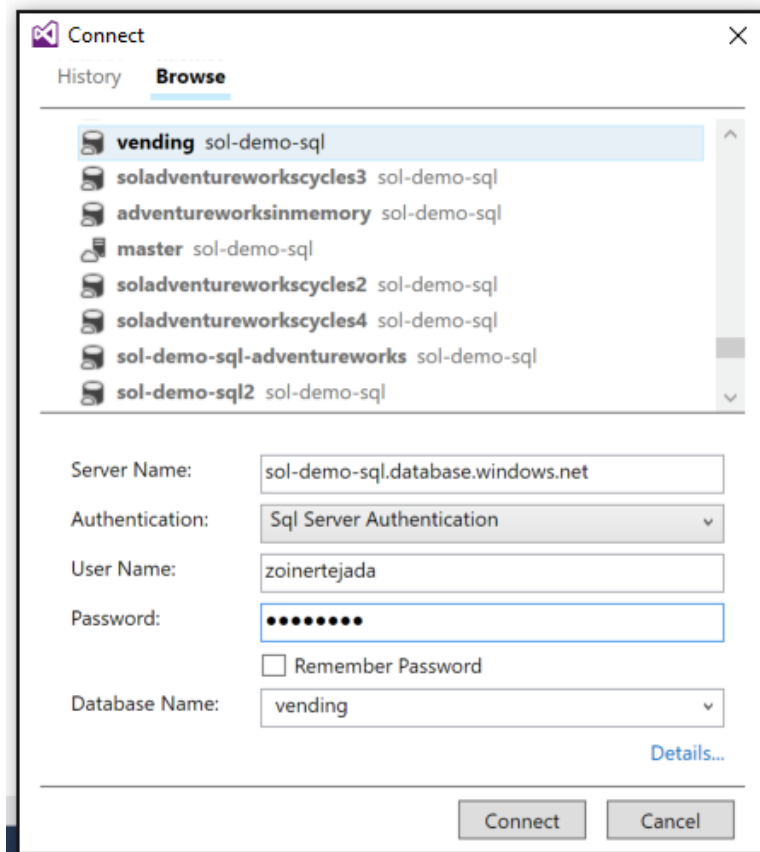
6. Save the script.
7. Execute the script by pressing the play icon.



8. When prompted, sign in with your Azure credentials.



- Expand the Azure node and select the database you created for the vending database.



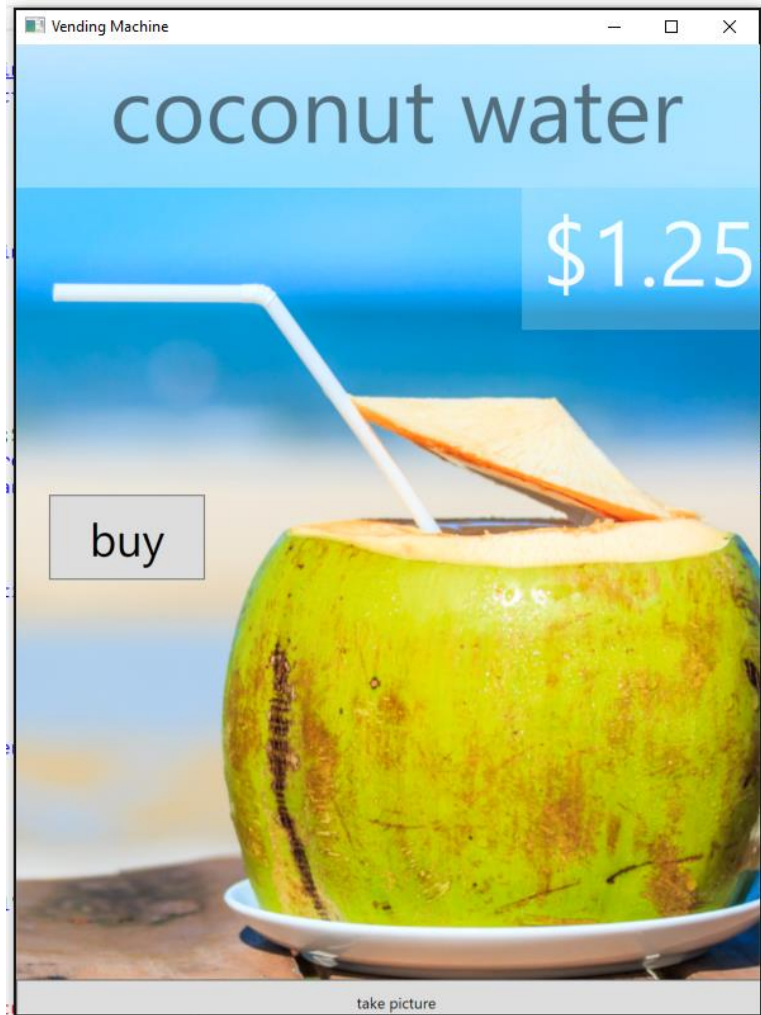
- In the in fields at the bottom, enter your user name and password for the SQL Server and select Connect. The script should run successfully.

Task 2: Configure the Simulator

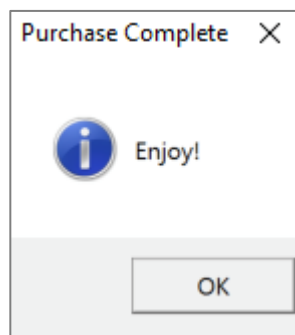
- In the Simulator project, open App.config.
- Within the connectionString section, set the following:
 - TransactionsModel: set the value of the connectionString attribute to the ADO.NET connection string to your SQL DB instance. When copying this value from the Azure Portal, do not forget to replace the values for {your_username} and {your_password} with your actual credentials.
- Save the App.config.

Task 3: Test purchasing

1. In solution explorer, right-click the Simulator project and select Build.
2. Ensure that your build generates no errors.
3. From the Debug menu, select Start Without Debugging.
4. In the Simulator, select buy.



5. You should see a confirmation dialog similar to the following:



Exercise 5: Implement device command and control

Duration: 30 minutes

In this exercise, you will implement the ability to push new promotions to the vending machine simulator using the command and control features of IoT Hub. You will update the simulator to listen for these messages. You will also update the console application DeviceControlConsole to send selected promotions.

Task 1: Listen for control messages

1. Within Visual Studio Solution Explorer, expand the Simulator project and open the file MainWindow.xaml.cs.
2. Scroll down to the ListenForControlMessages method.
3. Uncomment the body of the while(true) loop.
4. Replace TODO 1 with the following:

```
//TODO: 1. Receive messages intended for the device via the instance of _deviceClient.  
Microsoft.Azure.Devices.Client.Message receivedMessage = await  
_deviceClient.ReceiveAsync();
```

5. Replace TODO 2 with the following:

```
//TODO: 2. A null message may be received if the wait period expired, so ignore and  
call the receive operation again  
if (receivedMessage == null) continue;
```

6. Replace TODO 3 with the following:

```
//TODO: 3. Deserialize the received binary encoded JSON message into an instance of  
PromoPackage.  
string receivedJSON = Encoding.ASCII.GetString(receivedMessage.GetBytes());  
System.Diagnostics.Trace.TraceInformation("Received message: {0}", receivedJSON);  
PromoPackage promo =  
Newtonsoft.Json.JsonConvert.DeserializeObject<PromoPackage>(receivedJSON);
```

7. Replace TODO 4 with the following:

```
//TODO: 4. Acknowledge receipt of the message with IoT Hub  
await _deviceClient.CompleteAsync(receivedMessage);
```

8. Save the file.

Task 2: Send control messages

1. Within Visual Studio Solution Explorer, expand the DeviceControlConsole project and open the file Program.cs.
2. Scroll down to the PushPromo method.
3. Replace TODO 1 with the following:

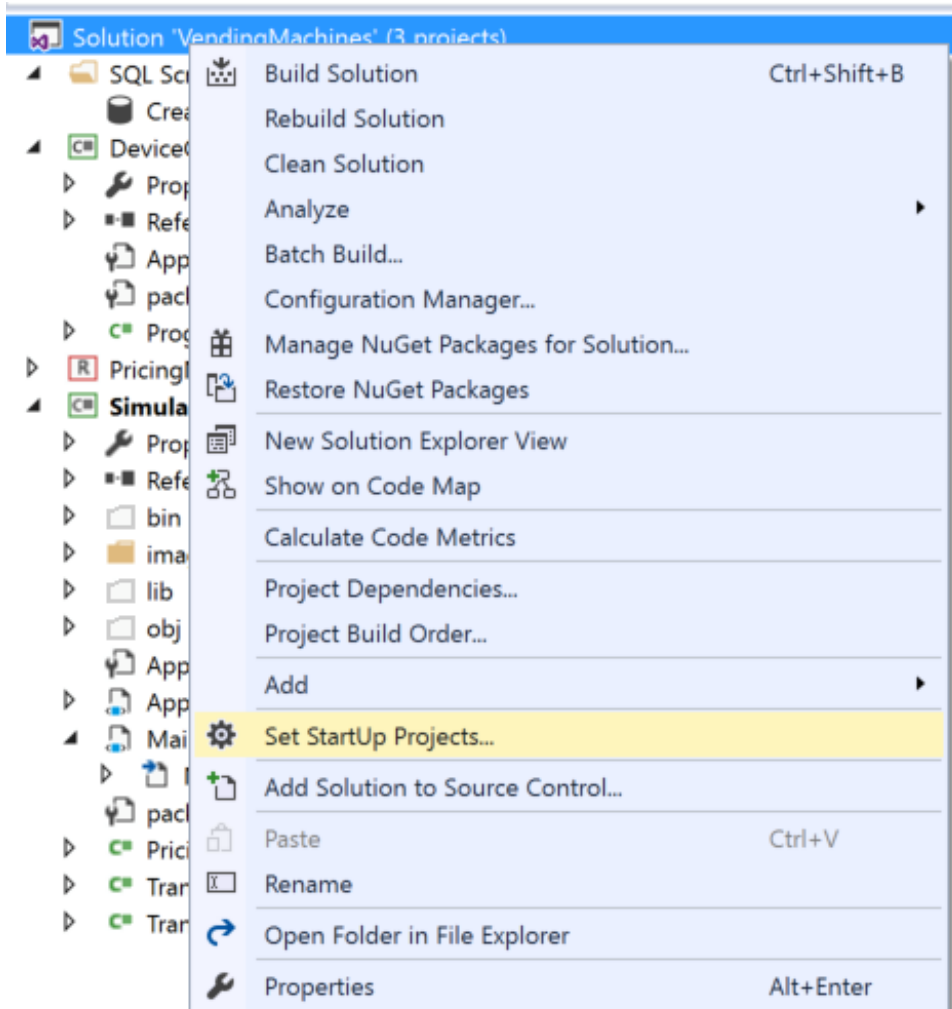
```
//TODO: 1. Create a Service Client instance provided the _IoTHubConnectionString
_serviceClient = ServiceClient.CreateFromConnectionString(_IoTHubConnectionString);
```

4. Replace TODO 2 with the following:

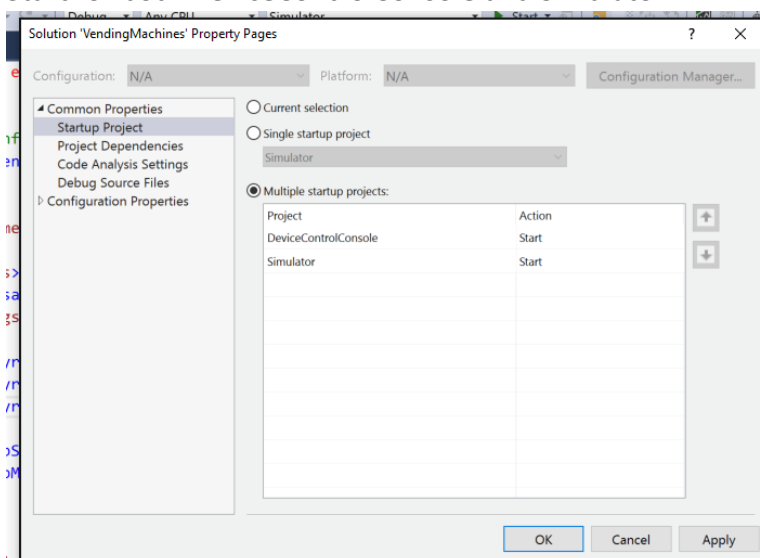
```
//TODO: 2. Send the command
await _serviceClient.SendAsync(deviceId, commandMessage);
```

Task 3: Configure the DeviceControlConsole and the Simulator

1. In DeviceControlConsole, open App.config.
2. Set the IoTHubConnectionString appSetting to have a value of the connection string for the service policy to your IoT Hub. (Recall you can get this from the Azure Portal IoT Hub blade, Shared access policies, and then select the policy.)
3. Save the file.
4. In Simulator, open App.config.
5. Set the IoTHubSenderConnectionString appSetting to have a value of the connection string for the device policy to your IoT Hub.
6. Set the IoTHubManagerConnectionString appSetting to have a value of the connection string for the iothubowner policy to your IoT Hub.
7. Save the file.
8. Build the Simulator and DeviceControlConsole projects.
9. In Solution Explorer, right-click Solution Vending Machines and select Set StartUp Projects.

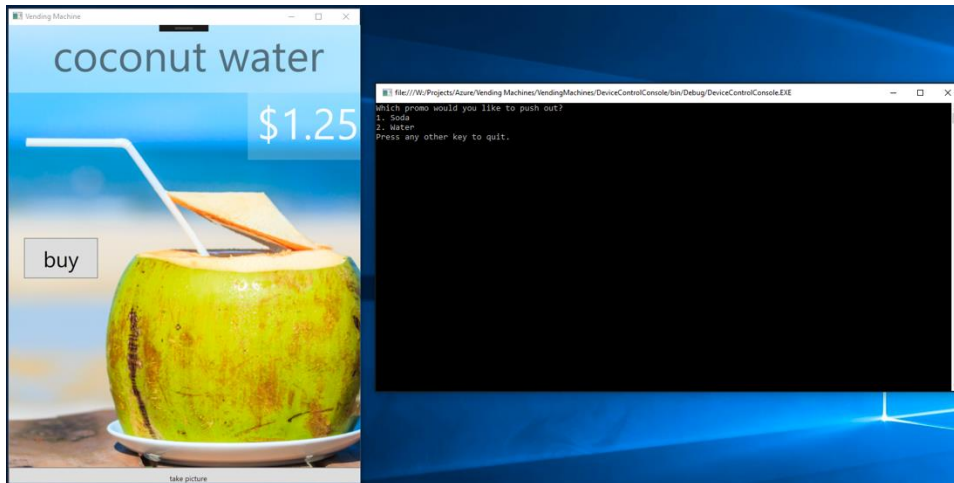


10. In the dialog, select the Multiple startup projects option and ensure that Action is set to Start for both DeviceControlConsole and Simulator.

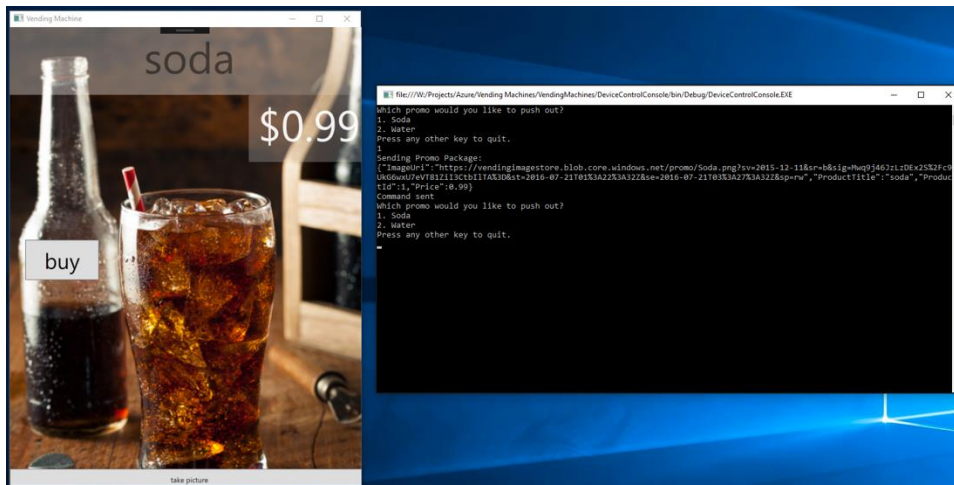


11. Select OK.

12. From the Debug menu, choose Start without Debugging.
13. Wait for both the Vending Machine Simulator and the DeviceControlConsole to appear.



14. In the DeviceControlConsole, press 1 to push the promotion for Soda.



15. Observe that the entire promotion surface of the vending machine changes (product name, price, and image).
16. Experiment sending the other promotion or toggling between promotions.
17. Experiment with making purchases and sending photos to verify the other functions still work with the new promoted products.

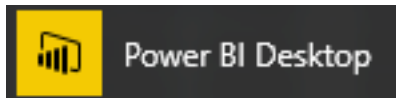
Exercise 6: Analytics with Power BI Desktop

Duration: 15 minutes

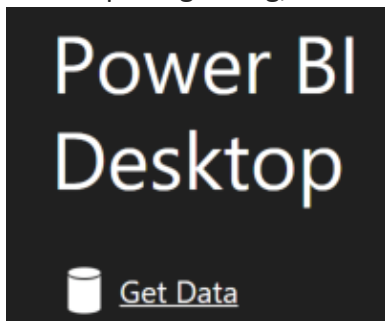
In this exercise, you will use Power BI Desktop to query purchase data from the in-memory table of SQL DB and visualize the result.

Task 1: Build the query and create the visualization

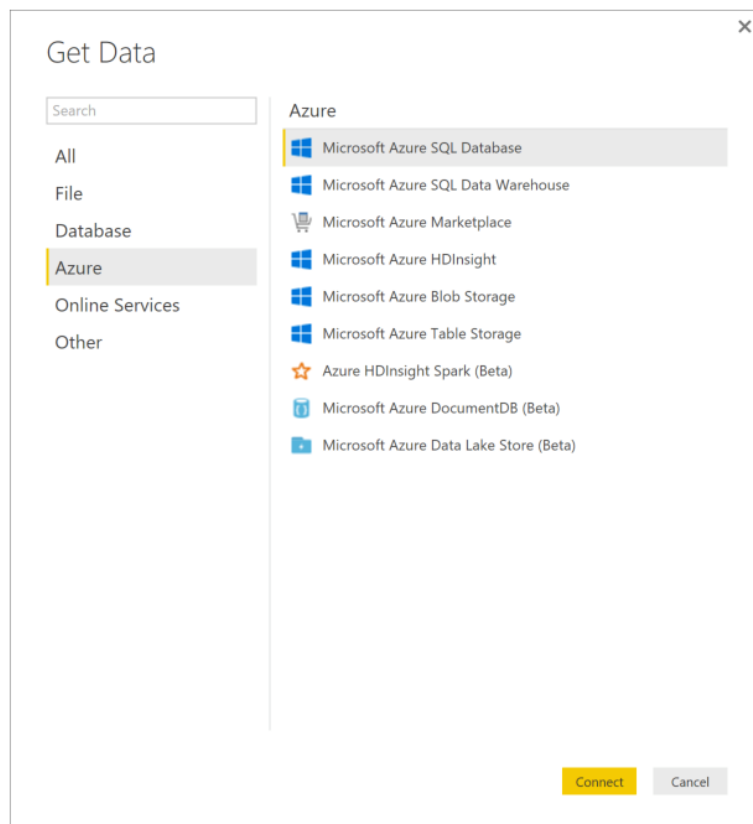
1. From your Start menu, open Power BI Desktop.



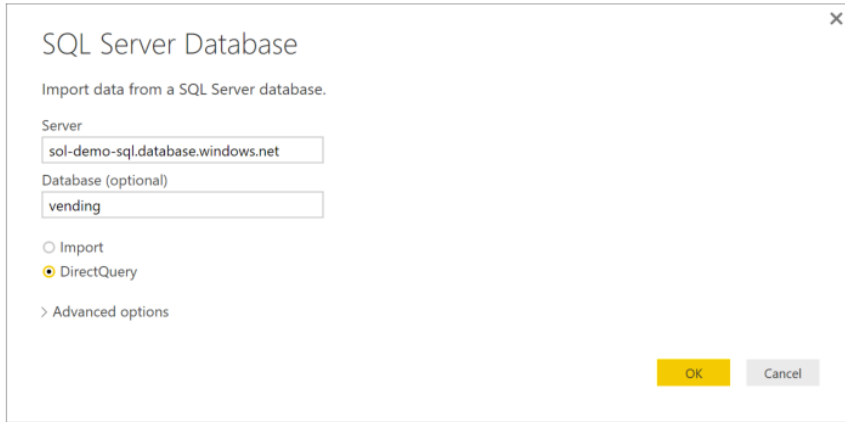
2. In the opening dialog, select Get Data.



3. In the Get Data dialog, select Azure in the categories list and then Microsoft Azure SQL Database.



4. Select Connect.
5. In the dialog, enter the name of your SQL Server (e.g., myserver.database.windows.net), the name of your vending database, and select the DirectQuery option. Select OK.



SQL Server Database

Import data from a SQL Server database.

Server
sol-demo-sql.database.windows.net

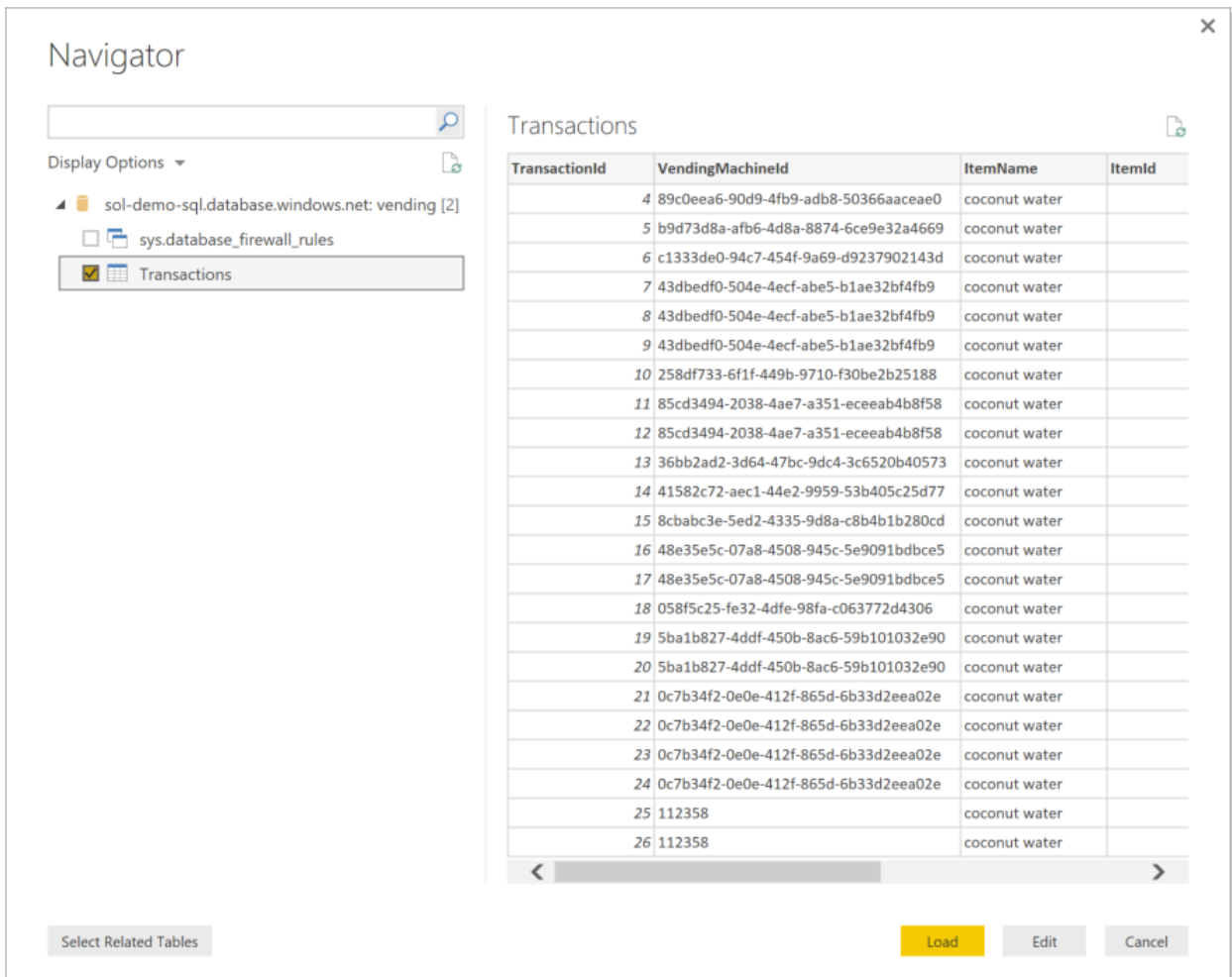
Database (optional)
vending

☐ Import
☒ DirectQuery

> Advanced options

OK Cancel

6. On the next screen, select the Database tab on the left.
7. Provide your SQL username and password and select Connect.
8. In the Navigator dialog, select the box next to Transactions.



Navigator

Display Options ▾

sol-demo-sql.database.windows.net: vending [2]

☐ sys.database_firewall_rules

☒ Transactions

Transactions

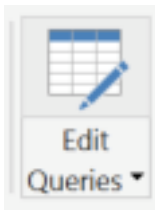
TransactionId	VendingMachineId	ItemName	ItemId
4	89c0eea6-90d9-4fb9-adb8-50366aaceae0	coconut water	
5	b9d73d8a-afb6-4d8a-8874-6ce9e32a4669	coconut water	
6	c1333de0-94c7-454f-9a69-d9237902143d	coconut water	
7	43dbedf0-504e-4ecf-abe5-b1ae32bf4fb9	coconut water	
8	43dbedf0-504e-4ecf-abe5-b1ae32bf4fb9	coconut water	
9	43dbedf0-504e-4ecf-abe5-b1ae32bf4fb9	coconut water	
10	258df733-6f1f-449b-9710-f30be2b25188	coconut water	
11	85cd3494-2038-4ae7-a351-ecceab4b8f58	coconut water	
12	85cd3494-2038-4ae7-a351-ecceab4b8f58	coconut water	
13	36bb2ad2-3d64-47bc-9dc4-3c6520b40573	coconut water	
14	41582c72-aec1-44e2-9959-53b405c25d77	coconut water	
15	8cbabc3e-5ed2-4335-9d8a-c8b4b1b280cd	coconut water	
16	48e35e5c-07a8-4508-945c-5e9091bdbce5	coconut water	
17	48e35e5c-07a8-4508-945c-5e9091bdbce5	coconut water	
18	058f5c25-fe32-4dfe-98fa-c063772d4306	coconut water	
19	5ba1b827-4ddf-450b-8ac6-59b101032e90	coconut water	
20	5ba1b827-4ddf-450b-8ac6-59b101032e90	coconut water	
21	0c7b34f2-0e0e-412f-865d-6b33d2eea02e	coconut water	
22	0c7b34f2-0e0e-412f-865d-6b33d2eea02e	coconut water	
23	0c7b34f2-0e0e-412f-865d-6b33d2eea02e	coconut water	
24	0c7b34f2-0e0e-412f-865d-6b33d2eea02e	coconut water	
25	112358	coconut water	
26	112358	coconut water	

Select Related Tables

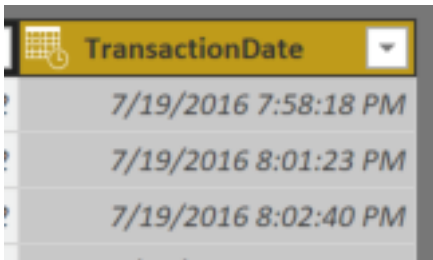
Load Edit Cancel

9. Select Load.

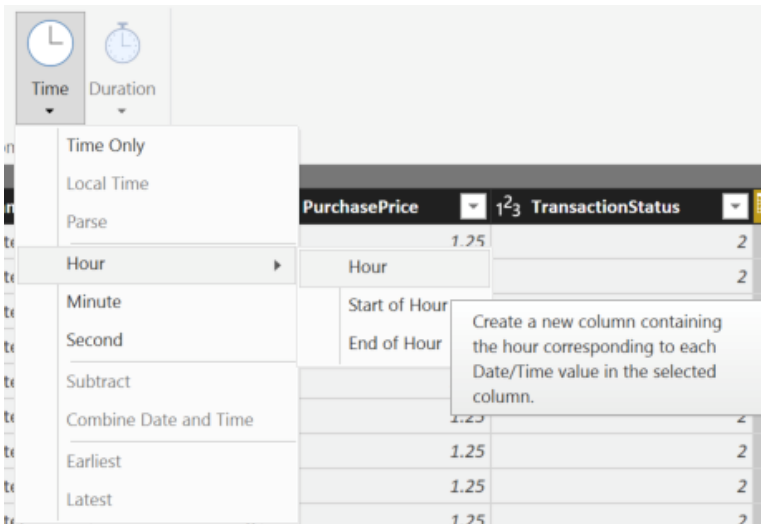
10. In the Ribbon, select Edit Queries.



11. In the Query Editor, select the Transaction Date column header to select the column.

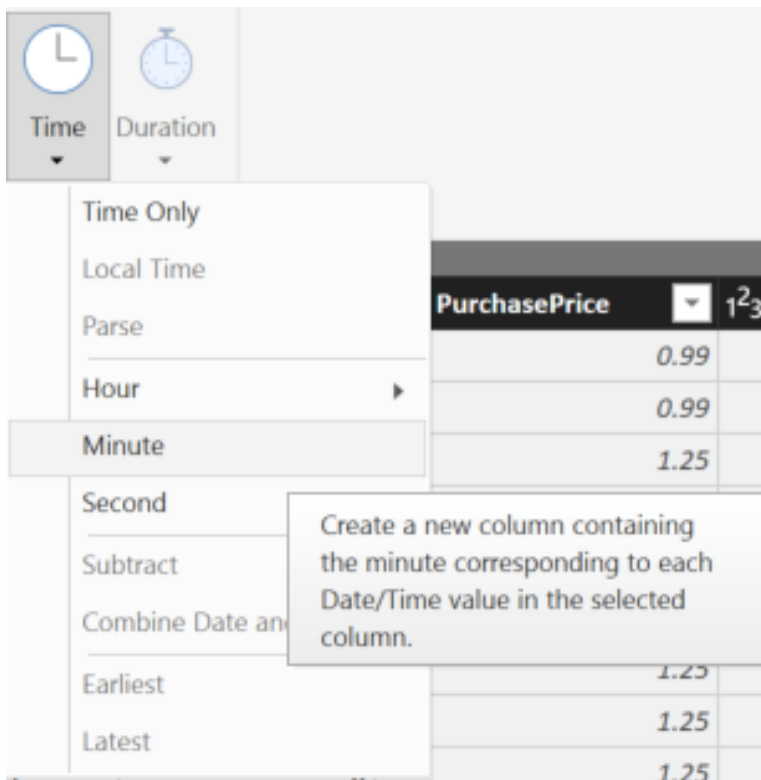


12. In the Ribbon, select the Add Column tab and select Time, Hour, Hour.



13. Select the TransactionDate column again.

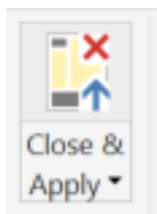
14. In the Ribbon, select Time, Minute.



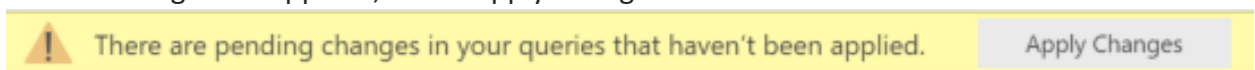
15. Select the TransactionDate one more time.

16. In the Ribbon, select Time, Second.

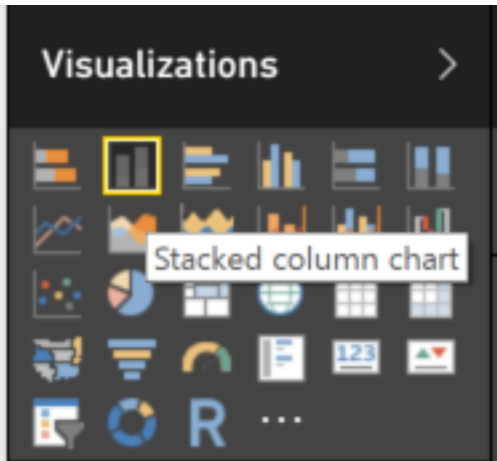
17. In the Ribbon, on the Home tab, select Close & Apply.



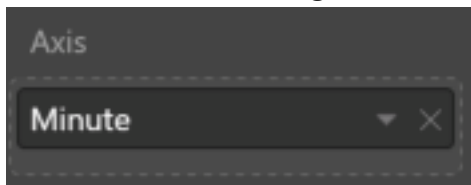
18. In the message that appears, select Apply Changes.



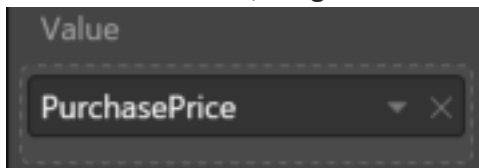
19. In the Visualizations, select Stacked column chart.



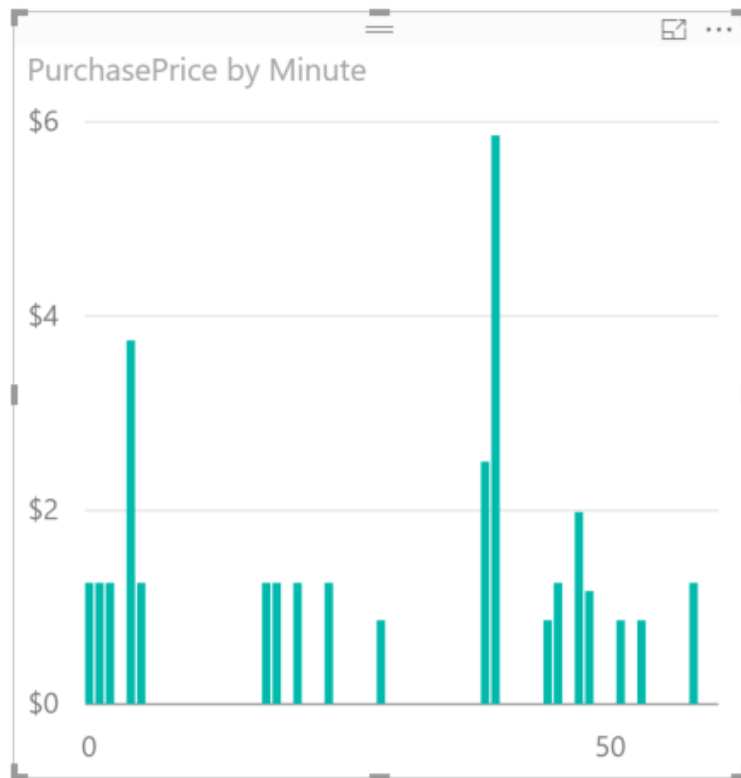
20. From the Fields list, drag the minute field over to the axis property.



21. From the Fields list, drag the PurchasePrice over to the value property.



22. Your completed visualization summarizing the most profitable minutes in each hour should appear as follows:

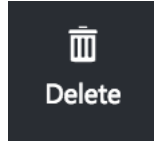


Exercise 7: Cleanup

Duration: 5 minutes

In this exercise, attendees will de-provision any Azure resources that were created in support of the hackathon.

1. Delete the Resource Group you created.
 - a. From the Portal, navigate to the blade of your Resource Group and select **Delete** in the **command bar** at the top.



2. When prompted to confirm the deletion, follow the prompts.