



Microsoft Cloud Workshop

Intelligent vending machines

Hands-on lab step-by-step

March 2018

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

Contents

Intelligent vending machines hands-on lab step-by-step	1
Abstract and learning objectives.....	1
Overview	1
Solution architecture.....	2
Requirements	2
Before the hands-on lab.....	3
Task 1: Provision an R Server on HDInsight with Spark cluster.....	3
Task 2: Setup a lab virtual machine (VM).....	7
Task 3: Install Power BI Desktop on the lab VM	10
Task 4: Prepare an SSH client.....	16
Task 5: Install R Tools for Visual Studio 2017	17
Exercise 1: Environment setup.....	20
Task 1: Download and open the vending machines starter project.....	20
Task 2: Provision IoT Hub.....	20
Task 3: Create Microsoft Machine Learning Server on Linux.....	24
Task 4: Create Storage Account	28
Task 5: Provision Cognitive Services Face API	32
Task 6: Provision SQL Database	34
Exercise 2: Create Dynamic Pricing Model	38
Task 1: Create a model locally	38
Task 2: Try a prediction locally.....	42
Task 3: Create the model in R Server on HDInsight.....	42
Task 4: Create predictive service in R Server Operationalization	45
Exercise 3: Implement dynamic pricing.....	50
Task 1: Implement photo uploads to Azure Storage	50
Task 2: Invoke Face API.....	51
Task 3: Invoke pricing model	52
Task 4: Configure the Simulator.....	53
Task 5: Test dynamic pricing in Simulator	53
Exercise 4: Implement purchasing	57
Task 1: Create the transactions table.....	57
Task 2: Configure the Simulator.....	58
Task 3: Test purchasing.....	59
Exercise 5: Implement device command and control	61
Task 1: Listen for control messages	61
Task 2: Send control messages.....	62
Task 3: Configure the DeviceControlConsole and the Simulator	62

Exercise 6: Analytics with Power BI Desktop.....	67
Task 1: Build the query and create the visualization	67
After the hands-on lab	73
Task 1: Delete the resource group	73

Intelligent vending machines hands-on lab step-by-step

Abstract and learning objectives

In this workshop, attendees will implement an IoT solution for intelligent vending machines, leveraging facial feature recognition and Azure Machine Learning, to gain a better understanding of building cloud-based machine learning app, and real-time analytics with SQL Database in-memory and columnar indexing.

In addition, attendees will learn to:

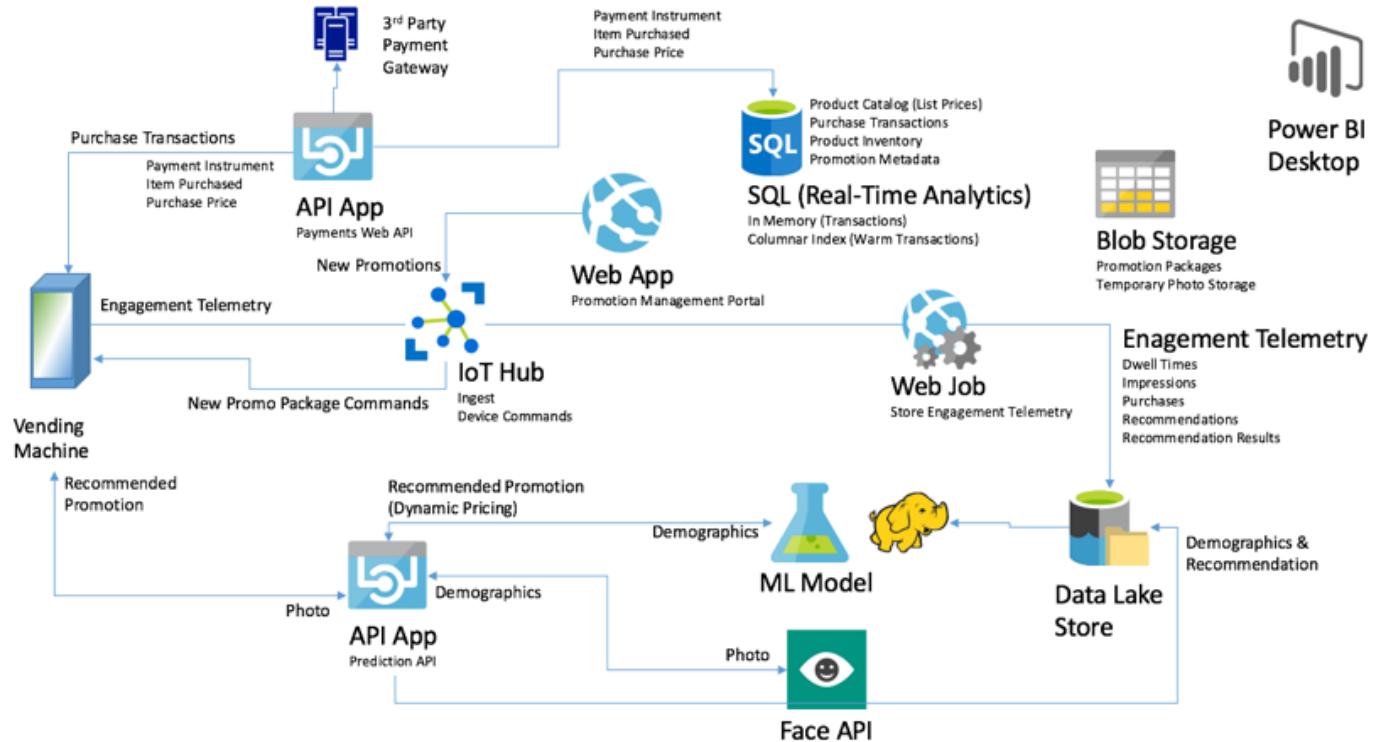
- Enable real-time analytics with SQL Database in-memory and columnar indexing
- Implement distributed machine learning with R Server for HDInsight & Microsoft R Server Operationalization
- Perform facial image processing
- Wrangle data in Power BI Desktop

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.

Solution architecture

Below are diagrams of the solution architecture you will build in this lab. Please study this carefully, so you understand the whole of the solution as you are working on the various components.



Requirements

- Microsoft Azure subscription must be pay-as-you-go or MSDN.
 - Trial subscriptions will not work.
- A virtual machine configured with:
 - Visual Studio Community 2017 15.6 or later
 - Azure SDK 2.9 or later (Included with Visual Studio 2017)
 - [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 [Before the Hands-on Lab](#)).

Before the hands-on lab

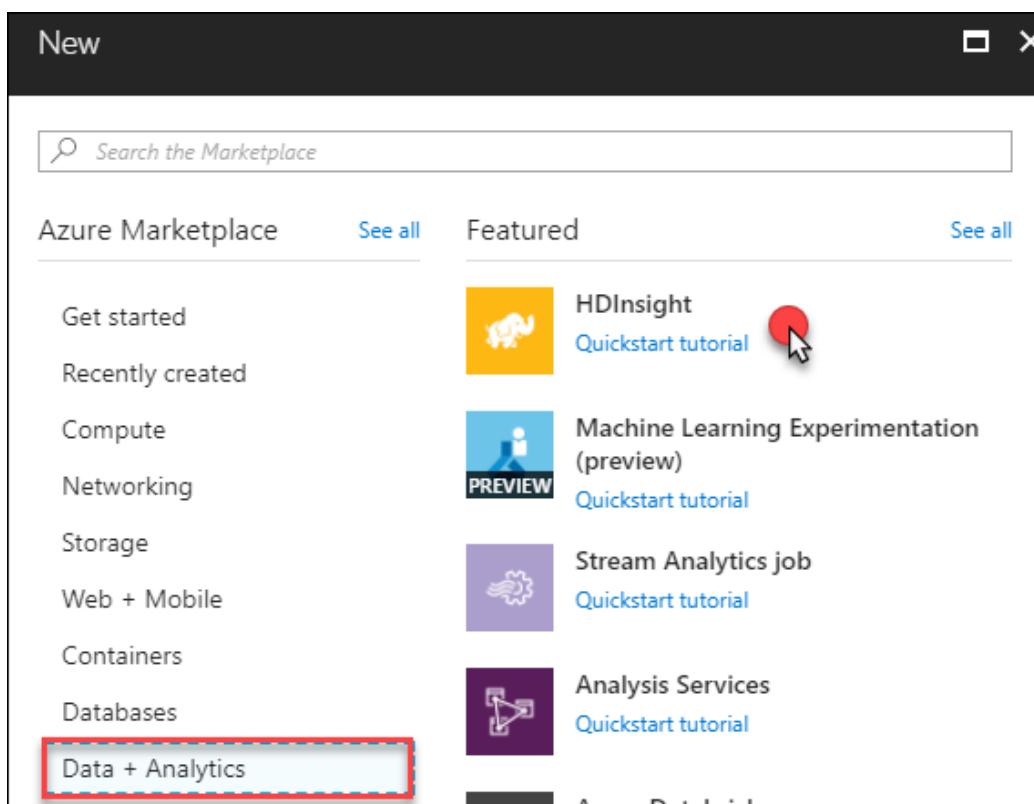
Duration: 75 minutes

Synopsis: In this exercise, you will set up your environment for use in the rest of the hands-on lab. You should follow all the steps provided in the Before the Hands-on Lab section to prepare your environment before attending the hackathon.

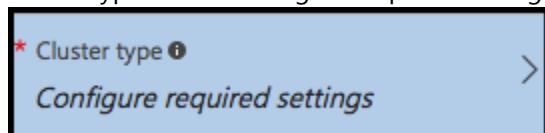
Task 1: Provision an R Server on HDInsight with Spark cluster

Using the Azure Portal, provision a new HDInsight cluster.

1. Open a browser and go to <https://portal.azure.com>.
2. Select **+Create a resource**, select **Data + Analytics, HDInsight**.

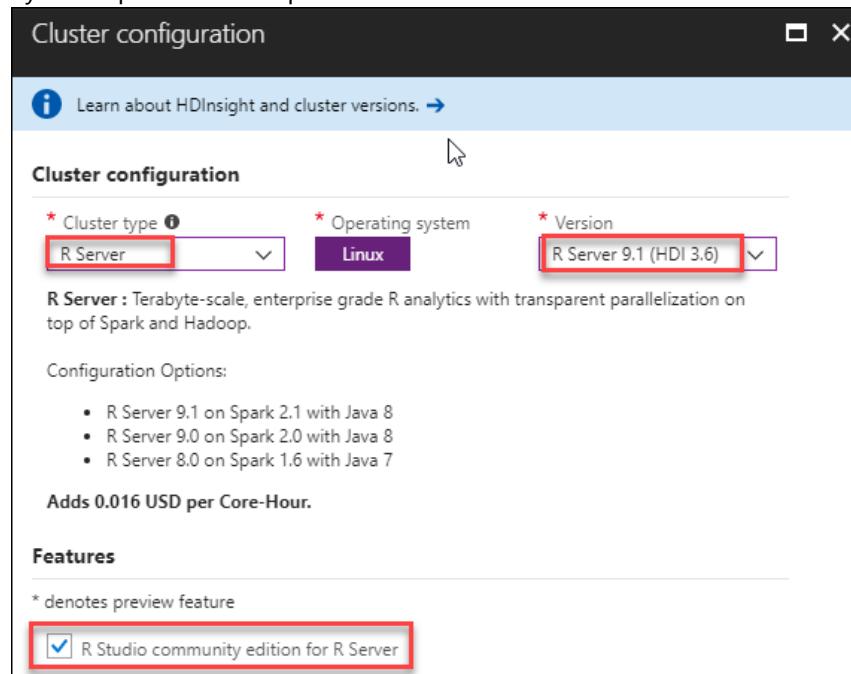


3. On the Basics blade, enter the following settings:
 - a. Cluster name: Enter a **unique name** (verified by the green checkmark).
 - b. Subscription: Select the Azure subscription into which you want to deploy the cluster.
 - c. Cluster type: Select Configure required settings.

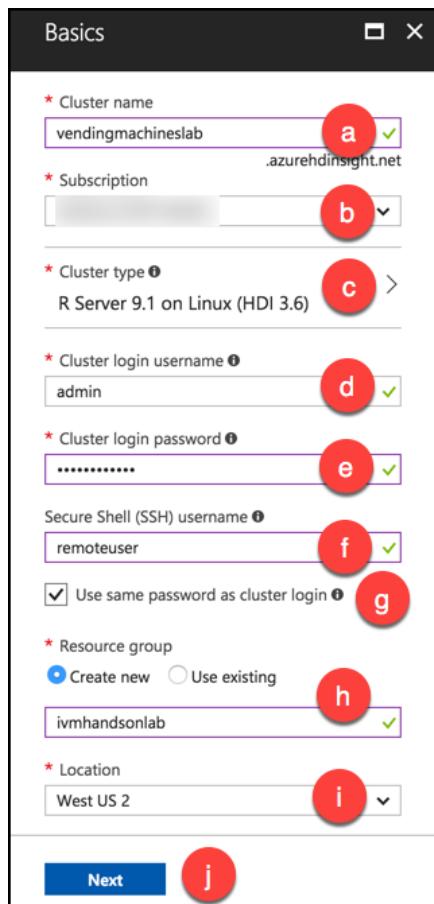


- i. On the Cluster configuration blade, set the Cluster type to **R Server** and the Version to **R Server 9.1 (HDI 3.6)**.

- ii. Check the box next to **R Studio community edition for R Server**. Note that the Operating System option for the Spark cluster is fixed to Linux.

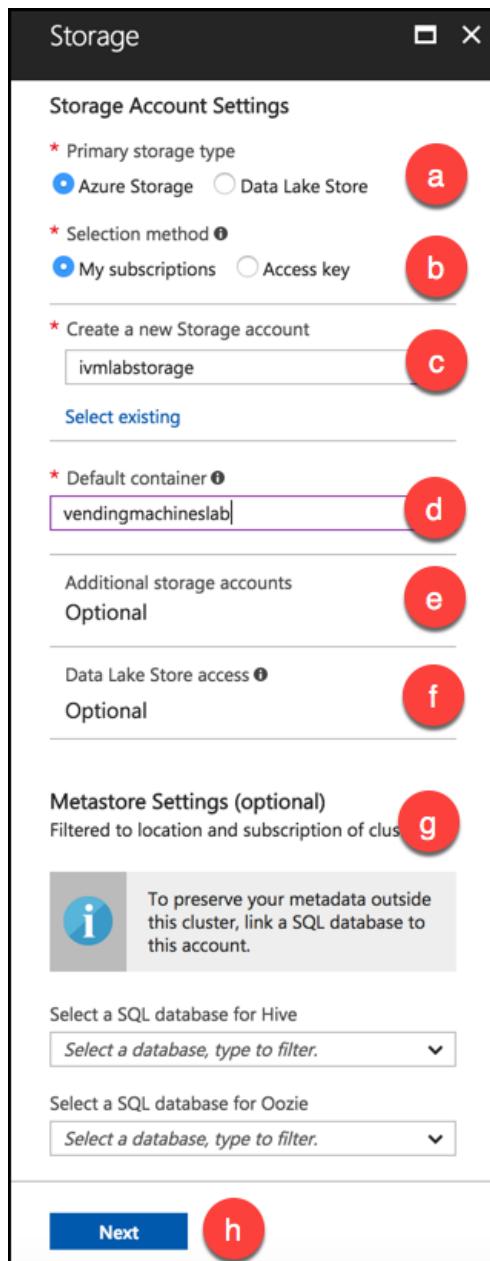


- iii. Click **Select** to close the Cluster configuration blade.
- d. Cluster login username: Leave as **admin**.
- e. Cluster login password: Enter **Password.1!!** for the admin password.
- f. Secure Shell (SSH) username: Set to **remoteuser** (this is required).
- g. Use same password as cluster login: Ensure the checkbox is checked.
- h. Resource group: Select the **Create new** radio button, and enter **ivmhandsonlab** for the resource group name.
- i. Select the desired location from the dropdown list.
- j. Select **Next** to move on to the storage settings.



4. On the Storage blade:

- a. Primary storage type: Leave set to **Azure Storage**.
- b. Selection Method: Leave set to **My subscriptions**.
- c. Select a Storage account: Select **Create new**, and enter a name for the storage account, such as ivmlabstorage.
- d. Default container: Set to the **name of your cluster**.
- e. Additional storage accounts: Leave unconfigured.
- f. Data Lake Store access: Leave unconfigured.
- g. Metastore Settings: Leave blank.
- h. Select **Next** to move on to the Cluster summary.

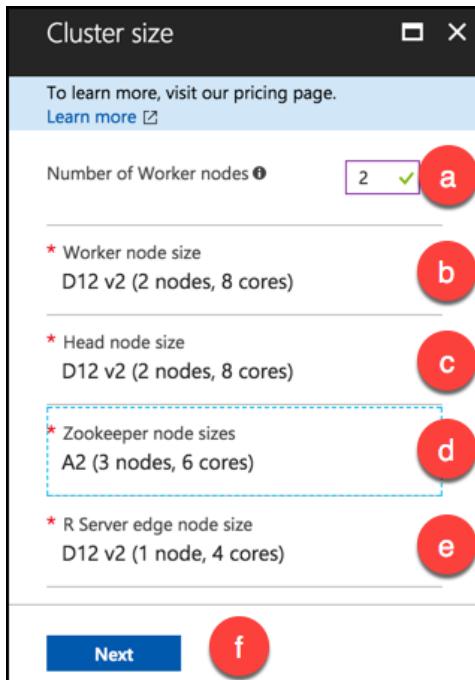


5. On the Cluster Summary blade, select Edit next to Cluster Size.

Cluster size (Edit)	
Nodes	Head (2 x D12 v2), Worker (4 x D4 v2), Zookeeper (3 x A2) RServer (1 x D4 v2)

6. On the Cluster size blade, enter the following:

- a. Number of worker nodes: Enter **2**.
- b. Select **Worker node size**, and select **D12 v2**, then click **Select**.
- c. Select **Head node size**, and select **D12 v2**, then click **Select**.
- d. Leave the Zookeeper node sizes set to A2.
- e. Select **R-Server edge node size**, and select **D12 v2**, then click **Select**.
- f. Select **Next**.



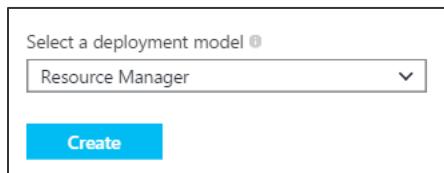
7. Select **Next** on the Advanced settings blade to move to the Cluster summary blade.
8. Select **Create** on the Cluster summary blade to create the cluster.
9. It will take approximately 20 minutes to create your cluster. You can move on to the steps below while the cluster is provisioning.

Task 2: Setup a lab virtual machine (VM)

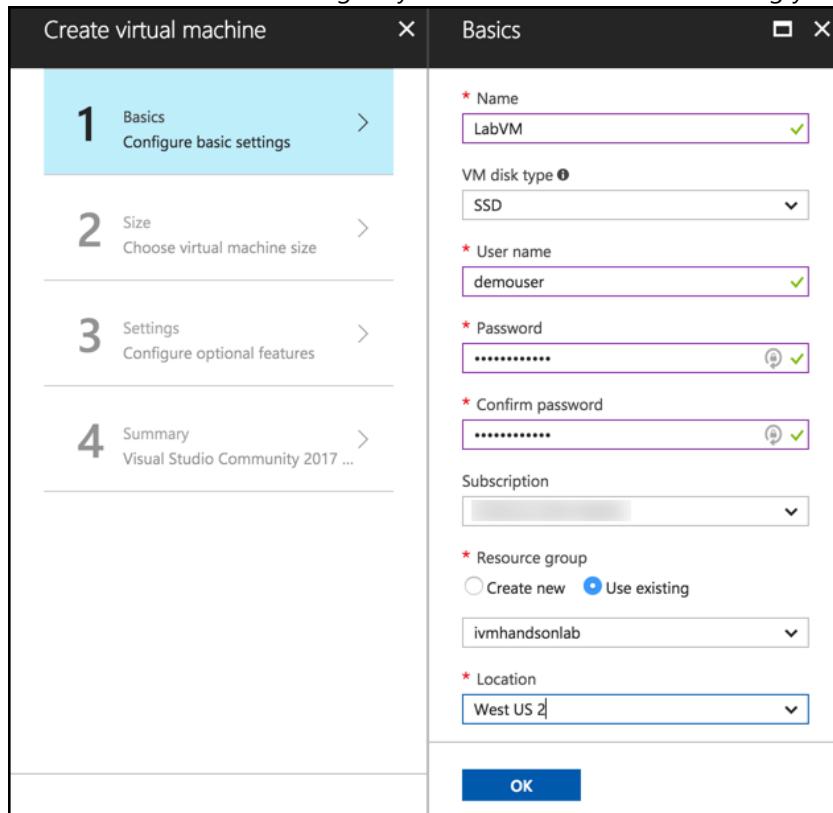
1. In the [Azure Portal](#), select **+Create a resource**, then type "Visual Studio" into the search bar. Select Visual Studio Community 2017 on Windows Server 2016 (x64) from the results.

NAME	PUBLISHER	CATEGORY
Visual Studio Community 2017 (latest release) on Windows Server 2016 (x64)	Microsoft	Compute

2. On the blade that comes up, at the bottom, ensure the deployment model is set to Resource Manager and select Create.



3. Set the following configuration on the Basics tab.
 - a. Name: Enter **LabVM**.
 - b. VM disk type: Select **SSD**.
 - c. User name: Enter **demouser**
 - d. Password: Enter **Password.1!!**
 - e. Subscription: Select the same subscription you used to create your cluster in [Task 1](#).
 - f. Resource Group: Select Use existing, and select the resource group you provisioned while creating your cluster in Task 1.
 - g. Location: Select the same region you used in Task 1 while creating your cluster.



4. Select **OK** to move to the next step.

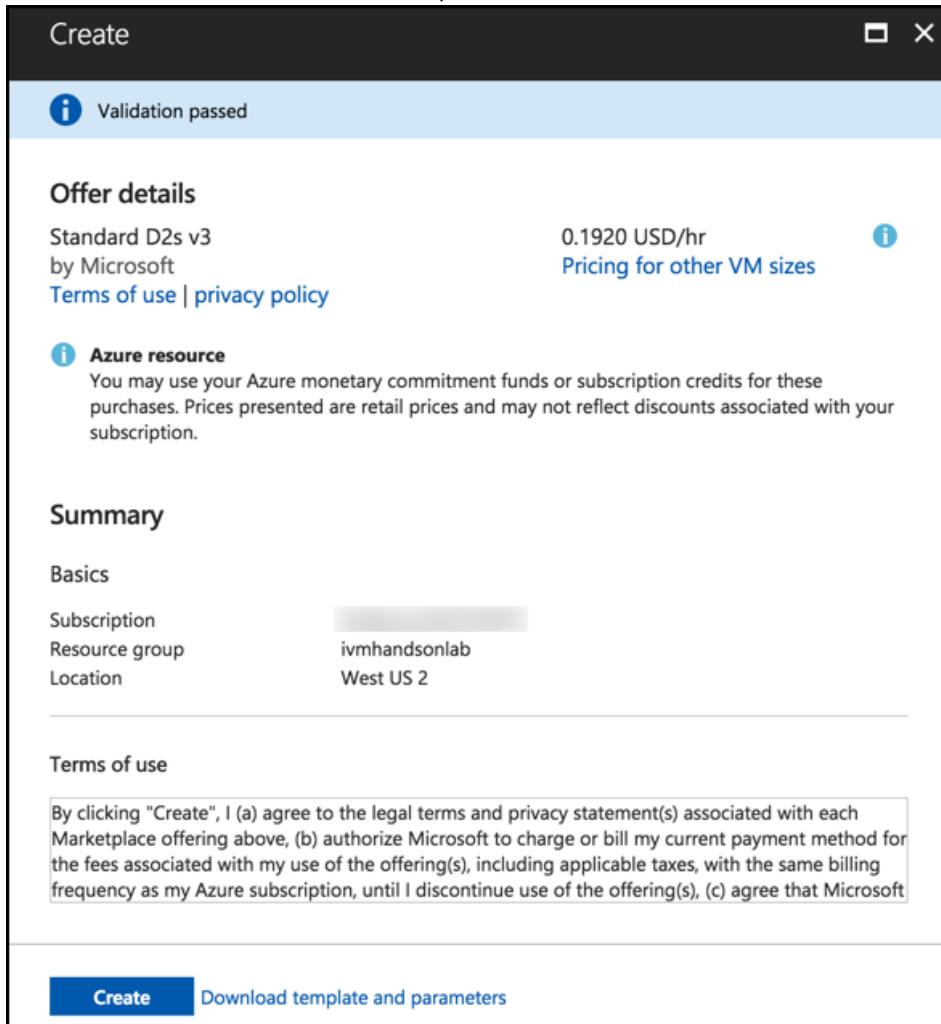
5. On the Choose a size blade, ensure the Supported disk type is set to SSD, and select View all. This machine won't be doing much heavy lifting, so selecting DS2_V3 Standard is a good baseline option.

Supported disk type	Minimum vCPUs	Minimum memory (GiB)
SSD	1	0
★ Recommended View all		
D2S_V3 Standard	D4S_V3 Standard	D8S_V3 Standard
2 vCPUs	4 vCPUs	8 vCPUs
8 GB	16 GB	32 GB
4 Data disks	8 Data disks	16 Data disks
4000 Max IOPS	8000 Max IOPS	16000 Max IOPS
16 GB Local SSD	32 GB Local SSD	64 GB Local SSD
Premium disk support	Premium disk support	Premium disk support
Load balancing	Load balancing	Load balancing
142.85 USD/MONTH (ESTIMATED)	285.70 USD/MONTH (ESTIMATED)	571.39 USD/MONTH (ESTIMATED)

Select

6. Click **Select** to move on to the Settings blade.
7. Accept all the default values on the Settings blade, and Select **OK**.

8. Select Create on the Create blade to provision the virtual machine.



9. It may take 10+ minutes for the virtual machine to complete provisioning.

Task 3: Install Power BI Desktop on the lab VM

1. Connect to the **LabVM**. (If you are already connected to your Lab VM, skip to Step 7.)

2. From the left side menu in the Azure portal, click on Resource groups, then enter your resource group name into the filter box, and select it from the list.

The screenshot shows the Azure Resource Groups blade. On the left, there's a navigation menu with options like New, Dashboard, All resources, Resource groups (which is highlighted with a red box), and App Services. The main area is titled "Resource groups" and shows a search bar with "ivm" typed in. Below the search bar, it says "Subscriptions: - Don't see a subscription". A list of resources is shown with one item: "ivmhandsonlab" (highlighted with a red box). The list includes columns for NAME, TYPE, LOCATION, and three dots for more options.

3. Next, select your lab virtual machine, LabVM, from the list.

The screenshot shows the Azure Resource List blade. It lists various resources under the "ivmhandsonlab" resource group. The "LabVM" resource is highlighted with a red box. The table has columns for RESOURCE, TYPE, LOCATION, and three dots for more options. Other resources listed include ivmhandsonlab-vnet, ivmlabstorage, LabVM_OsDisk_1, labvm789, LabVM-ip, LabVM-nsg, and vendingmachineslab.

RESOURCE	TYPE	LOCATION	
ivmhandsonlab-vnet	Virtual network	West US 2	...
ivmlabstorage	Storage account	West US 2	...
LabVM	Virtual machine	West US 2	...
LabVM_OsDisk_1_907fa90bf6a74507b518650...	Disk	West US 2	...
labvm789	Network interface	West US 2	...
LabVM-ip	Public IP address	West US 2	...
LabVM-nsg	Network security group	West US 2	...
vendingmachineslab	HDInsight cluster	West US 2	...

4. On your Lab VM blade, select Connect from the top menu.

The screenshot shows the Lab VM blade for the "LabVM" resource. At the top, there's a menu bar with "Connect" (highlighted with a red box), Start, Restart, Stop, Capture, Move, Delete, and Refresh. Below the menu, there's information about the resource group ("ivmhandsonlab") and the VM itself ("Computer name: LabVM, Operating system: Windows, Size: Standard D2s v3 (2 vcpus, 8 GB memory)").

5. Download and open the RDP file.

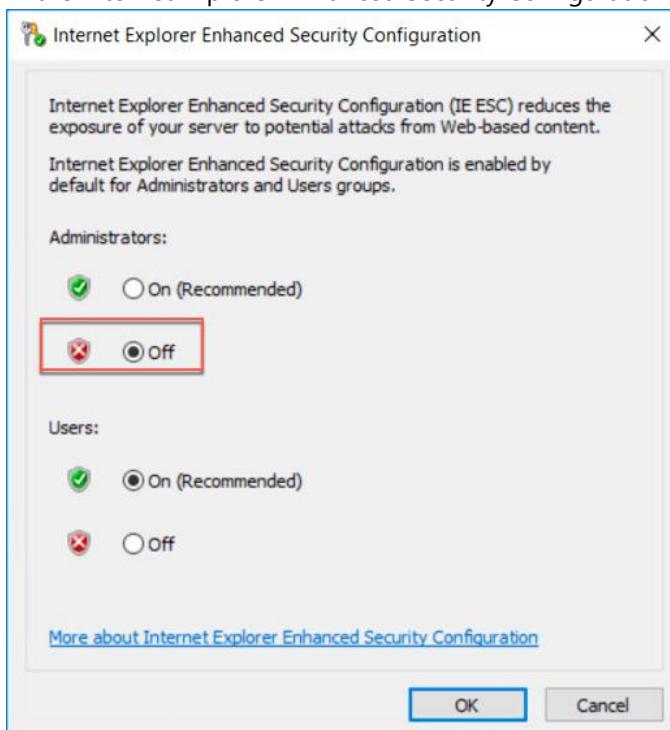
6. Select **Connect**, and enter the following credentials (or the non-default credentials if you changed them):

- User name: **demouser**
- Password: **Password.1!!**

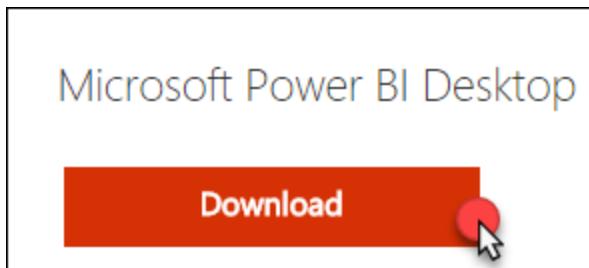
7. Once logged in, launch the **Server Manager**. This should start automatically, but you can access it via the Start menu if it does not start.
8. Select **Local Server**, then select **On** next to **IE Enhanced Security Configuration**.

PROPERTIES For LabVM			
Computer name Workgroup	LabVM WORKGROUP	Last installed updates Windows Update Last checked for updates	10/11/2017 6:20 PM Download updates only, using Windows Update 10/11/2017 6:20 PM
Windows Firewall Remote management Remote Desktop NIC Teaming Ethernet	Private: On Enabled Enabled Disabled IPv4 address assigned by DHCP, IPv6 enabled	Windows Defender Feedback & Diagnostics IE Enhanced Security Configuration	Real-Time Protection: On Settings On
		Time zone Product ID	(UTC) Coordinated Universal Time 00376-40000-00000-AA947 (activated)

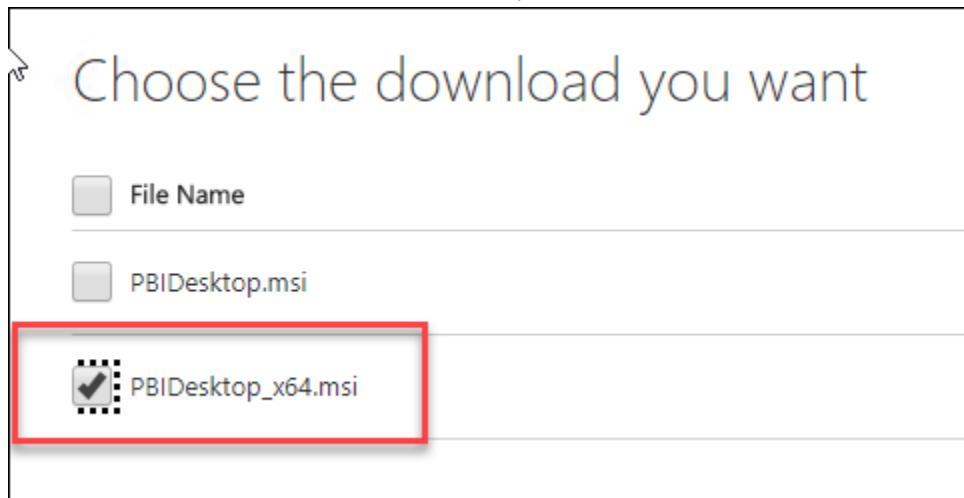
9. In the Internet Explorer Enhanced Security Configuration dialog, select **Off** under **Administrators**, then select **OK**.



10. Close the Server Manager.
11. In a web browser on the Lab VM navigate to the Power BI Desktop download page
<https://www.microsoft.com/en-us/download/details.aspx?id=45331>
12. Select the **Download Free** link in the middle of the page.



13. Select the **x64 bit version** of the download, then click **Next**.

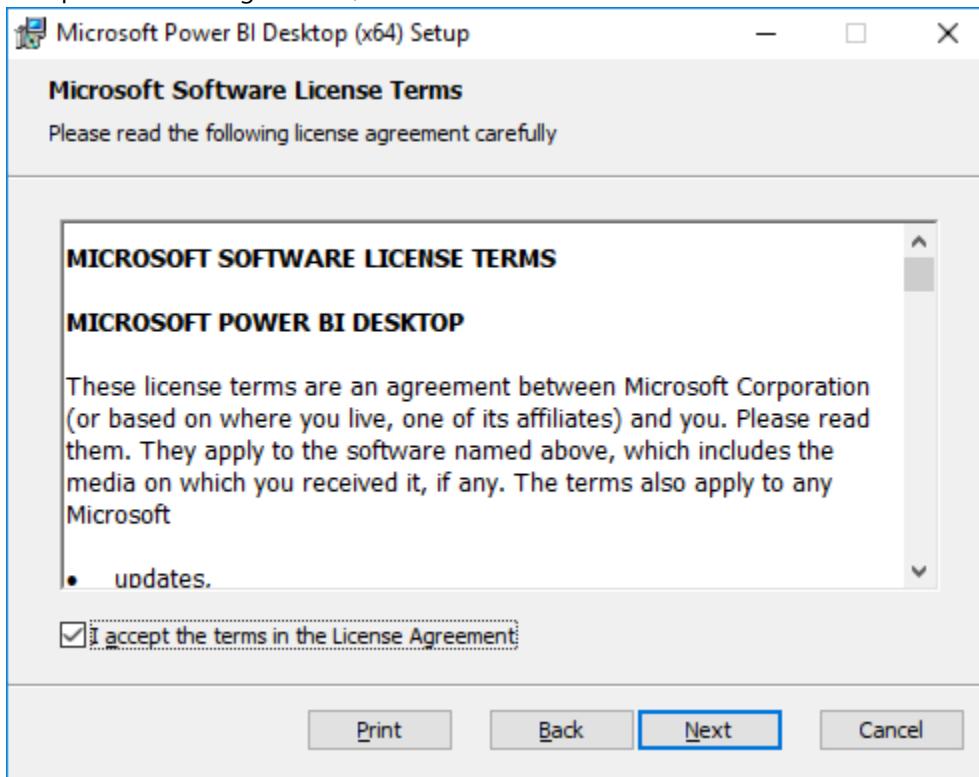


14. Run the installer once it downloads.

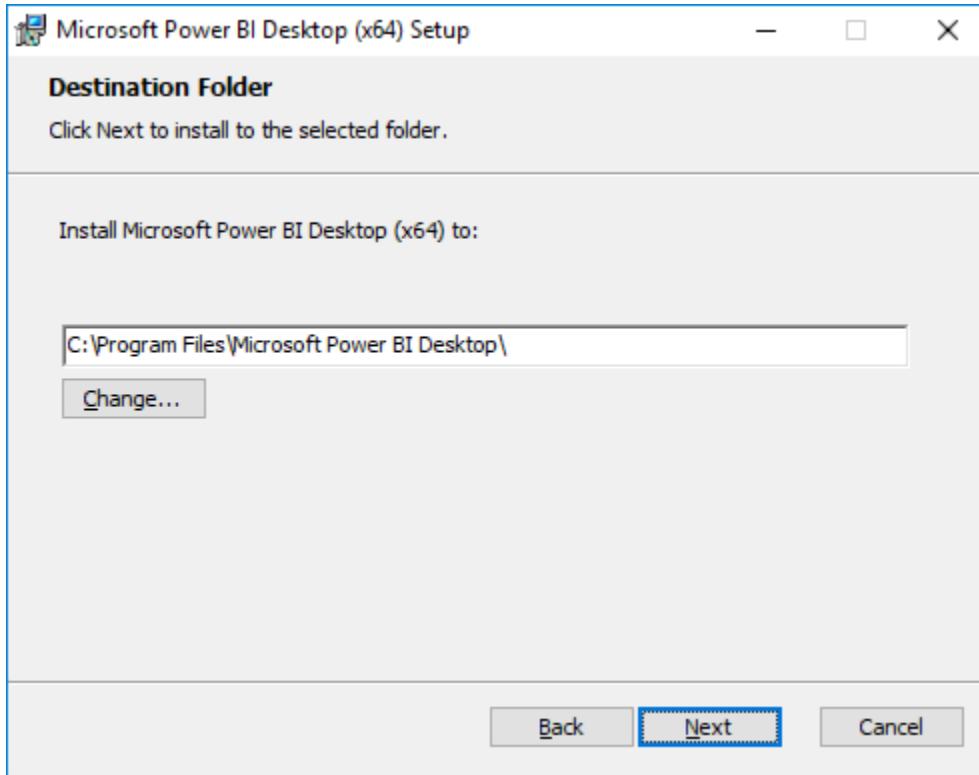
15. Select Next on the welcome screen.



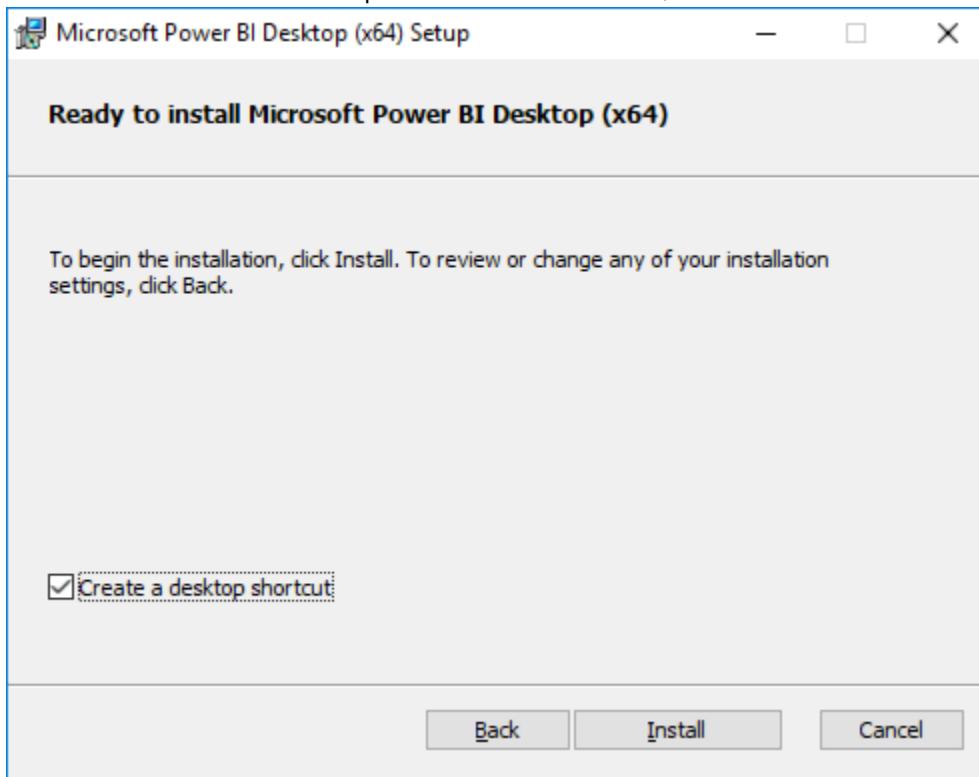
16. Accept the license agreement, and select Next.



17. Leave the default destination folder, and select Next.



18. Make sure the Create a desktop shortcut box is checked, and select Install.



19. Uncheck Launch Microsoft Power BI Desktop, and select Finish.



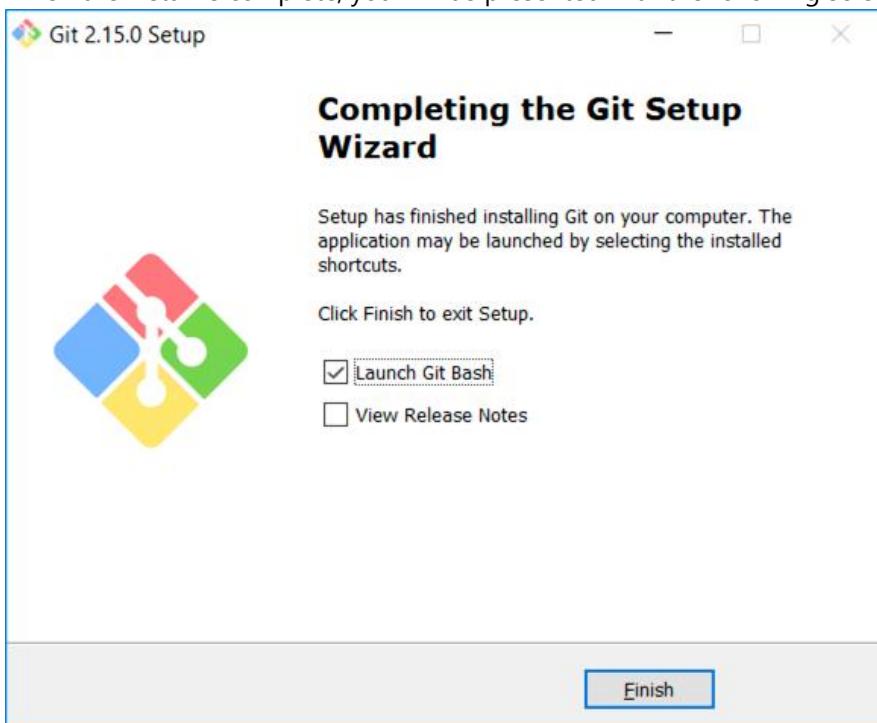
Task 4: Prepare an SSH client

In this task, you will download, install, and prepare the Git Bash SSH client that you will use to access your HDInsight cluster from your Lab VM.

1. On your Lab VM, open a browser, and navigate to <https://git-scm.com/downloads> to download Git Bash.



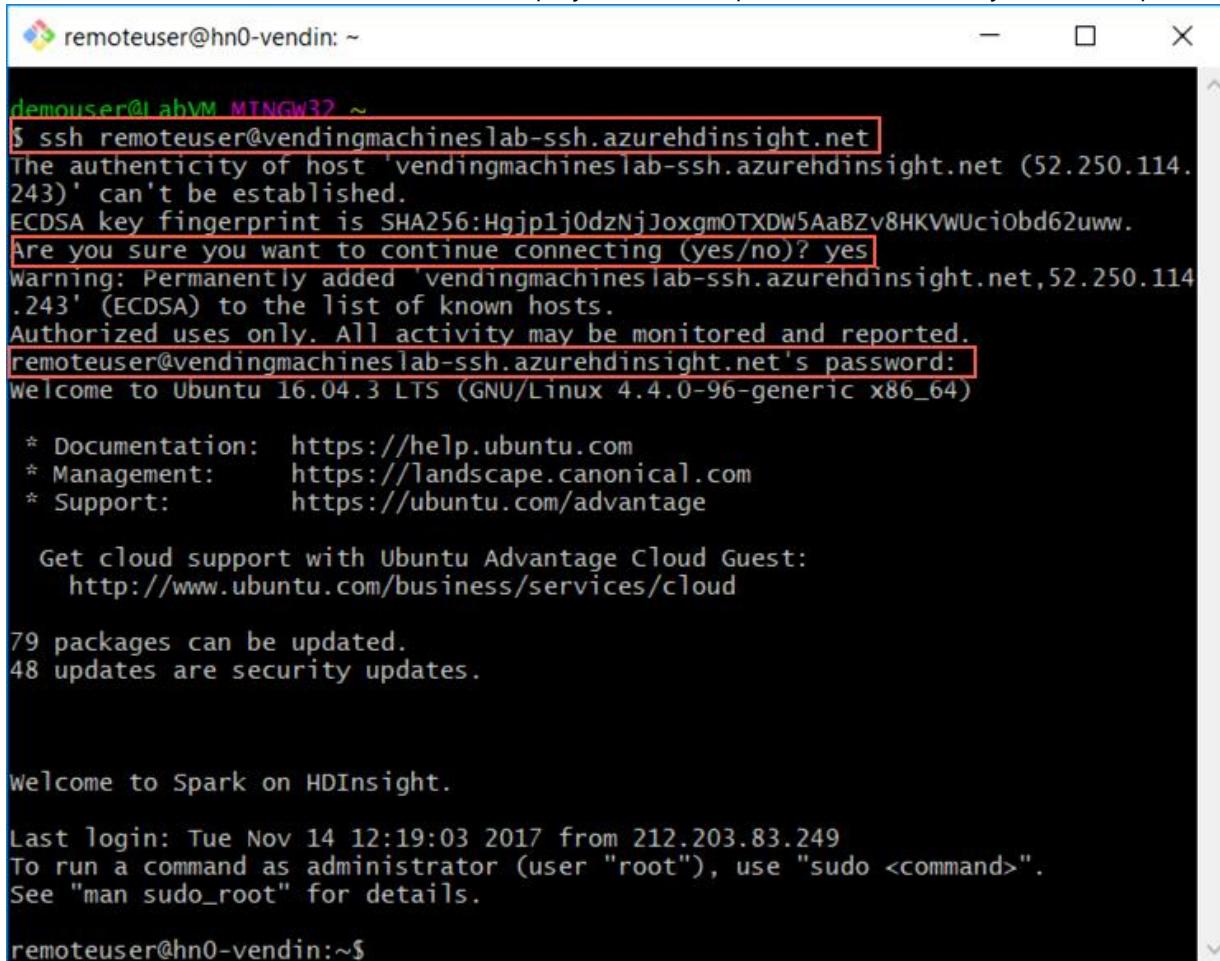
2. Select the download for your OS, and then select the Download button.
3. Run the downloaded installer, select Next on each screen to accept the defaults.
4. When the install is complete, you will be presented with the following screen:



5. Check the Launch Git Bash checkbox, and uncheck View Release Notes. Select **Finish**.
6. The Git Bash client should open in a new window.

- At the command prompt, enter **ssh remoteuser@<clustername>-ssh.azurehdinsight.net**, replacing <clustername> with the name of the HDInsight cluster created in [Task 1](#) above.

Note: You will need to wait for the cluster deployment to complete in Azure before you can complete this step.

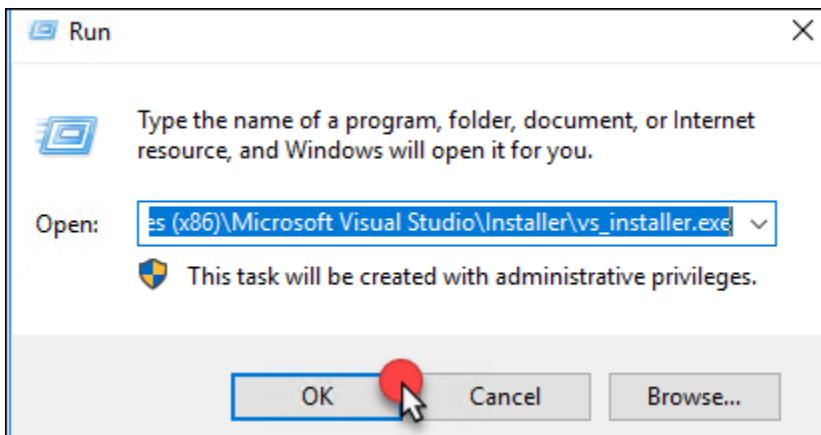


The screenshot shows a terminal window titled "remoteuser@hn0-vendin: ~". The user has run the command \$ ssh remoteuser@vendingmachineslab-ssh.azurehdinsight.net. The terminal displays the host key fingerprint and asks if the user wants to continue connecting (yes/no). The user responds with yes. It then shows a warning about adding the host to the list of known hosts and that authorized uses only. The password prompt for the user "remoteuser" is shown. The terminal then displays a welcome message for Ubuntu 16.04.3 LTS and provides documentation links for documentation, management, and support. It also mentions getting cloud support with Ubuntu Advantage Cloud Guest. Below this, it shows package update information: 79 packages can be updated, with 48 being security updates. Finally, it welcomes the user to Spark on HDInsight and provides login details. The session ends with the prompt remoteuser@hn0-vendin:~\$.

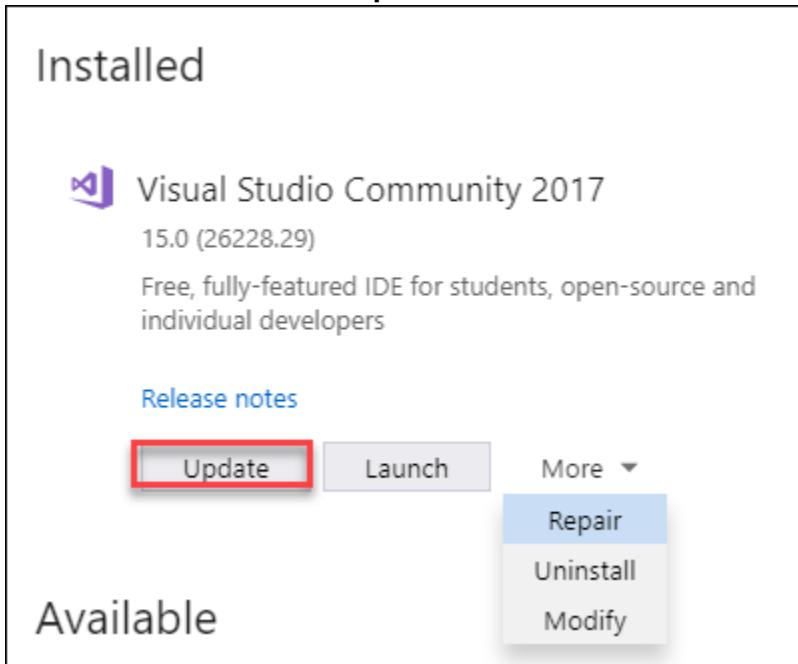
- Respond to any prompts in the SSH window, and enter the password for **remoteuser** when prompted.
- Use Git Bash for SSH during the hands-on lab for any instructions requiring an SSH connection. You can repeat these steps any time to re-connect.

Task 5: Install R Tools for Visual Studio 2017

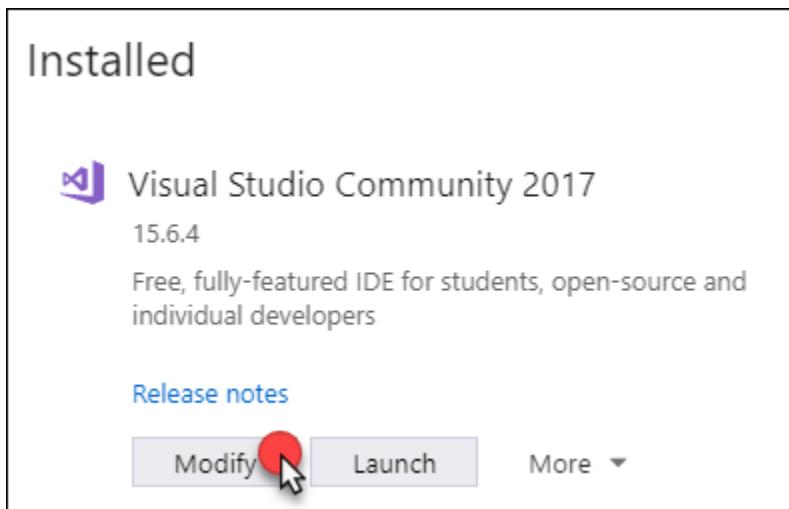
- Run the Visual Studio installer using Start, Run, C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe.



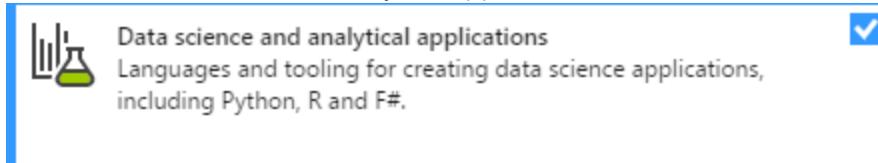
- Once the Installer starts, click **Update**. This will take some time to complete and require a reboot.



- Once the update is complete and the VM has restarted, run the installer again. Run the Visual Studio installer using Start, Run, C:\Program Files (x86)\Microsoft Visual Studio\Installer\vs_installer.exe.
- Next select the **Modify** option.



5. Select the Data science and analytical applications workload and then click Modify.



These steps should be completed prior to starting the rest of the Lab.

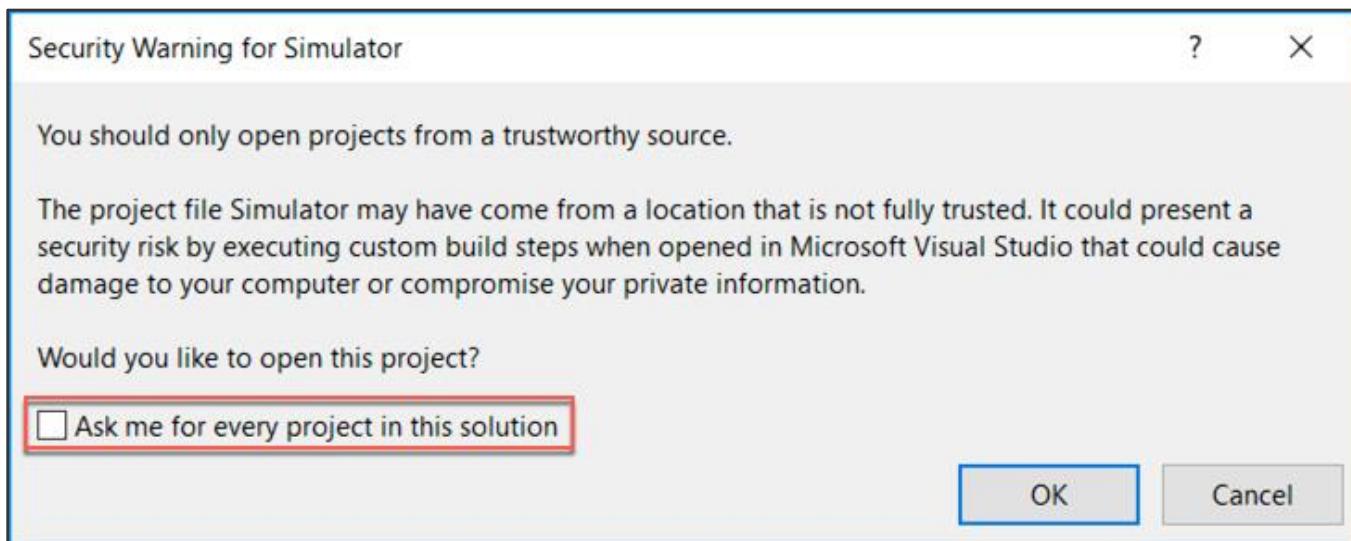
Exercise 1: Environment setup

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. From your LabVM, download the starter project from the following URL: <http://bit.ly/2w6t2qz>
2. Unzip the contents to the folder **C:\VendingMachines**.
3. Open **VendingMachines.sln** with Visual Studio 2017.
4. Sign in to Visual Studio or create an account, if prompted.
5. If the Security Warning for Simulator window appears, **uncheck Ask me for every project in this solution**, and select **OK**.



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.

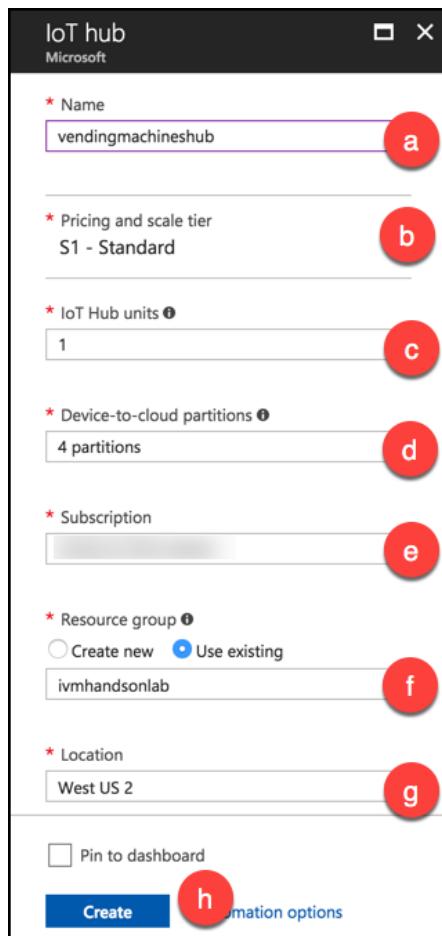
Task 2: Provision IoT Hub

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

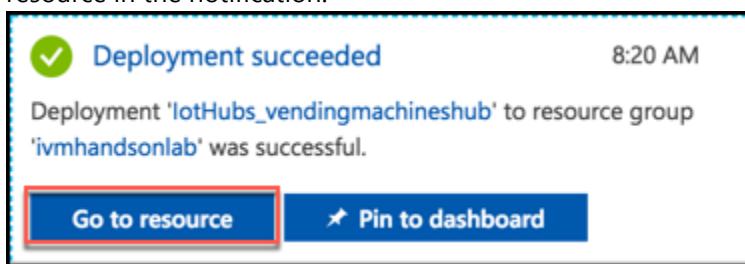
1. In your browser, navigate to the Azure Portal (<https://portal.azure.com>).
2. Select **+Create a resource**, then select Internet of Things, and select IoT Hub.

The screenshot shows the Azure Marketplace 'New' page. At the top, there is a search bar with the placeholder 'Search the Marketplace'. Below the search bar, there are two tabs: 'Azure Marketplace' and 'See all'. Underneath these tabs, there are several categories: 'Get started', 'Recently created', 'Compute', 'Networking', 'Storage', 'Web + Mobile', 'Containers', 'Databases', 'Data + Analytics', 'AI + Cognitive Services', and 'Internet of Things'. The 'Internet of Things' category is highlighted with a red rectangular border. To the right of each category, there is a blue square icon with a white symbol, followed by the service name and a 'Quickstart tutorial' link. The 'IoT Hub' section is currently selected, indicated by a red circle with a cursor icon over its 'Quickstart tutorial' link.

3. In the IoT Hub blade, enter the following:
 - a. Name: Provide a name for your new IoT Hub, such as **vendingmachingshub**
 - b. Pricing and scale tier: **Select S1 Standard**
 - c. IoT Hub units: **Set to 1**
 - d. Device-to-cloud partitions: **Select 4 partitions.**
 - e. Subscription: Select the same subscription you've been using for previous resources in this lab.
 - f. Resource group: Select Use existing, and select the **ivmhands-onlab** resource group you created previously.
 - g. Location: Select the location you used previously.
 - h. Select **Create**.



- When the IoT Hub deployment is completed, you will receive a notification in the Azure portal. Select **Go to resource** to view the newly created resource.



- From the IoT Hub's Overview blade, select **Shared access policies** under Settings on the left-hand menu.



6. Select **iothubowner** policy.

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

7. In the **iothubowner** blade, select the Copy button to the right of the Connection string - primary key field. Paste the connection string value into a text editor, such as Notepad, as this will be needed later in this lab.

iothubowner

vendingmachineshub

Save Discard Regen key Delete

Access policy name
iothubowner

Permissions
 Registry read
 Registry write
 Service connect
 Device connect

Shared access keys

Primary key
MZ2UbSulSpQW1GVFZkFBEJRSvtSnWnSPUQ9j5p038o=

Secondary key
4Jp5yn4iHyJS/LQ4NJMovivDPTKg+HWFkP09cCrtIgg=

Connection string—primary key
HostName=vendingmachineshub.azure-devices.net;SharedAc...

Connection string—secondary key
HostName=vendingmachineshub.azure-devices.net;SharedAc...

Task 3: Create Microsoft Machine Learning Server on Linux

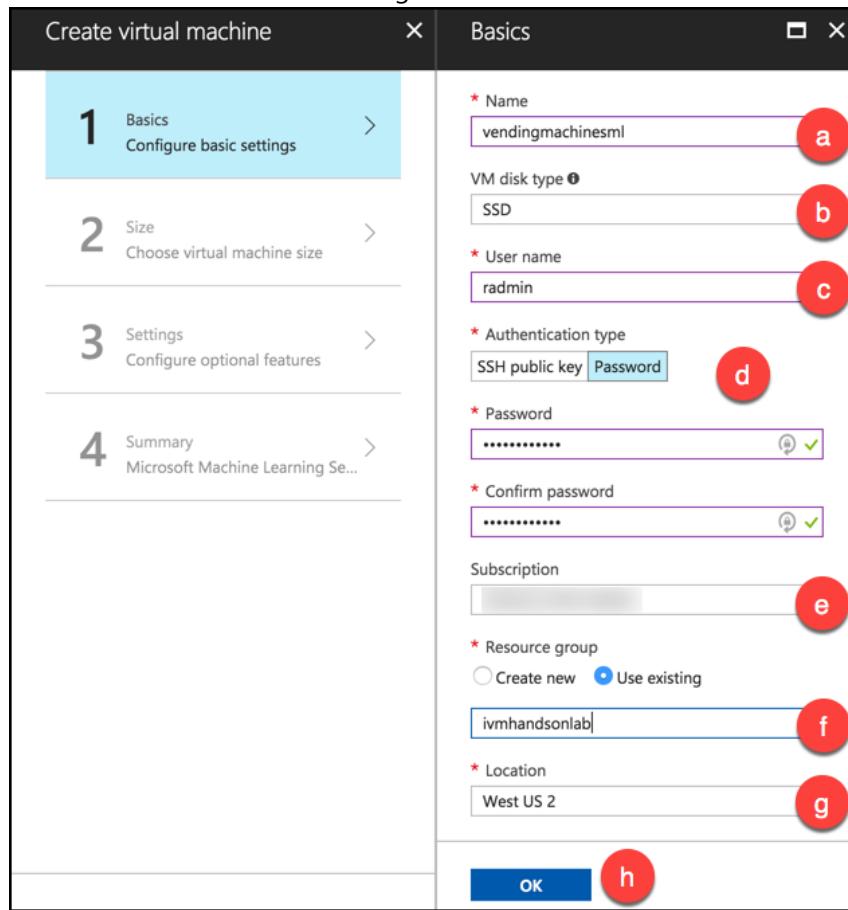
In these steps, you will provision and configure a Virtual Machine running Microsoft Machine Learning Server. You will use this machine to host the R Server Operationalization service.

1. Within the Azure Portal, select **+Create a resource**, then type **Machine Learning Server** into the Search field.
2. In the results list, select **Microsoft Machine Learning Server 9.3.0 on Ubuntu 16.04**.

NAME	PUBLISHER
Microsoft Machine Learning Server 9.3.0 on Red Hat Enterprise Linux 7.2	Microsoft
Microsoft Machine Learning Server 9.3.0 on Ubuntu 16.04	Microsoft
Microsoft Machine Learning Server 9.3.0 on Windows Server 2016	Microsoft
Microsoft Machine Learning Server 9.3.0 on CentOS Linux 7.2	Microsoft

3. On the blade that appears, select **Create**.
4. In the Basics blade, enter:
 - a. Name: Enter a **unique name** for the server.
 - b. User name: Enter **radmin**.
 - c. Authentication type: Select **Password**.
 - d. Password: Enter and confirm the password, **Password.1!!**
 - e. Subscription: Select the subscription you've been using for this lab.
 - f. Resource group: Select **Use existing**, and select the Resource Group you created earlier.
 - g. Location: Select the same location used previously.

- h. Select OK to move on to choosing a VM size.



5. On the Choose a size blade, select **E2S_V3 Standard**, and click **Select**.

The screenshot shows the 'Create virtual machine' wizard. Step 2, 'Choose virtual machine size', is active. A table lists three sizes: E2S_V3 Standard (selected), E4S_V3 Standard, and E8S_V3 Standard. The E2S_V3 Standard row is highlighted with a red box. It includes details like 2 vCPUs, 16 GB memory, 4 Data disks, 4000 Max IOPS, 32 GB Local SSD, Premium disk support, and Load balancing. The price is listed as 98.95 USD/MONTH (ESTIMATED). The 'Select' button is at the bottom of the size table.

	E2S_V3 Standard	E4S_V3 Standard	E8S_V3 Standard
2 vCPUs	2 vCPUs	4 vCPUs	8 vCPUs
16 GB	32 GB	64 GB	160 GB
4 Data disks	8 Data disks	16 Data disks	32 Data disks
4000 Max IOPS	8000 Max IOPS	16000 Max IOPS	32000 Max IOPS
32 GB Local SSD	64 GB Local SSD	128 GB Local SSD	256 GB Local SSD
Premium disk support	Premium disk support	Premium disk support	Premium disk support
Load balancing	Load balancing	Load balancing	Load balancing
98.95 USD/MONTH (ESTIMATED)	197.90 USD/MONTH (ESTIMATED)	395.81 USD/MONTH (ESTIMATED)	

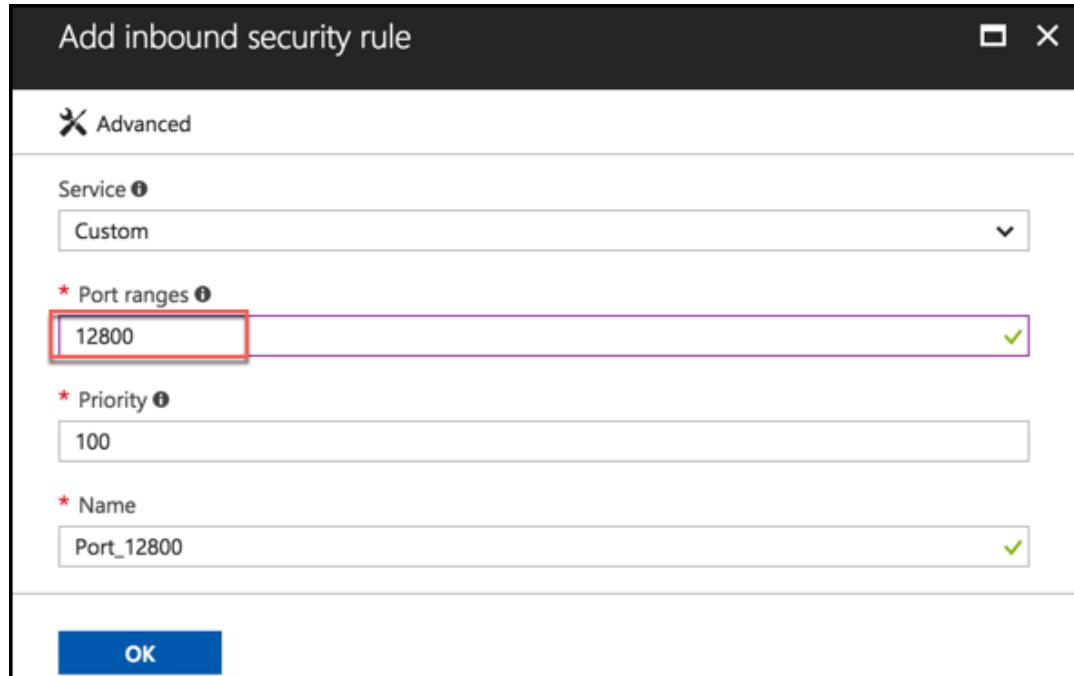
6. On the Settings blade, select **Network security group** (firewall).

- Click **Create new** under Choose network security group.
- Enter a name, such as **my-r-nsg**.
- Select **+Add an inbound rule**.

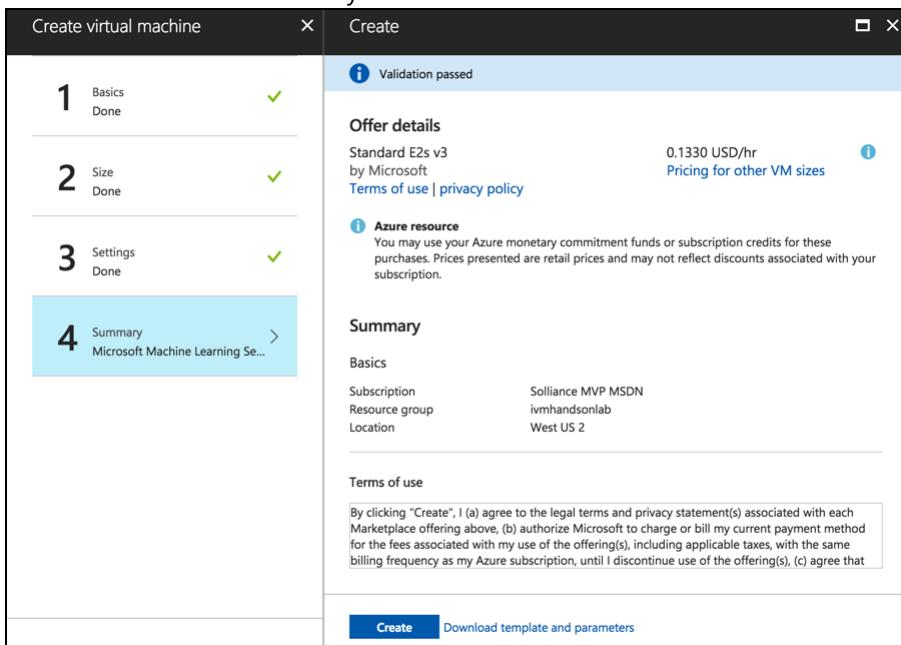
The screenshot shows the 'Create virtual machine' wizard. Step 3, 'Settings', is active. In the 'Network' section, the 'Network security group (firewall)' dropdown is set to '(new) vendingmachinesml-nsg'. The 'Create network security group' dialog is open on the right, showing the 'Create new' button highlighted with a red box. The dialog also shows existing network security groups: None, LabVM-nsg, and deep-learning-z-nsg.

- Select **Custom** under Service and enter the following:

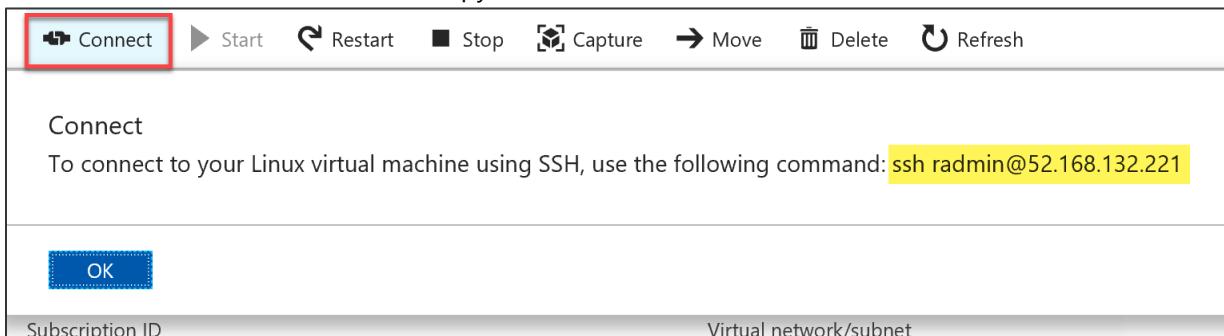
- i. Port range: **12800**
- ii. Priority: **100**
- iii. Name: **Port_12800** (should be auto-generated)
- iv. Select **OK**



7. Select **OK** on the Create network security group blade.
8. Select **OK** on the Settings blade.
9. Select **Create** on the Summary blade.



10. Once the machine has provisioned you will need to perform some configurations. On top of the Overview blade of the new server, select Connect, then copy the SSH command.



11. Using a new Git Bash window on your Lab VM, SSH into your Microsoft Machine Learning Server VM by pasting the SSH command you copied above at the command prompt. For example:
ssh radmin@<your-server-ip>.

12. When prompted if you want to continue connecting, enter **yes**.

13. Enter your password, **Password.1!!**

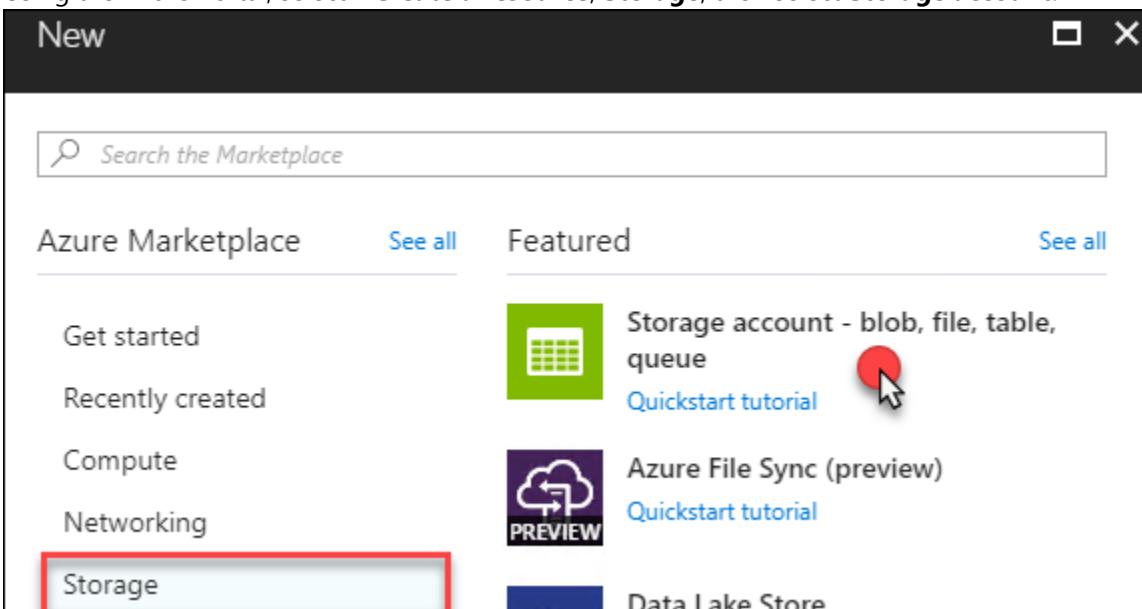
14. At the prompt, after successfully logging in, enter the following command: **sudo apt-get update -y**

15. Type **exit** twice to disconnect from the ssh session.

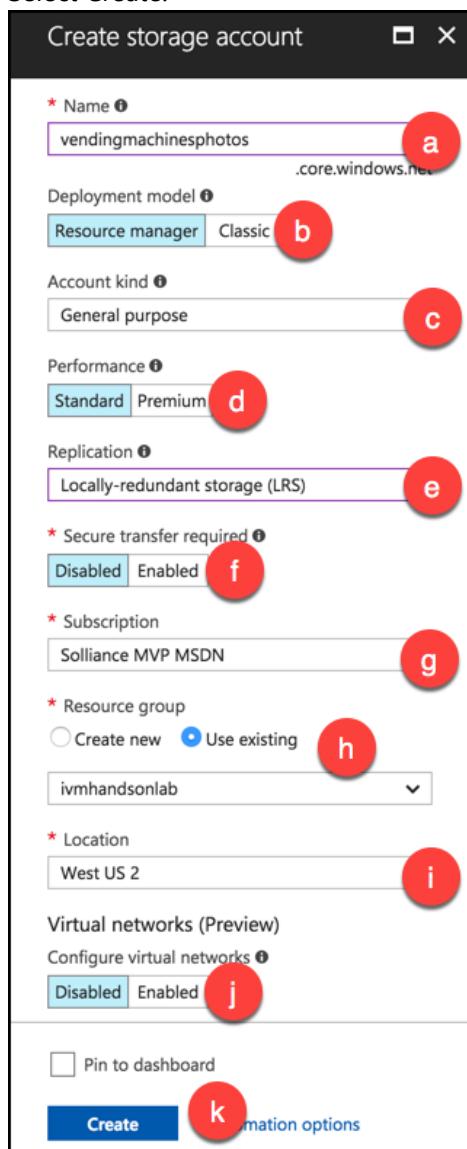
Task 4: Create Storage Account

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 **+Create a resource, Storage**, then select **Storage account**.

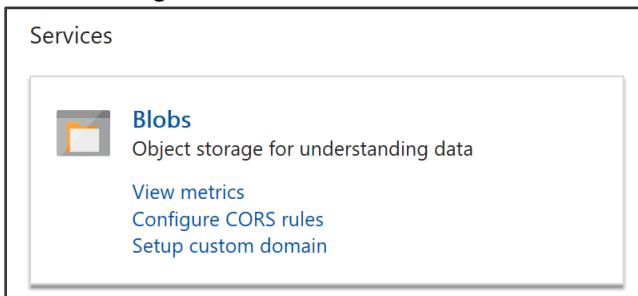


2. On the Create storage account blade, enter the following values:
 - a. Name: Enter a **unique name** for the storage account.
 - b. Deployment model: Leave **Resource Manager** selected.
 - c. Account kind: Leave set at **General purpose**.
 - d. Performance: Leave set to **Standard**.
 - e. Replication: Set to **Locally-redundant storage (LRS)**.
 - f. Storage service encryption: Leave as **Disabled**.
 - g. Subscription: Select your Subscription.
 - h. Resource group: Select the Use existing radio button, and select **ivmhandsonlab** from the resource group list.
 - i. Location: Select a Location to be consistent with the other resources you have created.
 - j. Virtual networks : Leave set to **Disabled**.
 - k. Select Create.

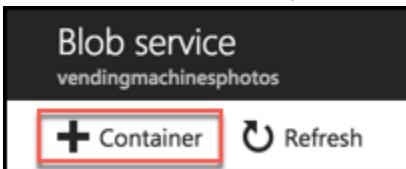


3. Navigate to the newly created storage account in the Azure Portal by clicking on Storage Accounts, and selecting it from the list of available storage accounts.

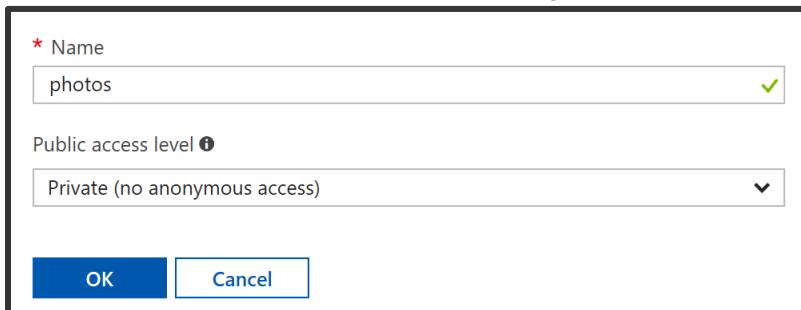
4. On the Storage account blade, select Blobs.



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



6. On the New container blade, set the name to "**photos**" and select **Private** as the Access type.



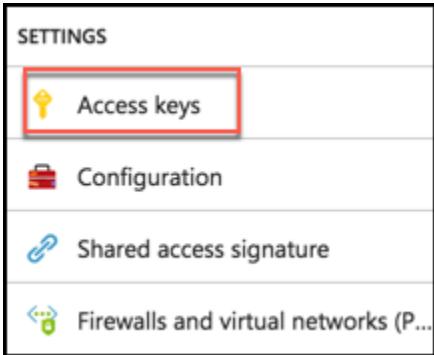
7. Click **OK**.

8. Repeat steps 6-8 to create another container named "**promo**".

9. You should now see both containers listed on the Blob service blade.

NAME	LAST MODIFIED	PUBLIC ACCESS...
photos	9/7/2017 4:42:58 PM	Private
promo	9/7/2017 4:49:17 PM	Private

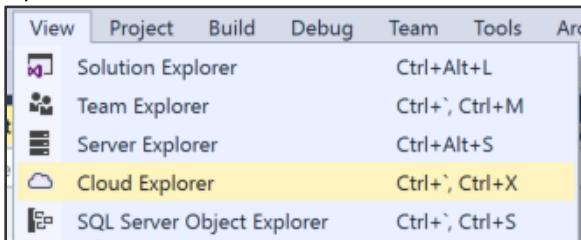
10. Close the Blob service blade to go back to the Storage blade. Select **Access Keys** from the left-hand menu.



11. Use the copy button to the right of the Connection String for key1 to copy your storage connection string. Save the copied value to a text editor, such as Notepad, as this will be used later on.

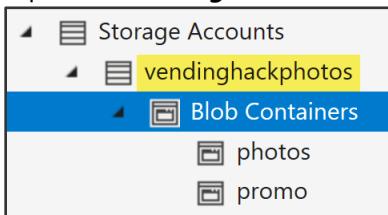
Default keys			
NAME	KEY	CONNECTION STRING	
key1	ac78tyMxtMBmPucDy9065071DU8s+dKTdq6MHZEXC3Ix+y5Wm1/eHejPbxMU+xOf...	DefaultEndpointsProtocol=https;AccountName=vendinghackphotos;AccountKey=ac78tyMxtMBmPucDy9065071DU8s+dKTdq6MHZEXC3Ix+y5Wm1/eHejPbxMU+xOf...	
key2	e/PPW/1bbqpQUVF5fZW9zxueanhxfqKnPMWFxhquB4RqFvwbCJaOWK1YN3YhuV6oqt...	DefaultEndpointsProtocol=https;AccountName=vendinghackphotos;AccountKey=e/PP... 	

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

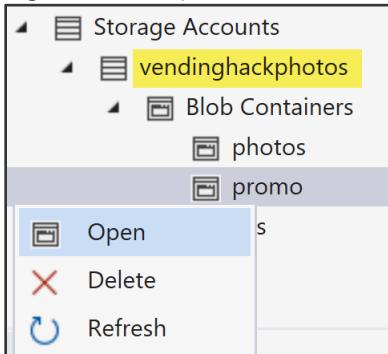


Note: You may need to select your subscription, but clicking the person icon and expanding the subscriptions.

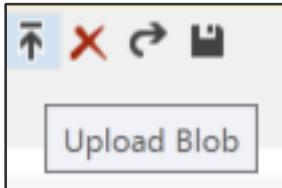
13. Expand the **Storage account** that you just created, and the **Blob Containers** item underneath it.



14. Right-click the promo container and select Open.



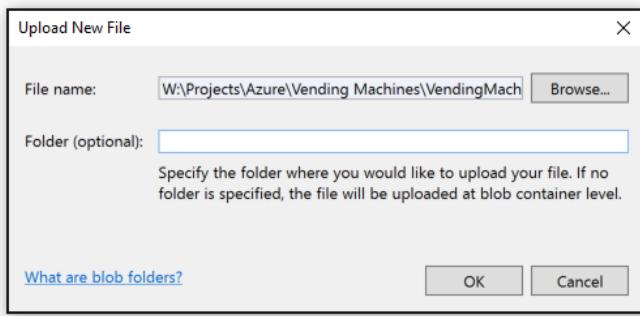
15. Select the Upload blob button.



16. Select **Browse**.

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

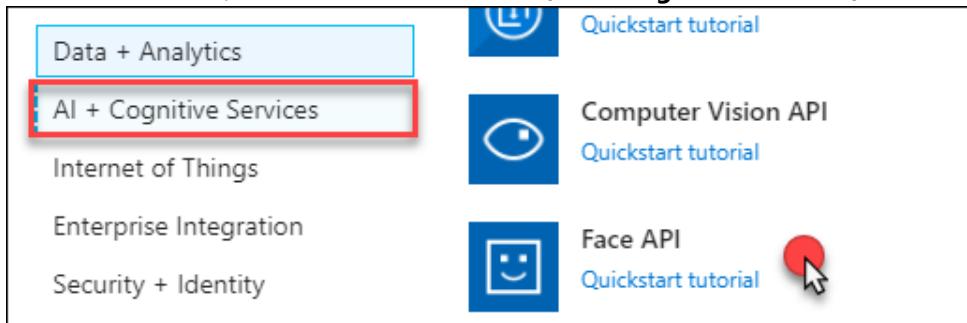
18. Select **OK** on the Upload New File Dialog to upload the images to the container.



Task 5: Provision Cognitive Services Face API

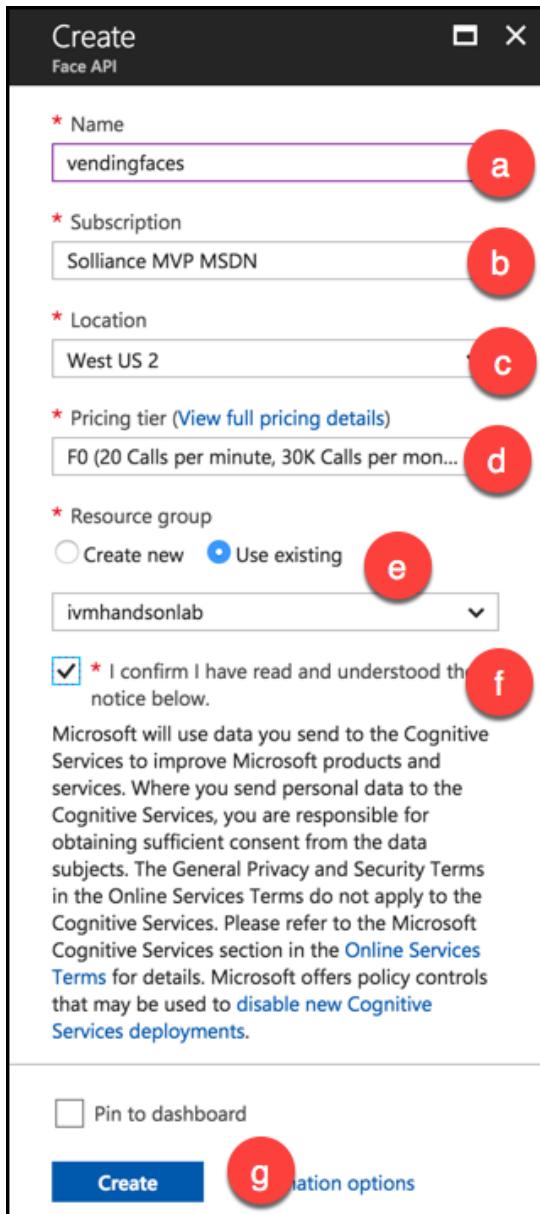
To provision access to the Face API (which provides demographic information about photos of human subjects), you will need to provision a Cognitive Services account.

1. In the Azure Portal, select **+Create a resource, AI + Cognitive Services**, and select **Face API**.

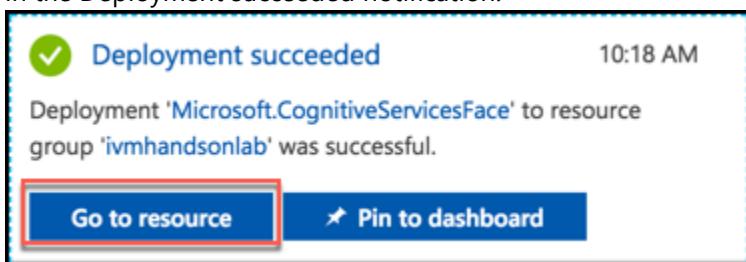


2. On the Create Face API blade:

- Name: Enter a name.
- Subscription: Choose your Subscription.
- Location: Choose the location you've been using for other resources in this lab.
- Pricing tier: Select the **Free tier (F0)** from the drop-down list.
- Resource group: Select Use existing, and select the **ivmhandsonlab** resource group from the list.
- Check the box confirming you have read and understand the legal terms.
- Click Create to provision the Cognitive Services account.



- When the Face API finishes provisioning, browse to the Cognitive Services Face API by clicking on Go to resource in the Deployment succeeded notification.



4. On top of the Cognitive Services overview blade, click the Copy button to the right of the Endpoint. Paste this value into a text editor, such as Notepad, for later use.

The screenshot shows the 'Essentials' section of a Cognitive Services resource. It includes fields for Resource group (ivmhandsonlab), Status (Active), Location (West US 2), Subscription name, API type (Face API), Pricing tier (Free), and Endpoint (<https://westus2.api.cognitive.microsoft.com/face/v1.0>). A 'Click to copy' button is located next to the Endpoint URL, which is highlighted with a red box. A small icon of a hand with a cursor is positioned over the 'Click to copy' button.

5. In the Cognitive Services blade, click on Keys under the Resource Management heading.

The screenshot shows the 'RESOURCE MANAGEMENT' section of the Cognitive Services blade. Under the 'Keys' heading, there is a 'Click to copy' button next to the Key 1 value, which is highlighted with a red box. A small icon of a hand with a cursor is positioned over the 'Click to copy' button.

6. Click the Copy button next to the value for Key 1. Paste this value into a text editor, such as Notepad, for later use.

The screenshot shows the 'NAME' section with the value 'vendinghackfaceapi'. Below it, the 'KEY 1' section shows the value 'a6ba98d95dae475398fe6ccae81eb2a2', with a 'Click to copy' button next to it. This value is also highlighted with a red box. A small icon of a hand with a cursor is positioned over the 'Click to copy' button. The 'KEY 2' section shows the value 'bdfc3e83025b43a2b742ad1b568a75ad'.

Task 6: Provision 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 **+Create a resource**, select **Databases**, then select **SQL Database**.

The screenshot shows the Azure Marketplace interface. On the left, there's a sidebar with links like 'Get started', 'Recently created', 'Compute', 'Networking', 'Storage', 'Web + Mobile', 'Containers', and 'Databases'. The 'Databases' link is highlighted with a red dashed box. On the right, there are four main categories: 'SQL Database' (with a 'Quickstart tutorial' link), 'SQL Data Warehouse' (with a 'Quickstart tutorial' link), 'SQL Elastic database pool' (with a 'Learn more' link), and 'Azure Database for MySQL' (with a 'Quickstart tutorial' link). A red circle with a cursor icon is positioned over the 'Quickstart tutorial' link for the SQL Database.

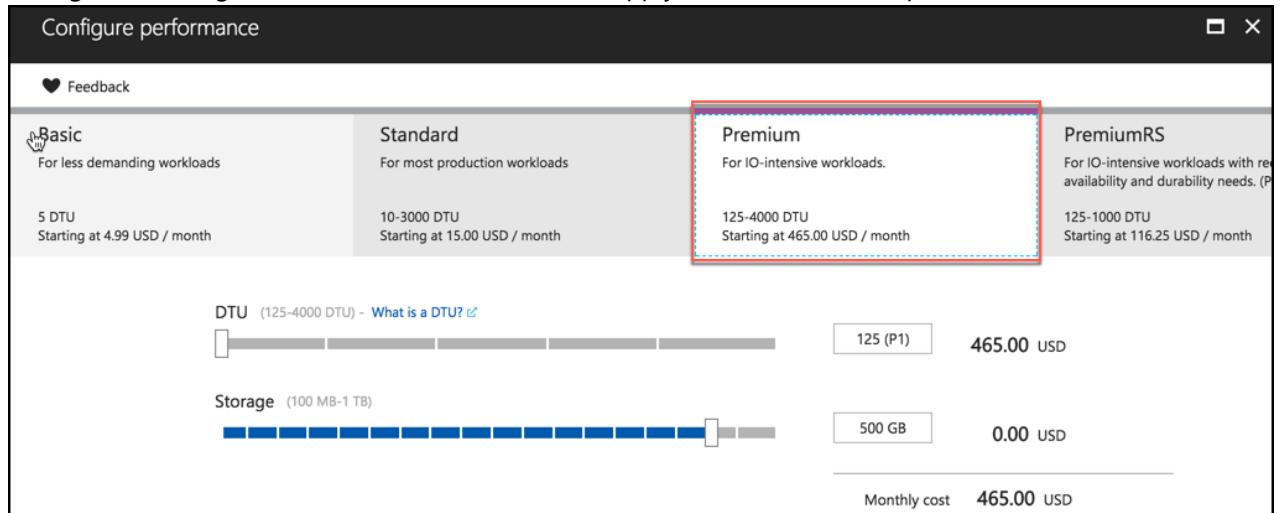
2. In the SQL Database blade, enter the following:
 - a. Database name: Enter **vending**.
 - b. Subscription: Choose your Subscription.
 - c. Resource Group: Select Use existing, and choose **ivmhandsonlab** from the resource group list.
 - d. Source: Leave source as **Blank** database.
 - e. Server: Select **configure required settings**
 - i. Server name: Enter a unique server name
 - ii. User name: Enter **demouser**
 - iii. Password: Enter **Password.1!!**
 - iv. Location: Select the same location you've used for other resources in this lab.
 - v. Click Select.

This screenshot shows the 'Configure required settings' step of the Azure SQL Database creation wizard. It includes fields for:

- * Server name: vendingmachineslab.database.windows.net
- * Server admin login: demouser
- * Password: (redacted)
- * Confirm password: (redacted)
- * Location: West US 2
- Allow azure services to access server

- f. Want to use SQL elastic pool: Leave set to Not now.

- g. Change the Pricing tier to **Premium P1**, and select Apply. Premium tier is required for Columnar indexes.



- h. Collation: Leave set to the default value (SQL_Latin1_General_CI_AS).

- i. Select **Create**.

The screenshot shows the 'SQL Database' creation dialog. The fields are labeled as follows:

- a. Database name: vending
- b. Subscription: Soliance MVP MSDN
- c. Resource group: ivmhandsonlab (radio button selected)
- d. Select source: Blank database
- e. Server: vendingmachineslab (West US 2)
- f. Want to use SQL elastic pool? (radio buttons): Not now (selected)
- g. Pricing tier: Premium P1: 125 DTU, 500 GB
- h. Collation: SQL_Latin1_General_CI_AS
- i. Create button

3. Once the SQL Database finishes provisioning, navigate to the database in the Azure portal and select the Show database connection strings near the top of the Overview blade.

The screenshot shows the 'Overview' blade for a SQL Database named 'vendingmachineslab'. At the top, there are several actions: Tools, Copy, Restore, Export, Set server firewall, and Delete. Below these, it displays the Resource group ('ivmhandsonlab'), Server name ('vendingmachineslab.database.windows.net'), Status ('Online'), and Location ('West US 2'). On the right side, under 'Connection strings', there is a link 'Show database connection strings' which is highlighted with a red box.

4. Copy the connection string on the ADO.NET tab of the Database connection string blade, and paste the value into a text editor, such as Notepad, for later reference.

The screenshot shows the 'Database connection string' blade for the 'vendingmachineslab' database. It has tabs for ADO.NET, JDBC, ODBC, and PHP. The ADO.NET tab is selected. It shows the connection string configuration for 'ADO.NET (SQL authentication)'. The connection string itself is highlighted with a blue box, and a copy icon (a blue square with a white arrow) is located to its right. Below the connection string, there is a link 'Download ADO.NET driver for SQL server'.

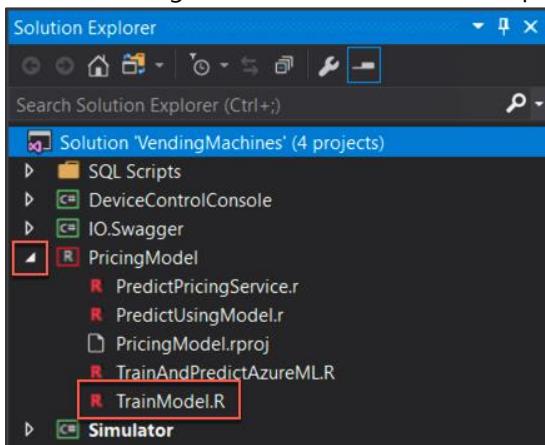
Exercise 2: Create Dynamic Pricing Model

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.

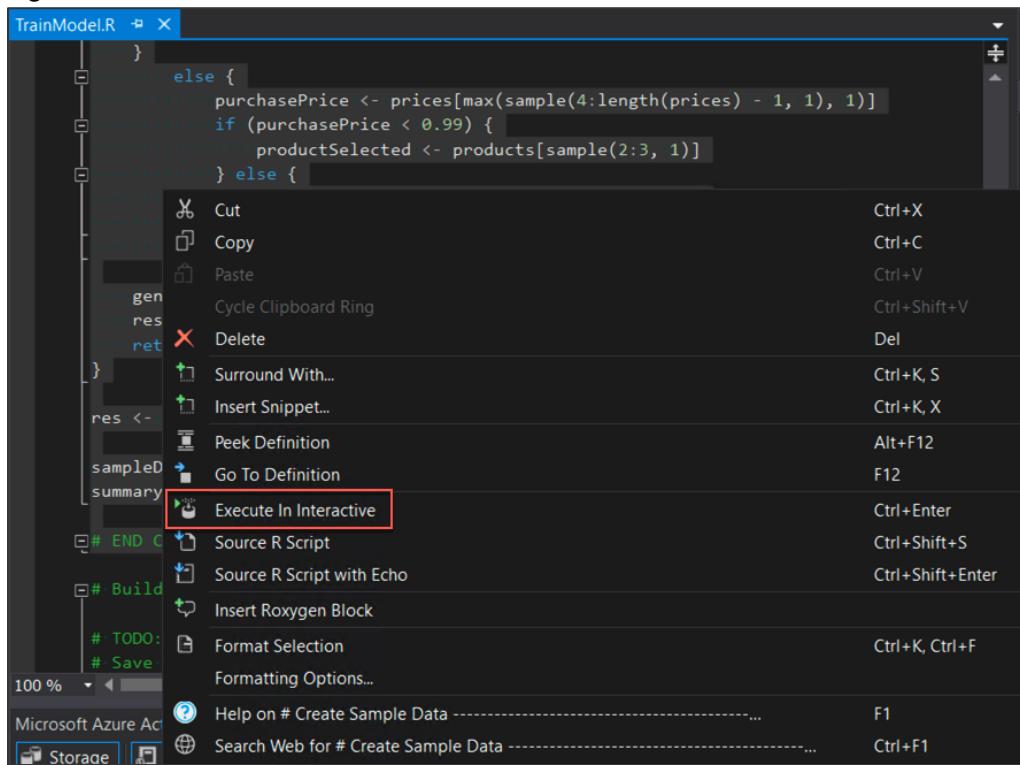
Task 1: Create a model locally

1. Within Visual Studio Solution Explorer, expand the **PricingModel** project and open the file **TrainModel.R** by double-clicking on the file in the Solution Explorer.



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.

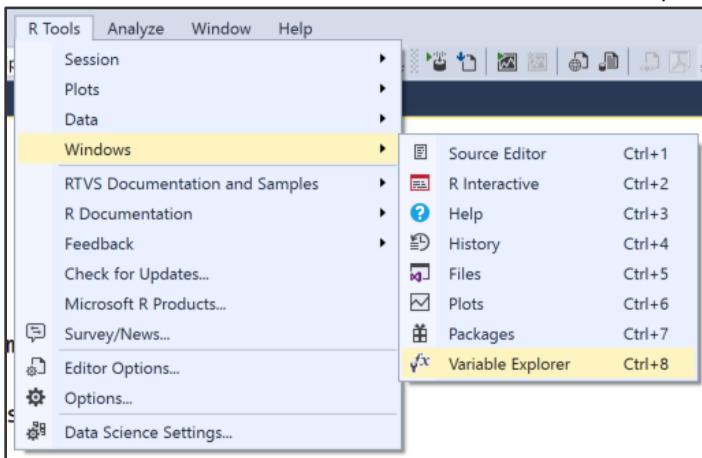


5. You should see it execute in the R Interactive Window, ending with a summary of the created data.

The screenshot shows the R Interactive window with the title 'R Interactive - Microsoft R Client (3.3.2.0)'. The window displays the output of the R code, specifically a summary of the 'productSelected' data frame. The output includes statistical summaries for 'age', 'purchasePrice', 'gender', and 'productSelected'.

age	purchasePrice	gender	productSelected
Min. :18.00	Min. :0.2500	F:491	coconut water:254
1st Qu.:26.00	1st Qu.:0.5000	M:509	soda :501
Median :35.00	Median :0.7500		water :245
Mean :34.49	Mean :0.7394		
3rd Qu.:43.00	3rd Qu.:0.9900		
Max. :50.00	Max. :1.0000		

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



7. Expand the variable **sampleData**, and explore the structure of the created data.

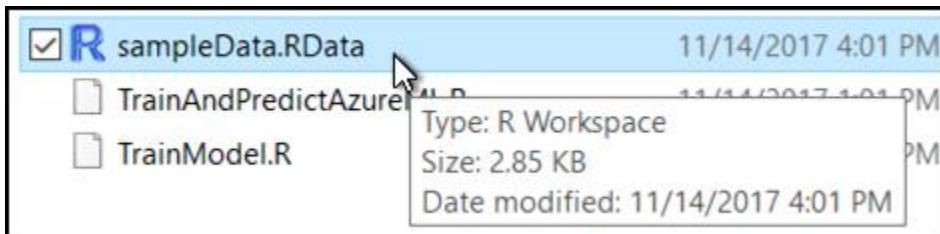
Variable Explorer				
.GlobalEnv				
Name	Value	Class	Type	
ages	int [1:33] 18 19 20 21 22 23 24 ;	[x]	integer	integer
distAges	function (x)		function	closure
genders	Factor w/ 2 levels "F","M": 2 1	[x]	factor	integer
generateExample	function (index)		function	closure
prices	num [1:7] 0.25 0.35 0.5 0.75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

8. Now save this sampleData to a file by replacing TODO 1 in the TrainModel.R script with the following code:

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

9. Highlight the **save** line, and select **Execute In Interactive**.

10. Open File Explorer and navigate to the location of the **PricingModel** (C:\VendingMachines\VendingMachines - Clean\PricingModel) project on disk. **You should see** the file **sampleData.RData** on disk.



11. 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)
```

12. 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.RData")
```

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.RData")
```

14. Save your changes to **TrainModel.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.RData**, **pricingModel.RData**, and **inputExample.RData**.

inputExample.RData	11/14/2017 4:04 PM	R Workspace
PredictPricingService.r	11/14/2017 1:01 PM	R File
PredictUsingModel.r	11/14/2017 1:01 PM	R File
pricingModel.RData	11/14/2017 4:04 PM	R Workspace
PricingModel.rproj	11/14/2017 1:01 PM	RPROJ File
PricingModel.rxproj	11/14/2017 1:01 PM	RXPROJ File
sampleData.RData	11/14/2017 4:01 PM	R Workspace

Task 2: Try a prediction locally

1. Within Visual Studio, open **PredictUsingModel.r** under the Pricing Model project in Solution Explorer.

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. On your LabVM, open a Git Bash shell like you did in the [Before the Hands-on Lab, Task 4](#), Step 7.
2. SSH into your deployed R Server in HDInsight cluster. (You can get the SSH connection string for your cluster from the HDInsight Blade in the Azure Portal).

Connect to cluster using secure shell (SSH)

You can securely connect to the below endpoints in the HDInsight cluster with an SSH client.

[Documentation](#)

Hostname
vendingmachineslab-ssh.azurehdinsight.net

ssh remoteuser@vendingmachineslab-ssh.azurehdinsight.net

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


```
sparkCluster <- RxSpark()
rxSetComputeContext(sparkCluster)
```
7. In Visual Studio, open **TrainModel.r**, and copy the entire script.
8. 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.)
9. When the script has finished executing, type the following:


```
dir()
```
10. You should see it list the three files created by the script, as follows:


```
> dir()
[1] "inputExample.RData" "pricingModel.RData" "sampleData.RData"
```
11. 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)
```

12. List the contents of the root ("/") directory, and confirm your **"/models"** 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("/")
[1] "Found 18 items"
[2] "drwxr-xr-x  - root      supergroup          0 2017-11-14 10:45 /hdinotebooks"
[3] "drwxr-xr-x  - root      supergroup          0 2017-11-14 10:49 /hdisamples"
[4] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /ams"
[5] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /amshbase"
[6] "drwxrwxrwx   - yarn     hadoop             0 2017-11-14 10:39 /app-logs"
[7] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /apps"
[8] "drwxr-xr-x  - yarn     hadoop             0 2017-11-14 10:39 /atshistory"
[9] "drwxr-xr-x  - root      supergroup          0 2017-11-14 10:48 /cluster-info"
[10] "drwxr-xr-x  - root     supergroup          0 2017-11-14 10:49 /custom-scriptaction-logs"
[11] "drwxr-xr-x  - root     supergroup          0 2017-11-14 10:49 /example"
[12] "drwxr-xr-x  - hbase    supergroup          0 2017-11-14 10:39 /hbase"
[13] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /hdp"
[14] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /hive"
[15] "drwxr-xr-x  - mapred   supergroup          0 2017-11-14 10:39 /mapred"
[16] "drwxr-xr-x  - remoteuser supergroup          0 2017-11-14 17:56 /models"
[17] "drwxrwxrwx   - mapred   hadoop             0 2017-11-14 10:39 /mr-history"
[18] "drwxrwxrwx   - hdfs     supergroup          0 2017-11-14 10:39 /tmp"
[19] "drwxr-xr-x  - hdfs     supergroup          0 2017-11-14 10:39 /user"
[1] TRUE
```

13. Copy the **pricingModel.RData** from the local directory to the **/models** folder in HDFS by running the following command:

```
rxHadoopCopyFromLocal("pricingModel.RData", modelExportDir)
```

14. Repeat the previous step for **inputExample.RData** and **sampleData.RData**.

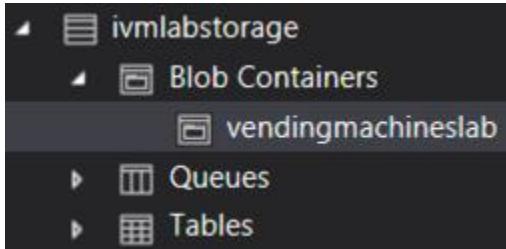
15. Run the following command to verify the three files now exist in HDFS (and Blob storage), under **/models**.

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

16. The output should look similar to the following:

```
> rxHadoopListFiles("/models")
[1] "Found 3 items"
[2] "-rw-r--r--  1 sshuser supergroup      243 2017-09-12 02:26 /models/inputExample.RData"
[3] "-rw-r--r--  1 sshuser supergroup    1895 2017-09-12 02:25 /models/pricingModel.RData"
[4] "-rw-r--r--  1 sshuser supergroup    2954 2017-09-12 02:26 /models/sampleData.RData"
[1] TRUE
```

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



18. Right-click the storage container, and select **Open**.

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

Name	Size	Last Modified (UTC)	Content Type	URL
inputExample.RData	243 bytes	9/12/2017 2:26:04 AM	application/octet-stream	https://hivelab.blob.core.windows.net/hackathoncluster2/models/inputExample.RData
pricingModel.RData	1.9 KB	9/12/2017 2:25:42 AM	application/octet-stream	https://hivelab.blob.core.windows.net/hackathoncluster2/models/pricingModel.RData
sampleData.RData	2.9 KB	9/12/2017 2:26:18 AM	application/octet-stream	https://hivelab.blob.core.windows.net/hackathoncluster2/models/sampleData.RData

20. Right-click **inputExample.RData** and select Save As... and choose the directory for your **PricingModel** project, overwriting files if prompted.

21. Repeat the previous step for **pricingModel.RData** and **sampleData.RData**.

22. You have now used R Server on HDInsight to train a model that you can then upload to R Server Operationalization to expose it as a web service.

Task 4: Create predictive service in R Server Operationalization

After training a model, you want to operationalize the model so that it becomes available for integration by developers. One way to operationalize a trained model is to take the model you trained in HDInsight, 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 the VM that is running R.

16. In the Azure portal, navigate to the Microsoft Machine Learning Server Virtual Machine you created in [Exercise 1, Task 3](#).

17. On top of the Overview blade, select Connect, then copy the SSH command.

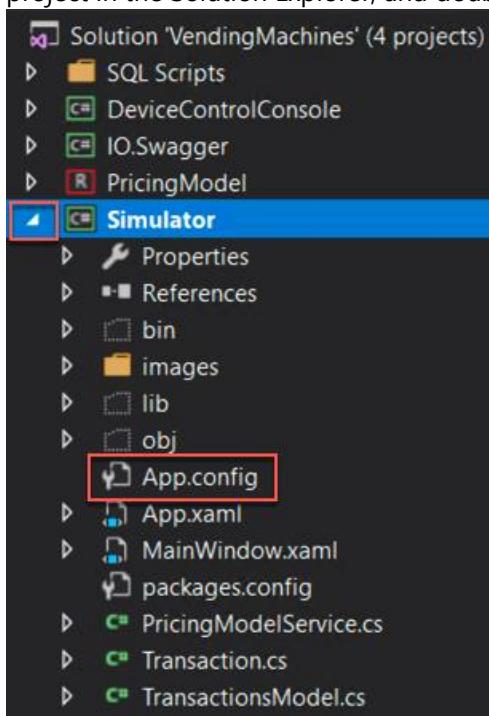
The screenshot shows the 'Overview' blade for a Linux virtual machine. At the top, there are several buttons: 'Connect' (highlighted with a red box), 'Start', 'Restart', 'Stop', 'Capture', 'Move', 'Delete', and 'Refresh'. Below these buttons, the word 'Connect' is displayed, followed by instructions: 'To connect to your Linux virtual machine using SSH, use the following command: ssh radmin@52.168.132.221'. A yellow box highlights the SSH command. At the bottom of the blade, there are two buttons: 'OK' and 'Subscription ID'.

18. Using a new Git Bash window on your Lab VM, SSH into your Microsoft Machine Learning Server VM by pasting the SSH command you copied above at the command prompt. For example:
ssh radmin@<your-server-ip>.
19. When prompted if you want to continue connecting, enter yes.
20. Enter your password, **Password.1!!**
21. At the prompt, after successfully logging in, you will need to complete a few tasks to configure and operationalize the environment.
22. Run the following command to act as root: **sudo -i**
23. Now that you are acting as root run the following command to: **az ml admin node setup --onebox --admin-password Password.1!! --confirm-password Password.1!!**
root@ivmMLserver:~# az ml admin node setup --onebox --admin-password Password.1!! --co
az ml admin node setup --onebox --admin-password Password.1!! clear --confirm-password Pa

Starting Compute Node
SUCCESS! Compute Node started. (PID: 25394, Listening on URI: http://localhost:12805/)
Starting Web Node
SUCCESS! Web Node started. (PID: 25657, Listening on URI: http://localhost:12800/)

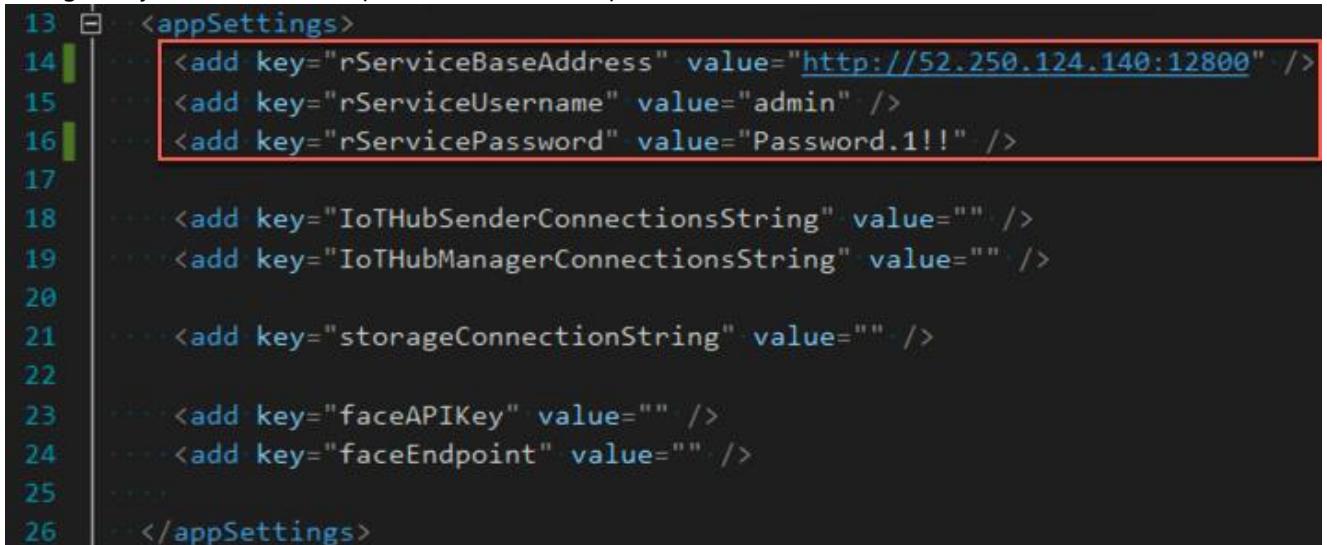
root@ivmMLserver:~# exit
logout
radmin@ivmMLserver:~\$ |

24. In Visual Studio, open the **App.config** for the Simulator project. This can be done by expanding the Simulator project in the Solution Explorer, and double-clicking on **App.config** under the **Simulator** project.



25. Locate the **appSetting rServiceBaseAddress** and enter **http://<your-server-public-ip>:12800** for the value.
(For example: <http://52.168.132.221:12800>). Your server IP address is the same IP address you used for the SSH connection in Step 3 of this task, above.

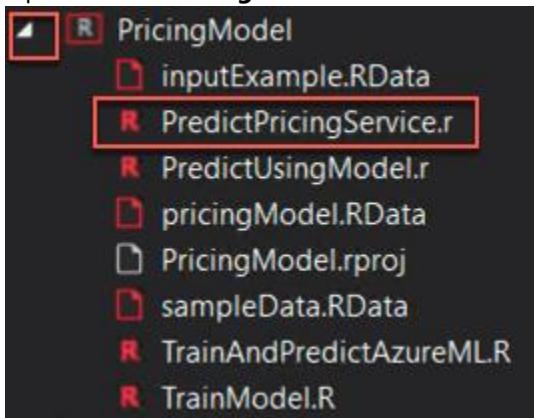
26. Locate the **appSetting rServicePassword** and update its value with the password you defined when you configured your R server for Operationalization (step 11 above), **Password.1!!**



```
13 <appSettings>
14   <add key="rServiceBaseAddress" value="http://52.250.124.140:12800" />
15   <add key="rServiceUsername" value="admin" />
16   <add key="rServicePassword" value="Password.1!!" />
17
18   <add key="IoTHubSenderConnectionString" value="" />
19   <add key="IoTHubManagerConnectionString" value="" />
20
21   <add key="storageConnectionString" value="" />
22
23   <add key="faceAPIKey" value="" />
24   <add key="faceEndpoint" value="" />
25
26 </appSettings>
```

27. Save **App.config**.

28. Open **PredictPricingService.r** within the **PricingModel** project in the Visual Studio Solution Explorer.



29. Find TODO 1, and replace with the following code block:

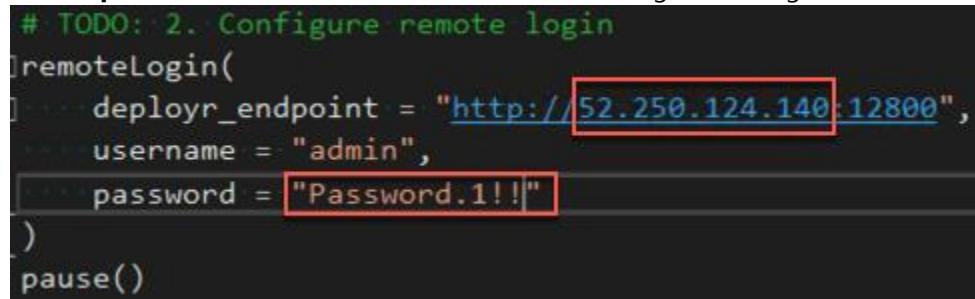
```
# TODO: 1. Load packages needed for operationalization
usePackage <- function(p) {
  if (!is.element(p, installed.packages()[, 1]))
    install.packages(p, dep = TRUE)
  library(p, character.only = TRUE)
}
```

```
usePackage("curl")
usePackage("ggplot2")
usePackage("mrsdeploy")
usePackage("RevoScaleR")
```

30. Find TODO 2, and replace with the following code block to remotely connect to the R Server Operationalization service:

```
# TODO: 2. Configure remote login
remoteLogin(
  deployr_endpoint = "http://<your-server-ip>:12800",
  username = "admin",
  password = "<your-admin-password>"
)
pause()
```

Note: Make sure to replace <your-server-ip> with the IP address with your VM's IP address, and enter the password you specified when you configured your R server for Operationalization (step 11) in place of <your-admin-password>. The TODO 2 section should look something like the following screen shot.



```
# TODO: 2. Configure remote login
remoteLogin(
  deployr_endpoint = "http://52.250.124.140:12800",
  username = "admin",
  password = "Password.1!!"
)
pause()
```

31. Highlight all the code in **PredictPricingService.r**, right-click and then **execute in interactive**. The last output status in the R Interactive window should be "Published service".

32. Find TODO 3 and replace with the following code block to consume the API as a test:

```
# TODO: 3. Consume the API as a test
services <- listServices("apiPredictPurchasePrice")
serviceName <- services[[1]]
api <- getService(serviceName$name, serviceName$version)
result <- api$apiPredictPurchasePrice(30, "F", "coconut water")
print("Result: ")
print(result$output("answer"))
```

```
result
```

33. Find TODO 4 and replace with the following code block to generate and save the Swagger JSON file for the API:

```
# TODO: 4. Generate the Swagger JSON file for the API  
swagger <- api$swagger()  
cat(swagger, file = "swagger.json", append = FALSE)
```

34. Highlight just the TODO 3 and TODO 4 code blocks you added and execute in interactive.

35. When you scroll up through the R Interactive window results, you should see an output with your prediction like the following:

```
$success  
[1] TRUE  
  
$errorMessage  
[1] ""  
  
$outputParameters  
$outputParameters$purchasePrice  
[1] 0.9348741
```

36. Also open the swagger.json file in your PricingModel project directory to view its contents. This file can be used within the swagger.io online editor to generate client code to connect to your service. We have already done this for you within the included IO.Swagger project.

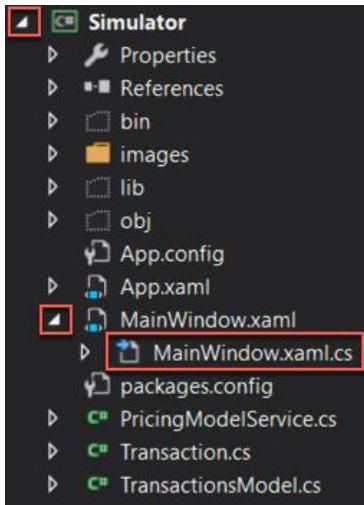
Exercise 3: Implement dynamic pricing

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**.

```
69
70     private async Task UpdateDynamicPrice(string filename)
71     {
72         // TODO 1. Retrieve storage account from connection string.
73         CloudStorageAccount storageAccount = /*Complete this*/
74         CloudBlobClient blobClient = storageAccount./*Complete this*/;
75         CloudBlobContainer container = blobClient.GetContainerReference("photos");
```

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);
```

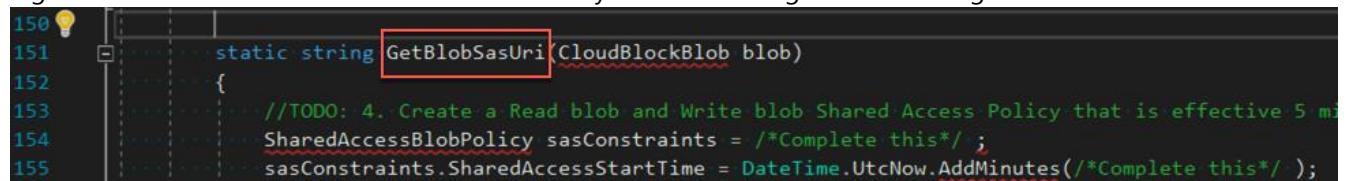
5. Replace TODO 3 with the following:

```
// 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.



A screenshot of the Visual Studio code editor showing a C# file. Line 150 contains a yellow lightbulb icon. Lines 151 through 155 define a static string method named 'GetBlobSasUri' that takes a 'CloudBlockBlob' parameter. The code includes a TODO comment and some placeholder code for creating a SharedAccessBlobPolicy.

```
150  static string GetBlobSasUri(CloudBlockBlob blob)  
151 {  
152     //TODO: 4. Create a Read blob and Write blob Shared Access Policy that is effective 5 mi  
153     SharedAccessBlobPolicy sasConstraints = /*Complete this*/;  
154     sasConstraints.SharedAccessStartTime = DateTime.UtcNow.AddMinutes(/*Complete this*/);  
155 }
```

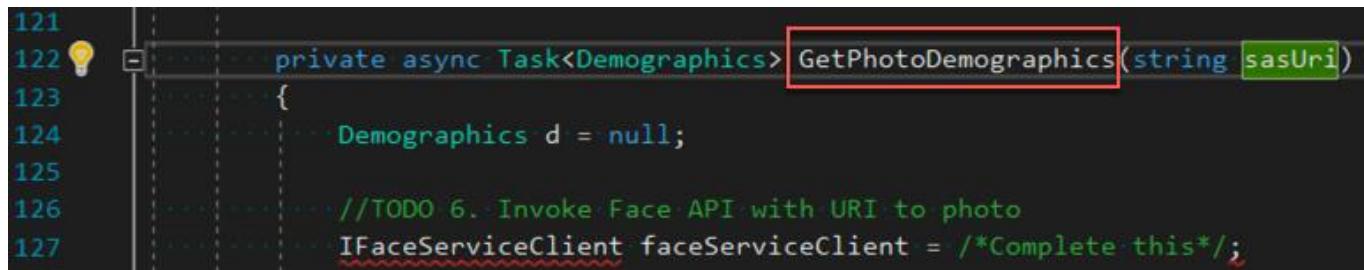
2. Replace TODO 4 with the following:

```
//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.



```
121
122     private async Task<Demographics> GetPhotoDemographics(string sasUri)
123     {
124         Demographics d = null;
125
126         //TODO 6. Invoke Face API with URI to photo
127         IFaceServiceClient faceServiceClient = /*Complete this*/;
```

5. Replace TODO 6 with the following:

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

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 **UpdateDynamicPrice** 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 (there were copied into a text edit previously):
 - a. **faceAPIKey**: set this to the KEY 1 value for your Face API as acquired from the Azure Portal.
 - b. **faceEndpoint**: set this to the ENDPOINT value for your Face API as acquired from the Azure Portal (for example: <https://eastus2.api.cognitive.microsoft.com/face/v1.0>)
 - c. **storageConnectionString**: set this to the connection string of the Storage Account you created with the photos container.
3. Save the **App.config**. The updated **App.config** file settings should look similar to the following:

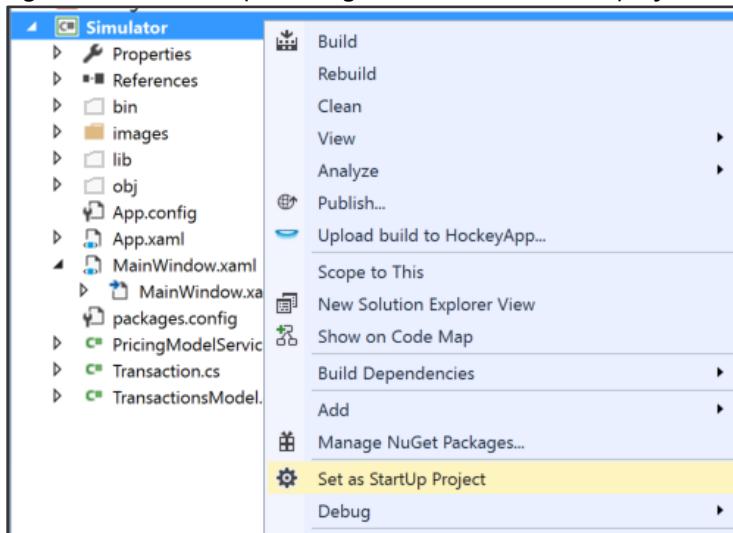
```

20 <add key="storageConnectionString" value="DefaultEndpointsProtocol=https;AccountName=vendingmachinesphotos;
21   <add key="faceAPIKey" value="beeead523483b4bfb9258abd9fda6db9b" />
22   <add key="faceEndpoint" value="https://westus2.api.cognitive.microsoft.com/face/v1.0" />
23 </appSettings>
24
25
26

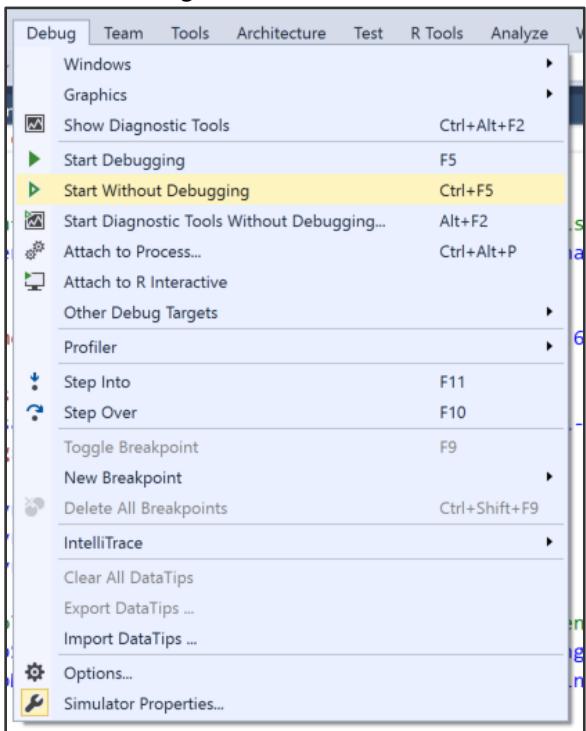
```

Task 5: Test dynamic pricing in Simulator

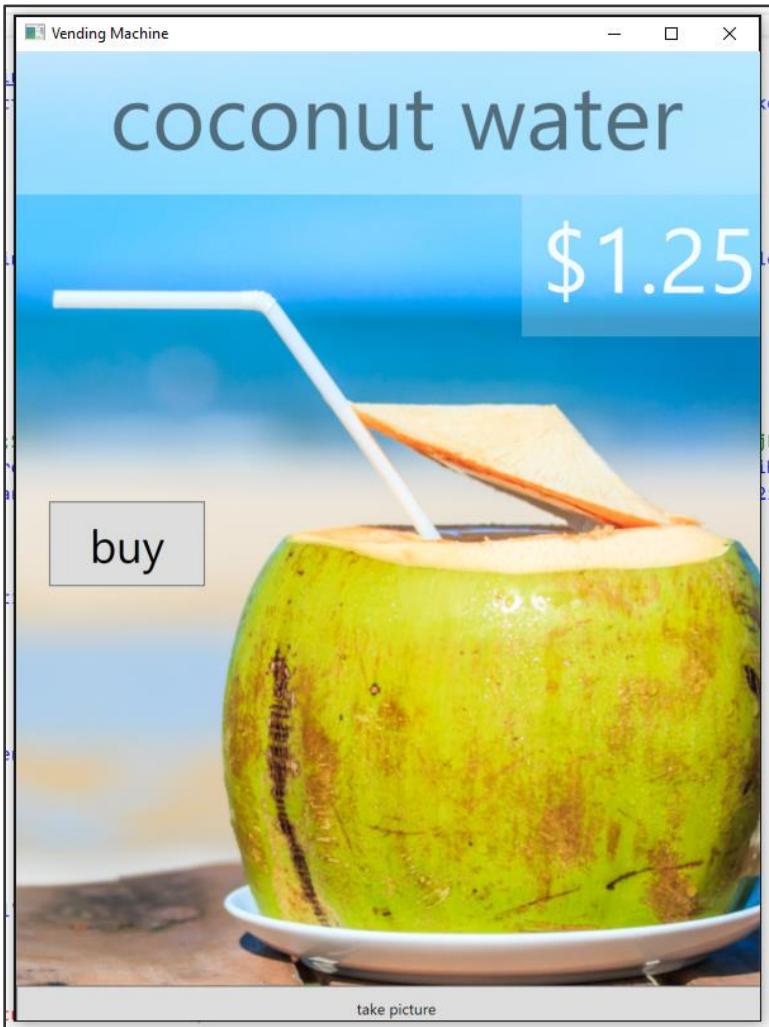
1. Before building the project in Visual Studio, we need to ensure all the NuGet packages are properly restored. In Visual Studio, go to **Tools->NuGet Package Manager->Package Manager Console** and enter the following command:
Update-Package -Reinstall
2. That should force the packages to get downloaded again.
3. Now, in Solution Explorer, right-click the Simulator project, and select **Build**.
4. Ensure that your build generates no errors (View the Output and Errors windows, under the View menu in Visual Studio).
5. Again, in solution explorer, right-click the Simulator project, and select Set as Startup Project.



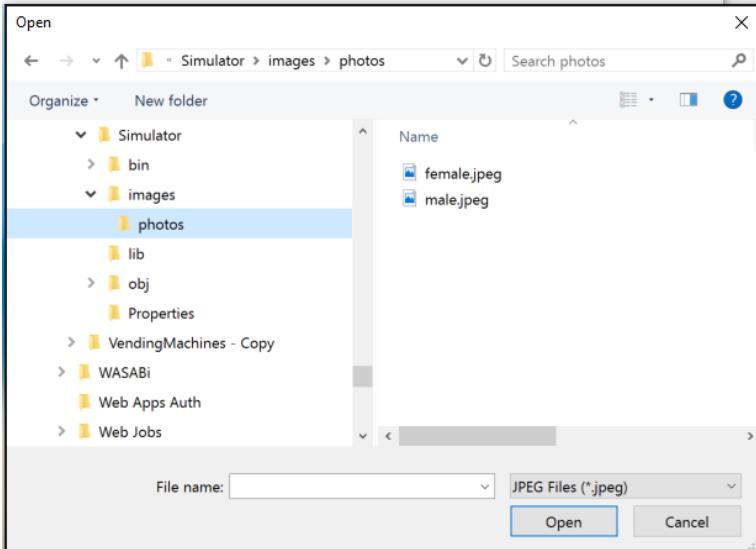
6. From the Debug menu, select Start Without Debugging.



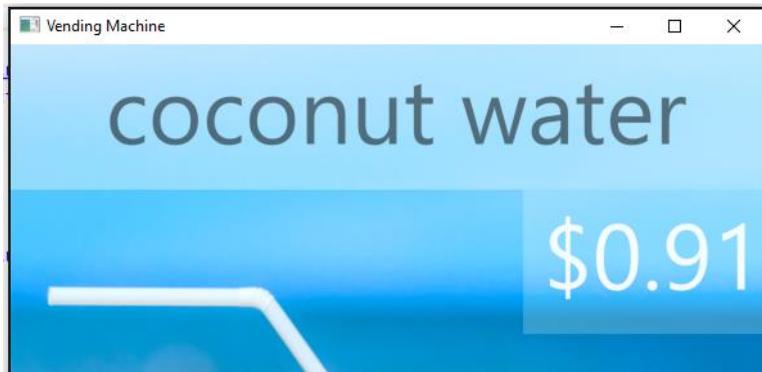
7. When the vending machine simulator appears, **select take picture** at the bottom.



8. In the dialog that appears, navigate to the images folder under **C:\VendingMachines\VendingMachines - Clean\Simulator\images\photos**, and pick the photo of either the man or woman to upload, and select Open.



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



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

11. Click the X at the top right of the application to stop it.

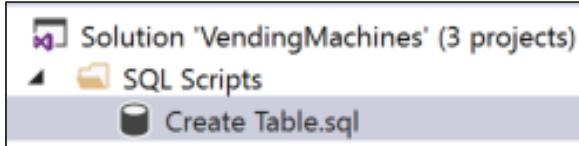
Exercise 4: Implement purchasing

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;
```

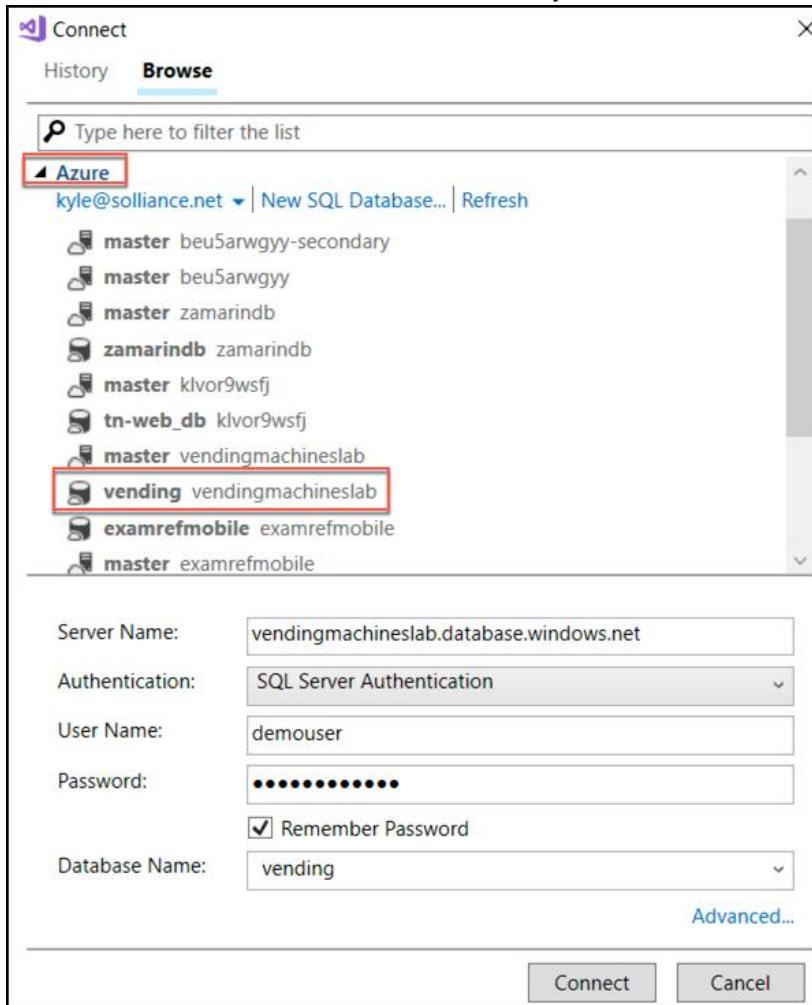
6. Save the script.

7. Execute the script by pressing the play icon.



8. Expand **Azure**, and if prompted, sign in with your Azure credentials.

9. From the Azure node, and select the database you created for the vending database.



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

Query executed successfully at 7:36:34 PM

Task 2: Configure the Simulator

1. In the Simulator project, open **App.config**.
2. Within the **connectionString** section, set the following:
 - a. TransactionsModel: set the value of the connectionString attribute to the ADO.NET connection string to your SQL DB instance. This value was copied to a text editor previously, or you can copy it from the Azure Portal. **Do not forget to replace the values for {your_username} and {your_password} with your actual credentials.**
 - i. User name: demouser
 - ii. Password: Password.1!!

3. Save the **App.config**.

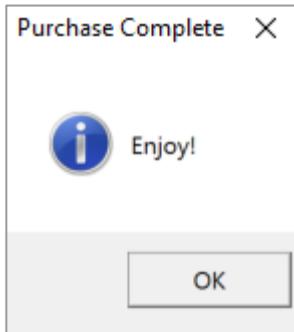
```
10 <connectionStrings>
11   <add name="TransactionsModel" connectionString="Server=tcp:vendingmachineslab.database.windows.net,1433;Init
12     ...>/connectionStrings>
```

Task 3: Test purchasing

1. In solution explorer, right-click the Simulator project, and select Build. Note: You may need to ensure the previous instance you started has been closed before rebuilding.
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:



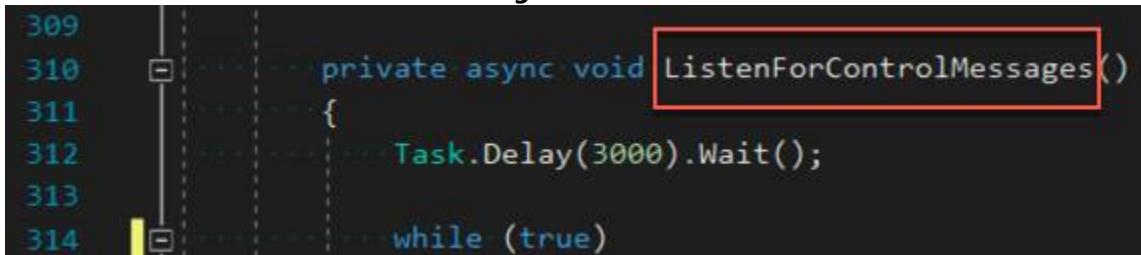
Exercise 5: Implement device command and control

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

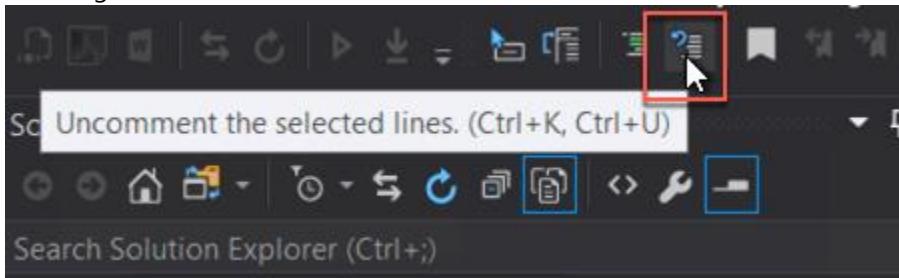
1. Within Visual Studio Solution Explorer, expand the Simulator project, and open the file **MainWindow.xaml.cs**.
2. Scroll down to the **ListenForControlMessages** method.



A screenshot of the Visual Studio code editor showing a C# file. The method `private async void ListenForControlMessages()` is highlighted with a red rectangle. The code within the method is as follows:

```
309  
310     private async void ListenForControlMessages()  
311     {  
312         Task.Delay(3000).Wait();  
313  
314         while (true)
```

3. Uncomment the body of the while(true) loop. You can uncomment a block of code by selecting the code, then selecting the Uncomment button on the toolbar.



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);

```

- Replace TODO 4 with the following:

```

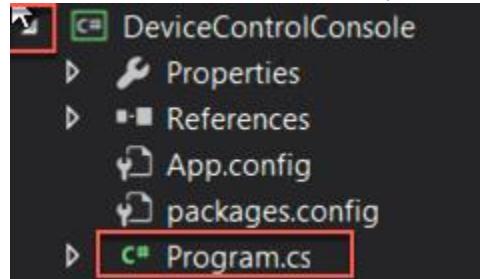
//TODO: 4. Acknowledge receipt of the message with IoT Hub
await _deviceClient.CompleteAsync(receivedMessage);

```

- Save the file.

Task 2: Send control messages

- Within Visual Studio Solution Explorer, expand the **DeviceControlConsole** project, and open the file **Program.cs**.



- Scroll down to the PushPromo method.

```

87
88     static async Task PushPromo(string deviceId, PromoPackage promoPackage)
89     {
90         //TODO: 1. Create a Service Client instance provided the _IoTHubConnectionString
91         _serviceClient = /*Complete this*/;
92

```

- Replace TODO 1 with the following:

```

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

```

- Replace TODO 2 with the following:

```

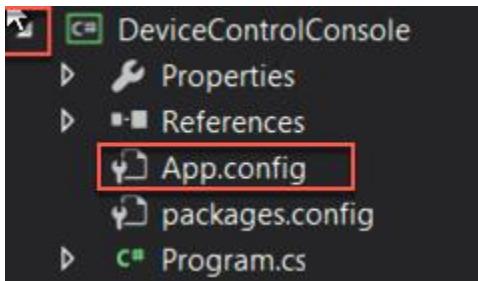
//TODO: 2. Send the command
await _serviceClient.SendAsync(deviceId, commandMessage);

```

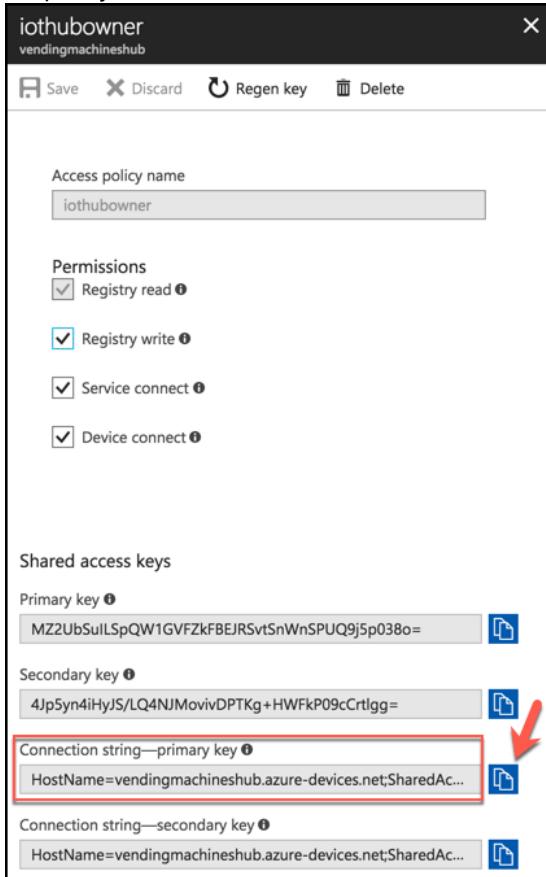
- Save Program.cs.

Task 3: Configure the DeviceControlConsole and the Simulator

- In **DeviceControlConsole**, open **App.config**.

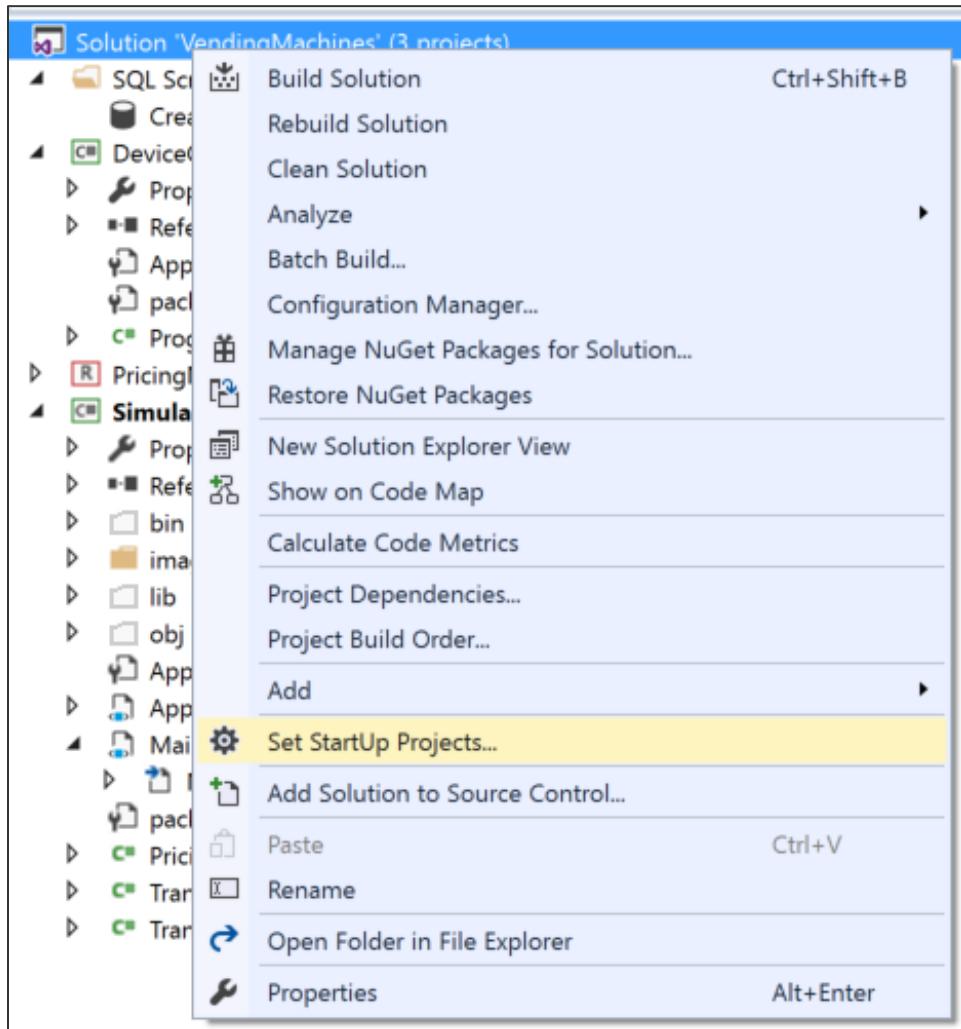


- Set the **IoTHubConnectionString appSetting** to have a value of the connection string for the service policy to your IoT Hub. (Recall you can get this from the Azure Portal IoT Hub blade, Shared access policies, and then select the policy.)

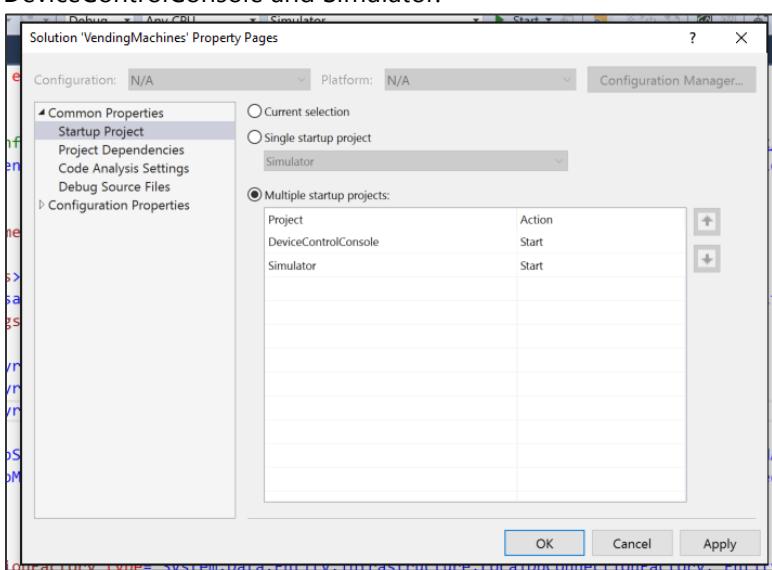


- Set the **storageConnectionString appSetting** to have the same connection string for your storage account that the **App.config** file in the Simulator project has.
- Save the file.
- Now, open the **App.config** file in the Simulator project.
- Set the **IoHubSenderConnectionString appSetting** to have a value of the connection string for the device policy to your IoT Hub.

7. Set the **IoTHubManagerConnectionString appSetting** to have a value of the connection string for the **iothubowner** policy to your IoT Hub.
8. Save the file.
9. Build the Simulator and DeviceControlConsole projects.
10. In Solution Explorer, right-click Solution 'Vending Machines,' and select Set StartUp Projects.



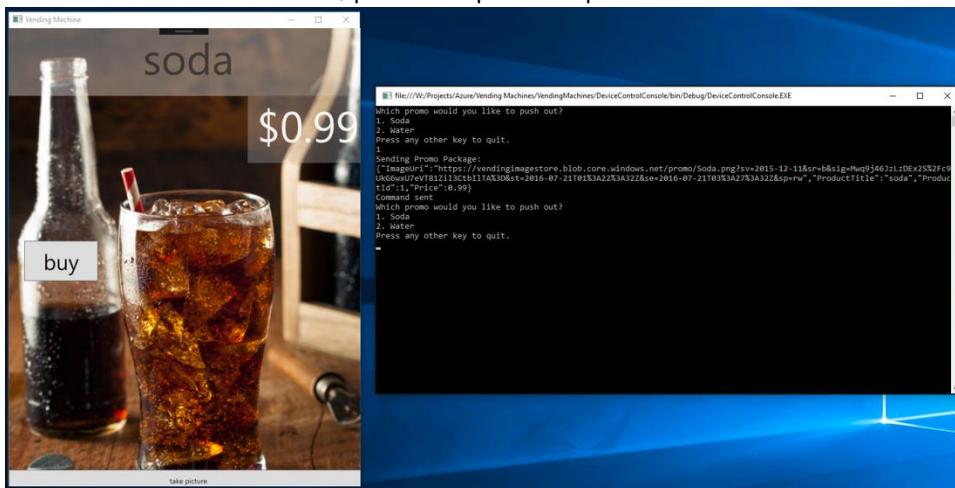
11. In the dialog, select the Multiple startup projects option, and ensure that Action is set to Start for both DeviceControlConsole and Simulator.



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



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



16. Observe that the entire promotion surface of the vending machine changes (product name, price, and image).

Note: If the photo does not change, and after a few minutes you receive a DotNetty.Transport... error, you will need to delete and recreate your IoT Hub in the Azure portal. The error is caused by a communication error between the application and your IoT Hub. Be sure to update your App.config file with the new IoT hub connection strings.

17. Experiment sending the other promotion or toggling between promotions.

18. Experiment with making purchases and sending photos to verify the other functions still work with the new promoted products.

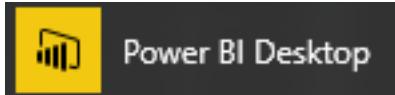
Exercise 6: Analytics with Power BI Desktop

Duration: 15 minutes

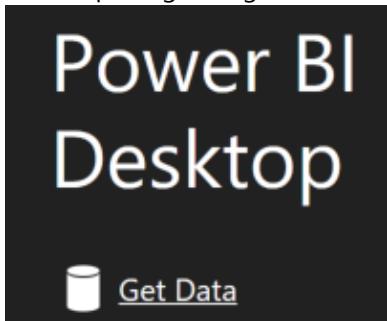
In this exercise, you will use Power BI Desktop to query purchase data from the in-memory table of SQL DB and visualize the result.

Task 1: Build the query and create the visualization

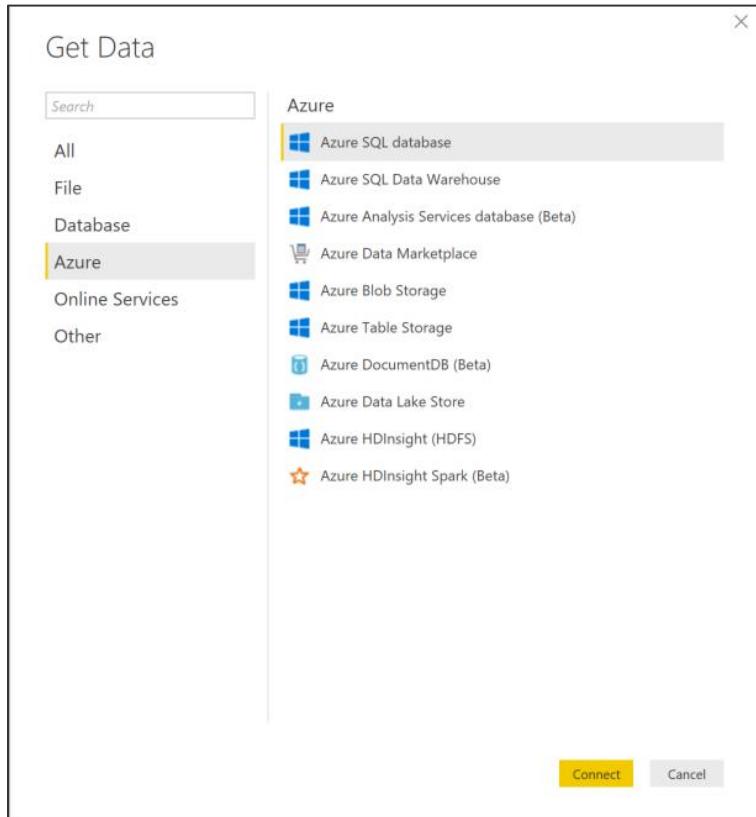
1. From your Start menu on your Lab VM, open Power BI Desktop.



2. In the opening dialog, select **Get Data**.

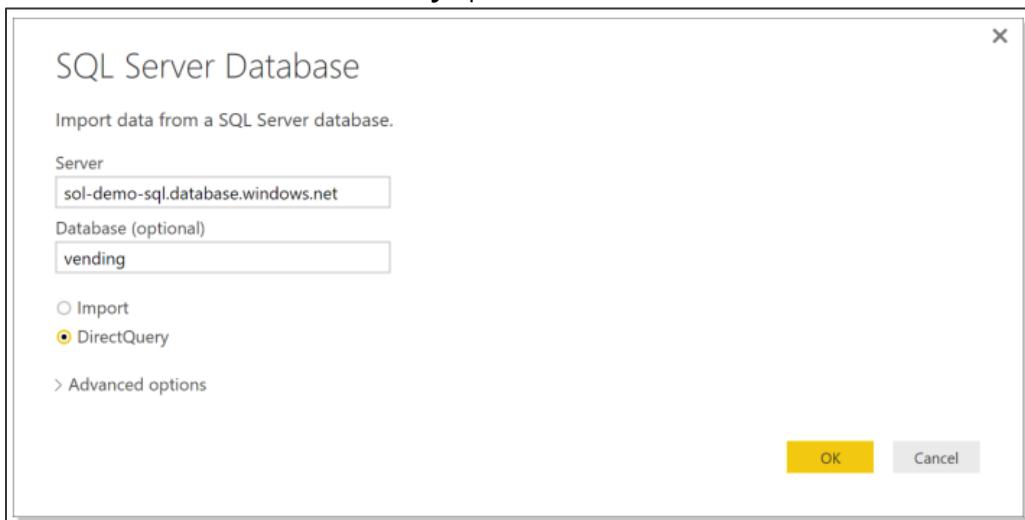


3. In the Get Data dialog, select **Azure** in the categories list and then **Azure SQL Database**.

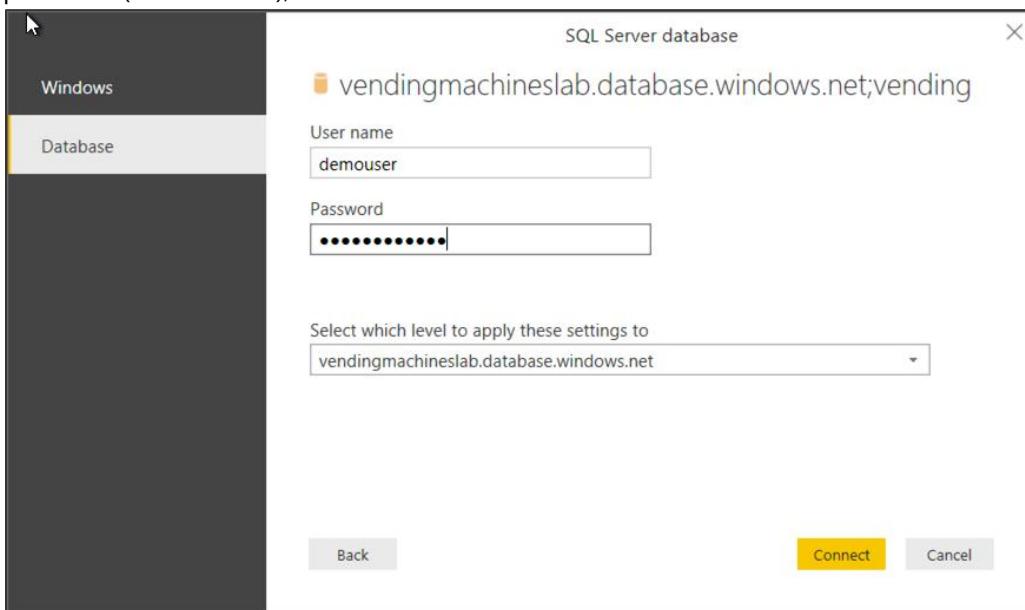


4. Select Connect.

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.



6. On the next screen, select the **Database** tab on the left, and provide your SQL username (demouser) and password (Password.1!!), then select **Connect**.



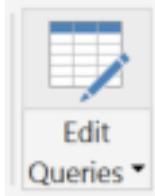
7. In the Navigator dialog, check the box next to **Transactions**.

The screenshot shows the 'Navigator' dialog box. On the left, there's a tree view under 'Display Options' showing 'sol-demo-sql.database.windows.net: vending [2]' with 'Transactions' selected. On the right, a table titled 'Transactions' is displayed with columns: TransactionId, VendingMachineId, ItemName, and ItemId. The table contains 26 rows of data, all of which have 'coconut water' listed under ItemName. At the bottom of the dialog are buttons for 'Select Related Tables', 'Load' (which is highlighted in yellow), 'Edit', and 'Cancel'.

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-abef-b1ae32bf4fb9	coconut water	
8	43dbedf0-504e-4ecf-abef-b1ae32bf4fb9	coconut water	
9	43dbedf0-504e-4ecf-abef-b1ae32bf4fb9	coconut water	
10	258df733-6f1f-449b-9710-f30be2b25188	coconut water	
11	85cd3494-2038-4ae7-a351-eceab4b8f58	coconut water	
12	85cd3494-2038-4ae7-a351-eceab4b8f58	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-5e9091bdce5	coconut water	
17	48e35e5c-07a8-4508-945c-5e9091bdce5	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	

8. Select **Load**.

9. In the Ribbon, select **Edit Queries**.



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

The screenshot shows the 'Query Editor' window. A column header 'TransactionDate' is highlighted with a yellow background. Below it, three rows of data are shown: '7/19/2016 7:58:18 PM', '7/19/2016 8:01:23 PM', and '7/19/2016 8:02:40 PM'. The rest of the table is visible but not highlighted.

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

The screenshot shows the Microsoft Power BI Query Editor interface. The ribbon at the top has the 'Add Column' tab selected, indicated by a red box. The main area shows a table with columns: Name, ItemId, \$ PurchasePrice, and TransactionDate. A context menu is open over the 'TransactionDate' column, with the 'Hour' option highlighted. The 'Time' icon in the ribbon is also highlighted with a red box.

12. Select the **TransactionDate** column again.

13. In the Ribbon, select **Time, Minute**.

The screenshot shows the Microsoft Power BI Query Editor interface. The ribbon at the top has the 'Time' tab selected, indicated by a red box. A context menu is open over the 'PurchasePrice' column, with the 'Minute' option highlighted. A tooltip window is displayed, explaining that it creates a new column containing the minute corresponding to each Date/Time value in the selected column. The table below shows three rows of data with values 0.99, 0.99, and 1.25.

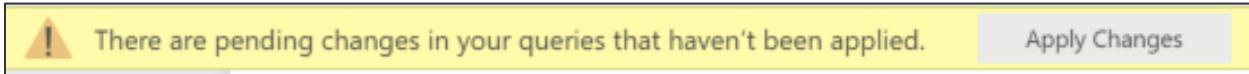
14. Select the **TransactionDate** one more time.

15. In the Ribbon, select **Time, Second**.

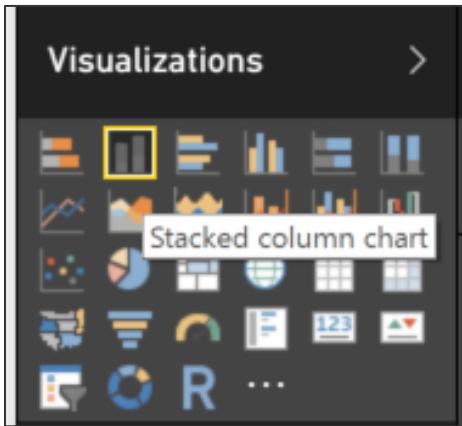
16. In the Ribbon, on the Home tab, select **Close & Apply**.



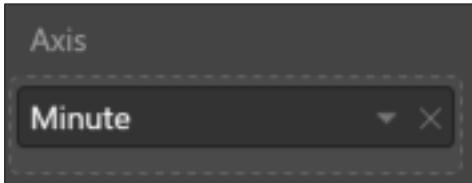
17. In the message that appears, select **Apply Changes**.



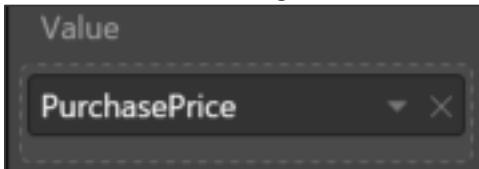
18. In the Visualizations, select **Stacked column chart**.



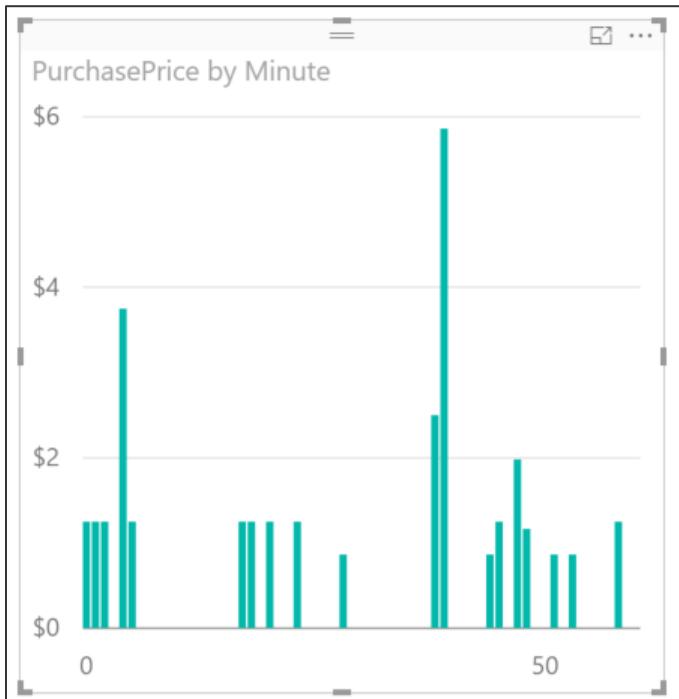
19. From the Fields list, drag the **minute** field over to the **axis** property.



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



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



After the hands-on lab

Duration: 10 mins

In this exercise, attendees will deprovision any Azure resources that were created in support of the lab. You should follow all steps provided after attending the Hands-on lab.

Task 1: Delete the resource group

1. Using the Azure portal, navigate to the Resource group you used throughout this hands-on lab by selecting Resource groups in the left menu.
2. Search for the name of your research group, and select it from the list.
3. Select Delete in the command bar, and confirm the deletion by re-typing the Resource group name, and selecting Delete.