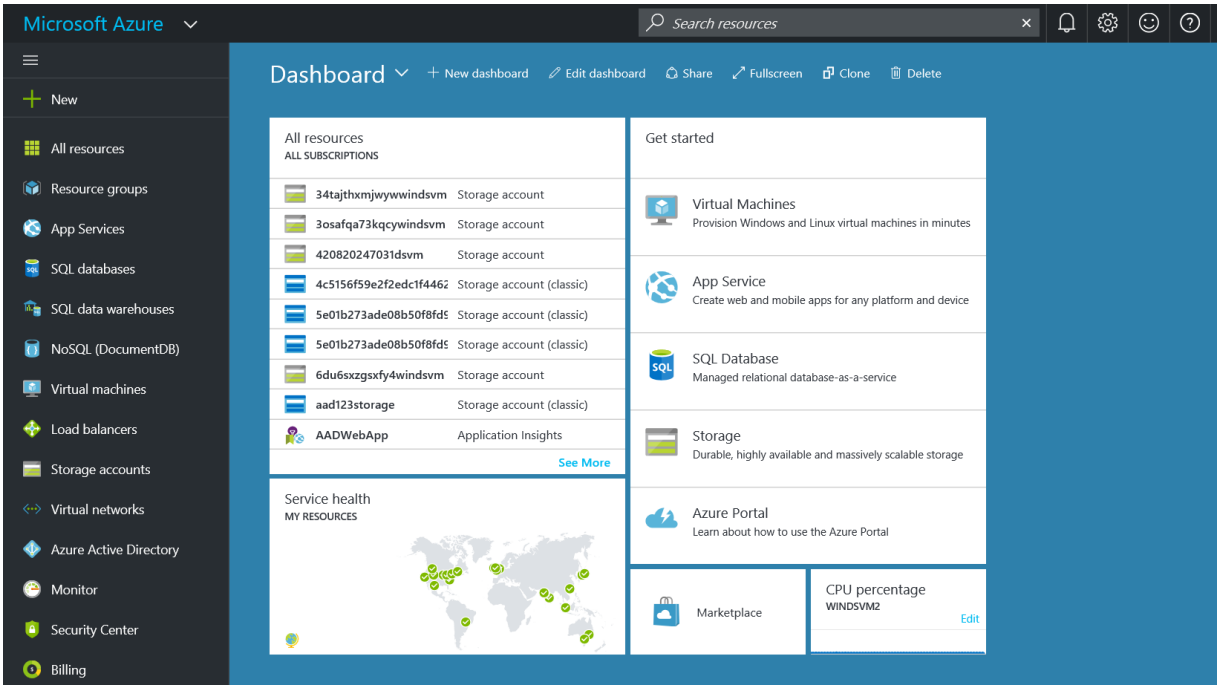
Data Science Virtual Machine – Introductory Hands-On Workshop

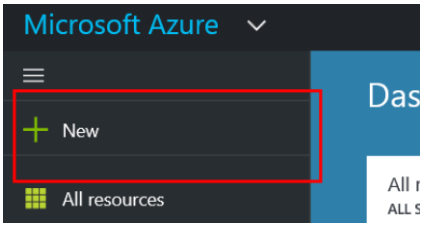
Contents

Deploying a Windows Data Science Virtual Machine – Using the Azure portal

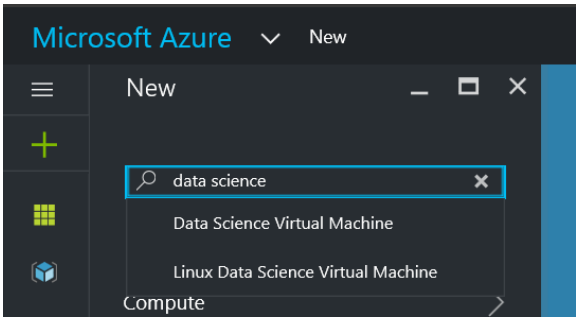
1. Go to http://portal.azure.com
   Sign in with your Azure subscription credentials and you will see the portal dashboard.



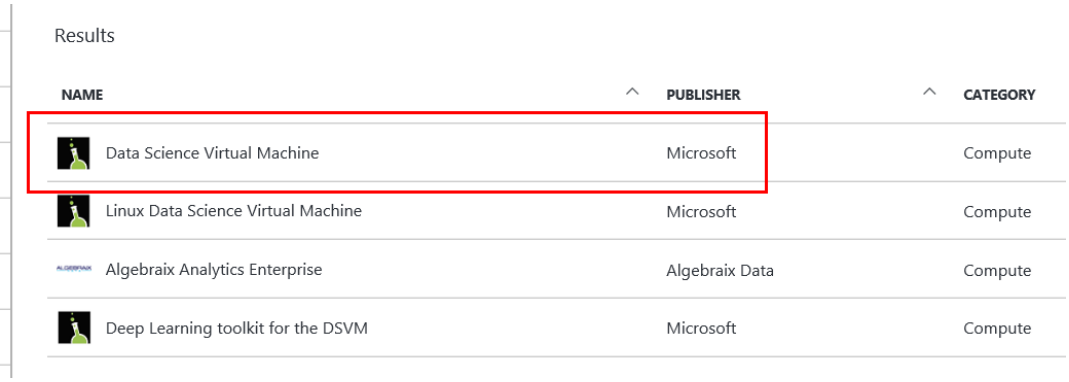
2. Click on New



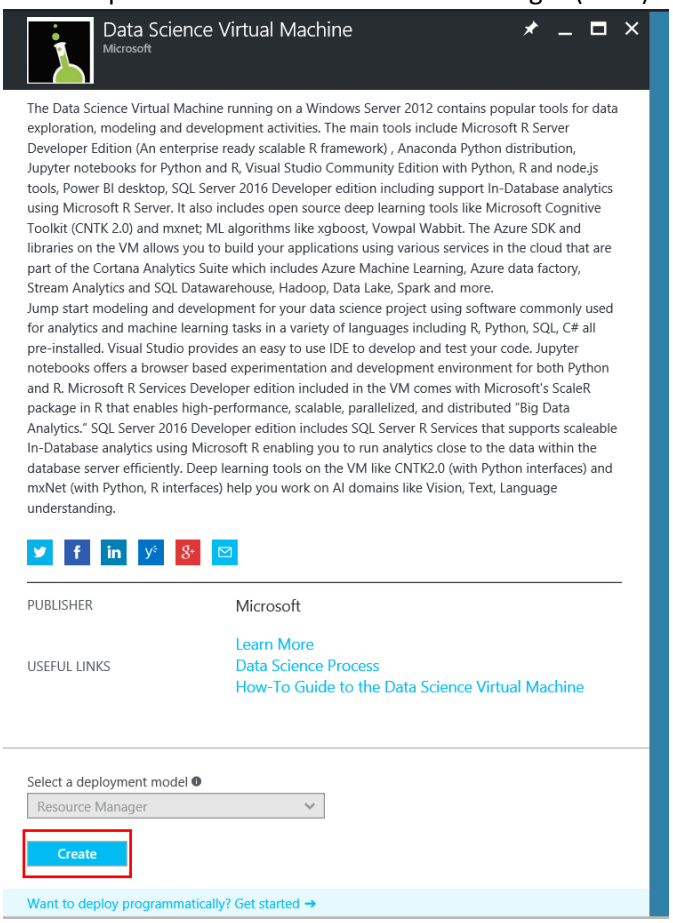
3. Type in data science in the search bar and then press Enter. You may also click on the suggestions, but if u just search for Data Science, you will see all options in the results.



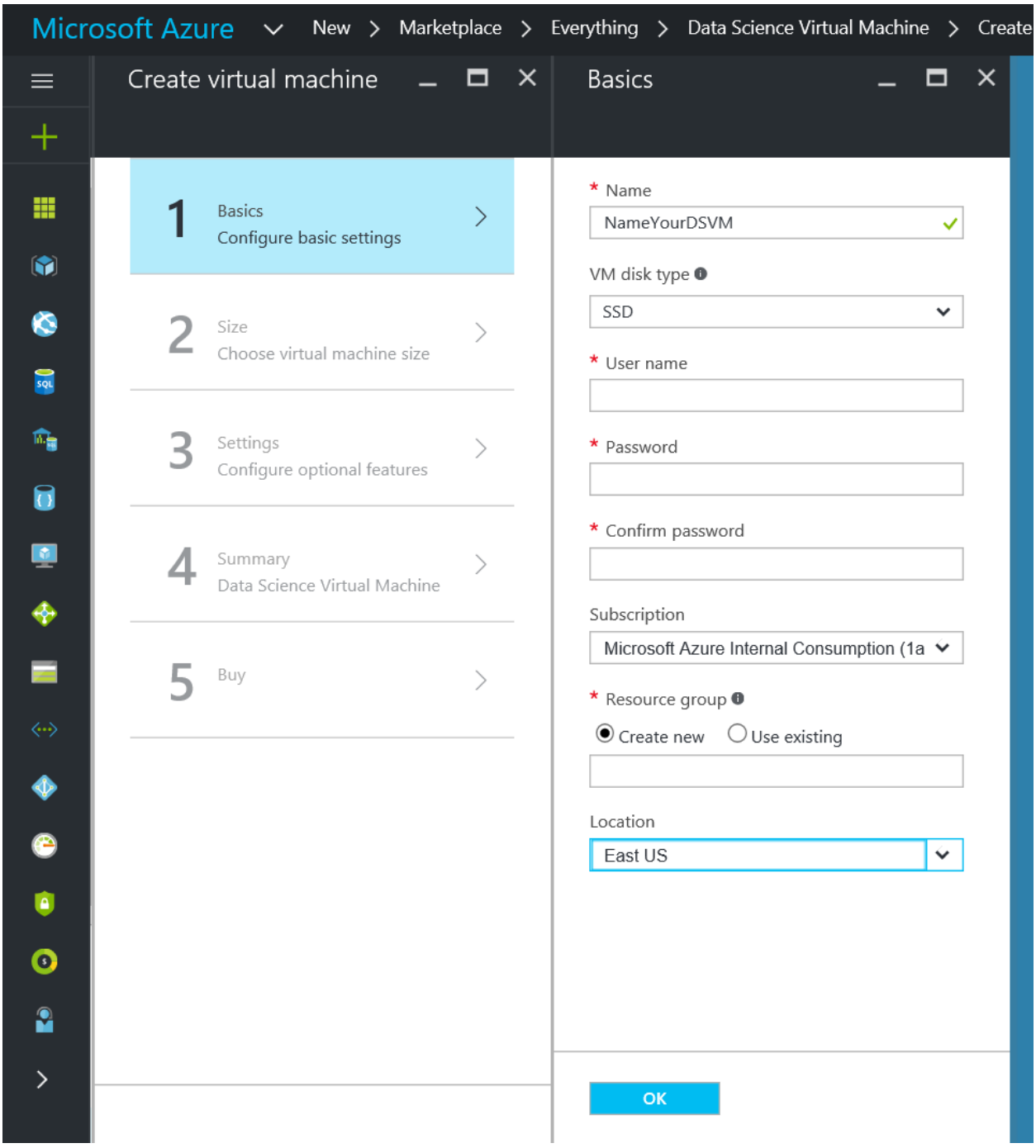
4. The search results should look like the figure below. Choose the version of the Data Science Virtual Machine (DSVM) that you wish to use, i.e. Windows or Linux (This walkthrough is for the Windows version, but the Linux version is very similar). The third option is the 'Deep Learning Toolkit for the DSVM', this is designed to be deployed on the NC class hardware instances in Azure that offer GPU based parallel processing. There are other specific walkthroughs for this.



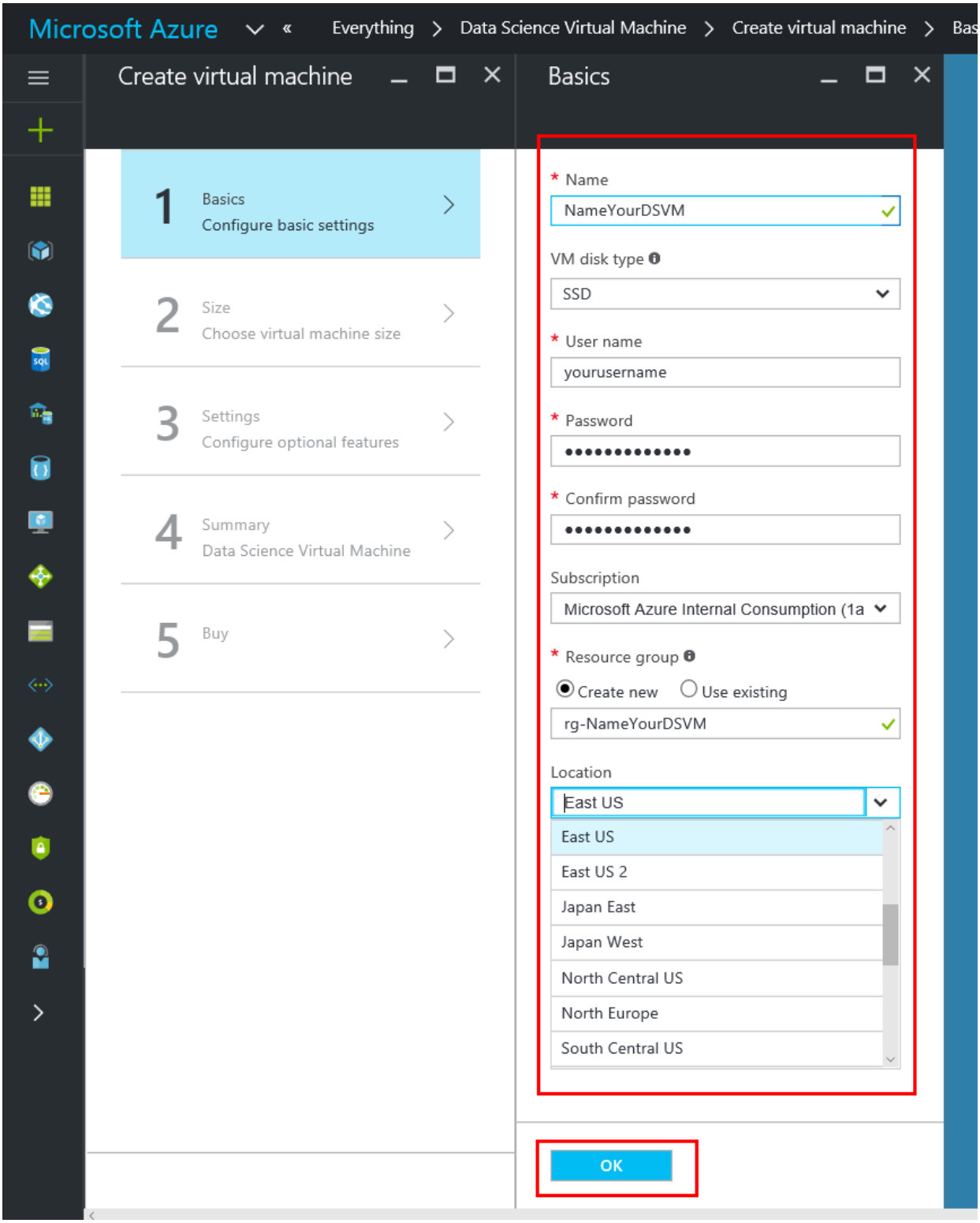
5. Once you close the VM version, you will see the next frame that has some specific information about the image you chose. You can click 'Create' to start the creation process. The deployment model for DSVM is pre-set to be Azure Resource Manager (ARM).



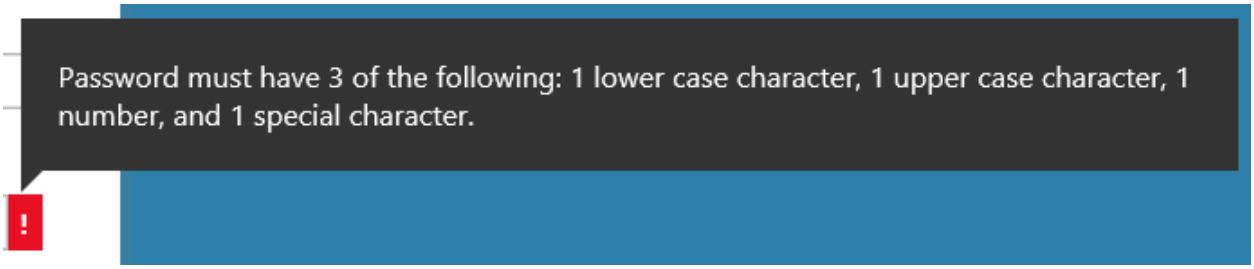
6. Once you press ‘Create’ you will arrive at the ‘Create Virtual Machine’ frame. Here, you will be going through 5 stages of creation: Basics >> Size >> Settings >> Summary >> Buy.



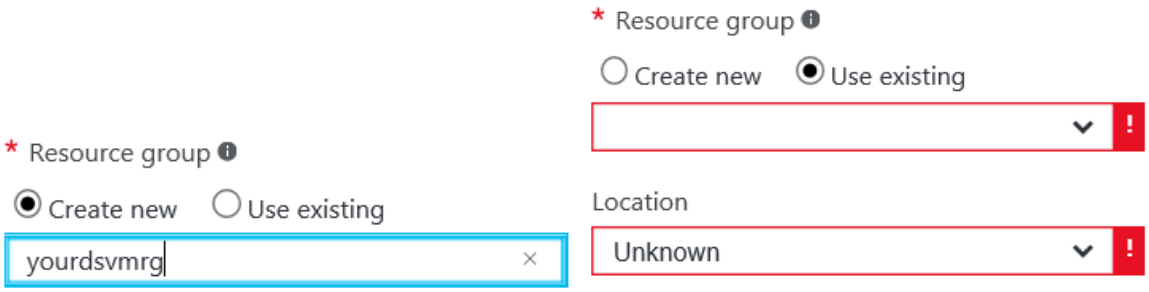
7. Fill out the fields in the 'Basics' tab, and chose the subscription if you have more than one and choose the data center region/location where you want your VM to be deployed to. Typically, geographically closest is the most common choice. Press 'OK' when you are done.



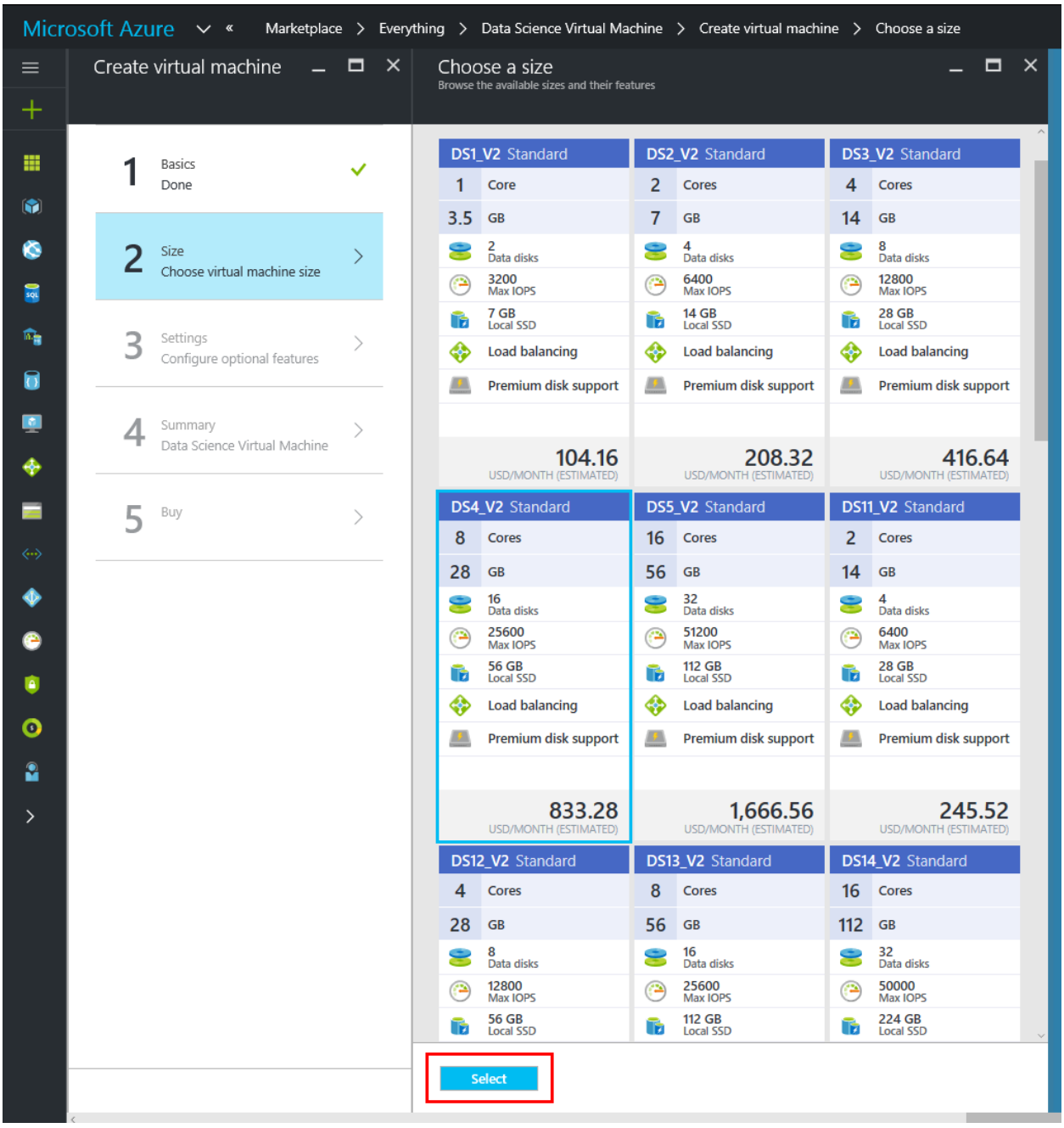
8. You may see a warning/error when entering your password as it would need to adhere to some complexity requirements, for security reasons, refer to the tooltip for exact requirements.



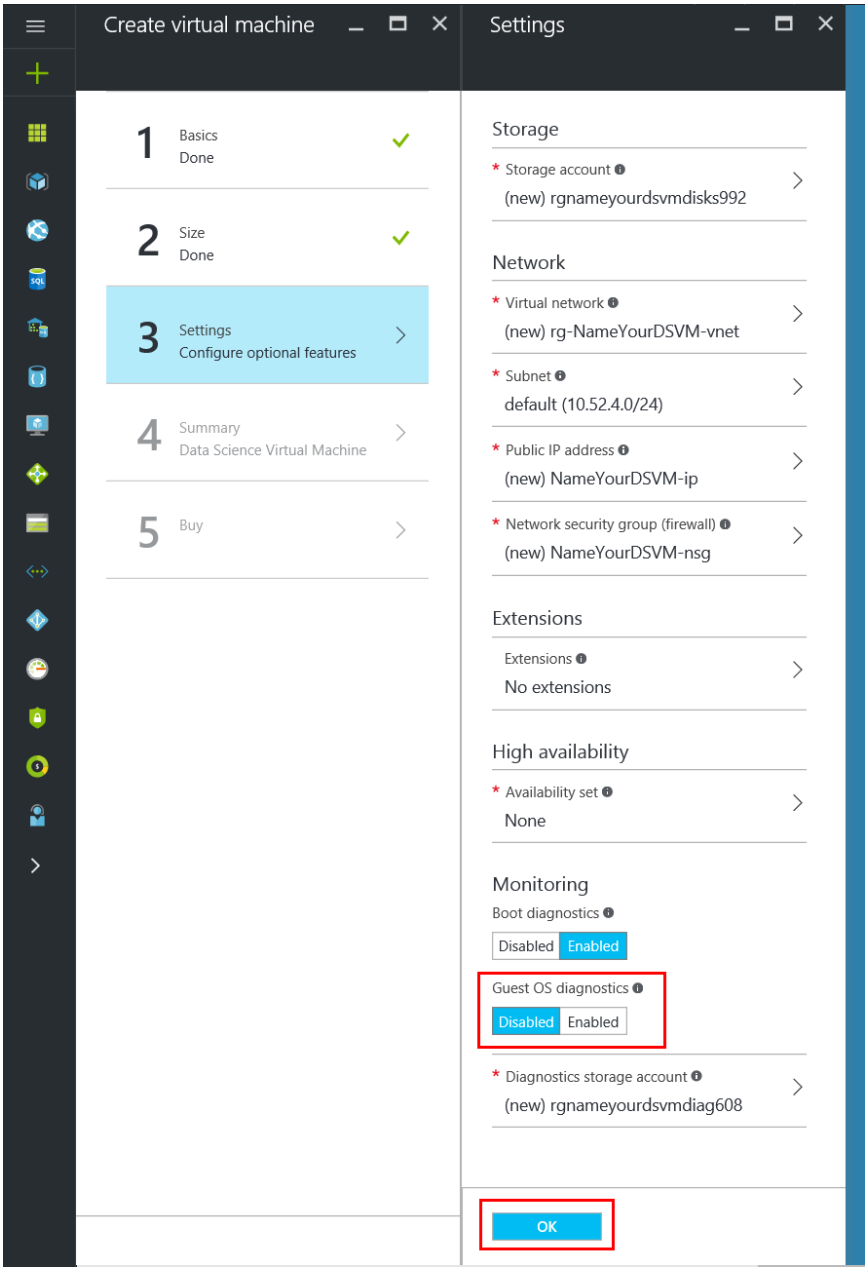
9. Additionally, you may either chose to create a new resource group to encapsulate all the related resources for your VM. This is a logical reference to collectively refer to related resources. You may also choose an existing resource group. Be sure to name resource groups in a manner where they are self-explanatory of the resources they encapsulate or their purpose. Click 'OK' when done.



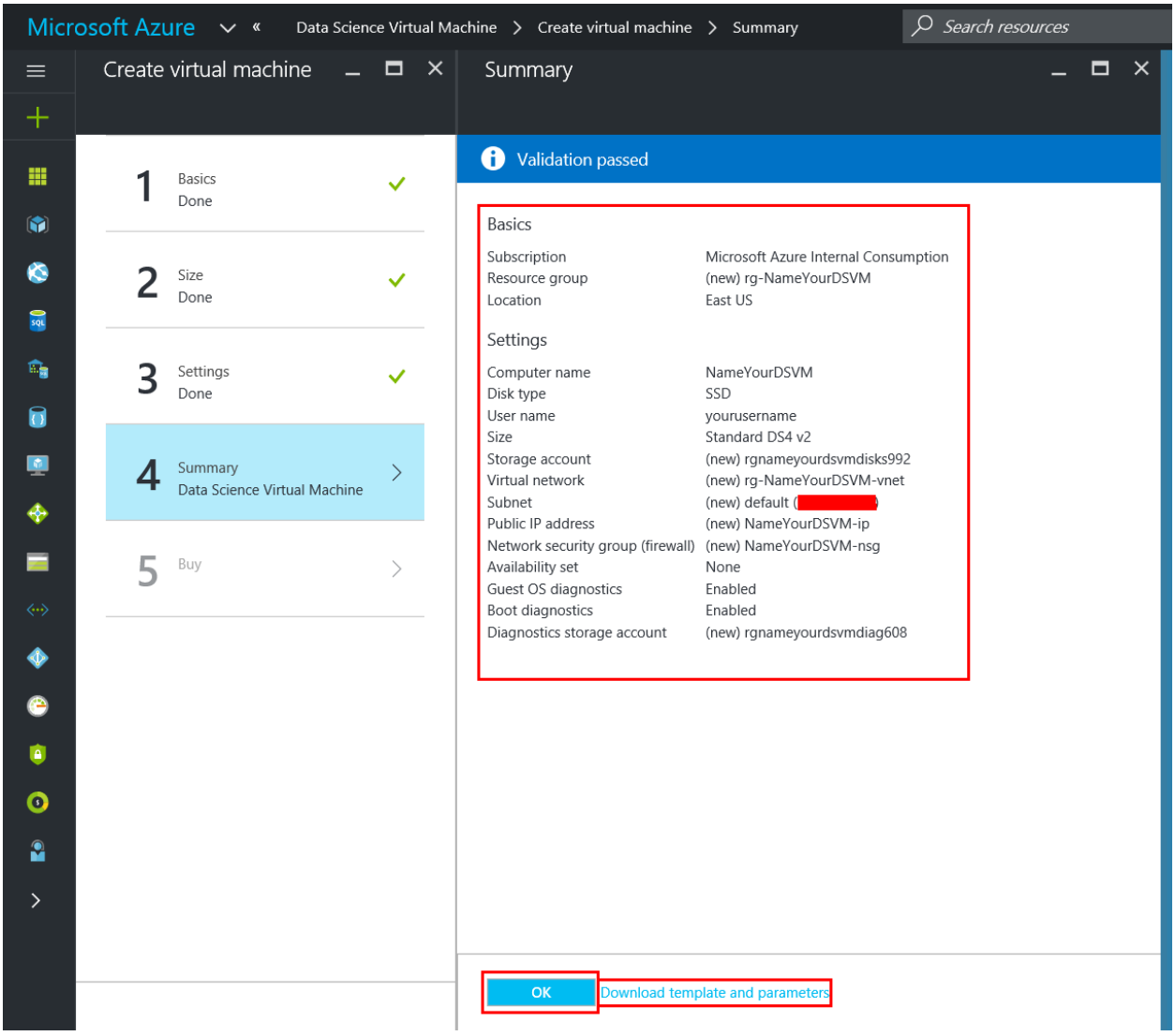
11. On the ‘Size’ tab, select the desired configuration of the machine that you want. Since the purpose of this instance is ‘Data Science’, we recommend choosing an instance size with 4 or more cores and 14GB or more of RAM. You can scale the instances up or down as per your requirement. Additionally, you can also shut down and deallocate the instances when not in use. This may be done programmatically as well. Click ‘Select’ to proceed to the ‘Settings’ tab.



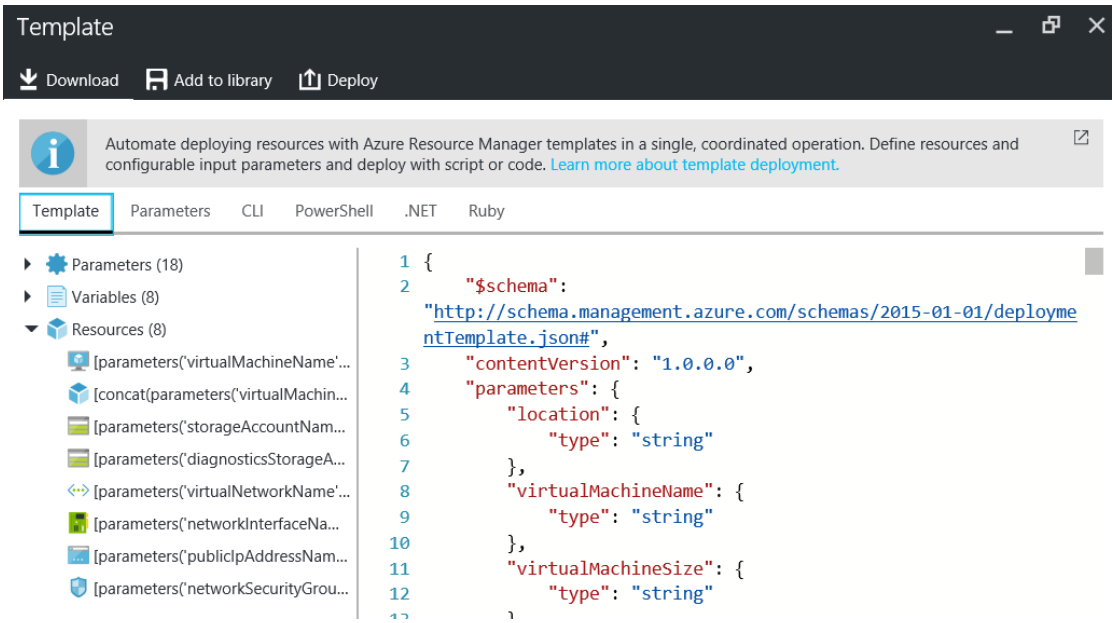
13. On the ‘Settings’ tab you can select or specify different locations or names etc. for some resources like storage and network. The form is pre-populated with some generated values that are automatically built from the initial names etc. that you specified in the ‘Basics’ tab. You can choose to change these or use these generated values. Additionally, there are a few settings to note. Here is where you can click on ‘Extensions’ and add a post provisioning extension like an Anti-Malware solution etc. or bring in your custom scripts and extensions. You can also enable ‘Guest OS diagnostics’ which is off by default. This gives you metrics every minute for the VM and can use them to create alerts and stay informed on your applications etc. A service like Operations Management Suite can ingest these feeds, they are stored in the specified diagnostic storage account. Press ‘OK’ when done.



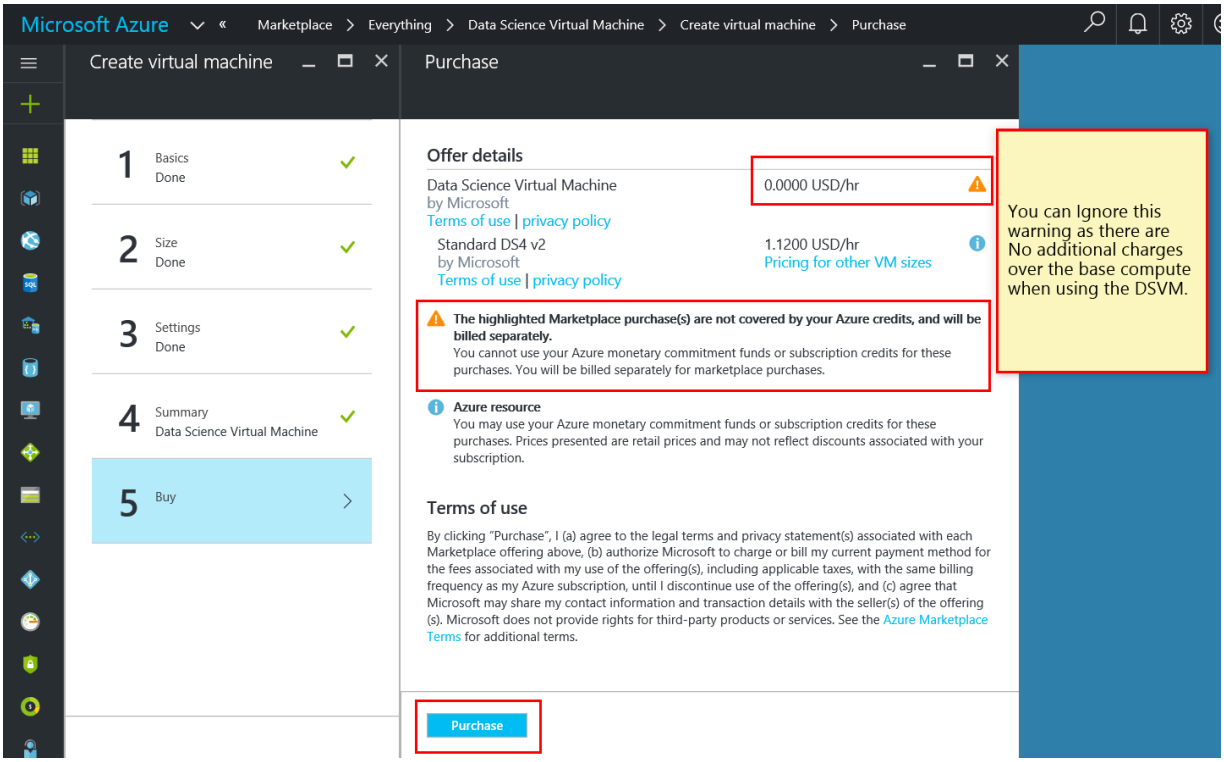
14. The Summary tab is mostly for review. Do verify the details that you have specified in the previous steps. This step also allows you to click on the link to “Download template and parameters”. This option allows you to procure the Azure Resource Manager (ARM) templates for your VM specification so you can deploy the same configuration programmatically without the use of the portal. Press ‘OK’ to continue to the ‘Buy’ tab once you have verified the information.



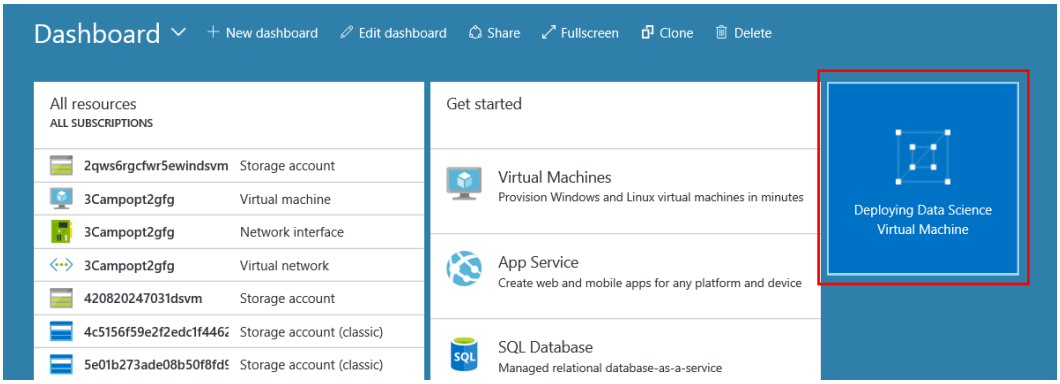
16. (OPTIONAL) If you do click on “Download Template and parameters”, the download interface will look like the figure below.



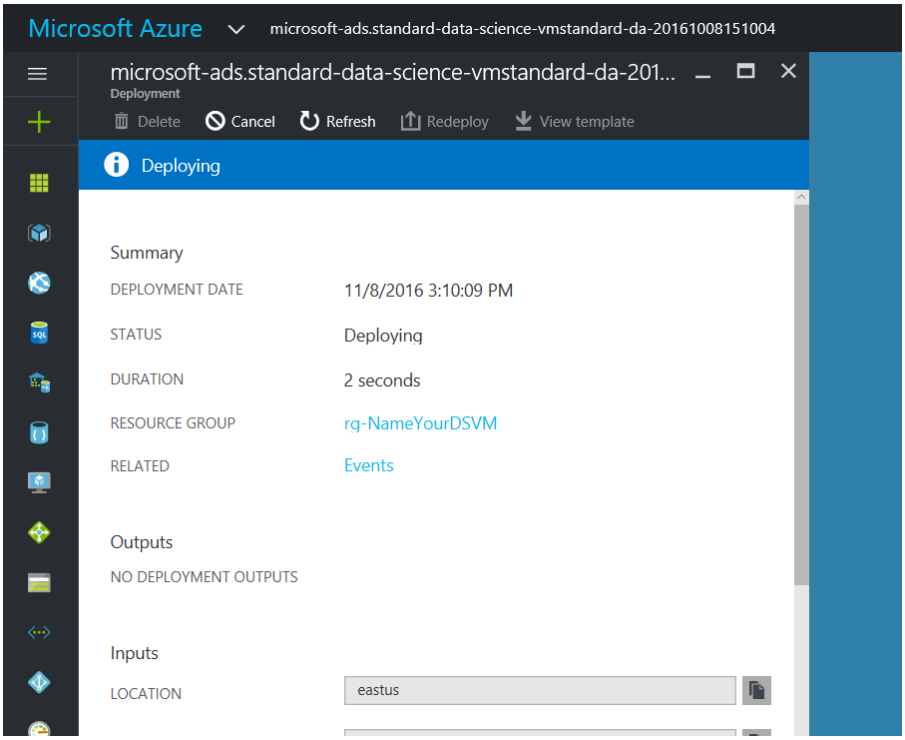
17. Once on the ‘Buy’ tab, you will see similar to the figure below. You will also see a ‘Marketplace Purchase’ charge warning. You can ignore this as this is a Zero-dollar warning and your actual per minute charges are only for the base compute as reflected in the compute charge line item. When ready, click ‘Purchase’.



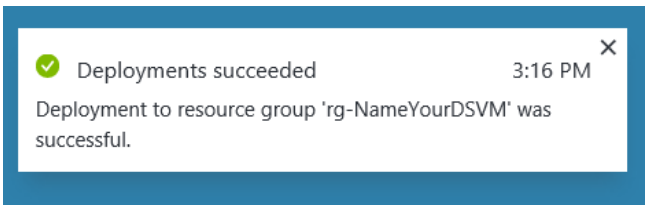
18. Once you click ‘Purchase’, you will be taken back to the portal dashboard and a new tile will show up like the figure below. This will read, “Deploying Data Science Virtual Machine”. Your DSVM is now being provisioned along with its related resources. This process will take approximately 5 minutes.

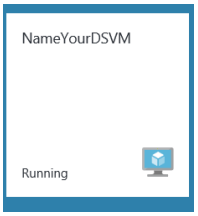


19. If you click on the “Deploying Data Science Virtual Machine” tile, you will see some of the deployment details and it will give you the options to ‘Cancel’ or ‘Refresh’.



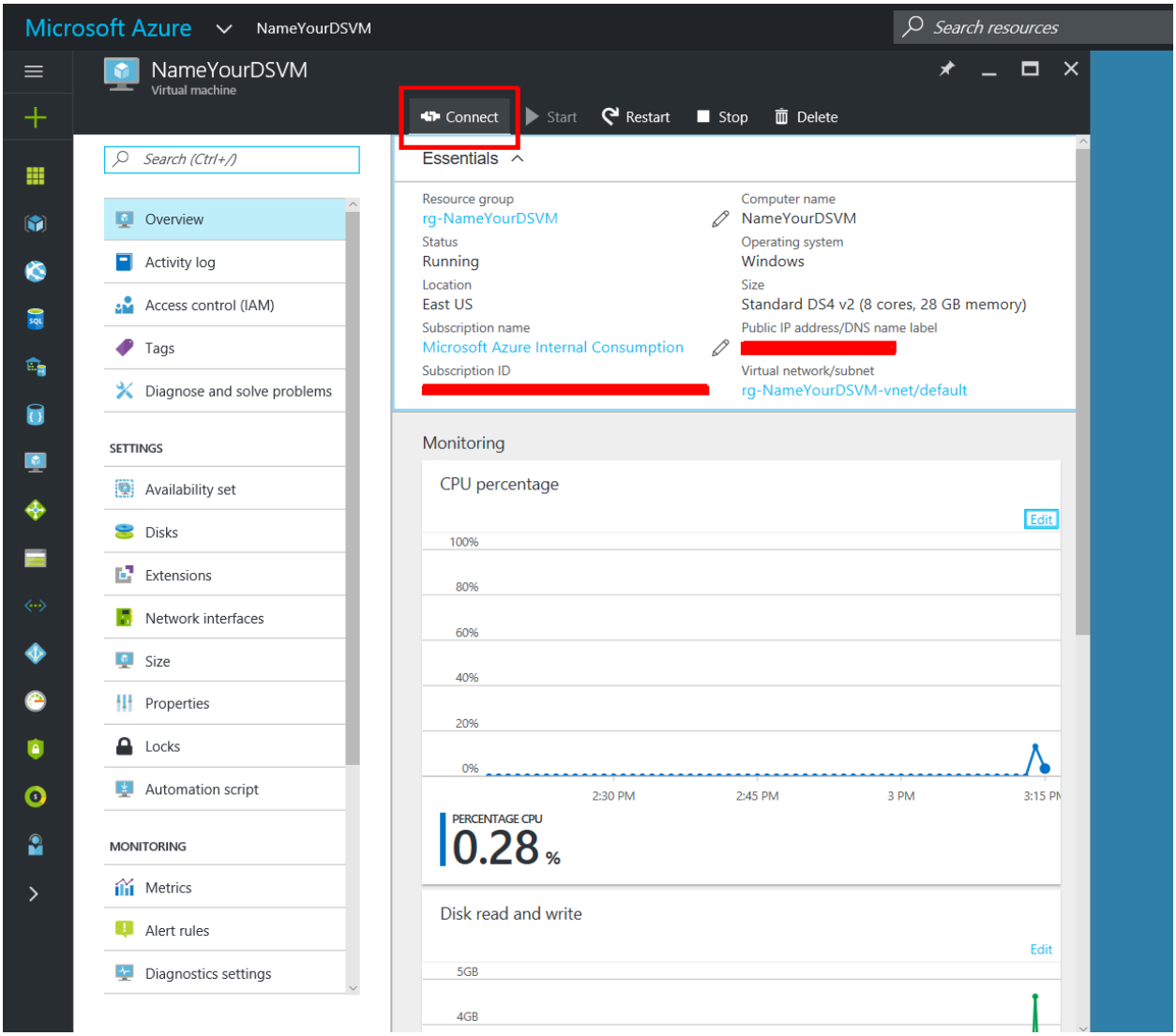
20. Once complete (after about 5 minutes) you will see a notification on the top right that reads “Deployment Successful” and your tile will change to the previously specified name for your DSVM.





Connecting to a Data Science Virtual Machine – Using Remote Desktop & Exploring the VM and some of the installed tools

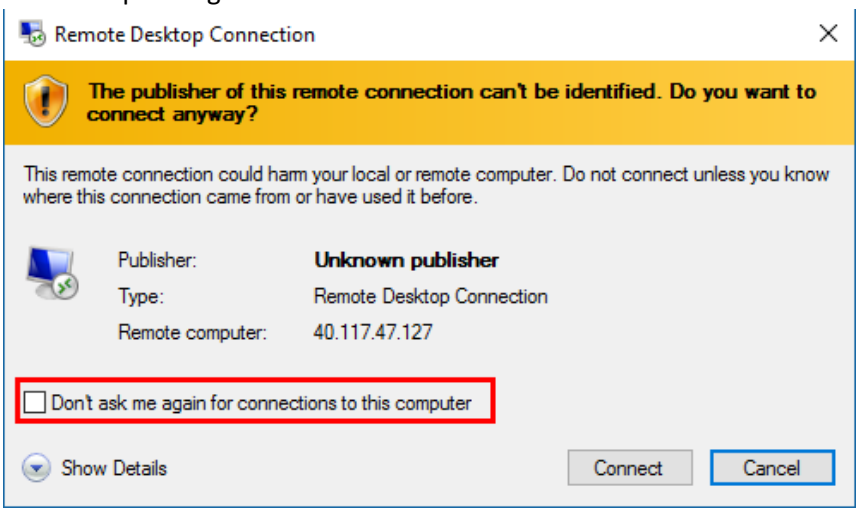
21. You can now Click on the Tile for your DSVM. This will open the frame for your VM and give you numerous options, links and charts with dashboards showing the per minute diagnostics etc. if you had turned them on. You can now Click on the "Connect" button to download the RDP (remote desktop) file which is pre-configured with your VM's details. You can also use this frame to restart, stop or delete the VM.



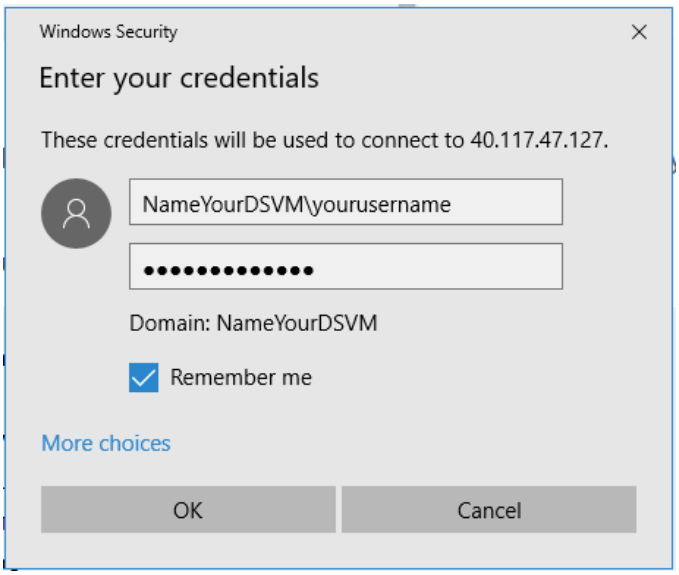
Save the RDP file to a location of your choice on your Local Machine.

Alternatively you can also connect to the VM using the public IP on the Windows Remote Desktop Connection Client.

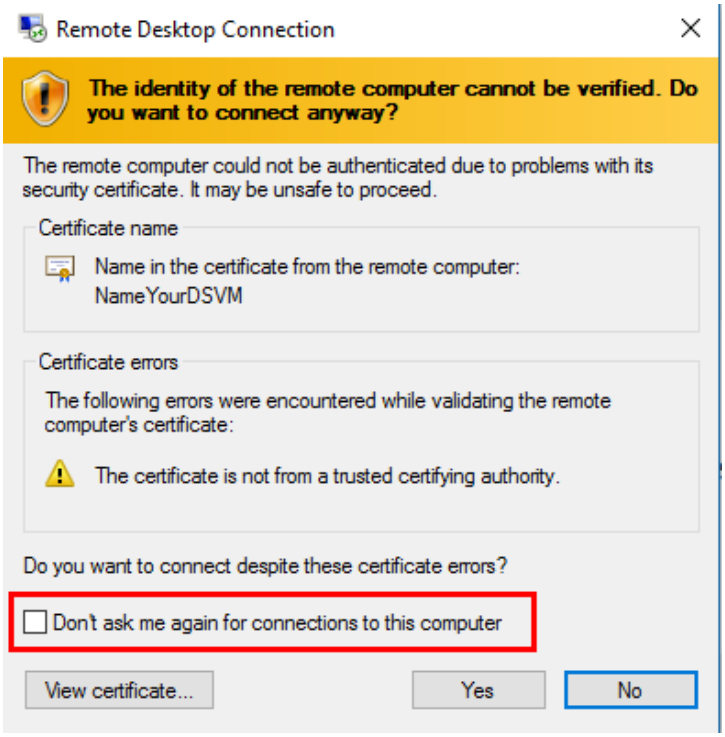
22. The first time you connect using this RDP connection to your created VM, you will see a publisher warning. Please ignore this and tick the 'Don't ask me again for connections to this computer' tock box to stop seeing this.



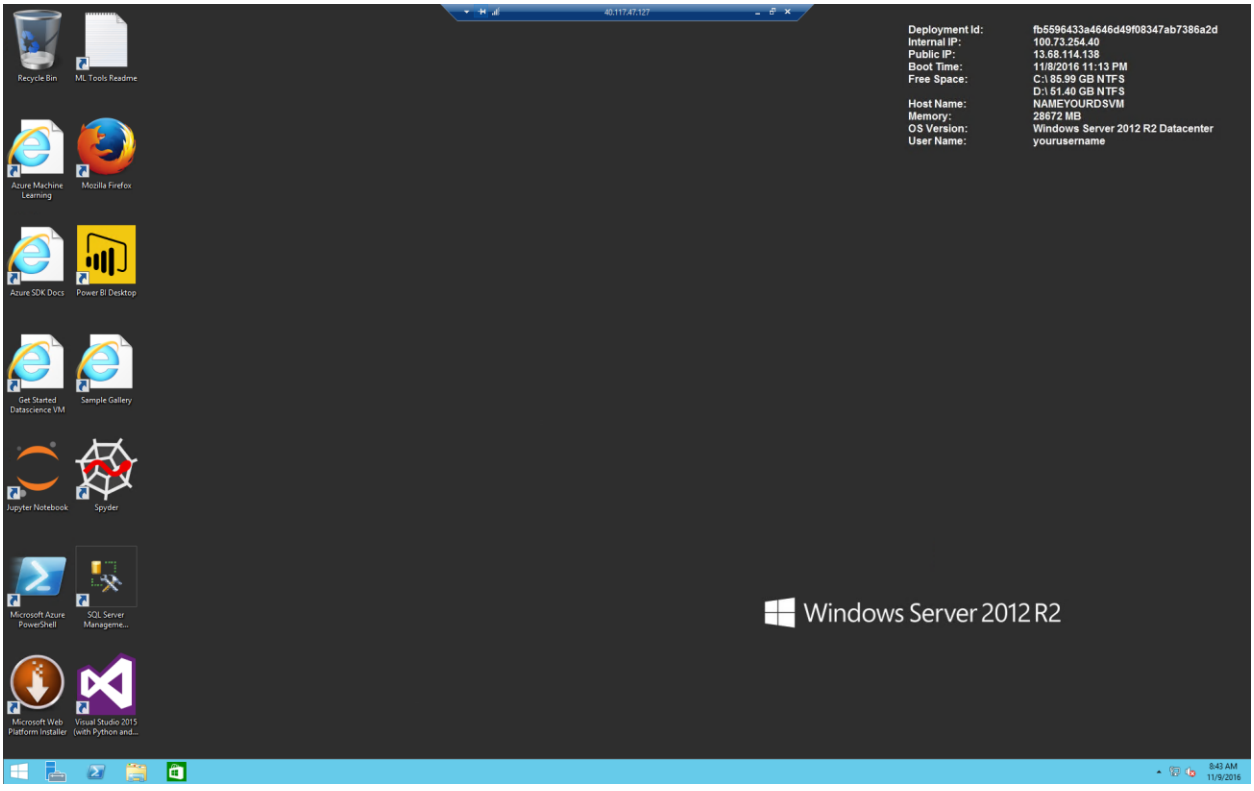
23. Enter your credentials.



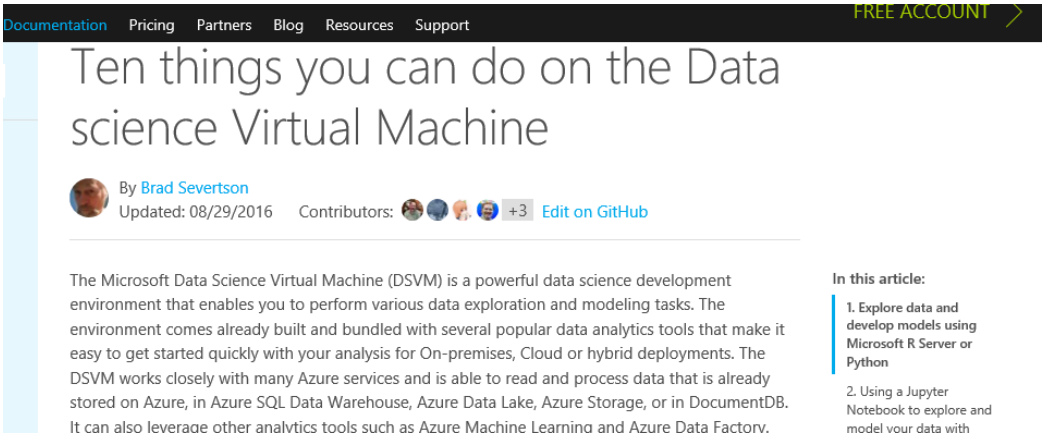
24. You will get another warning is because the VM is not in a domain yet where there is a trusted certificate provider. Please ignore this and tick the 'Don't ask me again for connections to this computer' tock box to stop seeing this.



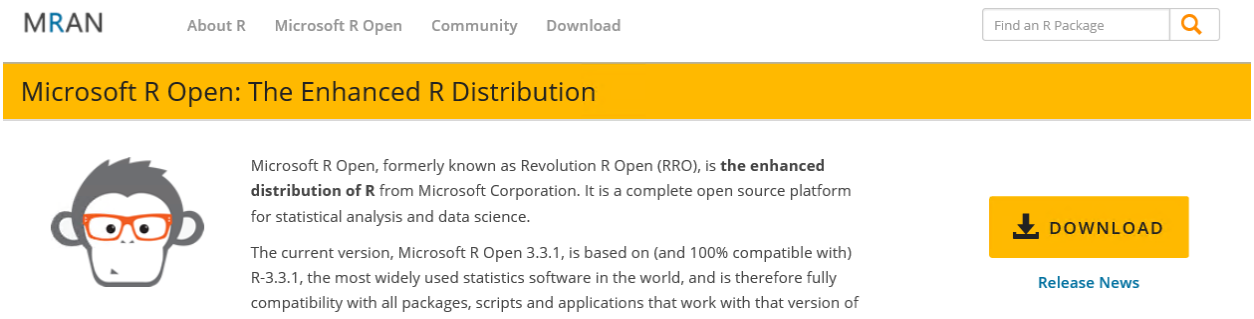
25. You will now be logged in to your DSVM:



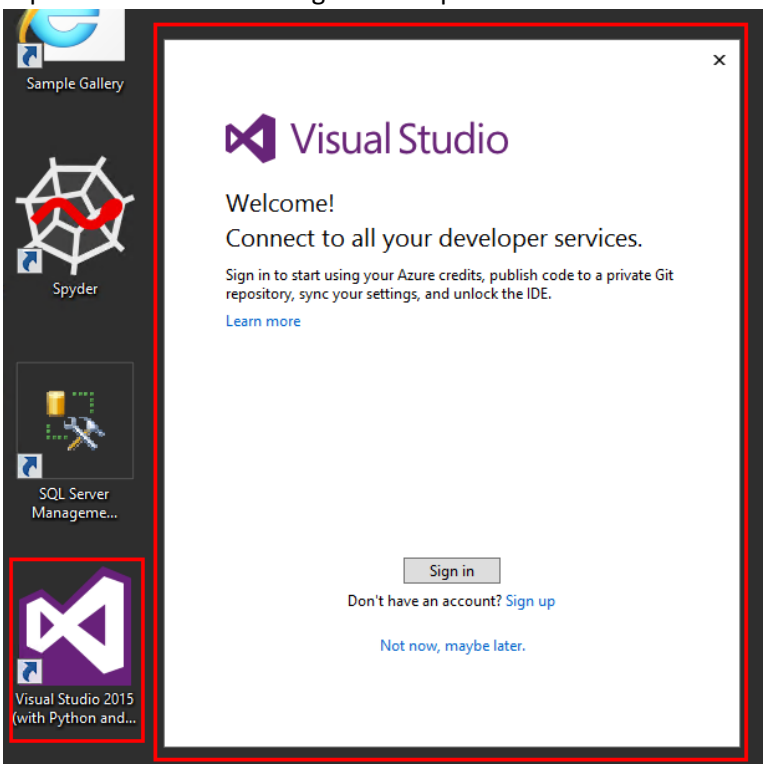
26. Check out the “Getting Started - 10 things You can do on DSVM” article by opening the shortcut link on the desktop, or going directly to http://aka.ms/dsvmtenthings.



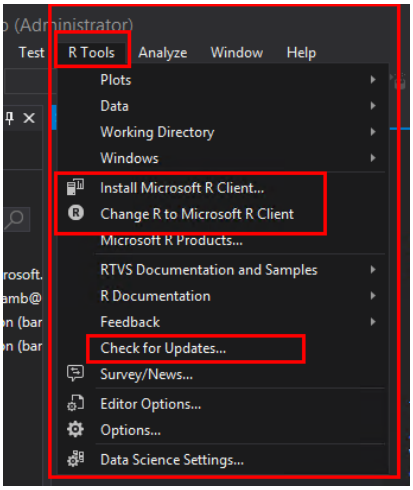
27. Update your underlying Microsoft R Open environment if you wish from: https://mran.microsoft.com/open

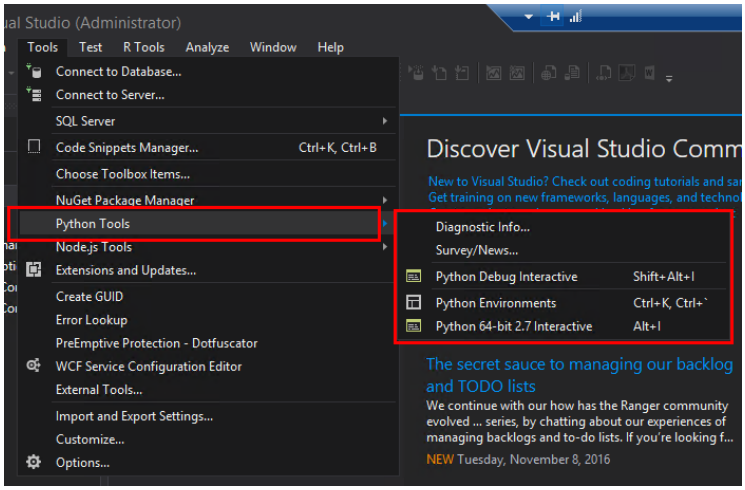


28. Open Visual Studio and sign in or skip.

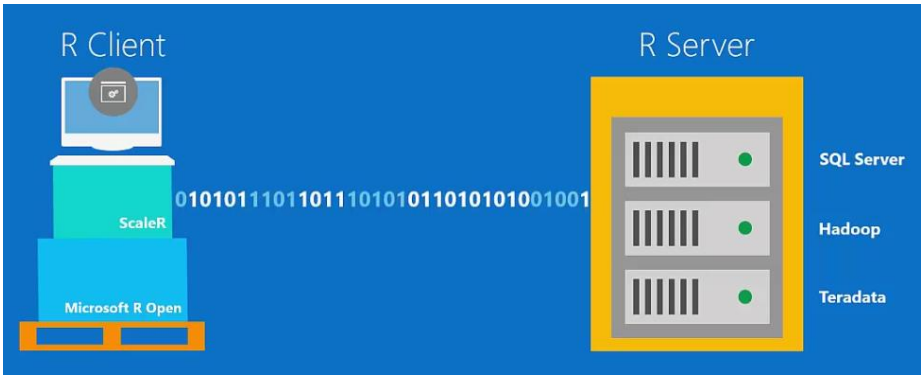


29. As you can see R & Python Tools for Visual Studio are pre-installed (RTVS & PTVS), you may check for updates. Please click Install Microsoft R Client and run through the installation to ensure you have the latest version.



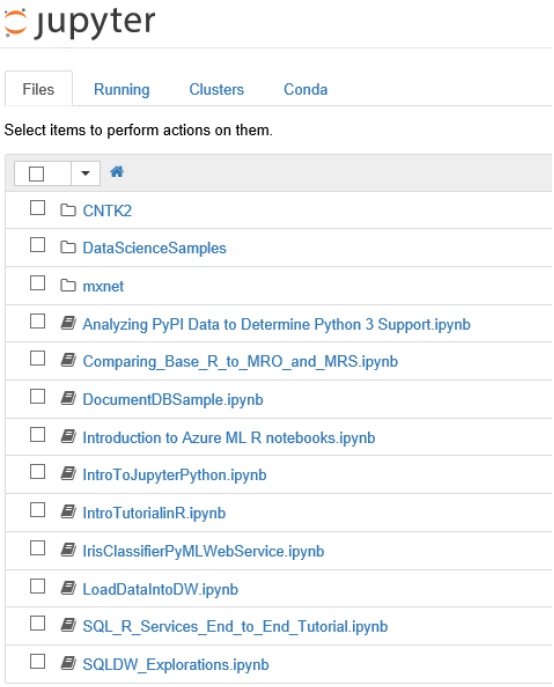


30. You may also install an IDE of choice like RStudio if you wish to use it. You may also wish to update the Version of Microsoft R client by going to this link: http://aka.ms/rclient/download. Some customers also prefer to use Microsoft R Server & R Services on SQL Server 2016. This introduction video for Microsoft R client should automatically start playing, but you will not get Audio from the VM so get the introductory video on your local machine from: http://aka.ms/rclient/intro.



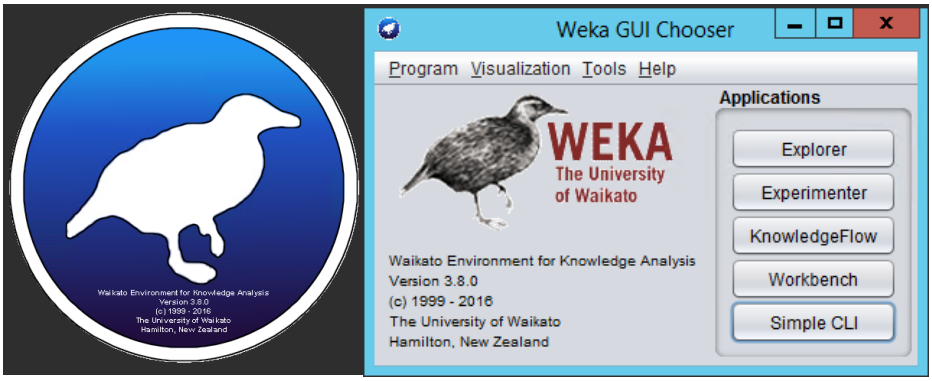
31. The DSVM is also a fully configured Jupyter Notebooks Server (Disabled By default for security, you have to first set up your own password). Please make sure you set up jupyter security after first log-on by using the utility on the desktop.

You can then open Jupyter from the desktop or browse to “https//localhost:9999/”.

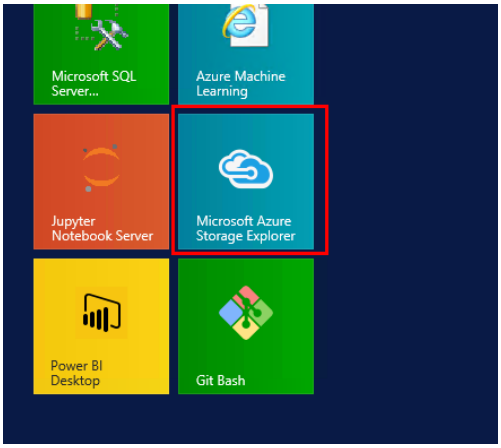


The DSVM comes Loaded with multiple examples.

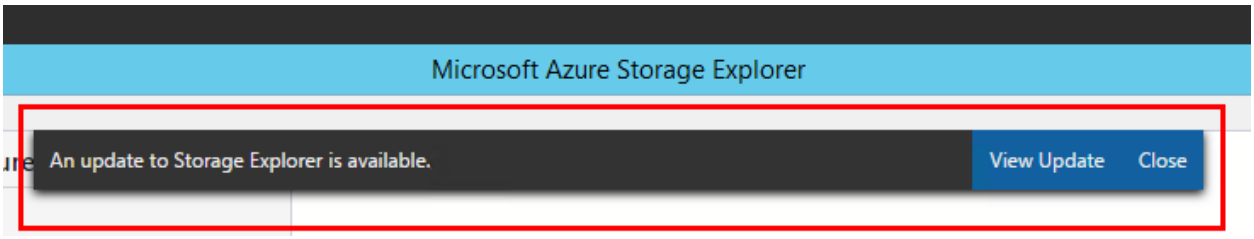
32. DSVM also has WEKA (Waikato Environment for Knowledge Analysis - popular data science tool) pre-installed. Please refer to http://www.cs.waikato.ac.nz/ml/weka/documentation.html for tutorial and resources.



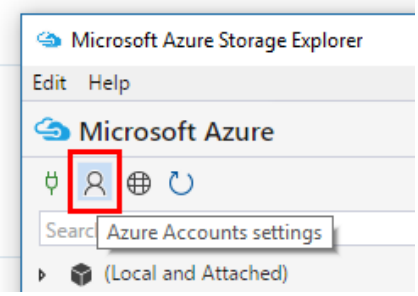
33. Try out Azure Storage Explorer:

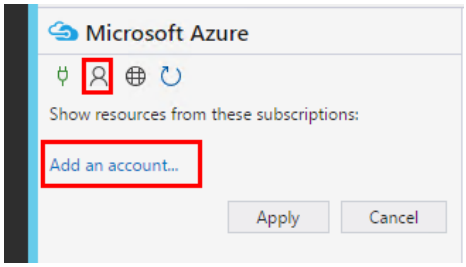


Install the update to Azure Storage Explorer if prompted:

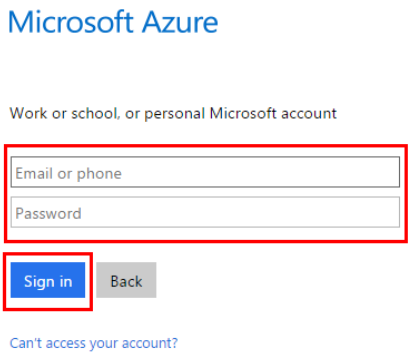


34. Add your Azure Account by signing in to see your subscription(s) and associated storage accounts:

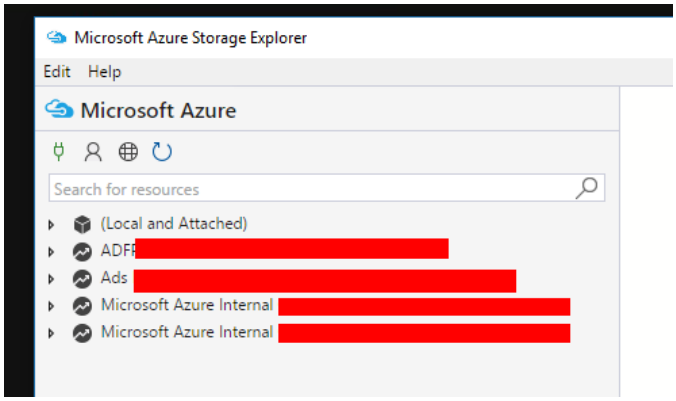




Fill out your credentials:

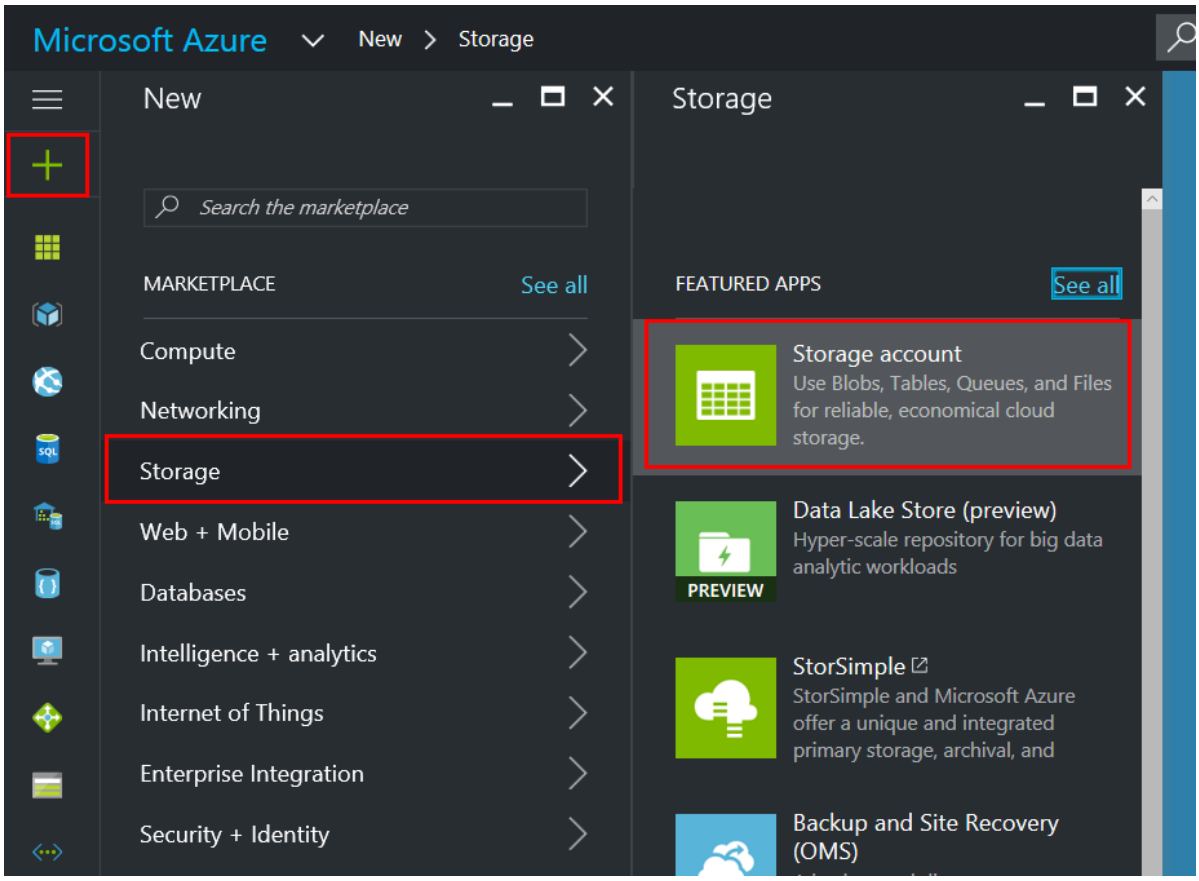


35. Once Signed in your view should look like the figure with the subscription(s) listed as dropdowns. When you expand the drop downs, you will see a list of all your storage accounts, which further expand to Blobs, Files, Queues and Tables. For a detailed introduction, refer to "Getting Started with Storage Explorer" (https://azure.microsoft.com/en-us/documentation/articles/vs-azure-tools-storage-manage-with-storage-explorer/).

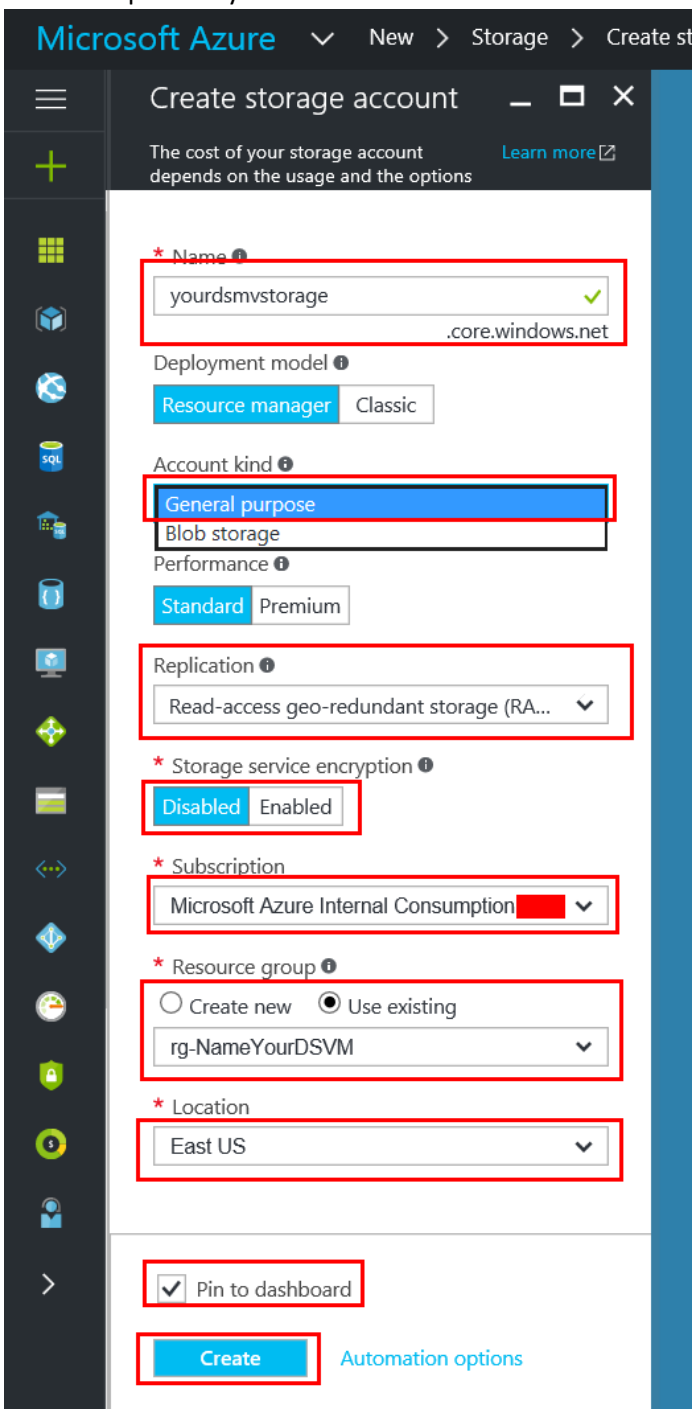


Creating an Azure Storage Account and a 5TB Shared File store and Mounting the Shared Storage as a drive on the VM

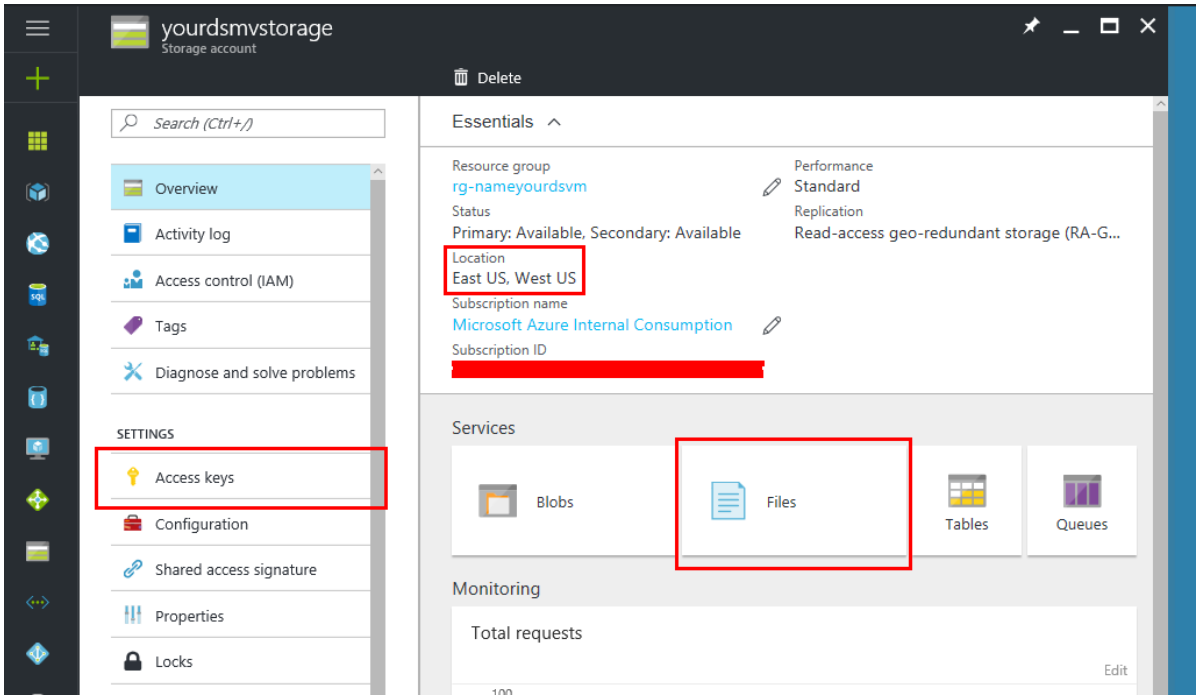
36. Confirm that you have an existing storage account. If not, you can go and create one using the Azure Portal: https://portal.azure.com. On the portal navigate to New >> Storage >> Storage Account



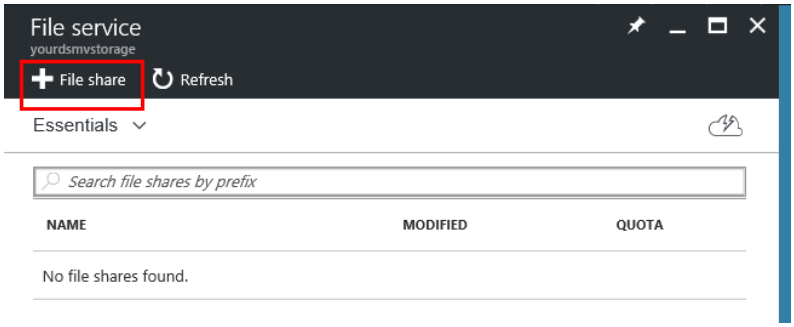
37. On the Create Storage account pane, enter your desired name for your storage account and change the default values if you wish to do so. You can change settings like Performance, Encryption, Replication etc. You may choose the same resource group as your VM that you created earlier or may choose another group or create a new one. It is best to have your storage in the same Azure location as your Data Science VM to avoid latency and data movement charges. You may also choose to pin it to your dashboard. Click Create when ready.



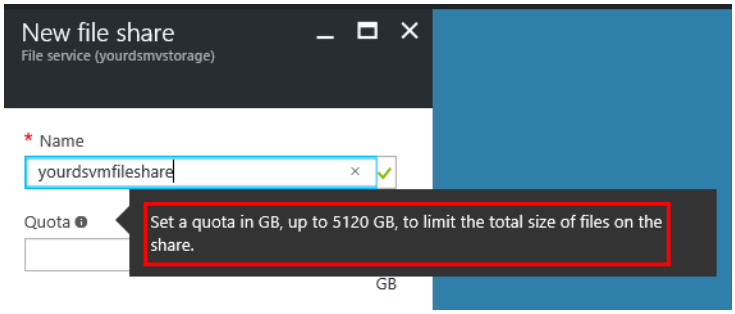
38. Once created you will see the storage account screen which should look like the figure below. If you select geo redundant storage, note that there will be a secondary location that will also be displayed. Since this is a general-purpose account you may create storage types of Blobs, Files, Tables & Queues.



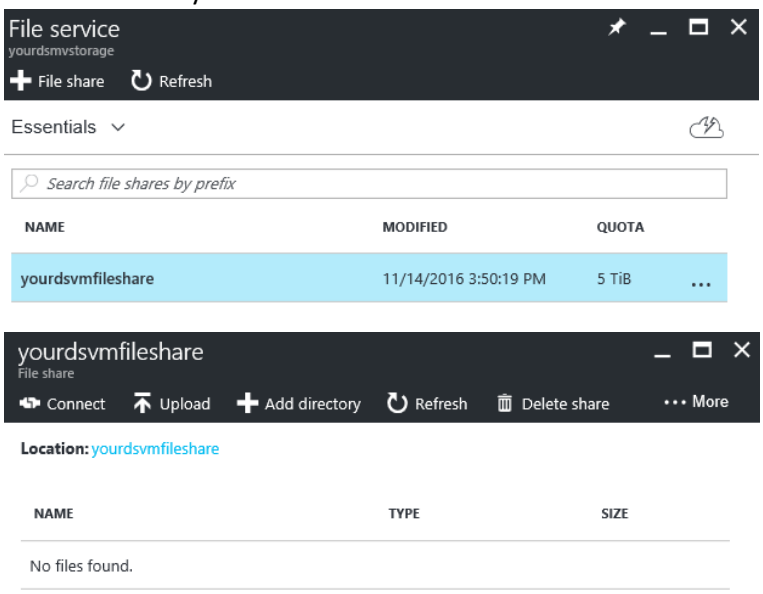
39. Click on 'Files' to show to show the File service pane and then click on '+ File share' to create a new File share.



Specify the 'Name' of your file share and optionally specify the maximum quota for size. By default, it chooses the maximum value of 5120 GB (5 TB). Press 'Create' to complete creation.



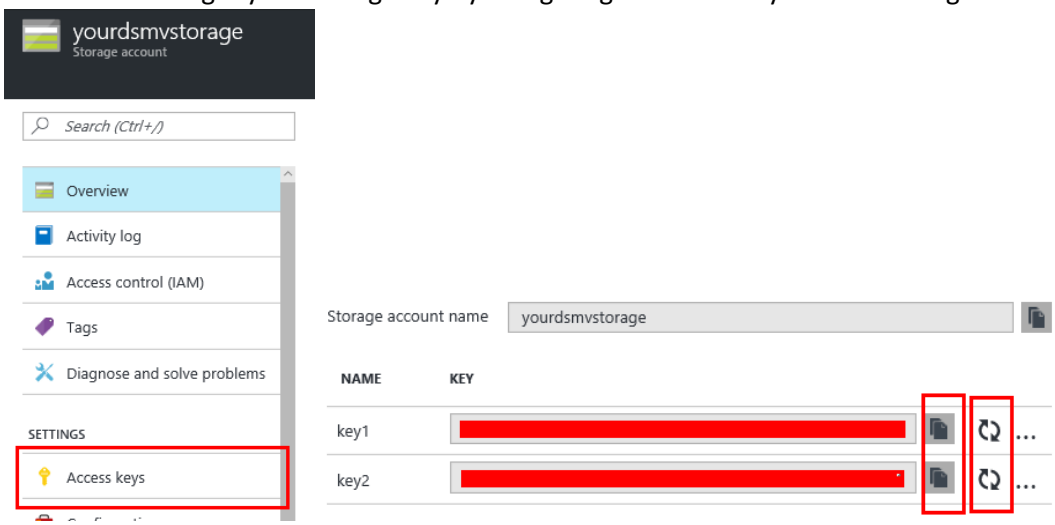
You should see your new file share.



40. If you wish to create File Shares programmatically you can use the following PowerShell script on the pre-installed PowerShell executable.

```
# Authenticate to Azure.
Login-AzureRmAccount
# Select your subscription
Get-AzureRmSubscription –SubscriptionName "<your subscription name>" | Select-
AzureRmSubscription
# Create a new resource group.
New-AzureRmResourceGroup -Name <rg-yourdsvmdata>
# Create a new storage account. You can reuse existing storage account if you wish.
New-AzureRmStorageAccount -Name <mydatadisk> -ResourceGroupName <rg-yourdsvmdata> -
Location "<Azure Data Center Name For eg. East US>" -Type "Standard_LRS"
# Set your current working storage account
Set-AzureRmCurrentStorageAccount –ResourceGroupName "< rg-yourdsvmdata >" –
StorageAccountName <mydatadisk>

# Create a Azure File Service Share
$s = New-AzureStorageShare <<teamsharename>>
# Create a directory under the FIle share. You can give it any name
New-AzureStorageDirectory -Share $s -Path <directory name>
# List the share to confirm that everything worked
Get-AzureStorageFile -Share $s
```

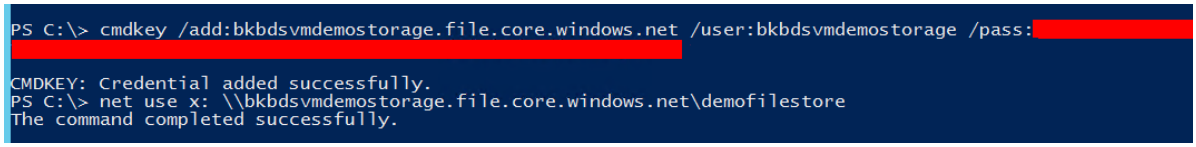
41. This 5TB File share can be easily mounted as a shared drive on the Data Science Virtual Machine. This is a very convenient tool for sharing moderately sized data assets across team members as they can be mounted to multiple machines at the same time. Additionally, you can also create multiple File shares and mount them simultaneously on the same machine. As a best practice guidance, we recommend starting from 'Z' and then coming backwards when mounting File Shares with Drive Letters. You can get your storage key by Navigating to Access Keys on the Storage account pane:



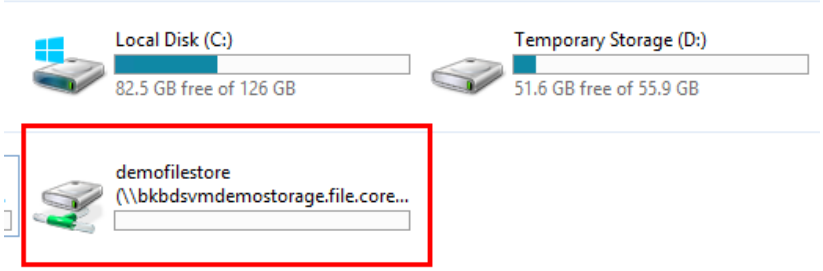
Use the copy buttons to copy one of the keys or the refresh button to generate new keys.
Here are the commands to mount the drive on the DSVM that you can run on Azure PowerShell.

Get storage key of the storage account that has the Azure file share from Azure portal. Store it securely on the VM to avoid prompted in next command.
cmdkey /add:<<mydatadisk>>.file.core.windows.net /user:<<mydatadisk>> /pass:<storage key>

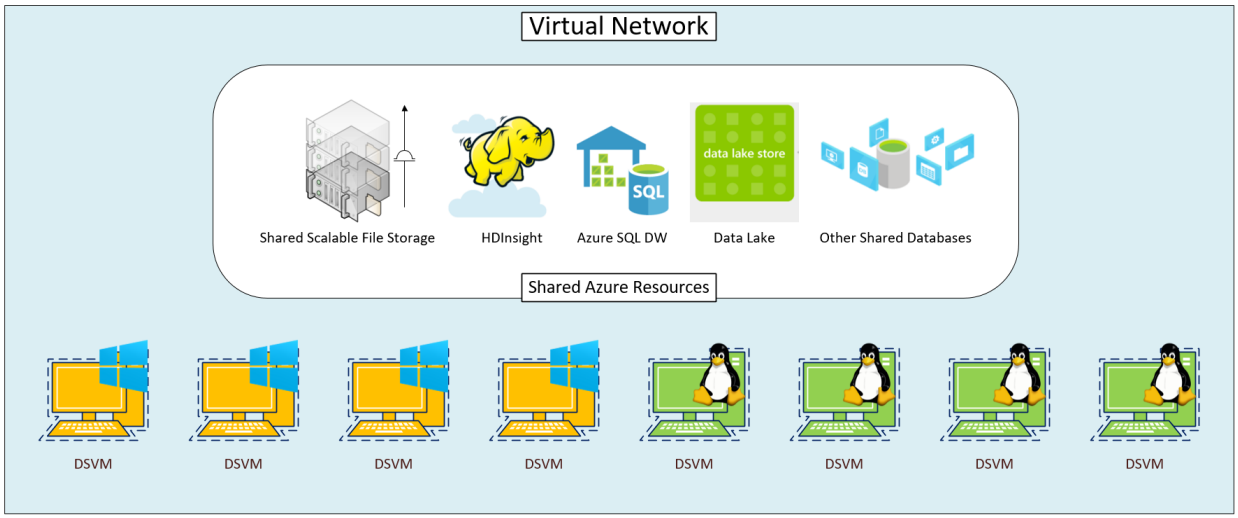
Mount the Azure file share as Z: drive on the VM. You can chose another drive letter if you wish
net use z: \\<mydatadisk>.file.core.windows.net\<<teamsharename>>



The Mounted Drive should now show in your explorer:



42. The Shared resource model is key to success and collaboration:

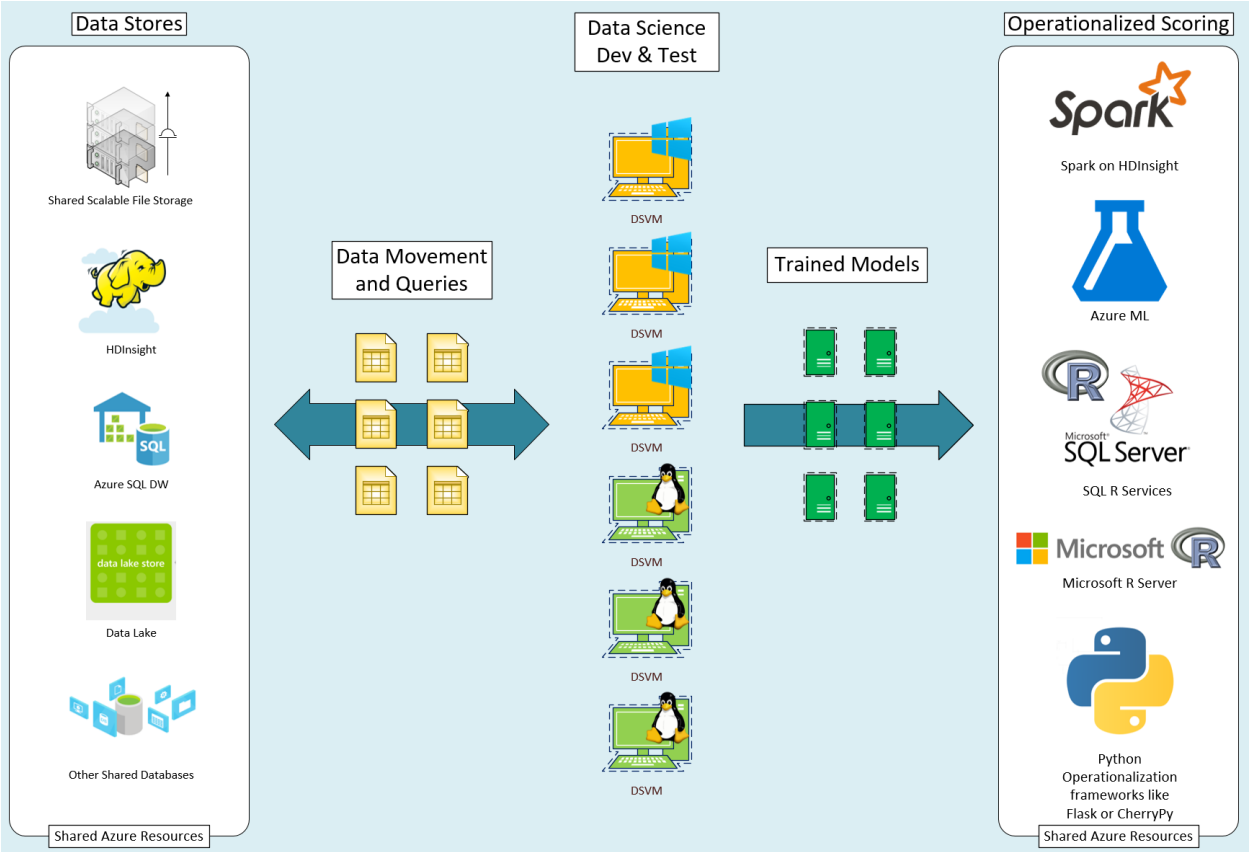


43. To copy files back and forth from Azure Blobs you may also use the pre-installed AzCopy utility:

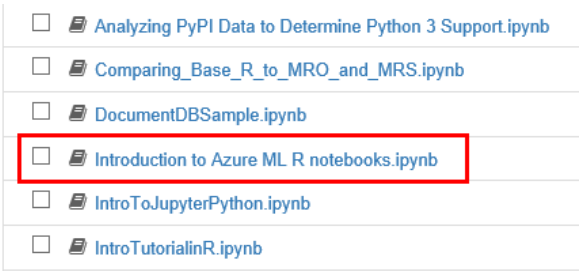
```
# Copy *.sql from local machine to a Azure Blob
"C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\azcopy" /Source:"c:\Aaqs\Data Science
Scripts" /Dest:https://[ENTER STORAGE ACCOUNT].blob.core.windows.net/[ENTER CONTAINER]
/DestKey:[ENTER STORAGE KEY] /S /Pattern:*.sql

# Copy back all files from Azure Blob container to Local machine
"C:\Program Files (x86)\Microsoft SDKs\Azure\AzCopy\azcopy" /Dest:"c:\Aaqs\Data Science
Scripts\temp" /Source:https://[ENTER STORAGE ACCOUNT].blob.core.windows.net/[ENTER
CONTAINER] /SourceKey:[ENTER STORAGE KEY] /S
```

Typical Data Science Walkthrough using the DSVM – From Getting Data and Creating Models to publishing Trained models as web services and consuming them for scoring – (R, Jupyter Notebooks, RTVS, Azure ML)

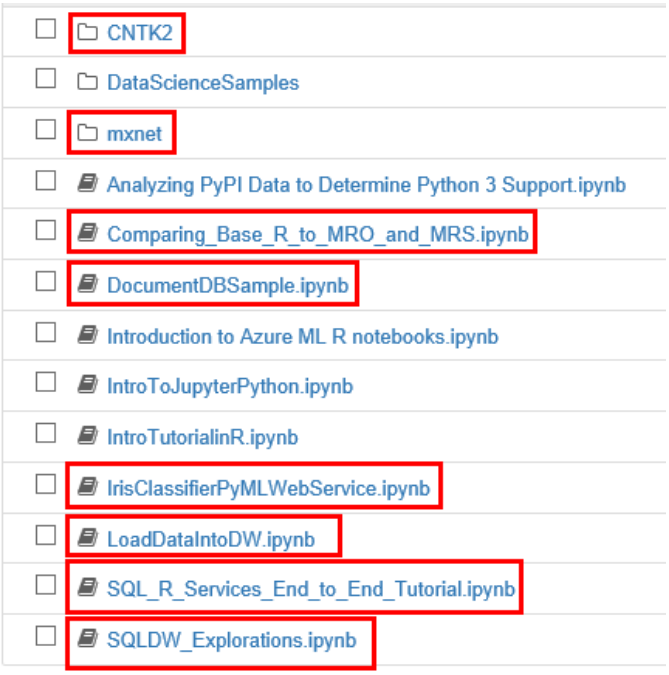
44. The DSVM Dev Test to Model Operationalization Workflow Looks like:



45. To better understand this workflow, the we can run through one of the examples already provided on the DSVM's Jupyter Notebook Server. The same can be done in R Tools for Visual Studio by copying the code over.
    We will go through the "Introduction to Azure ML R notebooks.ipynb" notebook.

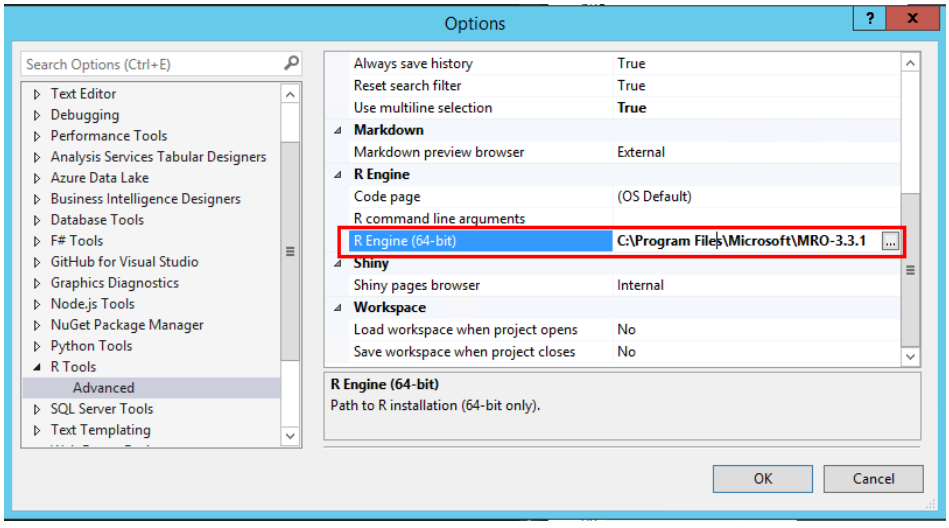


46. There Are Multiple Notebooks that show how to connect to several Azure and other Data Stores Like Document DB, SQL Server, Azure SQL Data Warehouse etc. Additionally, there are examples for doing Deep Learning tasks using the Deep Neural Network and Cognitive Frameworks like mxnet and CNTK2 (Also known as the Microsoft Cognitive Toolkit). These are already pre-installed on the DSVM.

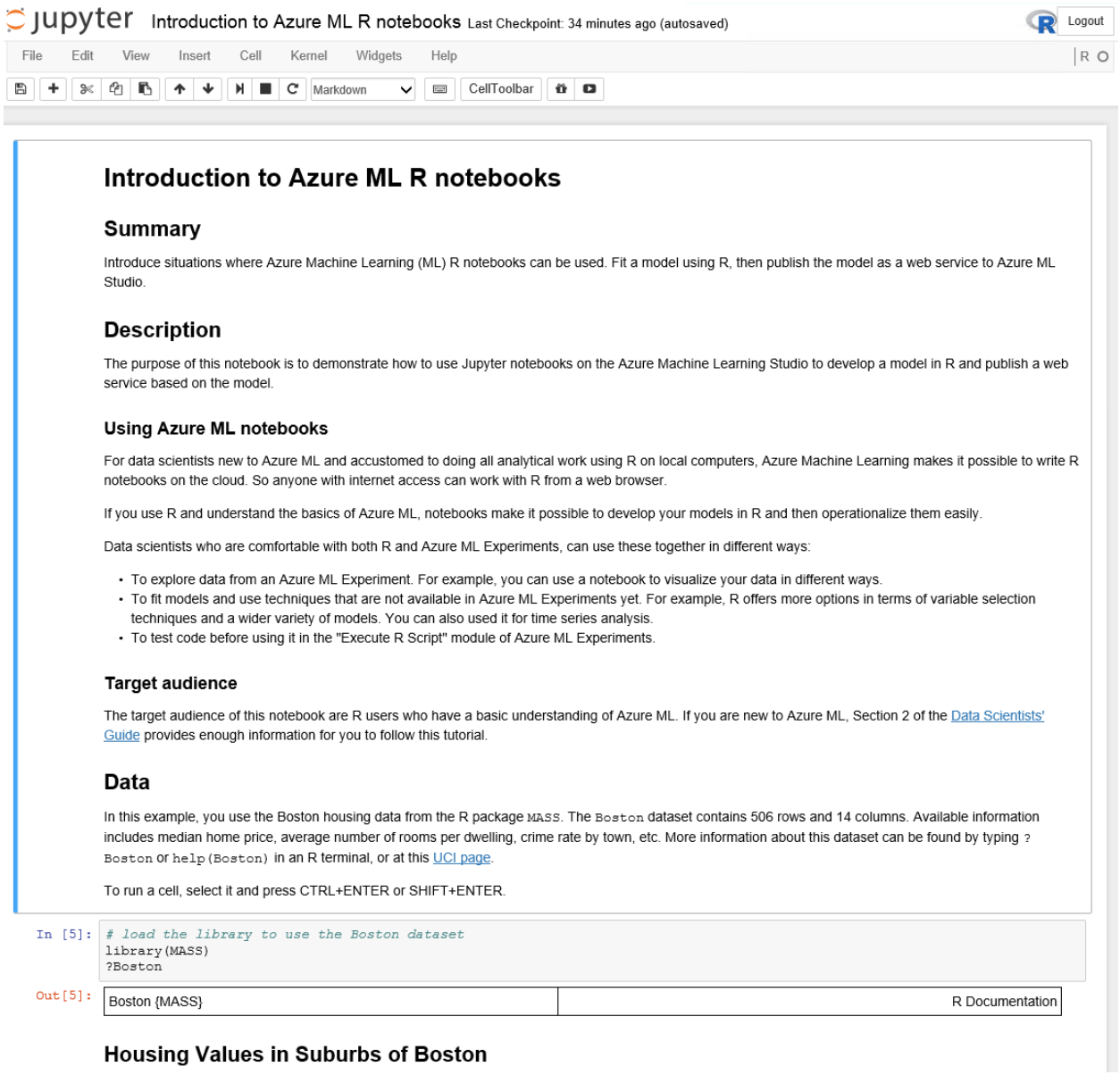


47. Open the "Introduction to Azure ML R notebooks.ipynb" notebook. This Notebook fits a model using R, then publishes the model as a web service to Azure ML Studio. If you are unfamiliar with Azure ML, Section 2 of the Data Scientists' Guide provides enough information for you to follow this tutorial: https://gallery.cortanaintelligence.com/Experiment/Tutorial-for-Data-Scientists-3.

48. (Optional)If you do the example from R Tools for Visual Studio, make sure to specify the R environment of your choice from the options window for R Tools.



49. In this example, you use the Boston housing data from the R package MASS. The Boston dataset contains 506 rows and 14 columns. Available information includes median home price, average number of rooms per dwelling, crime rate by town, etc.
    More information about this dataset can be found by typing ?Boston or help(Boston) in an R terminal, or at this UCI page https://archive.ics.uci.edu/ml/datasets/Housing.
    To run a cell, select it and press CTRL+ENTER or SHIFT+ENTER.
    The window should look like the figure below. Now you can proceed to running each Cell by placing the cursor in the cell and the Pressing CTRL+ENTER or SHIFT+ENTER and following the instructions.



50. The code in the first cell is to get an output that describes the Dataset which is a data frame with 506 rows and 14 columns. Click and make sure your cursor is in the cell and then press SHIFT + ENTER if you want to re run the code.

```
In [1]: # load the library to use the Boston dataset
        library(MASS)
        ?Boston
```

Even though for the sake of this example we are just using data from the MASS Library, this data can be from any of the typical data sources in our organization like Hadoop or SQL or Data Warehouse or just files on disk.

The Output on Jupyter should look like:

Out[1]:

Boston {MASS}	R Documentation

Housing Values in Suburbs of Boston

Description

The `Boston` data frame has 506 rows and 14 columns.

Usage

```
Boston
```

Format

This data frame contains the following columns:

`crim`
per capita crime rate by town.

`zn`
proportion of residential land zoned for lots over 25,000 sq.ft.

`indus`
proportion of non-retail business acres per town.

`chas`
Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

`nox`
nitrogen oxides concentration (parts per 10 million).

`rm`
average number of rooms per dwelling.

`age`
proportion of owner-occupied units built prior to 1940.

`dis`
weighted mean of distances to five Boston employment centres.

`rad`
index of accessibility to radial highways.

`tax`
full-value property-tax rate per $10,000.

`ptratio`
pupil-teacher ratio by town.

`black`
1000(Bk - 0.63)^2 where *Bk* is the proportion of blacks by town.

`lstat`
lower status of the population (percent).

`medv`
median value of owner-occupied homes in $1000s.

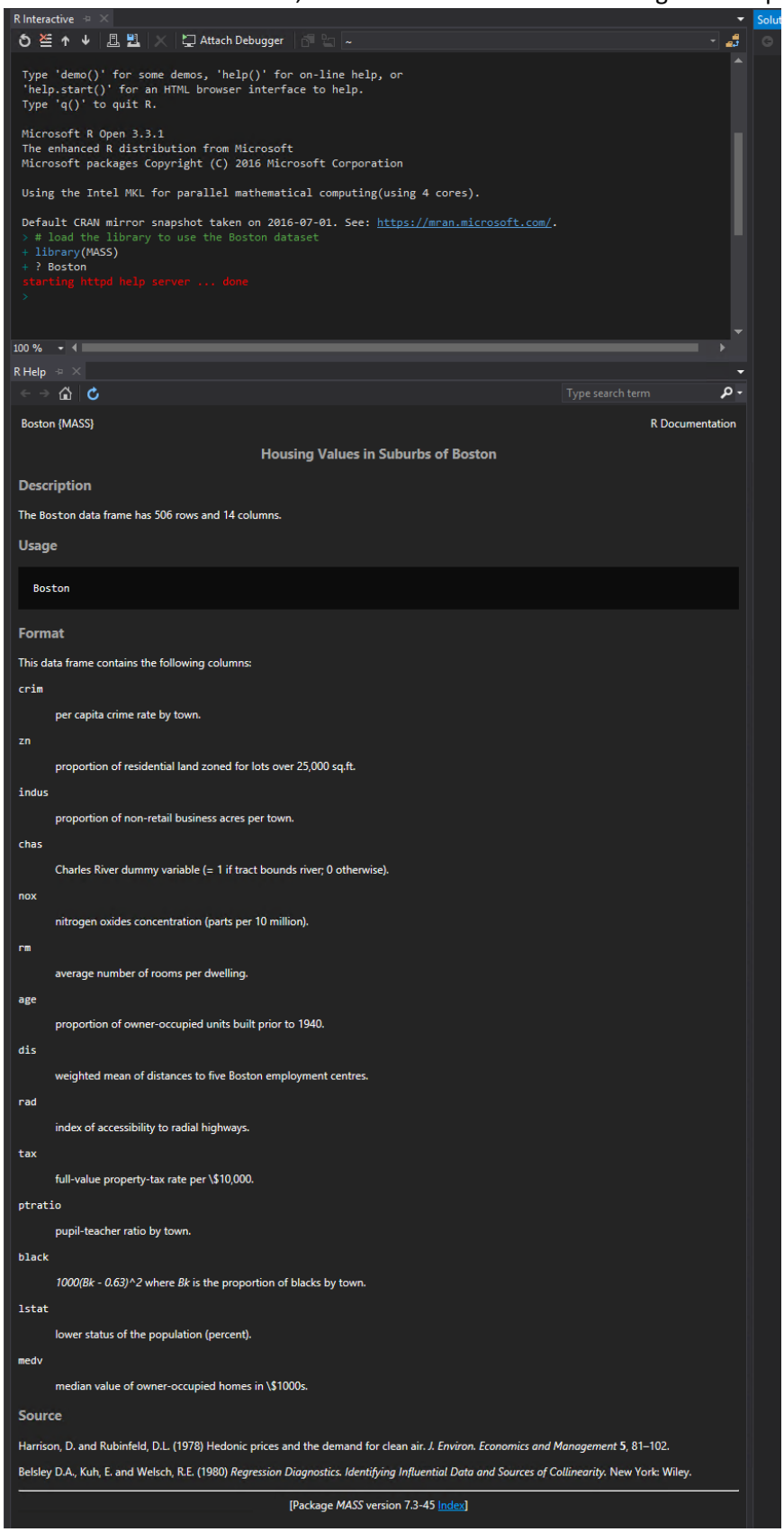
Source

Harrison, D. and Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. *J. Environ. Economics and Management* **5**, 81–102.

Belsley D.A., Kuh, E. and Welsch, R.E. (1980) *Regression Diagnostics. Identifying Influential Data and Sources of Collinearity*. New York: Wiley.

[Package *MASS* version 7.3-43]

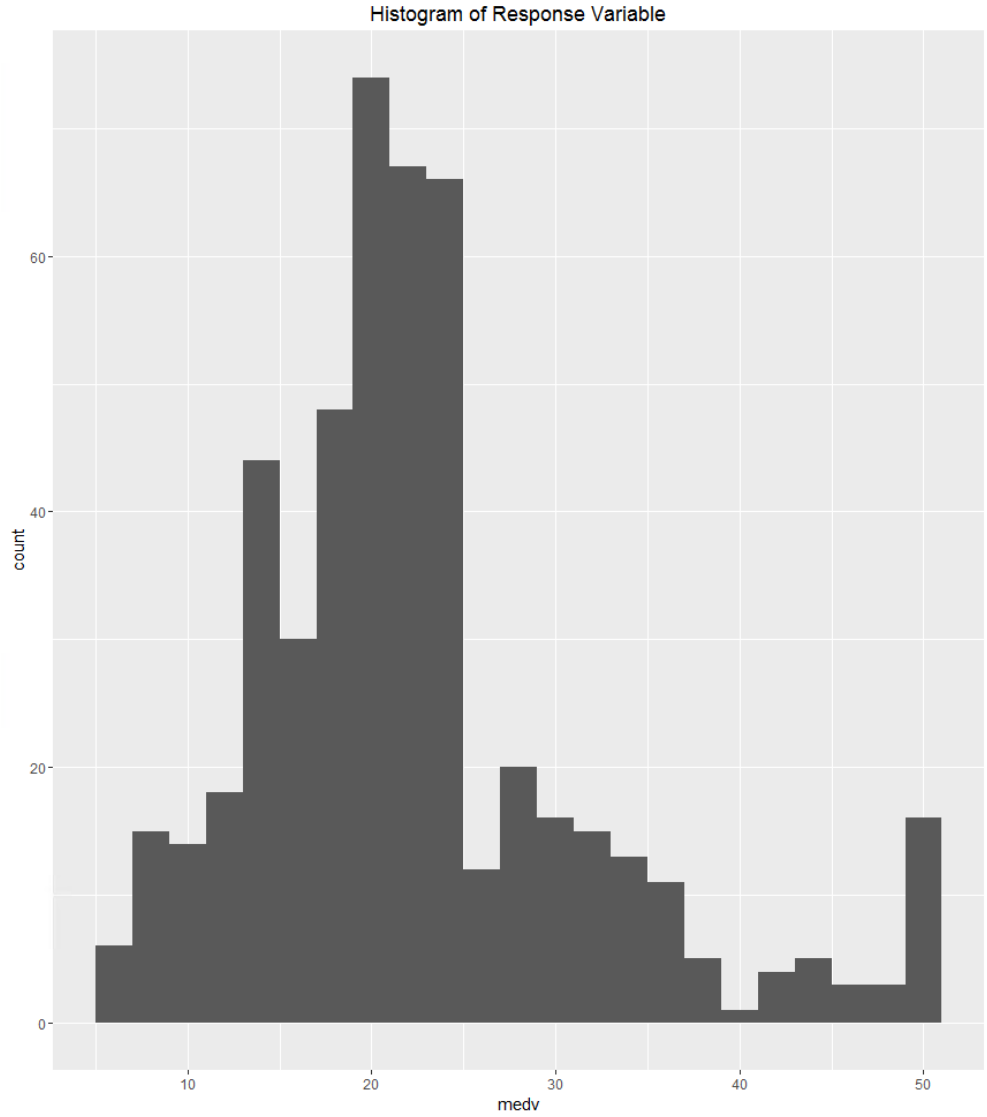
On R tools for Visual Studio, use the R Interactive window to get an output like:



51. In the previous step, we got a good verbal description of the data. For the purposes of illustration, we will be using the 'medv' variable (Median Value of Owner-Occupied homes in $1000s) as the target/response variable and the other 13 as the predictors. So, let us first see what the distribution of our response variable looks like. We will use the 'ggplot2' library for this:

```
In [6]: # library for plotting
        library(ggplot2)
        # plot distribution of the response variable
        ggplot(Boston, aes(x=medv)) +
          geom_histogram(binwidth=2) +
          ggtitle("Histogram of Response Variable")
```

Run the next cell on your notebook or Run these lines of code in RTVS. You should see a plot resembling the figure that follows:



52. Now let us fit a simple Linear Regression model to predict 'medv' using all the other 13 variables. You may use other models if you feel like, this is an illustration only to showcase the functionality and the typical workflow.

```
In [7]: # fit a model using all variables except medv as predictors
        lm1 <- lm(medv ~ ., data = Boston)

        # check model performance
        summary(lm1)
```

Your Output should look like either of the 2 following figures depending on whether you are running Jupyter or RTVS:

```
Out[7]:
        Call:
        lm(formula = medv ~ ., data = Boston)

        Residuals:
            Min      1Q  Median      3Q     Max
        -15.595  -2.730  -0.518   1.777  26.199

        Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
        (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
        crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
        zn           4.642e-02  1.373e-02   3.382 0.000778 ***
        indus        2.056e-02  6.150e-02   0.334 0.738288
        chas         2.687e+00  8.616e-01   3.118 0.001925 **
        nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
        rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
        age          6.922e-04  1.321e-02   0.052 0.958229
        dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
        rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
        tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
        ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
        black        9.312e-03  2.686e-03   3.467 0.000573 ***
        lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
        ---
        Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

        Residual standard error: 4.745 on 492 degrees of freedom
        Multiple R-squared:  0.7406,    Adjusted R-squared:  0.7338
        F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

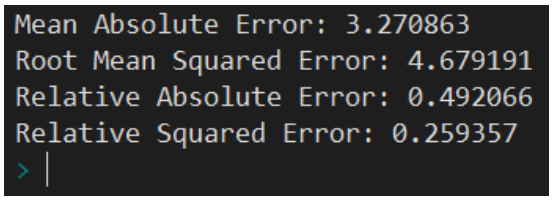

```
Call:
lm(formula = medv ~ ., data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-15.595  -2.730  -0.518   1.777  26.199

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288
chas         2.687e+00  8.616e-01   3.118 0.001925 **
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
black        9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-squared:  0.7406,    Adjusted R-squared:  0.7338
F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

53. Now, to calculate some error measure for the model, run the next cell/block of code.

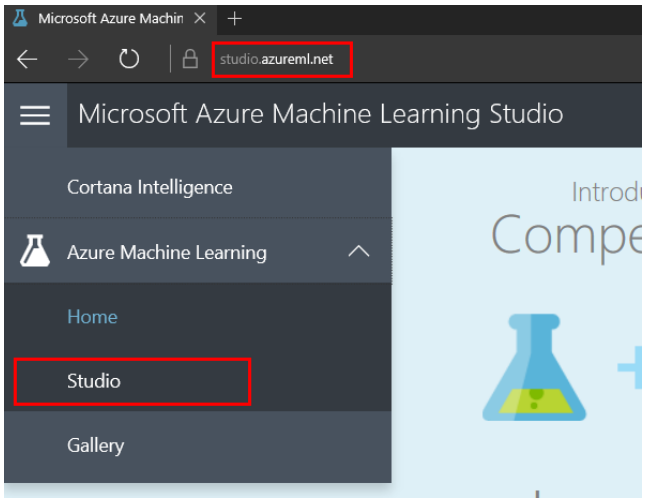
```
In [8]: pred <- predict(lm1)
        error <- pred - Boston$medv
        mae <- mean(abs(error))
        rmse <- sqrt(mean((error)^2))
        rae <- mean(abs(error)) / mean(abs(Boston$medv - mean(Boston$medv)))
        rse <- mean((error)^2) / mean((Boston$medv - mean(Boston$medv))^2)

        cat("Mean Absolute Error:", round(mae, 6), "\n")
        cat("Root Mean Squared Error:", round(rmse, 6), "\n")
        cat("Relative Absolute Error:", round(rae, 6), "\n")
        cat("Relative Squared Error:", round(rse, 6), "\n")
```

The Output should look like one of the following figures depending on the tool:

```
Mean Absolute Error: 3.270863
Root Mean Squared Error: 4.679191
Relative Absolute Error: 0.492066
Relative Squared Error: 0.259357
```

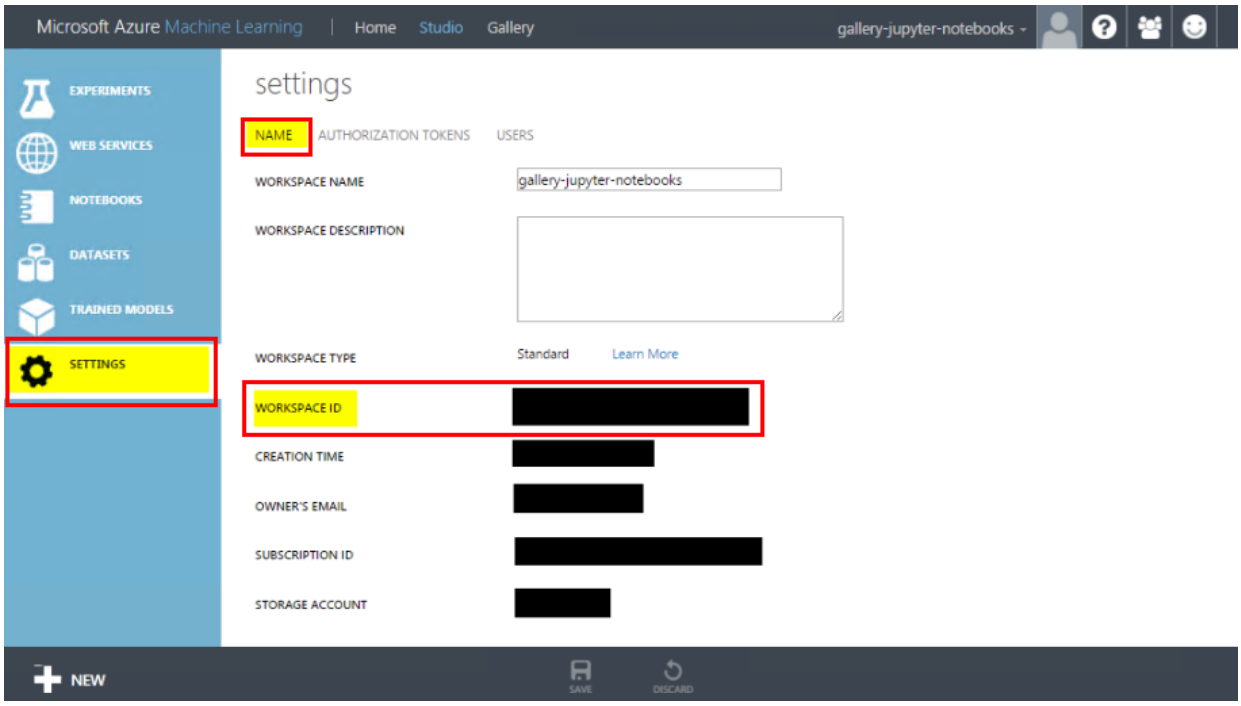


54. So, now we have a trained linear regression model that can predict one value based on 13 inputs. One of the easiest ways to operationalize this trained model so that one can use it programmatically for scoring workflows, is to deploy it as a web service in Azure Machine Learning (Azure ML). The `AzureML` R package will be used for this purpose. Please open a browser window and log in to Azure ML Studio https://studio.azureml.net/ :

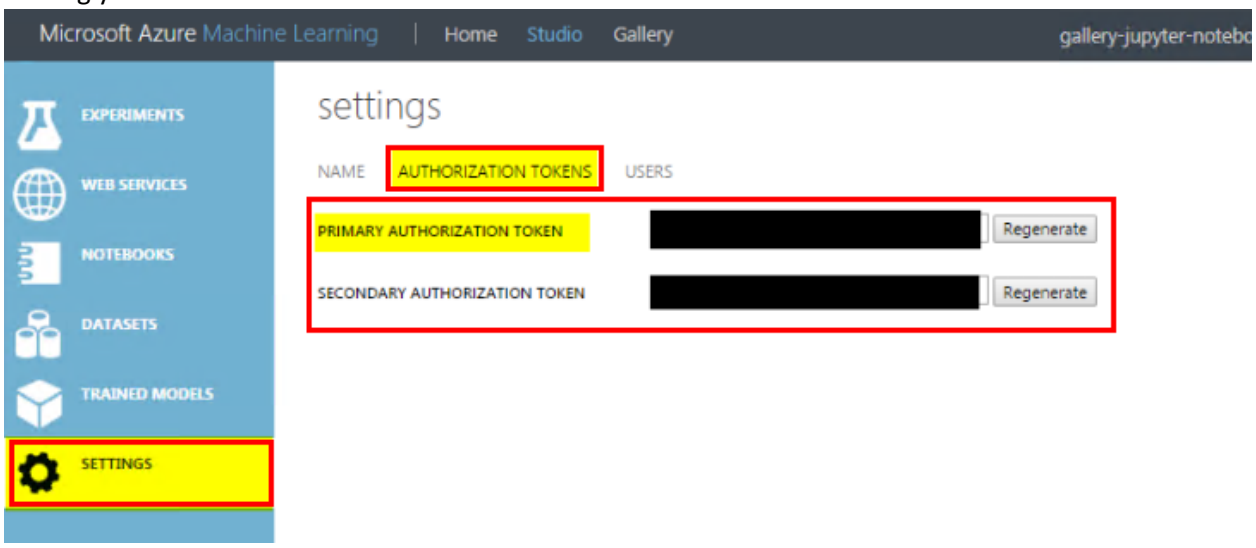


Important: Publishing and consuming a web service requires you to have valid Azure ML credentials. Concretely, if you logged in using a Guest account, you will get an error. If this is the case, please log in using your Microsoft or Azure account and try again.

You'll need to provide the **workspace ID** and **authorization token** for an Azure ML workspace. Finding your workspace ID:



Finding your authorization token:



55. If you are using RTVS, make sure you install the 'AzureML' package first:

```
> install.packages("AzureML")
also installing the dependencies `base64enc', `miniCRAN', `uuid'
```

56. We use the function `publishWebservice()` to deploy the model. Run the next code cell:

```
In [10]: # load the library
library(AzureML)

# If you use workspace() in a Jupyter notebook, you don't need to specify credentials,
# since the settings are stored for you in a local file.
# If you use this function on your own machine, specify your credentials. See ?workspace.

if(file.exists("~/.azureml/settings.json")){
    ws <- workspace()
} else {
    workspace_id <- "ENTER AZURE ML WORKSPACE ID (SEE ABOVE)"
    authorization_token <- "ENTER AZURE ML WORKSPACE AUTH TOKEN (SEE ABOVE)"
    ws <- workspace(workspace_id, authorization_token)
}

# define predict function
mypredict <- function(newdata){
  predict(lm1, newdata)
}

# a sample with predictor information
newdata <- Boston[1:5, ]

# test the prediction function
data.frame(
    actual = newdata$medv,
    prediction = mypredict(newdata))
```

```
> library(AzureML)
> # If you use workspace() in a Jupyter notebook, you don't need to specify credentials,
+ # since the settings are stored for you in a local file.
+ # If you use this function on your own machine, specify your credentials. See ?workspace.
+
+ if (file.exists("~/.azureml/settings.json")) {
+     ws <- workspace()
+ } else {
+     workspace_id <- "[redacted]"
+     authorization_token <- "[redacted]"
+     ws <- workspace(workspace_id, authorization_token)
+ }
+
+ # define predict function
+ mypredict <- function(newdata) {
+     predict(lm1, newdata)
+ }
+
+ # a sample with predictor information
+ newdata <- Boston[1:5,]
+
+ # test the prediction function
+ data.frame(
+     actual = newdata$medv,
+     prediction = mypredict(newdata))
```

The Output should look like one of the 2 figures that follow:

Out[10]:

	actual	prediction
1	24	30.00384
2	21.6	25.02556
3	34.7	30.5676
4	33.4	28.60704
5	36.2	27.94352

```
  actual prediction
1   24.0   30.00384
2   21.6   25.02556
3   34.7   30.56760
4   33.4   28.60704
5   36.2   27.94352
> |
```

Now use the publishwebservice() function to deploy the web service to the workspace in Azure ML. Run the next cell of code:

In [11]:
```
# publish the service
ep <- publishWebService(ws = ws,
                        fun = mypredict,
                        name = "HousePricePrediction",
                        inputSchema = newdata)
# str(ep)
```

```
converting `inputSchema` to data frame
```

```
> # publish the service
+ ep <- publishWebService(ws = ws,
+                         fun = mypredict,
+                         name = "HousePricePrediction",
+                         inputSchema = newdata)
+ # str(ep)
> |
```

57. The Web Service has now been published and is now ready to be consumed by scoring requests. After setting up a web service, you can use R to consume the model, either in the same session, or by saving some data about the web service, in any other (future) R session. We use the 'consume()' function to consume a web service from R and the arguments are 'endpoint' and 'newdata'.

58. **In-session consumption:**
If you are consuming the web service in the same session that the web service was set up, you can refer to the endpoint directly.

Important: Publishing and consuming a web service requires you to have valid Azure ML credentials. Concretely, if you logged in using a Guest account, you will get an error. If this is the case, please log in using your Microsoft or Azure account and try again.

Run the next 2 code cells to request a prediction from the web service:

```
In [12]: pred <- consume(ep, newdata)$ans
         # Note that this operation may take several seconds to complete.
         # You may see several retry attempts, before the service returns with a value
```

```
In [13]: data.frame(actual = newdata$medv, prediction = pred)
```

You should see a response like the 2 following figures depending on your choice of Jupyter or RTVS:

Out[13]:

	actual	prediction
1	24	30.00384
2	21.6	25.02556
3	34.7	30.5676
4	33.4	28.60704
5	36.2	27.94352

```
> pred <- consume(ep, newdata)$ans
+ # Note that this operation may take several seconds to complete.
+ # You may see several retry attempts, before the service returns with a value
> data.frame(actual = newdata$medv, prediction = pred)
  actual prediction
1   24.0   30.00384
2   21.6   25.02556
3   34.7   30.56760
4   33.4   28.60704
5   36.2   27.94352
>
```

59. **Out-of-session consumption:**

If you consume the web service in a new session, you need to save the workspace information - workspace id and authorization token - and web service ID. Such information can then be used by the `consume()` function:

Important: Publishing and consuming a web service requires you to have valid Azure ML credentials. Concretely, if you logged in using a Guest account, you will get an error. If this is the case, please log in using your Microsoft or Azure account and try again.

In the next section of the code, we will store the workspace information in a temporary file, then read it back into a variable and then consume the predictions from the web service using the stored endpoint data. Now, run the next cell of code to store the data into the file and read it back.

```
# Option 1: Create a temporary file to store the endpoint data
tf <- tempfile(fileext = ".rds")
saveRDS(ep, file = tf)

# Read the endpoint data from file
endpoint <- readRDS(tf)

# # Option 2: save workspace ID, authorization token and service ID
# ws_id <- ws$id
# ws_auth <- ws$.auth
# # save web service ID
# service_id <- ep$WebServiceId
# # define workspace, this is necessary if you are running outside of the service deployment session
# ws <- workspace(
#     id = ws_id,
#     auth = ws_auth
# )
# # defind endpoint based on workspace and service ID information
# endpoint <- endpoints(ws, service_id)
```

We can again consume the service by running the code in the next cell:

```
# consume
pred <- consume(endpoint, newdata)$ans
# check predictions
data.frame(
    actual = newdata$medv,
    prediction = mypredict(newdata))
```

You should see a response like the 2 following figures depending on your choice of Jupyter or RTVS:

	actual	prediction
1	24	30.00384
2	21.6	25.02556
3	34.7	30.5676
4	33.4	28.60704
5	36.2	27.94352

```
  actual prediction
1   24.0   30.00384
2   21.6   25.02556
3   34.7   30.56760
4   33.4   28.60704
5   36.2   27.94352
>
```

60. Now we know how to publish and consume modes from Azure as Web Service Endpoints. So if you now make improvements to a model, you can update the existing web service. For this purpose you can use the `updateWebService()` function by specifying the web service ID.

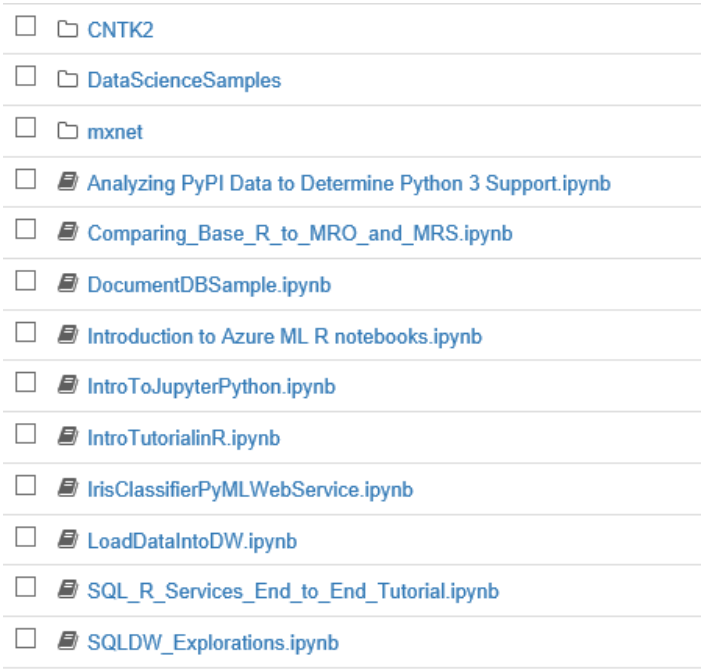
    The Next block has some sample code that you may chose to run to update the existing web service with a newer model.

```
In [11]: # To update a web service, you need the WebServiceId

         # define test function
         mypredictnew <- function(newdata){
           predict(lm1, newdata) + 100 # Add 100 for illustration purpose only
         }

         # update service with the new function
         ep_update <- updateWebService(
           ws = ws,
           fun = mypredictnew,
           name = "not necessary", # this does not matter since serviceId is provided
           inputSchema = newdata,
           serviceId = ep$WebServiceId
         )

         # str(ep_update)
```

61. We just completed a simple example for successfully building a model, publishing it as a web service and scoring using this web service all using R and Azure ML enabled on the DSVM. You may repeat this with Python and consume using other application programming frameworks, with the model deployed to numerous other operationalization architectures like R Services on SQL Server, Microsoft R Server, Spark, Python on Flask or CherryPy etc.

Conclusion + Next Steps

This guide was designed to get you introduced to the DSVM and how to set one up. Additionally, we wanted to get you familiar with the installed tools, SDKs etc. This was intended to walk you through a typical Data Science Workflow using the Data Science Virtual Machine in Azure and other related technologies. Please do go through the numerous other examples that we have already provided on the VM to get familiar with other combinations of resources.



If you have questions and or want to discuss, please visit the Data Science Virtual Machine Forum at:

http://aka.ms/dsvmforum

- *Barnam Bora*
Program Manager - Microsoft