

# Project Title: Data Engineering in Azure

Project Description: This script forms the basis of the Data Engineering in Azure Presentation and Demo sets

Date: January 23. 2020

Author: Steve Young

Table of Contents

[Project Title: Data Engineering in Azure - 1 -](#_Toc29902824)

[Overview - 3 -](#_Toc29902825)

[Demo Environment Element Listing - 3 -](#_Toc29902826)

[Asset Listing - 5 -](#_Toc29902827)

[Required resources - 5 -](#_Toc29902828)

[Script and Actions - 7 -](#_Toc29902829)

[Setup and Startup for Lab - 7 -](#_Toc29902830)

[Scene 1 - Introduction the environment and use case - 12 -](#_Toc29902831)

[Script and Actions - 16 -](#_Toc29902832)

[Scene 2 – Demo 1 -Sliding window in Azure Data Factory - 16 -](#_Toc29902833)

[SQL 2019 - Demo 2 - Data Virtualization - Using Polybase to query Azure SQL Database - 37 -](#_Toc29902834)

[Scene 3 - 37 -](#_Toc29902835)

[SQL 2019 - Demo 3 - Juypter style SQL Notebooks in Azure Data Studio - 47 -](#_Toc29902836)

[Scene 4 - 47 -](#_Toc29902837)

[Appendix - Sample Script and Actions - 52 -](#_Toc29902838)

[Scene 1 - 52 -](#_Toc29902839)

# Overview

This set of demonstrations that help support the Data Engineering in Azure presentation. The following labs are included

1. Azure Data Factory import to Data Lake of SQL Server Data using an Sliding Window pattern.
2. Data Virtualization using SQL 2019
3. Azure Synapse External Tables

# Demo Environment Element Listing

The demo environment is located in Azure and has the following components located in the resource group, rgSQL2019vm. The VM is used as the demo machine as this will provide the ability to move from machine to machine and keep the setup complete.

| Resources | Particulars |
| --- | --- |
|  | * **Azure Data factory** – used in the ETL processes |
|  | * **SQL2019** – This VM houses an on-premises style SQL server install. It is used for many different demos and is the source for these demonstrations. |
|  | * **Use Azure Storage Explorer –** Contains the Blob Containers sqlsynapselabs * This storage account has the source files for the Polybase labs * The windowtrigger blob has the lab content exporting using azure data factory |
|  | * **AdvetureworksLT** – this is the Azure SQL database used for various demos. Has the base AdventureworksLT database plus other tables etc * This is hosted on the **SQL2019labssey** sql logical server |
|  |  |

# Asset Listing

| Asset# | Asset File Name | Description | Length | Notes |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Required resources

| Scene | Resource Name | Description | Notes |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Script and Actions

\*\* For the azure portal and other items, the dark theme may not have the contrast on a large projection screen for everyone to see clearly.

## Setup and Startup for Lab

The demo environment is located in Azure and has the following components located in the resource group, rgSQL2019vm. The VM is used as the demo machine as this will provide the ability to move from machine to machine and keep the setup complete.

| Step | | Resources | Particulars |
| --- | --- | --- | --- |
|  |  | | * Open up <http://portal.azure.com>, login and go to the resource group, [rgSQL2019VM](https://ms.portal.azure.com/#@microsoft.onmicrosoft.com/resource/subscriptions/96c25bbf-7d5a-452e-8434-941b825c1ee0/resourceGroups/rgSQL2019VM/overview) * Select the element listed below and start them up. |
|  |  | | * Start the SQL 2019 VM * Connect to the VM and open a remote desktop session * Once connected, the next few steps will show you what to get connected to and started in the VM * Remember that the auto shutdown is set to 10pm EST |
|  |  | | * Remember to change your domain, this is a local sign on the machine is not on the Azure AD * PW is the demo password A7!! |
|  |  | | * SQL 2019 VM Start Ups * The SQL 2019 VM has Github, and other software that allows code to be transferred and brought into this environment. * The machine is setup as a developer machine |
|  |  | | * First Setup is to get/update the latest version of the code from github. * Log into Github and clode the repository * <https://github.com/steveyoungca/DataEngineeringInAzure> * Should be public at the time of the session |
|  |  | | * Call up SQL Server Management Studio and log into the local SQL instance * Log into the Azure SQL instance * SSMS Log into Local SQL 2019 instance * PW is the same as the local administrator |
|  |  | | * Listing of local databases |
|  |  | | * Log into * From the VM, as we can have the one interface for the demo rather than jumping back and forth. * Log into the Azure Portal and start up the Azure SQL DW (Azure Synapse) |
|  |  | | * Once Started, bring up the server in SSMS * sql2019labssey.database.windows.net * A7!! |
|  |  | | * Also connect with the Azure SQL DB * sql2019labssey.database.windows.net * A7!! * This was done in the previous step as both the DW and the DB are on the same server. May need to log in using different IDS for some of the demo scripts. |
|  |  | | * This also allows you to connect to Azure Synapse (SQL DW) * Note all the external resources. * For any Demo, change the object names to DEMO\_ as that will keep the other demo objects, where you do not create the new objects, untouched. |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |
|  |  | |  |

## Scene 1 - Introduction the environment and use case

|  |  |
| --- | --- |
| Product: |  |
| Date: | December 15. 2020 |
| Author: | Steve Young |

Purpose:

1. Overview of the Use case
2. Overview of the environment
3. Overview of the Labs

| # | Notes | Action on Screen | Audio |
| --- | --- | --- | --- |
|  | Use slide deck for showing this | Show the Demo Architecture slide | * Good day. We are going to use the following architecture for our use case. * Our use case is for the Smith and Smit retail company. They are based in Canada and provide retail and online shopping for various household products. They are currently in Azure and have some on-prem databases. * They want to expand their use of their data to forecast and provide insights into their business. * This is an old school BI project, but in Azure. * They have files and SQL Databases as sources, they want to be able to keep history and also leverage azue for reporting form their operational databases |
|  |  |  | * Features * They need an hourly process to migrate data to Azure blob storage * This blob store will be the source for all of their Azure working systems |
|  |  |  | * **Azure Data factory** – used in the ETL processes |
|  |  |  | * **SQL2019** – This VM houses an on-premises style SQL server install. It is used for many different demos and is the source for these demonstrations. |
|  |  |  | * **Use Azure Storage Explorer –** Contains the Blob Containers sqlsynapselabs * This storage account has the source files for the Polybase labs * The windowtrigger blob has the lab content exporting using azure data factory |
|  |  |  | * **AdvetureworksLT** – this is the Azure SQL database used for various demos. Has the base AdventureworksLT database plus other tables etc   This is hosted on the **SQL2019labssey** sql logical server |
|  |  |  | * Listing of local databases |
|  |  |  | * This also allows you to connect to Azure Synapse (SQL DW) * Note all the external resources. * For any Demo, change the object names to DEMO\_ as that will keep the other demo objects, where you do not create the new objects, untouched. |
|  |  |  |  |

# Script and Actions

## Scene 2 – Demo 1 -Sliding window in Azure Data Factory

|  |  |
| --- | --- |
| Demo: | Sliding window in Azure Data Factory |
| Date: | Dec 14, 2019 |
| Author: | Steve Young |

Purpose:

1. Show being able to capture new events and populate an Azure Data Lake
2. Show how to organize a data lake

Need

* SQL 2019 VM – SQL2019
* Azure Data Factory V2 - adfSQL2019Labs
* Azure Data Lake - sqlsynapselabs
* Show the SQL DB which is the source for the data - AdventureworksLT (sql2019labssey/AdventureworksLT)
* Load the SQL

| # | Notes | Action on Screen | Audio |
| --- | --- | --- | --- |
|  | Setup |  | Load up the SQL |
|  | Setup |  | Load Up the azure data factory and show the 5 minuteBI article that shows the How to. |
|  | Setup |  | <https://5minutebi.com/2018/06/03/how-to-use-azure-data-factory-v2-sliding-windows-for-sql-exports-to-azure-data-lake/> |
|  | Setup |  | * Load SSMS, attached to SQL Azure DB, open AdventureworkdLT. * We should already be logged into the server in SSMS during the demo setup. See the above section for details |
|  | Start Talk | Start on the Azure Portal | Good day. Our first demo is going to show you how to design and implement an hourly capture of sales facts from Azure SQL DB to Azure Data Lake. |
|  |  | Showing SSMS, AdventureWorksLT, show the tables   * Show the Azure SQL DB, the FactWindowHour table | * In our demo, we have a fact sales table that captures transactions from our web based sales application. |
|  |  | DEMO – Show a select of the table | * We created a table that will have records inserted at various time frames. |
|  |  |  | * We want these to be picked up and placed into folders by hour. |
|  |  |  | * Our first step is to create a blob container to house the hourly file feed from Azure Data factory. * Open up Azure Storage explorer, loginto your subscription and create the container * Go to your blob store, right click on the BloB containers, select Create Blob Container |
|  |  |  | * Enter a name, in all lower case |
|  |  |  | We now have our destination. |
|  |  |  | For a later step, we need to grab the Access Keys for the Data Lake.  Go to the RG |
|  | Get Keys |  | Go to the data Lake in the Azure portal, Select Access Keys |
|  |  | -Go to the Azure Data Factory – Select the Author and Monitor | -Go to the Azure Data Factory – Select the Author and monitor |
|  |  | -Go to the Azure Data Factory –  - Talk through the various options  - Select the Copy Data | Go to the Azure Data Factory –  Select the Copy Data from the screen  Reference  <https://docs.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory-copy-data-tool>  <https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview> |
|  |  | * After the screen opens, go over each item, point out the schedule and a good task name and description * Beginning first screen |  |
|  |  | * Ending first screen is | 1. We are going to enter a name that is meaningful. Types of names should include the step, or general task.  You can have a different Azure Data Factory by group, these should have a specific meaning and not one ADF to rule them all. 2. Put in a task description, again, one that is meaningful 3. You can run this once, such as a history load, or a tumbling window and also run on a schedule. Once you select schedule, you can select your type of schedule. 4. There are different options depending on which schedule you choose (Show difference) 5. Select the Date – Noting UTC time, will also be how the folders are spread out also 6. Recurred by day, hour, minute 7. End date can be used, such as for this demo, I will stop this in a couple of days. 8. The end date that is selected 9. How long should you delay after the start time 10. The number of simultaneous trigger runs that are run. For example, to back fill hourly runs for yesterday results in 24 windows. If maxConcurrency = 10, trigger events are fired only for the first 10 windows (00:00-01:00 - 09:00-10:00). After the first 10 triggered pipeline runs are complete, trigger runs are fired for the next 10 windows (10:00-11:00 - 19:00-20:00). Continuing with this example of maxConcurrency = 10, if there are 10 windows ready, there are 10 total pipeline runs. If there's only 1 window ready, there's only 1 pipeline run. - An integer, where the default is 0 (no retries). 11. Retry Policy Count - The number of retries before the pipeline run is marked as "Failed." - 12. Retry Policy Count – Interval in Seconds - The number of seconds, where the default is 30.   <https://docs.microsoft.com/en-us/azure/data-factory/how-to-create-tumbling-window-trigger> |
|  |  | The Source Data Store – We will create a new on here | Lets create a new connection. You see here that there are existing connections, use one if one is already available, we will create one now for this demo. |
|  |  |  | As you can see here, there are many linked services to choose from.  (Show on screen and review)  Ours is Azure SQL DB for this demo, (Select this one) |
|  |  |  | You will now see the Linked server properties pane.  We will fill this out, for this demo, we will keep it simple and use connection strings and SQL Authentication.   1. Provide a Name for this connection 2. Enter a description, again meaningful 3. Either create or use the AutoResolving Integration runtime. This is all in Azure. 4. Since this is all in azure and using a connection string, you can use the meta data in your subscription to select 5. Note, we could use service principle etc, fill this in with our items |
|  |  | Test | We can test connection, (Select test connection) and if all is well, we should have a good connection. |
|  |  |  | We can now select create, and should get a success  Select this Source, and click NEXT at the bottom of the panel. |
|  |  |  | We now have the Data Set selection. We can select a table or a query. Select the table, dbo.FactWindowHour  Notice the Preview, and Schema    Select NEXT at the bottom of the screen |
|  |  |  | We now have to select the column that we want to filter by selecting the drop down and Filter by Datatime column |
|  |  |  | We now we drop downs for the column selection, and the parameter names that we can select.  The Query time out defaults to 120 minutes, you can specify |
|  |  |  | Review the generated query  Select Next, which brings up the Destination |
|  |  |  | You can see all the connections here, we will create an new connection to an Azure for the Azure Data Lake |
|  |  |  | Select Azure data lake storage v2 |
|  |  |  | 1. Select a name that is meaningful 2. Enter a description, again that is meaningful 3. Keep the AutoResolveintegrationRuntime 4. For this |
|  |  |  | 1. Authentication method options. For SQL, we can use an account key also |
|  |  |  | 1. We will enter this manually, as when we use Polybase we will use the manual entry.   sqlsynapselabs. dfs.core.windows.net |
|  |  |  | Put in the account key and URL  Test connection    Should see success. As pictured to the left. |
|  |  |  | Select the newly created destination.  Select Next to go to the next screen. |
|  |  |  | 1. On this screen, we want to show the Folder and file name we want to use.   Use Browse. |
|  |  |  | Select the correct folder, DemoWindowTrigger |
|  |  |  | 1. Select the file name to use. Since we will use a different folder for each trigger, we can specify a nondynamic file name. |
|  |  |  | The moment you add in the variables, you get the variables to specify automatically.  demowindowtrigger/{year}/{month}/{day}/{hour}/{minute}    Select Next to go to the next screen |
|  |  |  | Add the header to the file if you need it. |
|  |  |  |  |
|  |  |  | Summary |
|  |  |  | When you hit the final next, your objects will deploy.  Hit Monitor to see how your first run went.  If you have “Add a Header” you will get an empty file with a header. |
|  |  | Show the monitor runs |  |
|  |  | Show Individual Run |  |
|  |  | Show the BLOB storage |  |
|  |  | OMG I come before the Tutorial – Jan 12 2020 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  | Ok, perhaps better that Docs Microsoft come before me on Google | I own the term  azure data factory copy data tool sliding window |

# SQL 2019 - Demo 2 - Data Virtualization - Using Polybase to query Azure SQL Database

This demo will go over setting up data virtualization.

## Scene 3

**Setup**

1. In the SQL VM call up SSMS and attach to Azure SQL DB and the Local SQL 2019 instance

|  |  |
| --- | --- |
|  | 1. Connection 1 - sql2019labssey.database.windows.net 2. Connection 2 – SQL2019 – Local SQL Server on the VM |
|  | 1. Load up the Demo 2 TSQL |
|  |  |

|  |  |
| --- | --- |
| Product: | SQL 2019 - Demo 2 - Data Virtualization - Using Polybase to query Azure SQL Database |
| Date: | Dec 02, 2019 |
| Author: | Steve Young |

Purpose:

1. Introduce main controls
2. Show ways to navigate and find information

**Script**

| # | Notes | Action on Screen | Audio |
| --- | --- | --- | --- |
|  |  | Introduction – in the slide deck go thorugh the 3 slides | This demo will use the new features in SQL 2019 to use the data where it resides, or as it is called data virtualization. You use the power of the source system  That being said, you and also Azure Synapse and polybase to use external tables, but this feature is in SQL 2019.  What benefits do you see by this from a data engineering point of view.   * No ETL, Use source system, no scheduling * BUT, you need access to the source data * We all know how difficult it can be to get permission and get * Think of it as data aggregation |
|  |  | * From inside the SQL VM show the script | This script will walk us through the steps required to use the data virtualization features in SQL 2019 to consume a table in Azure SQL DB and use this in a join query with WorldWideImporters.  Setting up the security and communications to do this are also covered  (Show the references in the script) |
|  |  | * Go to the Azure Demo DB and show the tables * Go to the Create Table in the script, also make sure you are connected to the correct server and also correct database.     🡨 correct connection | As you see here we will create the table in the AzureDemoDatabase. We run this command to create the table,  (Show the table successfully created)    (Refresh and show the table) |
|  |  | * Now create sample data | Now that we have a table, lets insert some records.  (Run the insert statements and show the results) |
|  |  | * <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-database-scoped-credential-transact-sql?view=sql-server-ver15> * <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-database-scoped-credential-transact-sql?view=sql-server-ver15> * Create a User to be used in the Credential for Polybase   Av6!! | We are now going to create a user to use as a credential for connection when using Ploybase. We will create a user in the Azure SQL DB to use as a connection.  Note that you cannot use a Shared Accesss Signature with Polybase. |
|  |  |  |  |
|  |  | Run the rest in the SQL 2019 Local Instance WideWorldImporters database | We are going to run the rest of the commands in the VM instance of SQL 2019    Validate in the menu bar below    Validate that you are pointing to WideWorldImporters. |
|  |  |  | We are going to validate that Poly base is enabled.  Run the following script command to enable and validate.  Lets validate first |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | * The database credential contains the IDENTITY (login) and SECRET (password) of the remote Azure SQL Database server or Managed Instance. Change this to the login and password for your server. Configure PolyBase to access external data in SQL Server   Ref <https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-configure-sql-server?view=sql-server-ver15> |
|  |  | <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql?view=sql-server-ver15> | Now that we have the credential, we have to create the external data source. Remember that this is using the UID and PW we have already setup  Pushdown=on is  (Run the command) |
|  |  |  | PUSHDOWN = ON | OFF  States whether computation can be pushed down to the external data source. It is on by default.  PUSHDOWN is supported when connecting to SQL Server, Oracle, Teradata, MongoDB, or ODBC at the external data source level.  Enabling or disabling push-down at the query level is achieved through a hint.  <https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-pushdown-computation?view=sql-server-ver15#force-pushdown> |
|  |  |  | From a data engineering point of view, if you create a schema for your external tables you will know the sources and secure access in your database accordingly.  Here I use the External database as the extension. Think about the number of tables that you will sue for example. Keeping them named as something meaningful will defiantly help. |
|  |  |  | An external table provides metadata so SQL Server knows how to map columns to the remote table. The name of the table for the external table can be your choice. But the columns must be specified in the same order with the same name as they are defined in the remote table. Furthermore, local data types must be compatible with the remote table.  The WITH clause specifies a LOCATION. This LOCATION is different than the EXTERNAL DATA SOURCE. This LOCATION indicates the [database].[schema].[table] of the remote table. The DATA\_SOURCE clauses is the name of the EXTERNAL DATA SOURCE you created earlier. |
|  |  |  | SQL Server allows you to store local statistics about specific columns from the remote table. This can help the query processing to make more efficient plan decisions.  Lets run this to create |
|  |  |  | Lets run a select of the remote table |
|  |  |  | Lets run a select with a where, |
|  |  |  | Now a query where we join local data and remote data  (Highlight the Query and show the estimated execution plan)  See the Remote query |
|  |  |  |  |

# Demo 3 - Juypter style SQL Notebooks in Azure Data Studio

This hands-on demo will detail the steps required to create the objects required for the labs. Not all steps will have a screen shot. This lab will walk you through the creation of the objects required for the “SQL Data Warehouse in a Day” workshops. The Goal is to Introduce the Azure SQL Notebooks

## Scene 4

|  |  |
| --- | --- |
| Product: | Azure SQL Notebooks |
| Date: | Jan 14, 2019 |
| Author: | Steve Young |

**Pre-requisites:** Demo files for this Lab located at https://github.com/steveyoungca/xxxxxxxxxxxxxxxxxxx downloaded to a local folder

There is a GitHub repo that contains a copy of the notebook so that we can show GITHUB

Setup:

1. Demo 1 start with creating a new notebook,
2. Then, load in the demo 1 notebook
3. Change and edit the note highlightec - Noted

Purpose:

1. Introduce main controls
2. Show ways to navigate and find information

| # | Notes | Action on Screen | Audio |
| --- | --- | --- | --- |
|  | Start | Start by opening up AZURE Data Studio from the toolbar. | I believe I have saved the best to last. How many people here are familiar with Jupter Notebooks. The are used with Machine Leaning and AI workloads. SQL Notebooks Azure Dat Studio bring the best to SQL also.  Azure Data Studio brings Gitgub and Juypter notebooks to SQL Server.  (Open in the SQL 2019 VM) |
|  |  | Show the connections | Doubleclicking on a server, brings up various dashboard options. |
|  | Setup | Bring up Azure Data Studio on the SQL VM and connect to 2 servers. | Sets open up an existing folder.    (Open the GitHub directory with the AzureSQLDemo and open the notebook. |
|  |  | GitHub - <https://docs.microsoft.com/en-us/sql/azure-data-studio/source-control?view=sql-server-ver15> | You can see that there are pending changes  (Open the source Control)    Open the GITHUB Directory in Github <https://github.com/steveyoungca/AzureSQLNotebookDemo> |
|  |  |  | We can now see SQL files and NotebookDemo1  We can combine notebooks and SQL and you have source control over any objects .  (Open the SQL and    Connect and Run a Query |
|  |  |  |  |
|  |  |  | * You can run through all of the parts etc. * Kernel * Click on +Code or + Text * Server connection * Terminal   This is VSCode in the background |
|  |  |  | * Add Markdown Text demo   What is Markdown#  (double click on the first paragraph and change the wording/spelling) |
|  |  |  | Note the preview sheet |
|  |  |  | Enter some code by adding a code text |
|  |  |  | * When you run, you will see the the results. You can save the workbook which will save the results. For example these use cases work week., * getting help on an error * QA results   Audit results, etc. pass a query to someone that does not have access. (Could be an issue, but you have that issue now) |
|  |  |  | * Working Query * Run cells on the tool bar   Vs run the cell. (CTRL Enter) Does not work |
|  |  |  | Export the tables. - |
|  |  | * Save back to GitHub | Now that we are finished, we can save all. |
|  |  |  | Once we save, you will see that you are notified that you have changes that need to be written back to your source control.  (Select Source Control, show the files waiting with modifications    (Enter a Commit message, and CTRL Enter. |
|  |  |  | You will see changes needed to be pushed back, click on the sync    You will see the following popup, select OK |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Appendix - Sample Script and Actions

This script was used for a video tutorial backup for a demo.

## Scene 1

|  |  |
| --- | --- |
| Product: |  |
| Date: |  |
| Author: |  |

Purpose:

1. Introduce main controls
2. Show ways to navigate and find information

| # | Notes | Action on Screen | Audio |
| --- | --- | --- | --- |
|  |  | Start on the BING home screen | Good day. This is a quick walk through of Azure Mobile Services. This view will allow you to get a feel for creating a mobile service and connecting up a web page from the sample that is provided after the creation process is finished. |
|  | Step 1 create Azure Data Lake repo | * In the SQL VM, create a Container to house the data by using Azure Data Studio |  |
|  |  | Zoom to screen location and highlight | On the menu bar on the left, you will see the various services available to you in Azure. We will concentrate on the Mobile Services.  Click on the mobile services tab, you will see all the mobile services that are currently setup. Yours might be empty, but any you create will display in the main window |
|  |  | Zoom to the screen size | It is important to note 3 of the columns on the screen. The Backend for this example will be Javascript as this will be an HTML5 application.  You can select either a .Net or Javascript backend, just that this example will be HTML5.  The Location will be the data centre closest to you.  The URL is important as this is the URL for you service. |
|  |  |  | At the bottom of the screen, you will see the NEW and Manage Keys.  The Manage keys is important, but will be assigned automatically when we create the service.  Lets start by clicking the “NEW” service |
|  |  |  | The next step in creating our service is fairly straight forward.  We select Compute, Movile Service, then Create.  This will start the multi step wizard. |
|  |  | Stoping Point? | Now we have to enter the first part of the url. What ever you select as your Something.azure-mobile.net has to be unique and will really become the name of your mobile service.  You have 3 selections in the drop down box for the database. If this is the first service you are creating, you can create a Free 20mb database. If not, you can use your existing SQL Database or create a new DB Instance.  Our example here will be to create a new database instance, but on a SQL Server I already have.    For my site, I will select Eastern US as my data center. Yours should be closer to you, or really your client activity.  Select your backend, Really, Javascript or .Net, we will use Javascript as our example. |
|  |  |  | For our example, I already have a DB Server setup, and have chosen to create a new db instance.  The Name will prepopulate with the name of the service url you created in the previous step with a \_DB attached to it.  We will select the server which will then ask me to log on.  We don’t have to, but we will select the advance setting just to take a look.  Lets move onto the next step |
|  |  |  | The advanced settings allow you to select the service tiers, size, collation and performance level.  The settings are beyond the quick walkthrough, however the MSDN site under mobile services has the explanations.  Click on the check mark to start the creation process. |
|  |  |  | The site will now be created. The status will change as the process goes on, however when “Ready” appears we will be ready for the next step. |
|  |  | Zoom in | Now that we have our service, click on the service name to bring up the details screen. |
|  |  |  | This is the main dashboard for the service. There are many features and setting available along the top screen. Hitting the “Cloud” will bring you back to this screen.  There are a number of platforms you can chose for your application. Since the RestAPIs are created, you can use any of the platforms listed for you application. We will use the “HTML/JavaScript” for our example.  Make sure the “HTML/JavaScript” is selected, and under the Get Started, click on the Create a New HTML APP. |
|  |  |  | There are 3 selections.  We first need to create the table to house our sample data.    We then download the application it will prompt us to save the application. This will save to the download directory.    Lets select the Configure a list of host names. |
|  |  |  | This is mainly so that the site will accept requests from the domain your application is running from. The list already has the LocalHost setup as this is where the site sample code will run from. |
|  |  |  | Now that we have everything ready, lets unzip the download file. |
|  |  |  | Once downloaded, going into the server directory will list the files that get the IIS Server running on this local machine.  There are other files that will do the same on Linux and the MAC. Depending on what machine you are on, start the server.  Note that this cannot be started from a file share. |
|  |  |  | We now have IIS Express running. Open up a browser and open the pate http://localhost:8080 |
|  |  |  | This will bring up the mobile service.  Add in some tasks,  Close it, then open, they are still there. |
|  |  |  | We have used the Mobile Service wizards to create our RestApi services, and downloaded the sample code and have our local server communicating and writing our data to our new services.  I encourage you to find out more at  http://azure.microsoft.com/en-us/documentation/services/mobile-services/ |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |