

Azure SQL Database

Hands-On Lab

Configuration Guide  
Version 2.7

6/13/2016

Prepared for: Guy Haycock

Azure SQL Database

Prepared by: Prowess Consulting

Table of Contents

[Architectural Overview 3](#_Toc425000559)

[Lab-Configuration Prerequisites 3](#_Toc425000560)

[Connecting Your Azure Account 5](#_Toc425000561)

[Deploying a New WingtipTickets Tenant Environment 8](#_Toc425000562)

[Appendix: Explanation of New-WTTEnvironment Parameters 13](#_Toc425000563)

[Appendix: Explanation of Web.config Properties 14](#_Toc425000564)

Table of Contents

[Architectural Overview 3](#_Toc452031719)

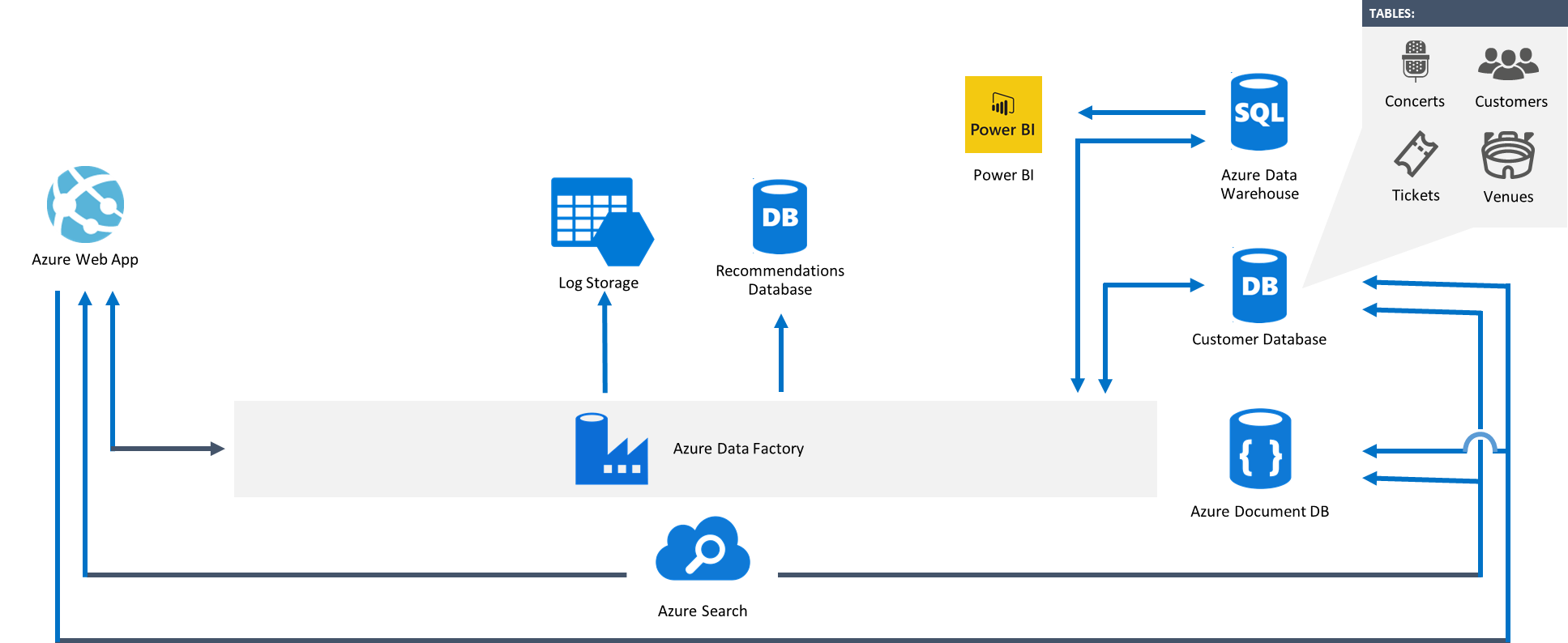
[Lab-Configuration Prerequisites 3](#_Toc452031720)

[Connecting Your Azure Account 9](#_Toc452031721)

[Deploying a New WingTipTickets Tenant Environment 11](#_Toc452031722)

[WingTip Tickets services deployed per PowerShell script 16](#_Toc452031723)

# Architectural Overview



**Figure 1** Overall architecture of the various lab components

# Lab-Configuration Prerequisites

**Table 1** Lab-configuration prerequisites and setup instructions

| **Prerequisite** | **Setup instructions** |
| --- | --- |
| **Microsoft account** | 1. If you don’t already have a Microsoft account, browse to [http://account.live.com](http://account.live.com/). 2. Click **Sign up now**. |
| **Microsoft Azure account** | 1. If you don’t already have an Azure account, browse to <http://azure.microsoft.com/en-us/pricing/free-trial/>. 2. Click **Try it now**. |
| **Microsoft Azure PowerShell** | 1. Browse to <http://azure.microsoft.com/en-us/downloads/>. 2. Under **Command-line tools**, **Windows PowerShell**, click **Install**. 3. Verify that the version being installed is at least 1.5.0 (released June 1, 2016). |
| **Microsoft SQL Server Management Studio** | 1. If SQL Server Management Studio is not installed, and you’d like to run queries against the databases, download and install SQL Server Management Studio from <https://www.microsoft.com/en-us/download/details.aspx?id=42299>. The package is listed as **SQLManagementStudio\_<X86/X64>\_ENU.exe**. |
| **Microsoft SQL Server PowerShell Tools** | 1. Browse to <https://www.microsoft.com/en-us/download/details.aspx?id=42295> 2. Click **Download** 3. Locate **ENU\x64\PowerShellTools.msi** 4. Click **Next** 5. Click **Run** |
| **Deployment scripts** | 1. Unzip **scripts.zip** to a local folder (for example, C:\scripts). |
| **Microsoft Power Query for Excel 2013 (optional)** | 1. If Power Query for Excel is not installed, and you’d like to complete the Auditing section in the hands-on lab (HOL) manual, download and install Power Query from <https://www.microsoft.com/en-us/download/details.aspx?id=39379>. |
| **Microsoft Visual Studio (optional)** | 1. If Visual Studio is not installed, and you’d like to explore any of the source code, download and install Visual Studio from <http://go.microsoft.com/?linkid=9832446&clcid=0x409>. 2. Unzip **sourcecode.zip** to a local folder (for example, C:\sourcecode). |
| **Microsoft Azure .NET software-development kit (SDK) (if installing Visual Studio)** | 1. Browse to <http://azure.microsoft.com/en-us/downloads/>. 2. Under **SDKs**, **.NET**, select the installer for your version of Visual Studio (for example, **VS 2013 Install**). |

Known issue with downloading WingTip Tickets zip files from Github

There is an issue where if the WTT project has been downloaded through the download zip option that some of the files are corrupted.

These files include:

\PowerShell\Packages\PrimaryPackage.zip

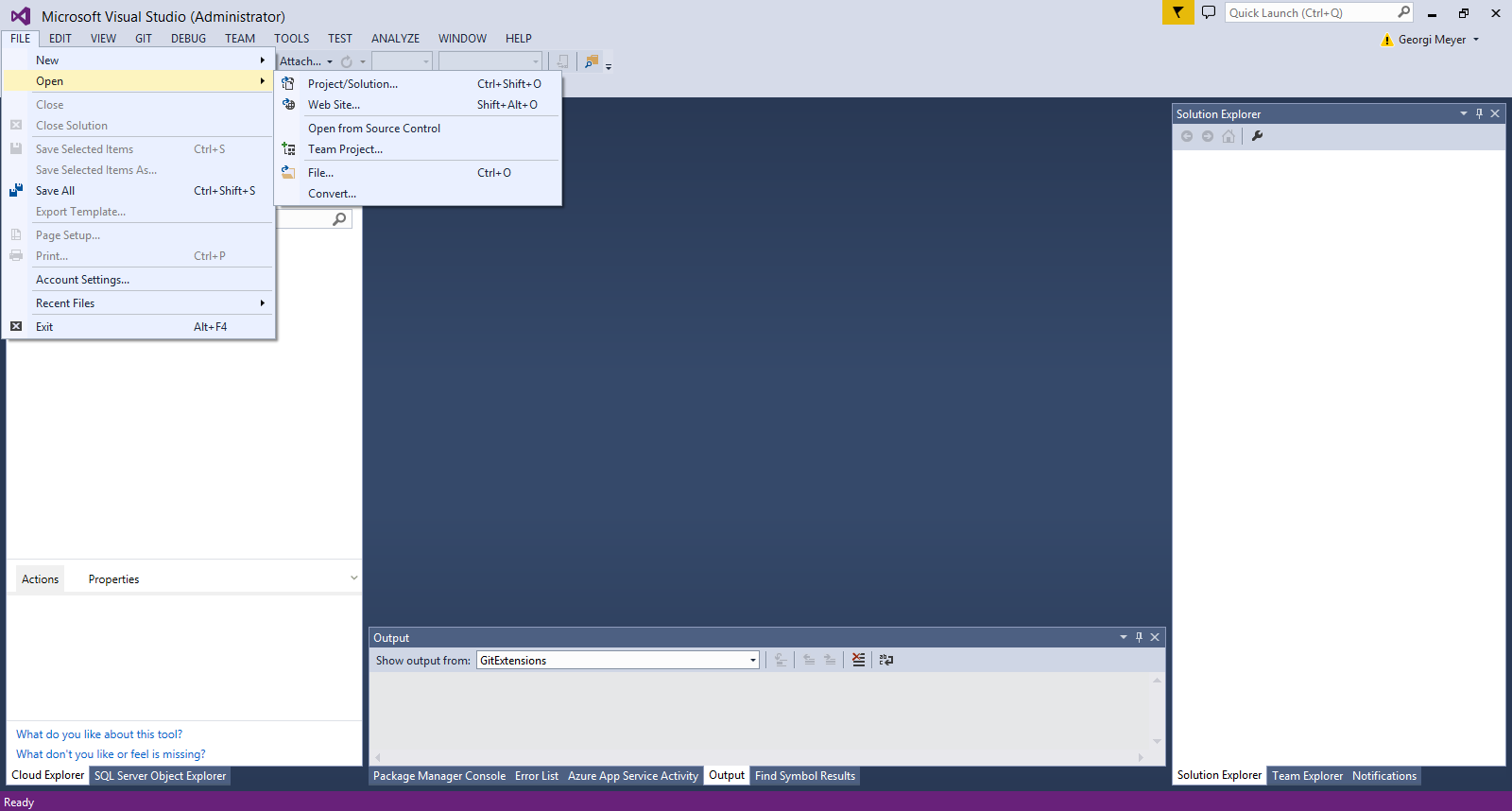
\PowerShell\Packages\SecondaryPackage.zip

\PowerShell\Packages\ProductRecDataGenerator.zip

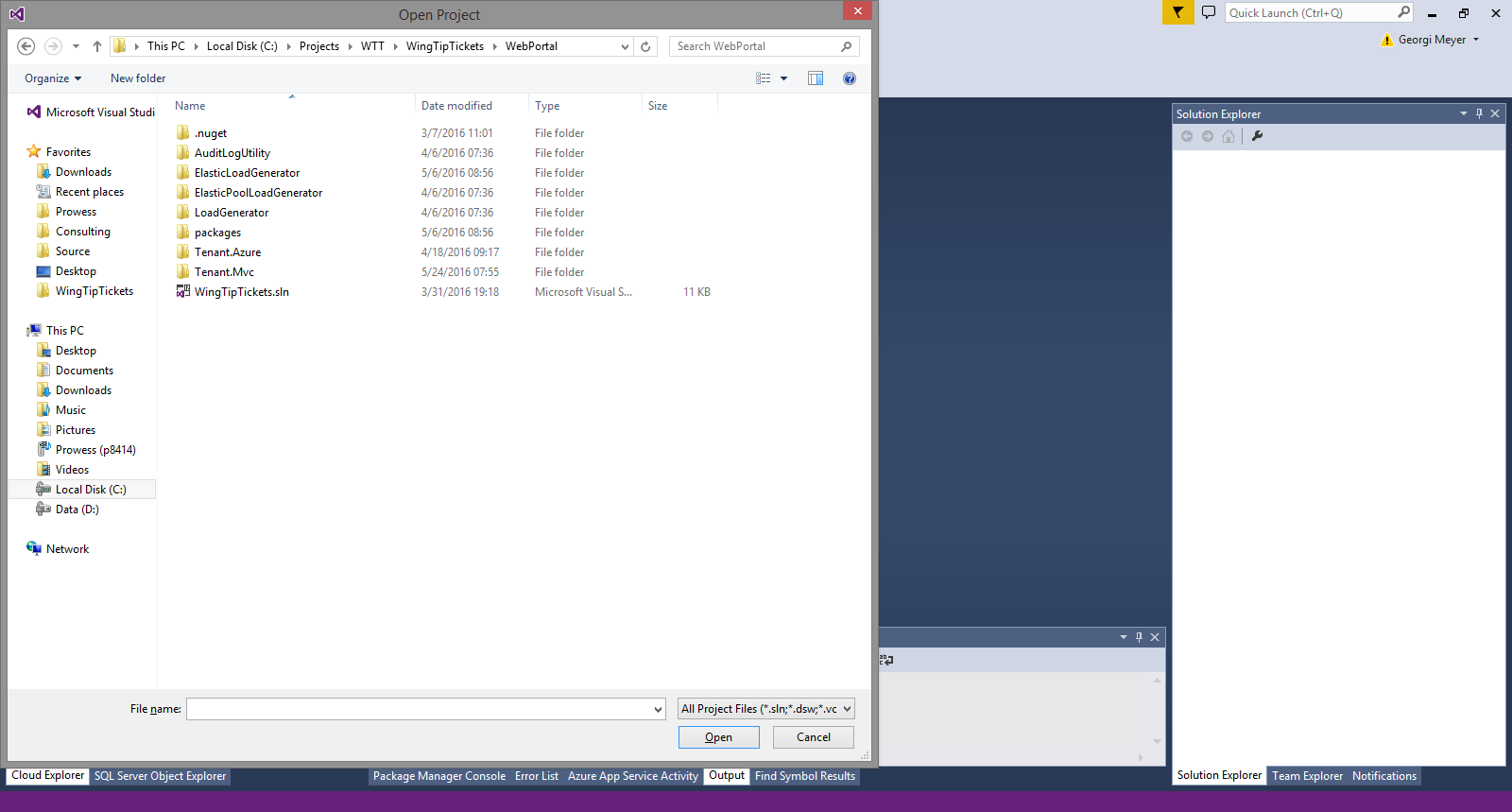
Follow these steps to repackage these files for use for deployment.

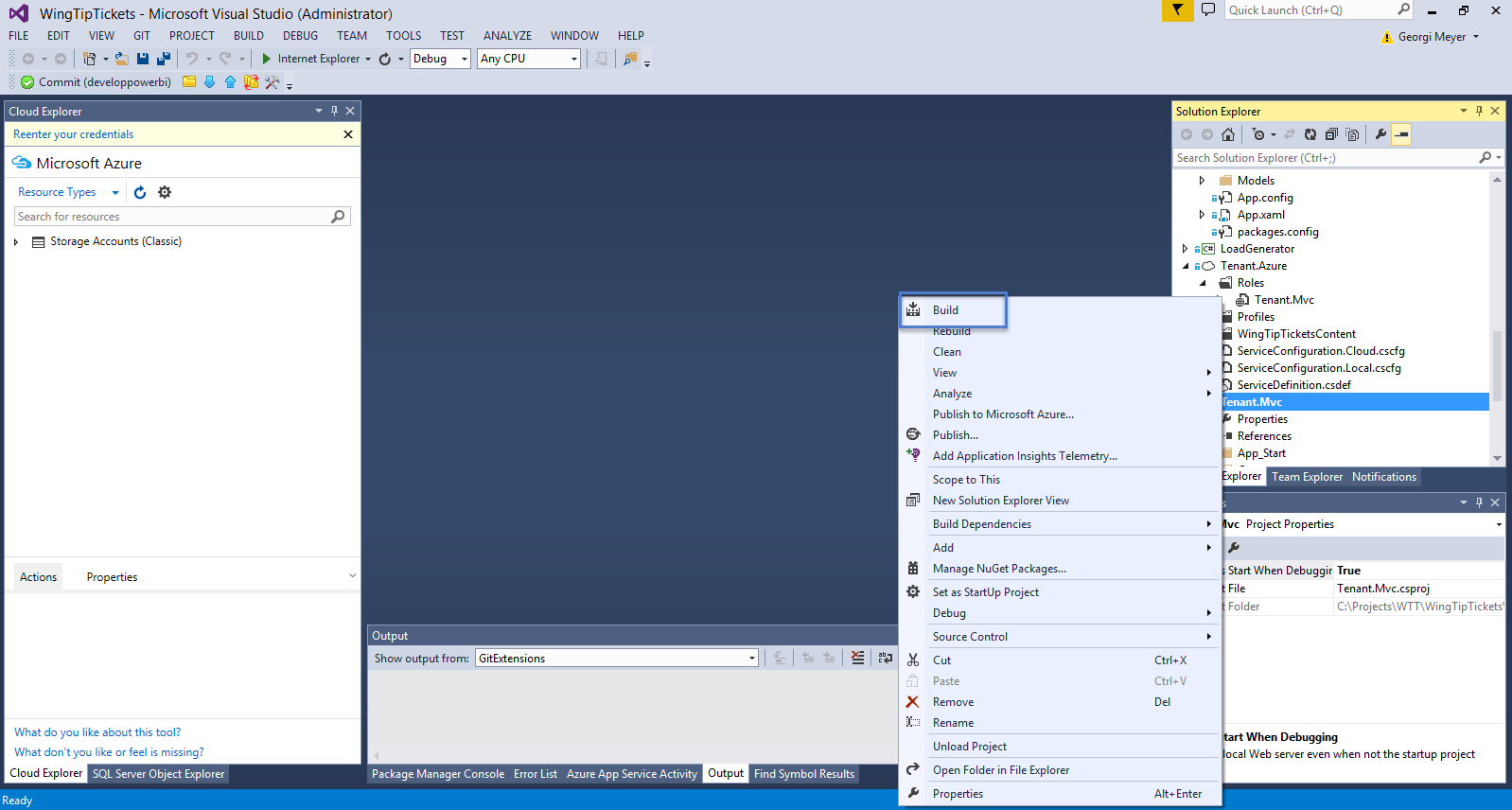
Web App packages

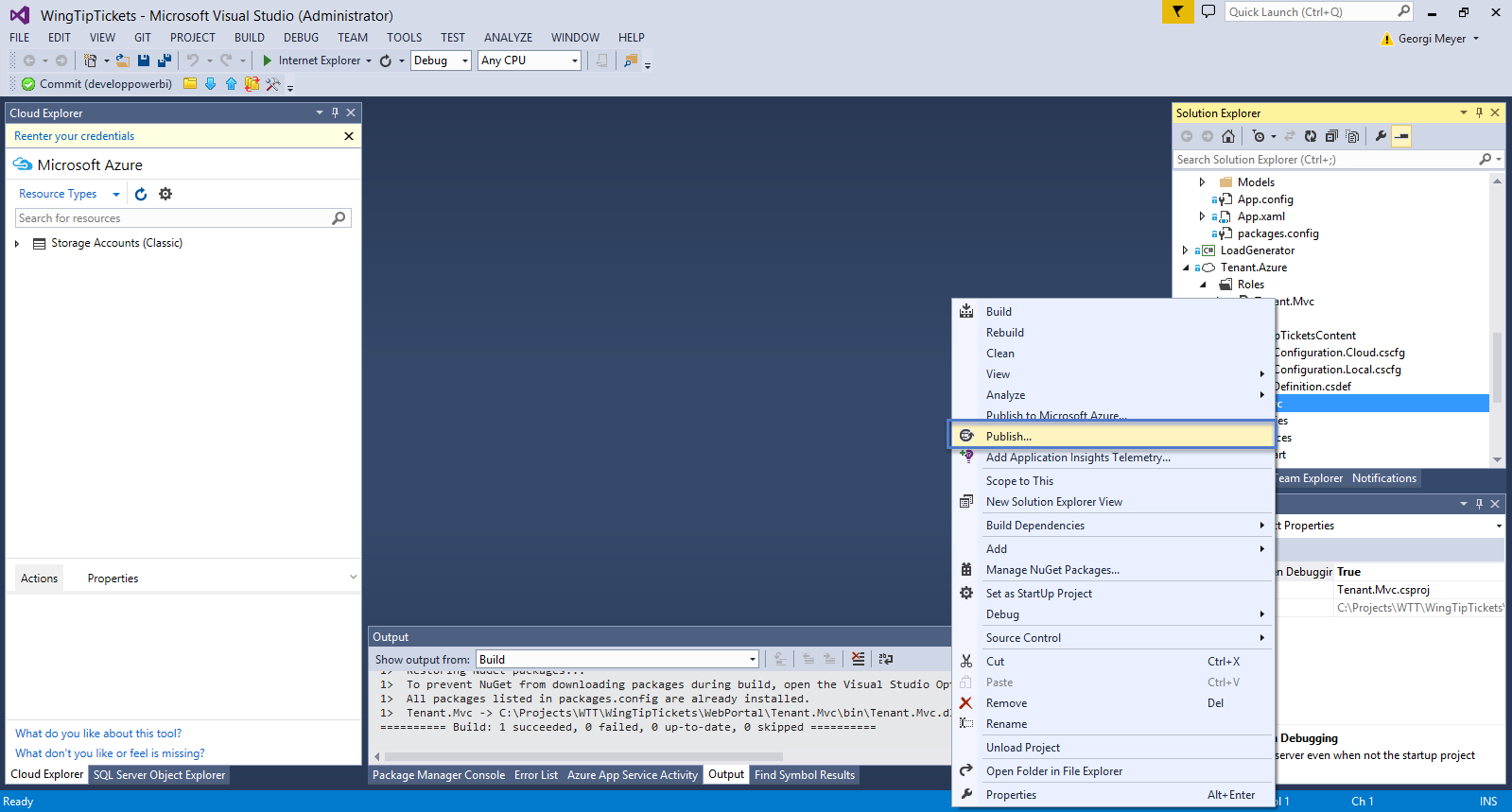
1. Launch **Visual Studio 2013/2015**

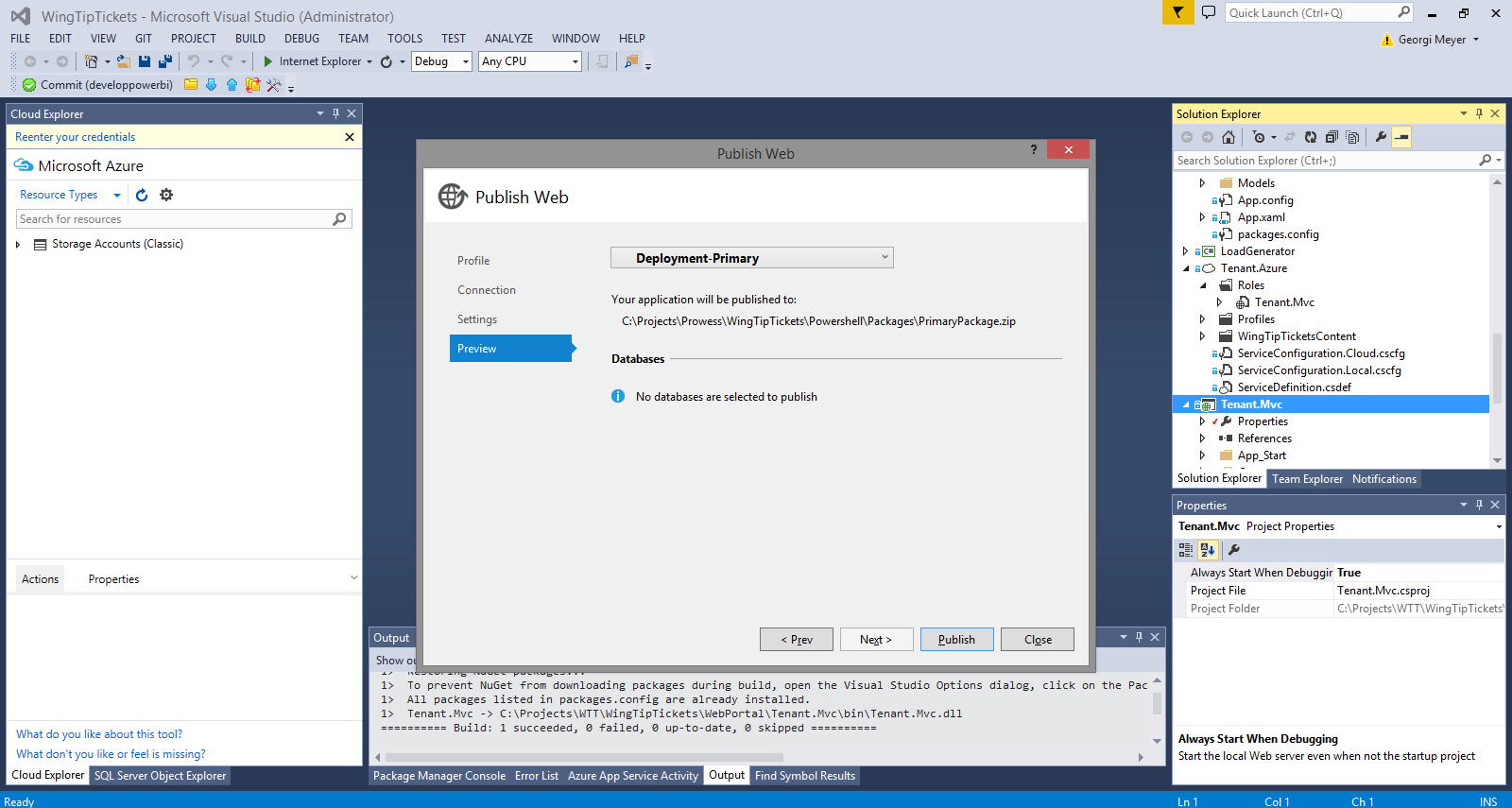
2. Select **File** > **Open Project/Solution**   
  
**Figure 2** Open Visual Studio Project

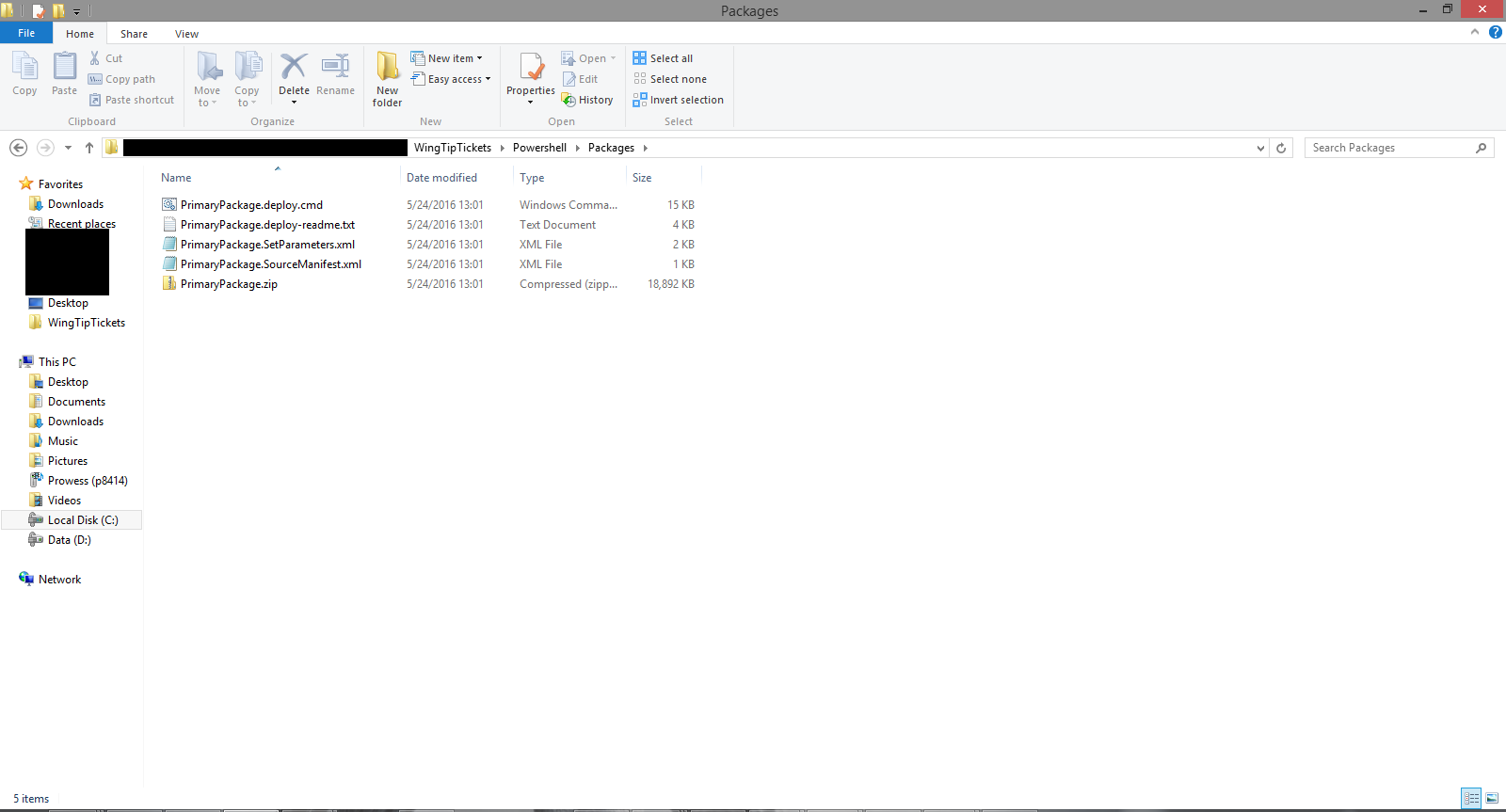
3. Browse to the downloaded WTT files

4. Open \WebPortal, select **WingTipTickets.sln** **Figure 2** Open WingTipTickets.sln project solution

5. Right click **Tenant.mvc**, select **Build** **Figure 3** Build WingTipTickets Tenant.mvc web application

6. Right click **Tenant.mvc**, select **Publish** **Figure 4** Publish WingTipTickets Tenant.mvc web application

7. Click **Publish** ***Figure 5*** *Publish WingTipTickets Tenant.mvc web application*

8. Locate the published **PrimaryPackage.zip** file  
  
**Figure 6** Location of published zip file

9. Copy the new file to **\PowerShell\Packages\**

10. Rename the published file to **SecondaryPackage.zip**

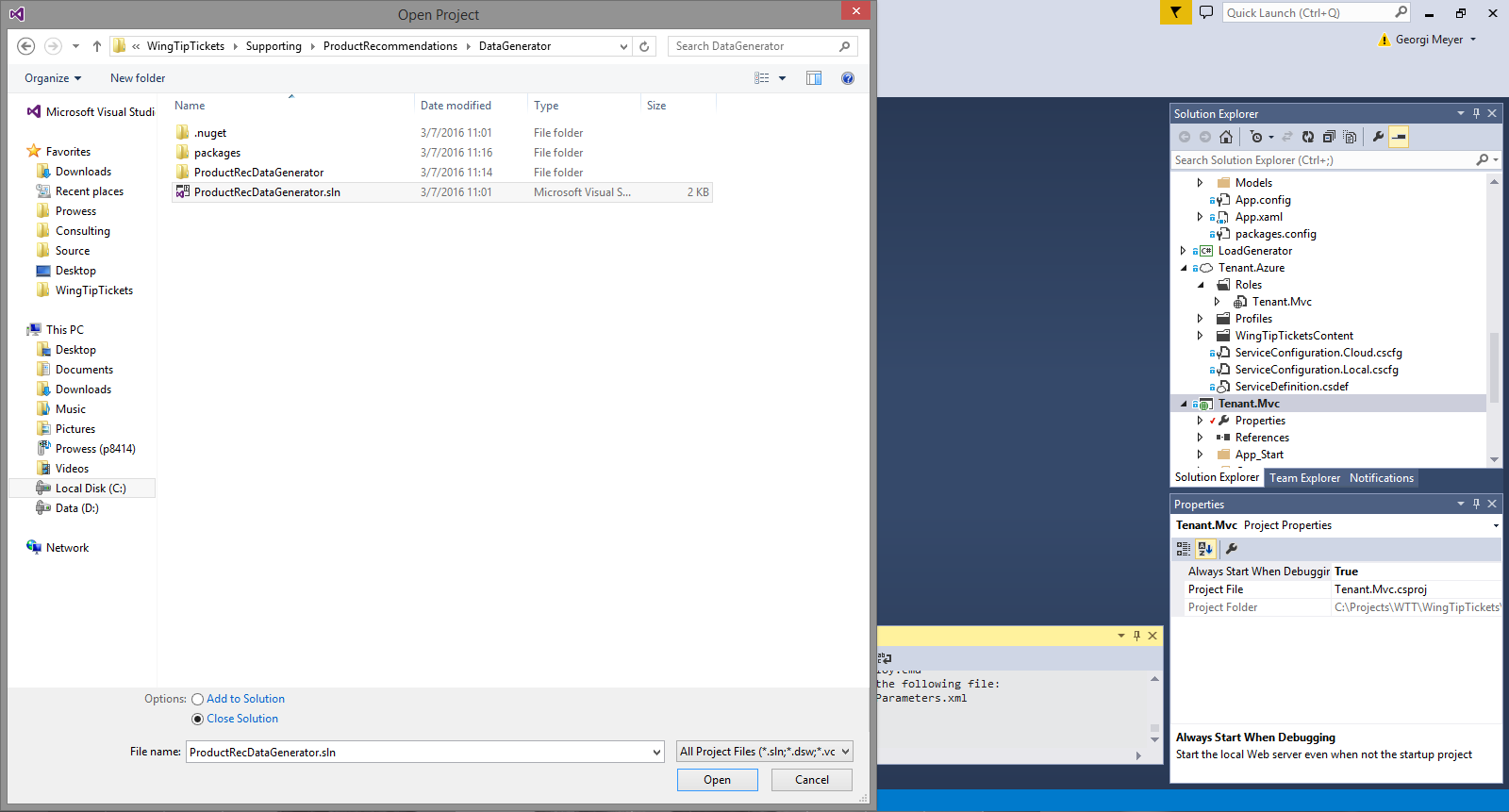
11. Copy the renamed SecondaryPackage.zip file to **\PowerShell\Packages\**

ProductRecDataGenerator

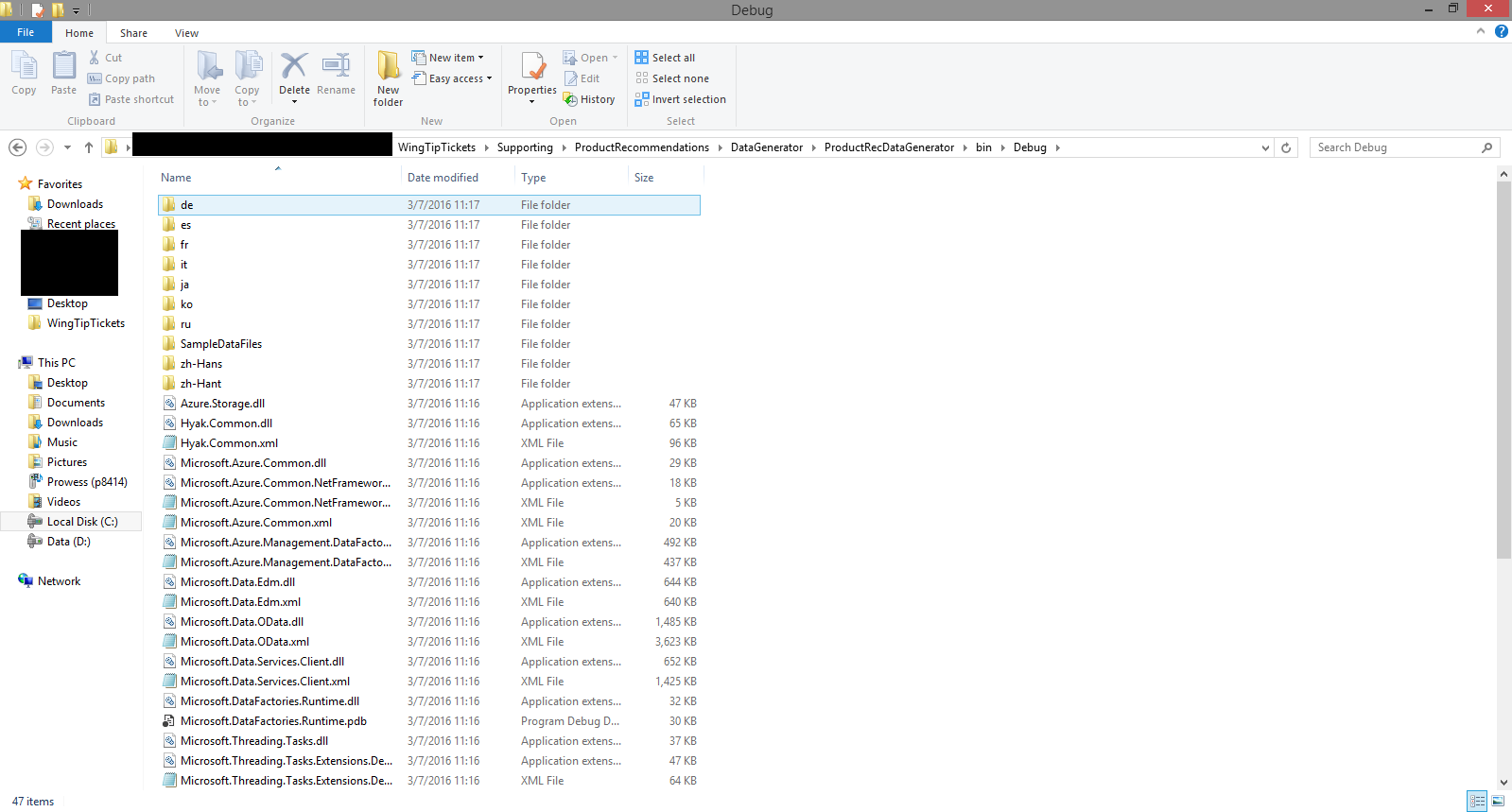
1. Launch **Visual Studio 2013/2015**

2. Select **File** > **Open** **Project/Solution**

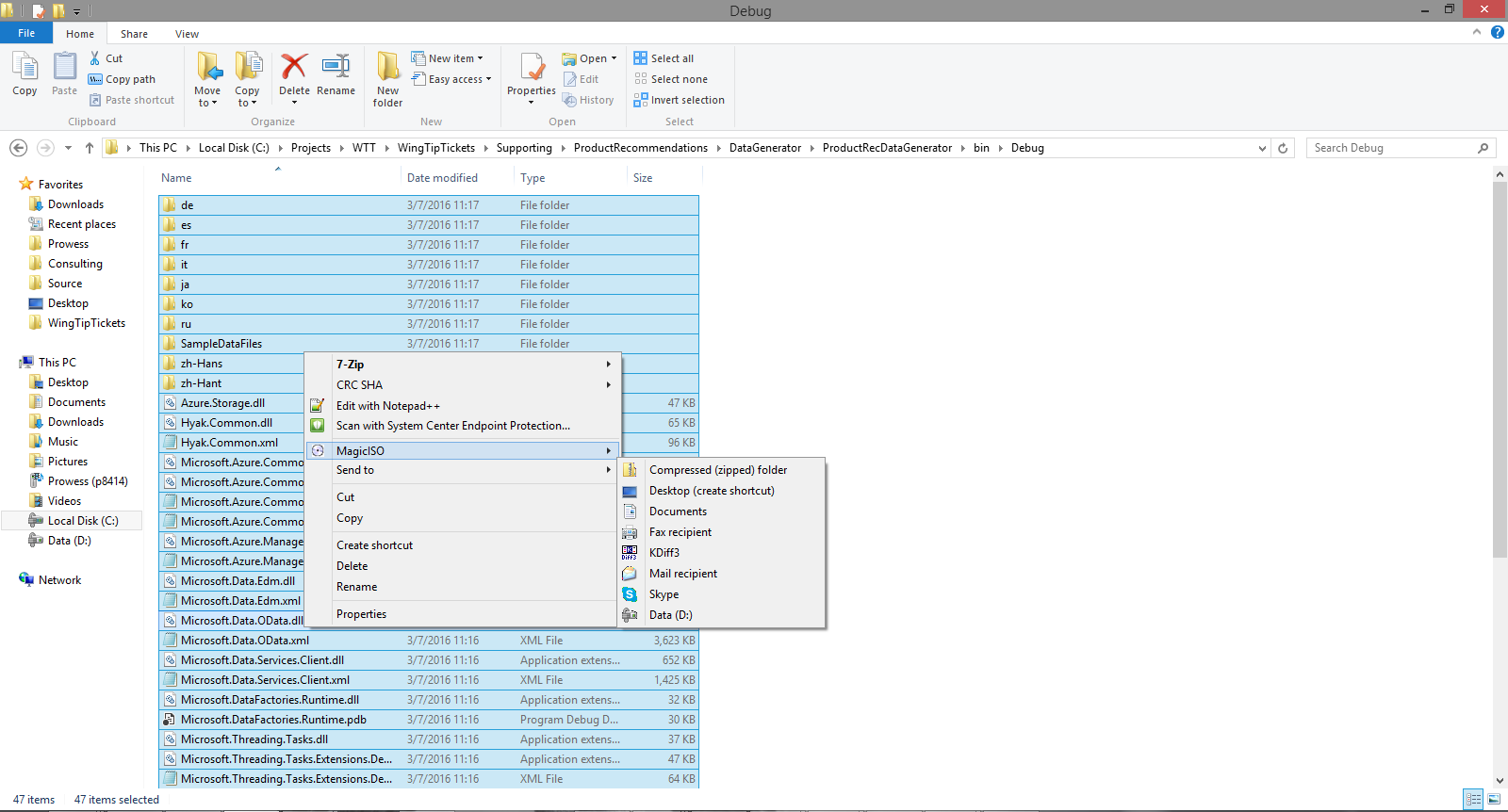
3. Browse to **\Supporting\ProductRecommendations\DataGenerator**

4. Select ProductRecDataGenerator.sln **Figure 7** Open ProductRecDataGenerator.sln project solution

5. Right click **ProjectRecDataGenerator**, select **Build**

6. Browse to the location of the build files  
  
**Figure 8** Location of ProductRecDataGenerator build files

7. Select all the files

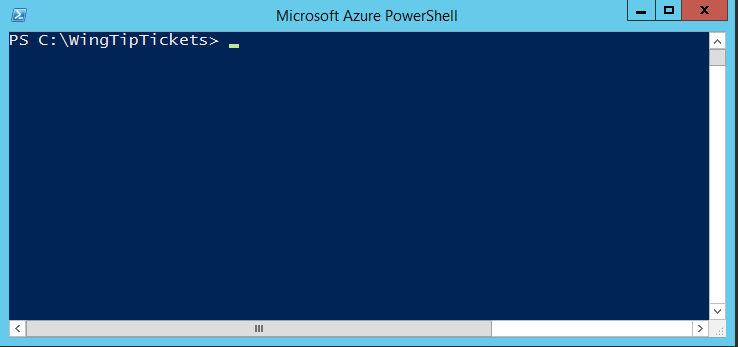
8. Right click, select Send To, select compressed (zipped) folder **Figure 9** Compress ProductRecDataGenerator build files

9. Rename the folder **ProductRecDataGenerator.zip**

10. Copy the zip file to **\PowerShell\Packages\**

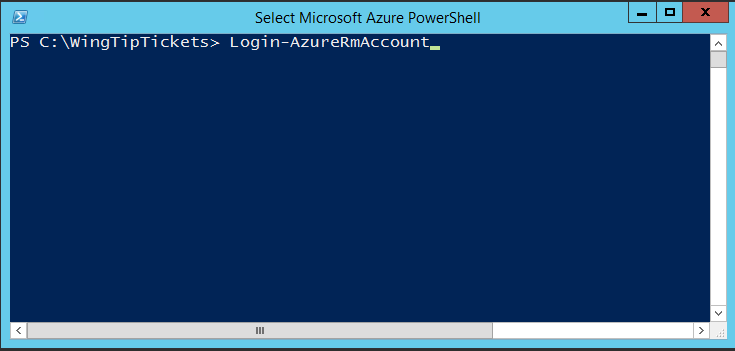
# Connecting Your Azure Account

1. Launch a Microsoft Azure PowerShell session as an administrator (click **Run as administrator**), and then browse to the folder where you’ve saved the scripts.



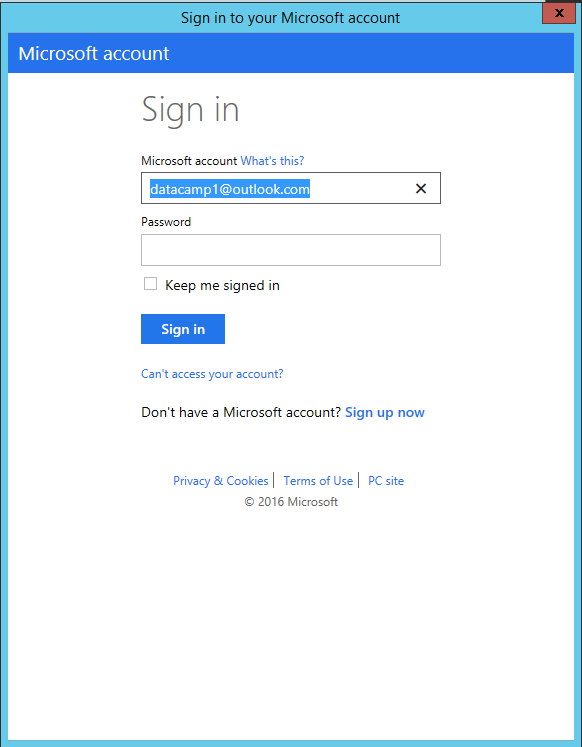
**Figure 10** A Microsoft Azure PowerShell command-line interface session

1. Connect to your Azure account by typing **Login-AzureRMAccount**.



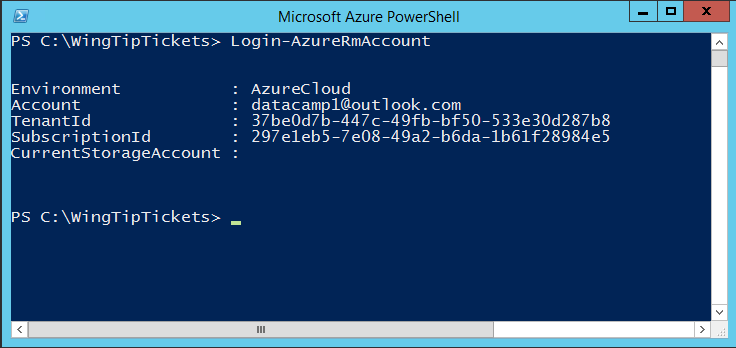
**Figure 11** Connect to your Azure account through Azure PowerShell

1. Type your Azure account credentials.



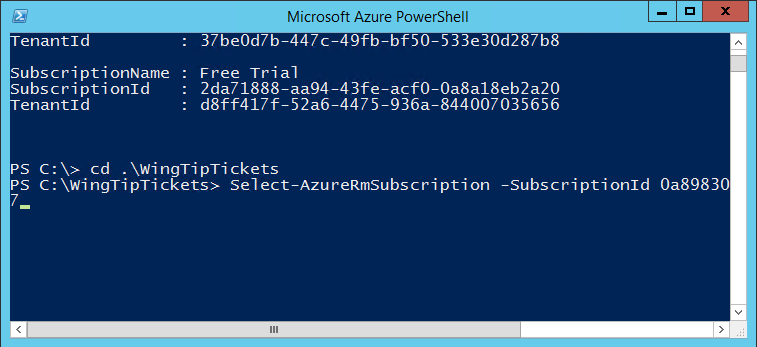
**Figure 12** Type your Azure account credentials

Your account should be linked to your primary subscription, as shown in Figure 5.



**Figure 13** Subscription confirmation in Azure PowerShell

1. If you have more than one Azure Subscription, enter Get-AzureRMSubscription
2. On the command line, type Select-AzureRMSubscription –SubscriptionId *YOUR SUBSCRIPTON ID* (as shown in Figure 6).

  
**Figure 14** You can also change or supply your Microsoft Azure credentials through the Select-AzureRMSubscription command in Microsoft Azure PowerShell

# Deploying a New WingTipTickets Tenant Environment

1. Run the following commands to load the Azure PowerShell script:
   1. **PS C:\Scripts>Set-ExecutionPolicy -Scope LocalMachine -ExecutionPolicy Unrestricted -Force**
   2. **PS C:\Scripts>Unblock-file .\New-WTTEnvironment.ps1**
   3. **PS C:\Scripts>. .\New-WTTEnvironment.ps1**



**Figure 15** Deploy a new tenant environment using the New-WTTEnvironment Azure PowerShell script

**Note:** To load the Azure PowerShell script, you must type a period and then a space before the path to the script: .\New-WTTEnvironment.ps1

1. To verify the script is loaded, type **New-WTTEnvironment -W**, and then press Tab, which should autocomplete to WTTEnvironmentApplicationName.

**Note**: -WTTEnvironmentApplicationName is the most important value because it’s used to prefix Azure resources—for example, storage accounts, web apps, and database servers.

Because this HOL uses the Azure public cloud, it is crucial that you choose a name that is unique in order for the lab to work. For example: your initials followed by julieandtheplantes (xx*julieandtheplantes*).

1. Run the cmdlet using syntax similar to this (remember to substitute the *xx* in *xx*julieandtheplantesfor your initials): **New-WTTEnvironment -WTTEnvironmentApplicationName *xx*julieandtheplantes**

Depending on your network connection, setup should take no more than 15 minutes.



**Figure 16** Example of creating a new deployment in Azure PowerShell

**Note:** During setup, Azure PowerShell will output a number of success messages in green and informational messages in yellow.



**Figure 17** Azure PowerShell success and informational messages that you might encounter during setup (this example shows creating a new deployment)

**Note:** If you have issues setting up the environment, you’ll need to review the errors that are returned to determine which Azure PowerShell function is causing an issue and, subsequently, what is causing the error.

The most common issues are due to name conflicts, webapp-package upload failures, and subscription quota limits.

For name conflicts, you can remove a deployment by running: **PS C:\Scripts> Remove-WTTEnvironment -WTTEnvironmentApplicationName *xx*julieandtheplantes**. This will delete any resources that were created by the New-WTTEnvironment cmdlet. Then try to re-run the New-WTTEnvironment cmdlet again as in step 6, using a different   
-WTTEnvironmentApplicationName (for example, *xx1*julieandtheplantes).

For package upload failures, re-running the same command that you previously ran in step 6 will try the upload again.

For pay-as-you-go subscription quota-limit-related issues, check the following:

* Typically there is a limit of six Azure SQL Database servers. Make sure there are no more than four Azure SQL Database servers in your subscription before running the New-WTTEnvironment cmdlet.
* Typically there is a limit of one free Azure Search service. Make sure there are no Azure Search services in your subscription before running the New-WTTEnvironment cmdlet.
* For other issues, please refer to [FAQ section](#FAQ) of this document.

1. Once the deployment is completed, open a browser and browse to [http://*xx*julieandtheplanets.trafficmanager.net](http://xxjulieandtheplanets.trafficmanager.net) (remember to replace *xx* with your initials).



**Figure 18** Web site running correctly on the primary web app

## Appendix: Explanation of New-WTTEnvironment Parameters

**Table 2** New-WTTEnvironment parameters

| **Parameter** | **Purpose** | **Default Value** |
| --- | --- | --- |
| -WTTEnvironmentApplicationName | Name that will differentiate your WingTipTickets tenant environment from others running in the Azure public cloud (in this example, *xx*julieandtheplantes) |  |
| -AzureSqlDatabaseServerAdministratorUserName | Database server-administrator user name | developer |
| -AzureSqlDatabaseServerAdministratorPassword | Database server-administrator password | P@ssword1 |
| -AzureSqlDatabaseServerVersion | Azure SQL Database server version | 12.0 |
| -AzureSqlDatabaseName | Name of the tenant database | Customer1 |
| -AzureWebSiteWebDeployPackagePath | Path to the Azure Web App Web Deploy packages | <unzip location>\Scripts\Packages |
| -AzureWebSitePrimaryWebDeployPackageName | Primary Web App Web Deploy package name | primarypackage.zip |
| -AzureWebSiteSecondaryWebDeployPackageName | Secondary Web App Web Deploy package name | secondarypackage.zip |
| -WTTEnvironmentPrimaryServerLocation | Azure datacenter region | <auto configured based on capacity> |

## Appendix: Explanation of Web.config Properties

**Table 3** Web.config property names and descriptions

|  |  |  |
| --- | --- | --- |
| **Property name** | **Description** | **Value** |
| TenantEventTypeGenre | Tenant event type used to dynamically theme the site; valid values include: pop, rock, and classical | Pop |
| TenantEventName | Tenant (application) name, used to dynamically theme the site | Set from  –WTTEnvironmentApplicationName <value> |
| PrimaryDatabaseServer | Name for the primary Azure SQL Database server where concerts, customers, venues, and tickets tables exist | Set from  –WTTEnvironmentApplicationName <value> + primary |
| SecondaryDatabaseServer | Name of the Azure SQL Database server that is configured as the target server (also known as the secondary) for geo-replication | Set from  –WTTEnvironmentApplicationName <value> + secondary |
| DatabaseUserName | User name to be used for all application-related connections to the Azure SQL Database servers | Set from  -AzureSqlDatabaseServerAdministratorUserName |
| DatabaseUserPassword | Password to be used for all application-related connections to the Azure SQL Database servers | Set from  -AzureSqlDatabaseServerAdministratorPassword |
| TenantDbName | Name for the Azure SQL Database tenant database, where concerts, customers, venues, and tickets tables exist | Set from  -AzureSqlDatabaseName |
| SearchServiceName | Name of the Azure Search service that indexes a view for the tables in the TenantDbName database | Set from  –WTTEnvironmentApplicationName <value> |
| SearchServiceKey | Azure Search service key | Dynamically retrieved during setup |
| DocumentDbUri | URI of the DocumentDB service that stores ad-hoc venue information. | Dynamically retrieved during setup |
| DocumentDbKey | Primary access key used to access the DocumentDB Service. | Dynamically retrieved during setup |
| RecommendationDatabaseServer | Name for the primary Azure SQL Database server where concerts tables exist | Dynamically retrieved during setup |
| RecommendationDatabase | Name for the Azure Data Factory Azure SQL Database where concerts exist. | Dynamically retrieved during setup |
| powerbiSigningKey | Primary access key of the Azure Power BI Workspace Collection | Dynamically retrieved during setup |
| powerbiWorkspaceCollection | Name of the Azure Power BI Workspace Collection | Dynamically retrieved during setup |
| powerbiWorkspaceId | Name of the Azure Power BI Workspace | Dynamically retrieved during setup |
| SeatMapReportId | Report ID of the SeatMapReport in the Azure Power BI Workspace | Dynamically retrieved during setup |

## 

## Frequently Asked Questions

1. **Q**: I am getting ”account expired” or “400 bad request” errors in the deployment after the Add-AzureAccount step, or an account is listed that doesn’t belong to me when calling Get-AzureSubscription in Azure PowerShell.

**A**: You need to type **Get-AzureAccount | Remove-AzureAccount** in Azure PowerShell. Close the Azure PowerShell console, open a new Azure PowerShell, and then try the **Add-AzureAccount** command again.

1. **Q**: I am getting a “Key Not Found in Dictionary” error when running the Add-AzureAccount command.

**A**: Try the following steps:

* **Remove-AzureAccount**
* **Clear-AzureProfile**
* **Add-AzureAccount**

1. **Q**: I have an active Microsoft Developer Network (MSDN) subscription, and it seems I have enough resources, but my provisioning failed somehow, and I am not sure why.

**A**: Please check that you have Azure Search in your subscription by going to [http://portal.a**zure.com**](http://portal.azure.com). Log on with your account, and then click **Browse All**. If Azure Search is not shown as available, you need to switch to a subscription that does include Azure Search service.

# WingTip Tickets services deployed per PowerShell script

Contents

[New-WTTEnvironment.ps1 18](#_Toc450891633)

[Get-WTTSqlDatabaseServerV12RegionAvailability.ps1 19](#_Toc450891634)

[Get-WTTAzureSqlDatabaseServerRegionCapabilities.ps1 20](#_Toc450891635)

[New-WTTAzureResourceGroup.ps1 20](#_Toc450891636)

[New-WTTAzureStorageAccount.ps1 20](#_Toc450891637)

[New-WTTAzureDocumentDb.ps1 21](#_Toc450891638)

[New-WTTAzureSqlDatabaseServer.ps1 21](#_Toc450891639)

[Deploy-DBSchema.ps1 22](#_Toc450891640)

[Populate-DBSchema.ps1 23](#_Toc450891641)

[Populate-Tickets.ps1 23](#_Toc450891642)

[New-WTTAzureSearchService.ps1 24](#_Toc450891643)

[Deploy-WTTWebApplication.ps1 25](#_Toc450891644)

[New-WTTAzureTrafficManagerProfile.ps1 25](#_Toc450891645)

[Add-WTTAzureTrafficManagerEndpoint.ps1 25](#_Toc450891646)

[Deploy-WTTAzureDWDatabase.ps1 26](#_Toc450891647)

[New-WTTADFEnvironment.ps1 27](#_Toc450891648)

[New-WTTPowerBI.ps1 27](#_Toc450891649)

[Set-WTTEnvironmentWebConfig.ps1 28](#_Toc450891650)

[Test-WTTAzureSQLConnection.ps1 29](#_Toc450891651)

New-WTTEnvironment.ps1  
Command:  
New-WTTEnvironment –WTTEnvironmentApplicationName <Environment Name> -WTTEnvironmentPrimaryServerLocation <Primary Server Location> -AdminUserName <Developer> -AdminPassword <P@ssword1> -AzureSqlServerVersion <12.0> -AzureSqlDatabaseName <Customer1> -AzureSqlDWDatabaseName <CustomerDW> -WebAppPackagePath <C:\wingtiptickets> -webAppPrimaryPackageName <primarypackage.zip> -webAppSecondaryPackageName <secondarypackage.zip> -tenantEventType <pop, rock, classical>   
What’s Deployed:  
This script is the main script that runs through and deploys all the services that are need to run the WingTip tickets demo.   
Parameters:

**Table 4** New-WTTEnvironment parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| –WTTEnvironmentApplicationName | XX00JulieandthePlantes XX is name initials | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -WTTEnvironmentPrimaryServerLocation | Data Center Location West US East US  West Europe East Europe  Etc. | Primary Azure SQL Server data center location |
| -adminUserName | Developer | Azure SQL Server administrator user name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlServerVersion | v2.0 or v12.0  v12.0 is default | Default Azure SQL Server version |
| -AzureSqlDatabaseName | Customer1 | Azure SQL Server database name |
| -AzureSqlDWDatabaseName | CustomerDW | Azure SQL Server Data Warehouse name |
| -WebAppPackagePath | Path to files | Path to where the PowerShell scripts and supporting files are stored |
| -webAppPrimaryPackageName | PrimaryPackage.zip | Name of the web application primary package |
| -webAppSecondaryPackageName | SecondaryPackage.zip | Name of the web application secondary package |
| -tenantEventType | Pop, rock, or classical | Web application theme |

Get-WTTSqlDatabaseServerV12RegionAvailability.ps1  
Command:  
Get-WTTSqlDatabaseServerV12RegionAvailability –azureResourceGroupName $azureResourceGroupName

What’s Deployed:  
This script goes through the list of available data centers and verifies if the data center is available for deployment. It also assists in automating the selection of a data center in the new-wttenvironment.ps1 script. This helps in that the end user does not have to provide the –WTTEnvironmentPrimaryServerLocation parameter. Both the primary and secondary Azure SQL Server locations are selected through this script.  
Parameters:

**Table 5** Get-WTTSQLDatabaseServerV12RegionAvailability parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| –azureResourceGroupName | XX00JulieandthePlantes XX is name initials | This is the resource group used throughout the PowerShell scripts to name the services that are deployed |

Get-WTTAzureSqlDatabaseServerRegionCapabilities.ps1  
Command:  
Get-WTTAzureSqlDatabaseServerRegionCapabilities

What’s Deployed:  
This script is used in the Get-WTTSqlDatabaseServerV12RegionAvailability.ps1 PowerShell script. It gets the list of locations that is available to SQL deployment and verifies that is available for deployment.  
Parameters:  
No parameters needed for this PowerShell script.

New-WTTAzureResourceGroup.ps1  
Command:  
New-WTTAzureResourceGroup -AzureResourceGroupName $azureResourceGroupName -AzureResourceGroupLocation $primaryServerLocation   
What’s Deployed:  
This script deploys the Azure Resource Group that all the Azure Resources.  
Parameters:

**Table 6** New-WTTAzureResourceGroup parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -AzureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -AzureResourceGroupLocation | Global | Location of the Azure Resource Group, does not necessarily have to be in the same location as the rest of the resources. |

New-WTTAzureStorageAccount.ps1  
Command:  
New-WTTAzureStorageAccount -azureResourceGroupName $azureResourceGroupName -AzureStorageAccountName $azureStorageAccountName -AzureStorageAccountType "Standard\_GRS" -AzureStorageLocation $primaryServerLocation

What’s Deployed:  
This script deploys the Azure Storage Account that is used to deploy the web applications and ADF services from. It also hosts the Azure SQL Server Audit logs.  
Parameters:

**Table 7** New-WTTAzureStorageAccount parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -AzureStorageAccountName | XX00JulieandthePlantes | Name of the Azure Storage Account |
| -AzureStorageAccountType | Standard\_GRS | Azure Storage Account type, Standard\_GRS is default |
| -AzureStorageLocation | Global | Azure Storage Account location, same location as the Azure SQL Server primary server |

New-WTTAzureDocumentDb.ps1  
Command:  
New-WTTAzureDocumentDb -azureResourceGroupName $azureResourceGroupName -WTTDocumentDbName $azureDocumentDbName -WTTDocumentDbLocation $WTTDocumentDbLocation

What’s Deployed:  
This script deploys the DocumentDB service used in the Web Application.  
Parameters:

**Table 8** New-WTTAzureDocumentDB parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -WTTDocumentDbName | XX00JulieandthePlantes | Name of the Azure DocumentDB service |
| -WTTDocumentDbLocation | 'West US', 'South Central US', ‘East US’, 'Central US', 'East Asia', 'West Europe', 'Japan East', 'Japan West', 'North Europe', 'Southeast Asia', 'Australia East', 'Australia Southeast' | Location to deploy the DocumentDB service to |

New-WTTAzureSqlDatabaseServer.ps1  
Command:  
New-WTTAzureSqlDatabaseServer -azureSqlServerName $azureServerPrimaryName -azureSqlServerLocation $primaryServerLocation -adminUserName $adminUserName -adminPassword $adminPassword -AzureSqlServerVersion $AzureSqlDatabaseServerVersion –azureResourceGroupName $azureResourceGroupName  
What’s Deployed:

This script deploys the Azure SQL Server. Run it twice to setup a primary Azure SQL Server in the primary region and a second time to deploy a secondary Azure SQL Server in the secondary region. The primary Azure SQL Server hosts the Customer1, Customer2, Customer3, Recommendations, and CustomerDW databases.   
Parameters:

**Table 9** New-WTTAzureSQLDatabaseServer parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureSqlServerName | XX00JulieandthePlantesPrimary  XX00JulieandthePlantesSecondary | Azure SQL Server primary or secondary server names |
| -azureSqlServerLocation | 'East US', 'West US', 'South Central US', 'North Central US', 'Central US', 'East Asia', 'West Europe', 'East US 2', 'Japan East', 'Japan West', 'Brazil South', 'North Europe', 'Southeast Asia', 'Australia East', 'Australia Southeast' | Azure SQL Server location for the primary or secondary server |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlServerVersion | 12.0 | Azure SQL Server version to deploy, v12 is default |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |

Deploy-DBSchema.ps1  
Command:  
Deploy-DBSchema -azureResourceGroupName $azureResourceGroupName

-azureSqlServerName $azureSqlServerPrimaryName -DatabaseEdition "Basic" -adminUserName $adminUserName -adminPassword $adminPassword -azureSqlDatabaseName $AzureSqlDatabaseName

What’s Deployed:  
Customer1, Customer2, and Customer3 DBs Schema is deployed to the database.  
Parameters:

**Table 10** Deploy-DBSchema parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -azureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -DatabaseEdition | Basic Standard Premium | Azure SQL Database edition |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -azureSqlDatabaseName | Customer1 Customer2 Customer3 | Azure SQL Database name |

Populate-DBSchema.ps1  
Command:  
Populate-DBSchema -azureResourceGroupName $azureResourceGroupName -azureSqlServerName $azureSqlServerPrimaryName -adminUserName $adminUserName -adminPassword $adminPassword -AzureSqlDatabaseName $AzureSqlDatabaseName

What’s Deployed:

This script populates the Customer1, Customer2, and Customer3 databases with the content needed for the WingTip Tickets application.  
Parameters:

**Table 11** Populate-DBSchema parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -azureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlDatabaseName | Customer1  Customer2  Customer3 | Azure SQL Database containing content for the WTT application |

Populate-Tickets.ps1  
Command:  
Populate-Tickets -azureResourceGroupName $azureResourceGroupName -adminUserName $adminUserName -adminPassword $adminPassword -AzureSqlDatabaseName $AzureSqlDatabaseName -AzureSqlServerName $azureSqlServerPrimaryName

What’s Deployed:  
This script populates the tickets table with data to display tickets purchased in the Seat Chart Map shown in the venue.  
Parameters:

**Table 12** Populate-Tickets parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -AzureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlDatabaseName | Customer1 | Azure SQL Database containing content for the WTT application |

New-WTTAzureSearchService.ps1  
Command:  
new-wttazuresearchservice -wttenvironmentapplicationname $wttenvironmentapplicationname -azureResourceGroupName $azureResourceGroupName -azuresearchservicelocation $primaryServerLocation -AzureSqlServerName $azureSqlServerPrimaryName -adminUserName $adminUserName -adminPassword $adminPassword -AzureSqlDatabaseName $AzureSqlDatabaseName

What’s Deployed:  
This script deploys the Azure Search Service. It checks if there are any free search services current in the Azure subscription and if there is deploys a Standard Azure Search Service. This PowerShell script also uses a series of REST commands to setup the index on the Customer1 Database.  
Parameters:

**Table 13** New-WTTAzureSearchService parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -wttenvironmentapplicationname | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -azuresearchservicelocation | West US, East US, North Central US, South Central US, North Europe, West Europe, East Asia, Southeast Asia, Japan West, Brazil South, Australia East | Azure Search Service location. |
| -AzureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlDatabaseName | Customer1 | Name of the Azure SQL Database used in the Azure Search Service |

Deploy-WTTWebApplication.ps1  
Command:  
Deploy-WTTWebApplication -azureStorageAccountName $azureStorageAccountName -azureResourceGroupName $azureResourceGroupName -Websitename $azureSqlServerPrimaryName -WebAppPackagePath $WebAppPackagePath -webAppPackageName $webAppPrimaryPackageName

What’s Deployed:  
This script deploys the web application package to the web application that is created through the new-wttenvironment.ps1 PowerShell script.   
Parameters:

**Table 14** Deploy-WTTWebApplication parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -Websitename | XX00JulieandthePlantesPrimary  XX00JulieandthePlantesSecondary | Name of the web app name, either primary or secondary |
| -WebAppPackagePath | Path to files | Path to where the PowerShell scripts and supporting files are stored |
| -webAppPackageName | PrimaryPackage.zip  SecondaryPackage.zip | Name of the web application package |

New-WTTAzureTrafficManagerProfile.ps1  
Command:  
New-WTTAzureTrafficManagerProfile -AzureTrafficManagerProfileName $wTTEnvironmentApplicationName -AzureResourceGroupName $azureResourceGroupName

What’s Deployed:  
This script deploys the traffic manager to show fault tolerance with Azure Web Applications.  
Parameters:

**Table 15** New-WTTAzureTrafficManagerProfile parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -AzureTrafficManagerProfileName | XX00JulieandthePlantes | Traffic Manager Name |
| -AzureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |

Add-WTTAzureTrafficManagerEndpoint.ps1  
Command:  
Add-WTTAzureTrafficManagerEndpoint -AzureTrafficManagerProfileName $wTTEnvironmentApplicationName -azurePrimaryWebAppName $azureSqlServerPrimaryName -azureSecondaryWebAppName $azureSqlServerSecondaryName -AzureTrafficManagerEndpointStatus "Enabled" -AzureResourceGroupName $azureResourceGroupName

What’s Deployed:  
Adding the Web Applications to the Traffic Manager  
Parameters:

**Table 16** New-WTTAzureTrafficManagerEndpoint parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -AzureTrafficManagerProfileName | XX00JulieandthePlantes | Traffic Manager Name |
| -azurePrimaryWebAppName | XX00JulieandthePlantesPrimary | Primary web app name |
| -azureSecondaryWebAppName | XX00JulieandthePlantesSecondary | Secondary web app name |
| -AzureTrafficManagerEndpointStatus | Enabled Disabled | Status of the web application in the Azure Traffic Manager |
| -AzureResourceGroupName | XX00JulieandthePlantes | Azure Traffic Manager Resource group name, usually same as -WTTEnvironmentApplicationName |

Deploy-WTTAzureDWDatabase.ps1  
Command:  
Deploy-WTTAzureDWDatabase -azureResourceGroupName $azureResourceGroupName -azureSqlServerName $azureSqlServerPrimaryName -DatabaseEdition "DataWarehouse" -adminUserName $adminUserName -adminPassword $adminPassword -azureDWDatabaseName $AzureSqlDWDatabaseName

What’s Deployed:  
Azure Data Warehouse with CustomerDW data warehouse database. This data base is also populated with the schema and data needed for the hands on lab. The data warehouse data is loaded via Polybase from a preconfigured storage account.  
Parameters:

**Table 17** Deploy-WTTAzureDWDatabase parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -azureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -DatabaseEdition | DataWarehouse | Azure SQL Database Data Warehouse edition parameter. This specifies that we are deploying a Data Warehouse and not a normal Azure SQL Database |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -azureDWDatabaseName | CustomerDW | Name of the Azure SQL Data Warehouse |

New-WTTADFEnvironment.ps1  
Command:  
New-WTTADFEnvironment -ApplicationName $WTTEnvironmentApplicationName -azureResourceGroupName $azureResourceGroupName -azureSqlServerName $azureSqlServerPrimaryName -azureSQLDatabaseName "Recommendations" -DatabaseEdition "Basic" -adminUserName $adminUserName -adminPassword $adminPassword

What’s Deployed:  
This script deploys Azure Data Factory, Recommendations database for the Azure Data Factory, the necessary Azure Storage Account blob storage, and the datasets, pipelines, and linked services for the Azure Data Factory data processing.  
Parameters:

**Table 18** New-WTTADFEnvironment parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -ApplicationName | XX00JulieandthePlantes | Name of the Azure Data Factory environment |
| -azureResourceGroupName | XX00JulieandthePlantes | Name of the Azure Resource Group |
| -azureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -azureSQLDatabaseName | Recommendations | Azure SQL Database name to process Azure Data Factory |
| -DatabaseEdition | Basic | Azure SQL Database edition |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |

New-WTTPowerBI.ps1  
Command:  
New-WTTPowerBI -azureResourceGroupName $azureResourceGroupName -AzurePowerBIName $azurePowerBIWorkspaceCollection -azurePowerBILocation $azurePowerBILocation -AzureSqlServerName $azureSqlServerPrimaryName -adminUserName $adminUserName -adminPassword $adminPassword -AzureSqlDatabaseName $AzureSqlDatabaseName -azureDWDatabaseName $AzureSqlDWDatabaseName

What’s Deployed:  
This script deploys the Azure Power BI Embedded service, including the workspace. This script also uploads the created reports view in the web application and sets the data source to the correct Azure SQL Server.  
Parameters:

**Table 19** New-WTTPowerBI parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -AzurePowerBIName | XX00JulieandthePlantes | Name of the Power BI Workspace Collection Service |
| -azurePowerBILocation | West US, North Europe, West Europe, East US 2, North Central US, South Central US, Brazil South, Southeast Asia, Australia Southeast | Azure Power BI Embedded service location |
| -AzureSqlServerName | XX00JulieandthePlantesPrimary | Azure SQL Server primary name |
| -adminUserName | Developer | Azure SQL Server administrator name |
| -adminPassword | P@ssword1 | Azure SQL Server administrator password |
| -AzureSqlDatabaseName | Customer1 | Azure SQL Database containing content for the WTT application |
| -azureDWDatabaseName | CustomerDW | Azure SQL Data Warehouse database containing content for the WTT application. This data does not directly reflect any ticket purchases or customers from the Customer1 database |

Set-WTTEnvironmentWebConfig.ps1  
Command:  
Set-WTTEnvironmentWebConfig -WTTEnvironmentApplicationName $wTTEnvironmentApplicationName -azureResourceGroupName $azureResourceGroupName -Websitename $azureSqlServerPrimaryName -SearchName $searchName -SearchServicePrimaryManagementKey $searchServicePrimaryManagementKey -AzureSqlServerPrimaryName $azureSqlServerPrimaryName -AzureSqlServerSecondaryName $azureSqlServerSecondaryName -azureDocumentDbName $azureDocumentDbName -documentDbPrimaryKey $documentDbPrimaryKey -powerbiSigningKey $powerbiSigningKey -powerbiWorkspaceCollection $powerbiWorkspaceCollection -powerbiWorkspaceId $powerbiWorkspaceId -seatMapReportID $seatMapReportID -TenantEventType $TenantEventType

What’s Deployed:  
This script sets the app settings in the deployed web applications.  
Parameters:

**Table 20** New-WTTEnvironmentWebConfig parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -WTTEnvironmentApplicationName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -azureResourceGroupName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -Websitename | XX00JulieandthePlantesPrimary  XX00JulieandthePlantesSecondary | Name of the web app name, either primary or secondary |
| -SearchName | XX00JulieandthePlantes | Name of the Azure Search Service |
| -SearchServicePrimaryManagementKey | xxxx-1234-xxxx-1234-xxxx | Azure Search Service primary access key |
| -AzureSqlServerPrimaryName | XX00JulieandthePlantesPrimary | Primary Azure SQL Server name |
| -AzureSqlServerSecondaryName | XX00JulieandthePlantesSecondary | Secondary Azure SQL Server name |
| -azureDocumentDbName | XX00JulieandthePlantes | Azure DocumentDB name |
| -documentDbPrimaryKey | xxxx-1234-xxxx-1234-xxxx | Azure DocumentDB primary access key |
| -powerbiSigningKey | xxxx-1234-xxxx-1234-xxxx | Azure Power BI Embedded Service primary access key |
| -powerbiWorkspaceCollection | XX00JulieandthePlantes | Azure Power BI Embedded Service Workspace Collection Name |
| -powerbiWorkspaceId | xxxx-1234-xxxx-1234-xxxx | Azure Power BI Embedded Service Workspace ID |
| -seatMapReportID | xxxx-1234-xxxx-1234-xxxx | Azure Power BI Embedded Service Seat Mat report ID |

Test-WTTAzureSQLConnection.ps1  
Command:  
Test-WTTAzureSQLConnection -ServerName $ServerName -UserName $UserName -Password $Password -DatabaseName $DatabaseName -WTTEnvironmentApplicationName $WTTEnvironmentApplicationName

What’s Deployed:  
This script tests that the Azure SQL Server is available and accessible.   
Parameters:

**Table 21** Test-WTTAzureSQLConnection parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| -ServerName | XX00JulieandthePlantesPrimary | Primary Azure SQL Server name |
| -UserName | Developer | Azure SQL Server administrator name |
| -Password | P@ssword1 | Azure SQL Server administrator password |
| -DatabaseName | Customer1 | Azure SQL Database containing content for the WTT application |
| -WTTEnvironmentApplicationName | XX00JulieandthePlantes | This name is the name used throughout the PowerShell scripts to name the services that are deployed |
| -TenantEventType | Pop, rock, classical | Web application theme |