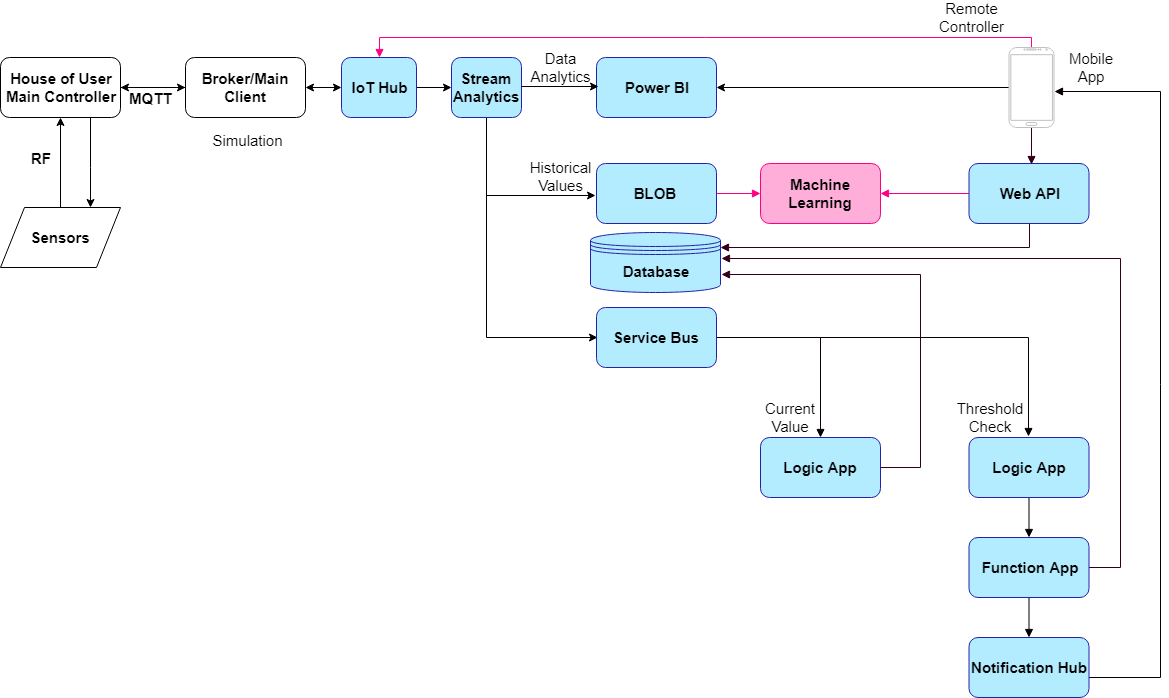
**Updated System Architecture Diagram**

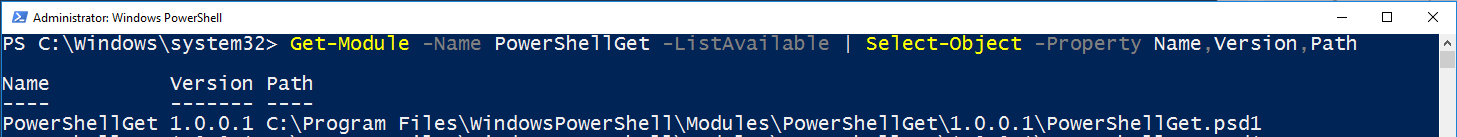


**Install and Configure Azure PowerShell**

1. Install PowerShellGet

* Run Windows PowerShell as administrator.
* Run the following command to see if you have PowerShellGet installed on your system.

Get-Module -Name PowerShellGet -ListAvailable | Select-Object -Property Name,Version,Path

* You should see something similar to the following output:

2. Install Azure PowerShell

* Run the following command:

# Install the Azure Resource Manager modules from the PowerShell Gallery

Install-Module -Name AzureRM -AllowClobber

**Prepare to Authenticate Azure Resource Manager requests**

1. Log in to you Azure subscription using the following command:

Login-AzureRmAccount

2. If you have multiple Azure subscriptions, use the following command to list the Azure subscriptions available for you to use:

Get-AzureRMSubscription

Use the following command to select subscription that you want to use to run the commands to manage your IoT hub. You can use either the subscription name or ID from the output of the previous command.

Select-AzureRMSubscription `

-SubscriptionName "{your subscription name}"4.

3. Make a note of your TenantId and SubscriptionId. You’ll need them later.

4. Create a new Azure Active Directory application using the following command, replacing the place holders:

* {Display name}: a display name for your application such as MySampleApp
* {Home page URL}: the URL of the home page of your app such as <http://mysampleapp/home>. This URL does not need to point to a real application.
* {Application identifier}: A unique identifier such as [http://mysampleapp](http://mysampleapp/). This URL does not need to point to a real application.
* {Password}: A password that you use to authenticate with your app.

$secureString = convertto-securestring "{Your password}" -asplaintext -force

New-AzureRmADApplication -DisplayName {Display name} -HomePage {Home page URL} -IdentifierUris {Application identifier} -Password $secureString

5. Make a note of the ApplicationId of the application you created. You’ll need this later.

6. Create a new service principal using the following command, replacing {MyApplicationId} with the ApplicationId from the previous step:

PowerShellCopy

New-AzureRmADServicePrincipal -ApplicationId {MyApplicationId}

7. Set up a role assignment using the following command, replacing {MyApplicationId}with your ApplicationId.

PowerShellCopy

New-AzureRmRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName {MyApplicationId}

You have now finished creating the Azure AD application that enables you to authenticate from your custom C# application. You need the following values later:

* TenantId
* SubscriptionId
* ApplicationId
* Password

**ARM (Azure Resource Manager) Templates**

Prerequisites:

* Microsoft Azure account and subscription.
* Visual Studio 2017
* A Power BI account
* A Power BI workspace - cannot use ‘My Workspace’

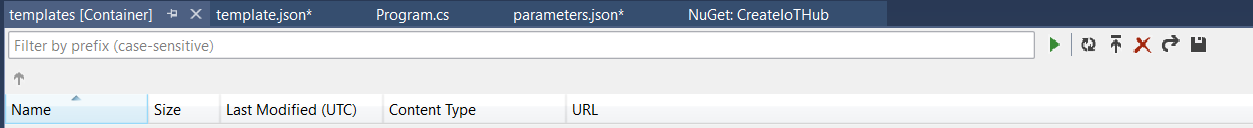
IoT Hub and Stream Analytics Setup

1. Open the ‘CreateAzureResources’ solution using Visual Studio.
2. Open Server Explorer. View -> Server Explorer.
3. Right click on Azure -> Connect to Microsoft Azure Subscription and log in using your credentials.
4. In Server Explorer, expand Azure -> right click on Storage -> select Create Storage Account. Choose your subscription, write the name of your new Storage Account (must be lowercase and between 3 and 24 characters) and select East US. Leave the default value for Replication.
5. Expand your new Storage Account. Right click on Blobs -> select Create Blob Container. Name the new blob container ‘templates’. Create another Blob Container and call it ‘sa-data’.
6. Go to parameters.json and replace the placeholder values.

*Note:* Your IoT Hub name must be globally unique. Also, IoT Hub will be publicly discoverable as a DNS endpoint, so make sure to avoid any sensitive information while naming it.

1. Go to **templates.json** and replace the placeholder values.

*Note:* azureblob\_1\_accessKey is obtained from the Azure Portal. Go to Azure Portal -> Resource groups -> click on your resource group -> select your storage account -> Access Keys and use key1.

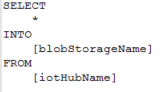
1. Go back to **Server** **Explorer**. Right click on **templates** -> select **View** **Blob** **Container**.
2. Click the upload button and upload both **template.json** and **parameters.json**. 
3. In Program.cs, replace the placeholder values. You made a note of **ApplicationId**, **SubscriptionId**, **TenantId**, and **Password** earlier. **Your Azure Storage account name** is the name of the Azure Storage account you created in Step 4. **Resource group name** is the name of the resource group you want to use when you create the IoT hub. The name can be a pre-existing or a new resource group. **Deployment name** is the name of the deployment, such as **Deployment\_01**.
4. Build and Run the application (may take several minutes).

*Note:* If successful, the console application should display details about the resources that have been provisioned. Also, check Azure Portal (portal.azure.com -> Resource groups -> your Resource group).

**To Do:**

1. Edit Stream Analytics SQL Query.

Open StreamAnalyticsQuery.txt and replace all the placeholders with your values. e.g. [blobStorageName] to the name of your “Stream Analytics to Blob Storage” connection. Copy the whole content into your **Stream** **Analytics** query.



2. Power BI token.

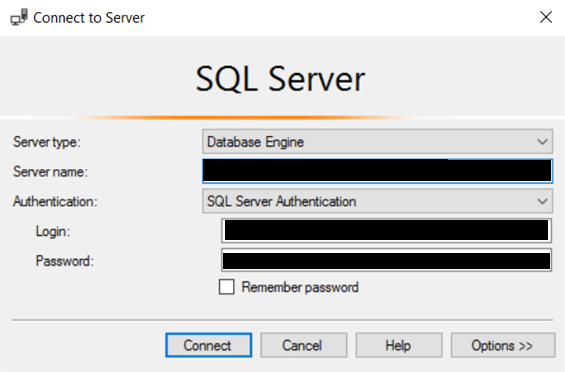
To generate a Power BI token, open the AuthToken file and open the AuthToken solution. You need to sign in to your Azure account to generate a token. The token can be used to generate reports, dashboards, etc. It’s also used for the Mobile App.

Power BI API documentation: https://powerbi.docs.apiary.io/#reference/

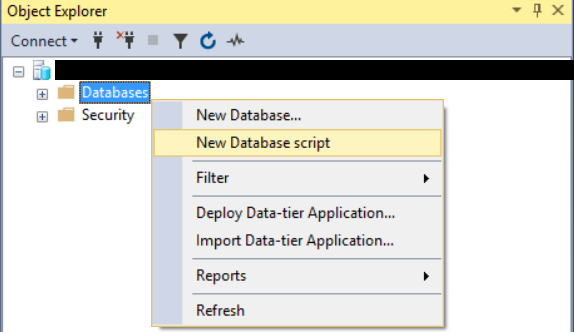
3. Create an SQL Database.

Download **Microsoft** **SQL** **Server** **Management** **Studio**. Create an SQL Server on Azure (remember your credentials).

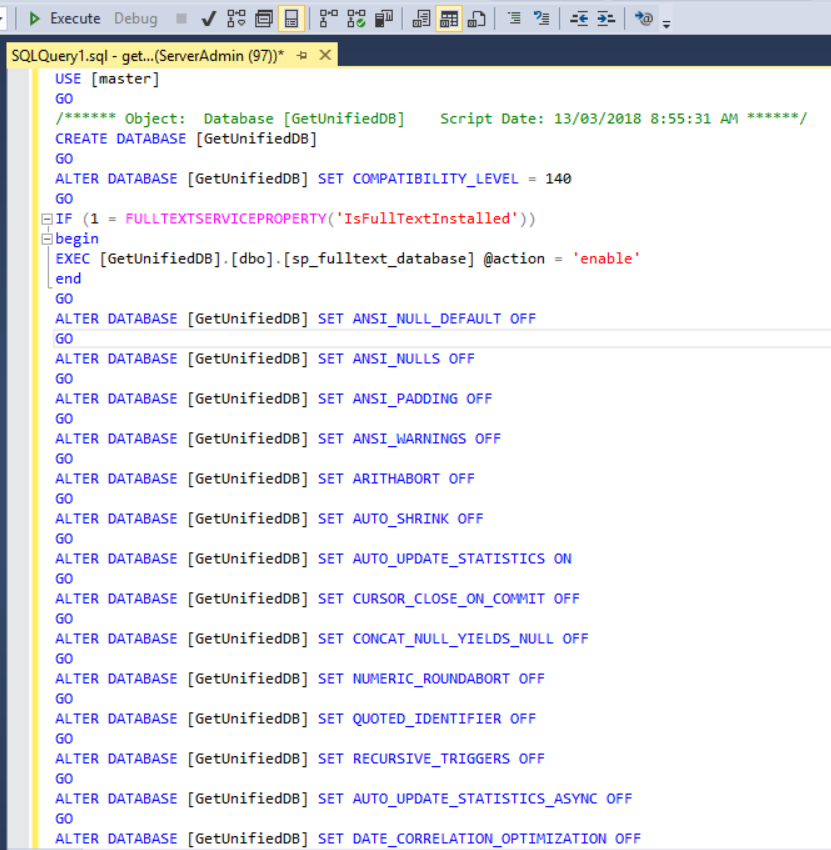
* Open **Microsoft** **SQL** **Server** **Management** **Studio** -> Connect to your Azure **SQL** **Server** by entering your credentials.



* Open **Object** **Explorer** (**View** -> **Object** **Explorer**). Expand your server -> Right click on **Databases** -> select **New** **Database** **script.** Enter

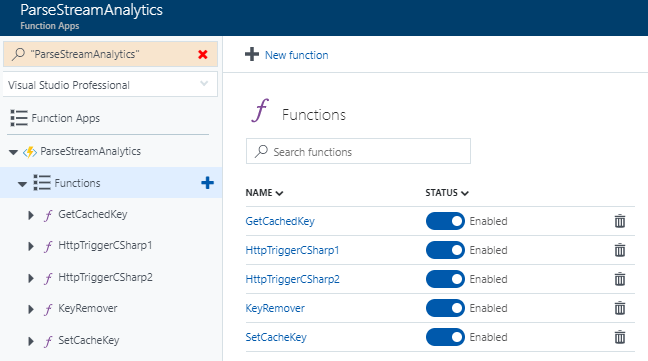


* Open **SQLDBScript**.**sql**. Copy everything and paste it into the new SQLQuery.sql window. Click **Execute**.

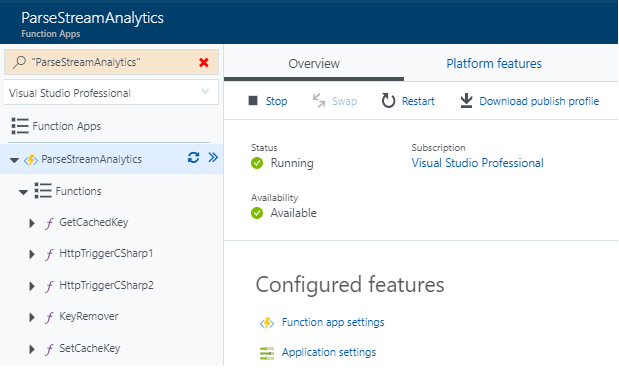


4. Create an Azure Function App.

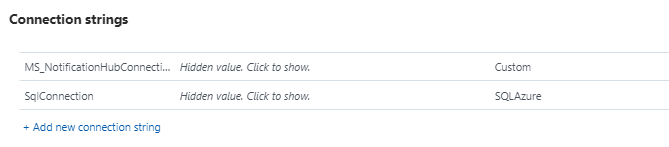
* Go to Azure Portal. Create a Function App and call it **ParseStreamAnalytics**.
* Open your new Function App and create 5 new functions that use an **HTTP** trigger:
  + GetCachedKey
  + HttpTriggerCSharp1
  + HttpTriggerCSharp2
  + KeyRemover
  + SetCacheKey
* The code for each function can be found in the ParseStreamAnalytics folder. Just copy the code into the function editor. You should end up with the following functions:



* Go to **ParseStreamAnalytics** **Overview** -> **Application** **settings**



* Add the following connection information (use the same connection names).



5. Create logic apps.

* Go to **Azure** **Portal** -> **Create a resource** -> Search for **Logic** **App** -> Select the first one -> **Create**.
* Call it ‘**notifications**’ and save it in the same resource group and location.
* Go to your **notifications** logic app -> **Logic App Code View**.
* Open notifications\_logicapp.txt -> copy the content -> paste into the logic app editor. Replace all the placeholder values (should be in { }).
* Click **Run**.
* Create another logic app and call it ‘**updateDB**’. Copy the content from updated\_logicapp.txt into the logic app editor. Replace all the placeholder values -> click **Run**.

6. Push Notifications.

Please follow the ‘Create a project that supports Firebase Cloud Messaging’ and ‘Configure a new notification hub’ parts of the tutorial below.

https://docs.microsoft.com/en-us/azure/notification-hubs/notification-hubs-android-push-notification-google-fcm-get-started

7. App Service

Open the REST\_API visual studio solution. Right click on the project -> Publish -> set up your credentials -> click Publish.

8. Finally, you should end up with the following resources:

