

# Create Windows 8.1 Store App using Azure Mobile Services

In this lab, you will learn about Microsoft Azure Mobile Services and using it to store data in an Azure SQL Table.

In order to complete this lab you will need the Visual Studio 2013, and an Active Windows Azure account.

This set of hands-on-labs use a Windows Store application to store a To-Do list and use Windows Azure Mobile Services to store the data remotely in a SQL database.

In this set of hands-on labs, you will walk through creating a Windows Azure Mobile service, build a database table to store your application's data, and then connect the Windows Store application to the Azure Mobile Service and save data to the database.

This hands-on lab includes the following exercises:

Prepare a Windows Store application that stores data locally

Create new Azure Mobile Service

Leverage Azure Mobile Services for Data

Update Store App to use Azure Mobile Service

**Expected duration: 60 minutes.**

## Exercise 1: Prepare a Windows Store application that stores data in-memory

In this exercise, you are going to use an existing Windows Store Project.

1. Launch **Visual Studio 2013** from the taskbar and then open the **AzMoSvc** solution from **C:\Projects\AzMoSvc\C#** (File | Open | Project/Solution...).

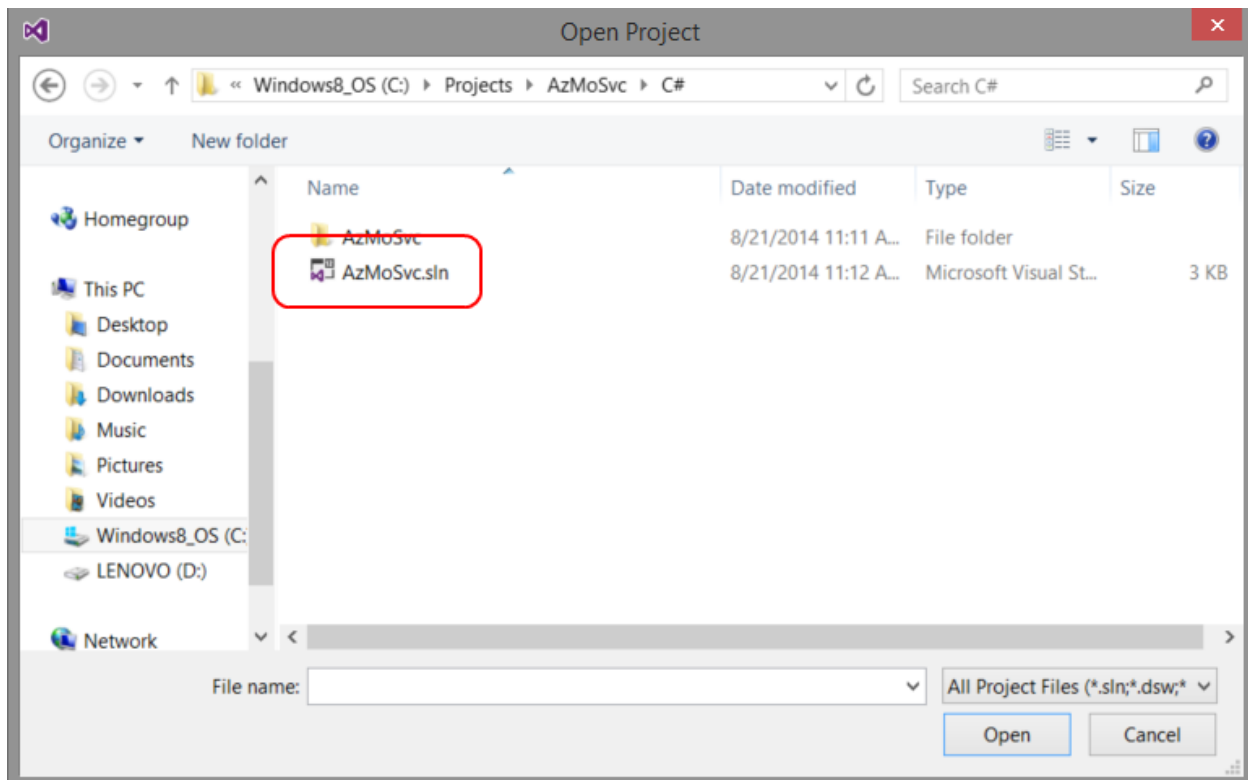


Figure 1: Loading sample solution

2. This Windows Store application currently stores the ToDo information in memory. By default, when you create a new Windows Store project, it is set to run on the Local Machine. For this lab, we will run the application through the Simulator. To change from the Local Machine to the Simulator, click on the **down** arrow of **Local Machine** and select **Simulator**.

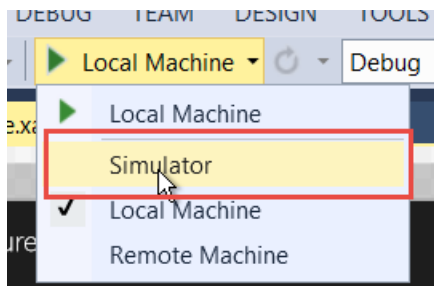


Figure 2: Select the Simulator to debug your Windows application on

3. Once you have selected the Simulator, click on the **Simulator** debug button to run the application.

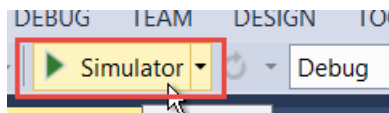


Figure 3: Debug the application through the emulator

4. The Simulator will launch and the application will load for you to test it out.

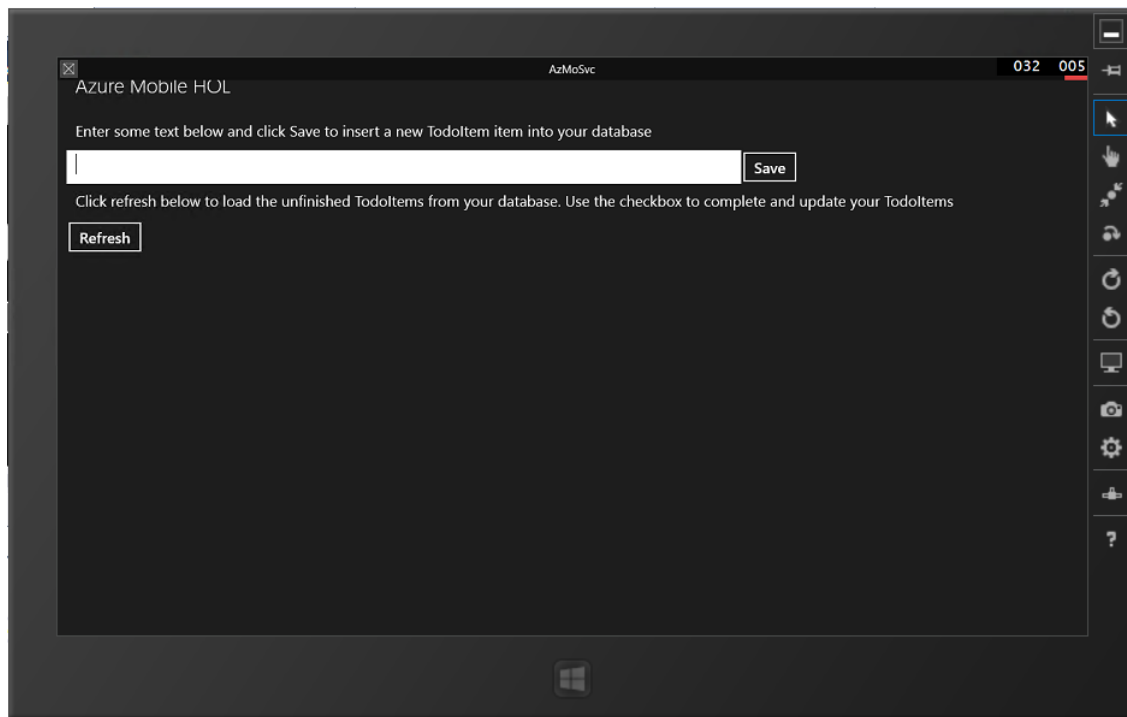


Figure 4: Sample application running in the Simulator

5. Place the cursor in the input field, enter a value, and click on **Save**. You will see the information added below the **Refresh** button. Go ahead and enter in a few more “To Do” items to build up a list.

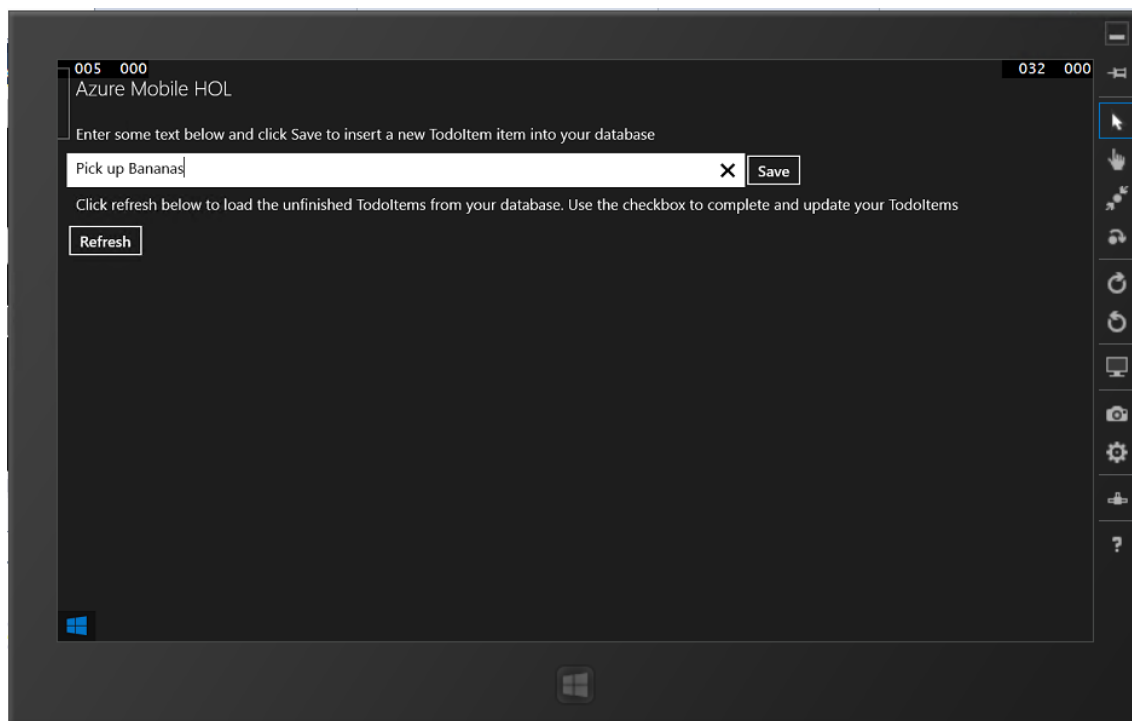


Figure 5: Enter value in the text box and press Save

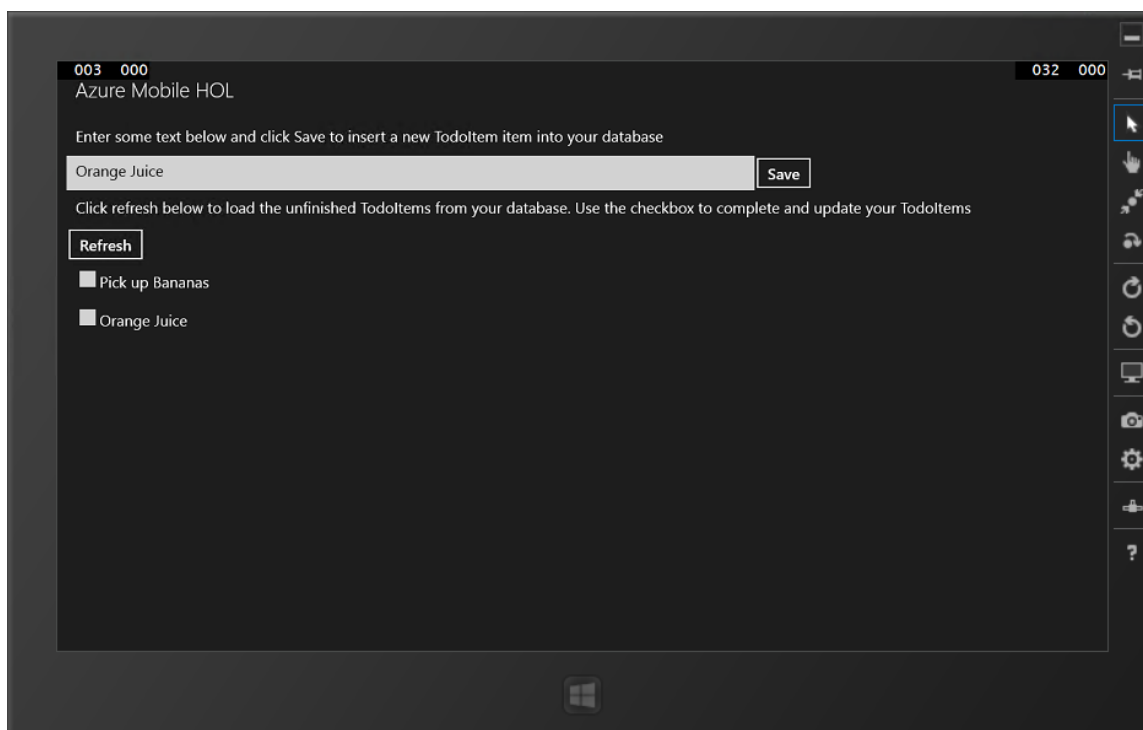


Figure 6: List of To Do items that have been entered showing under the Refresh button

6. When a To Do item is completed, click on the check box next to the item. This will remove it from the list.
7. To stop debugging the application, go back to Visual Studio and click on the **Stop Debugging** button.

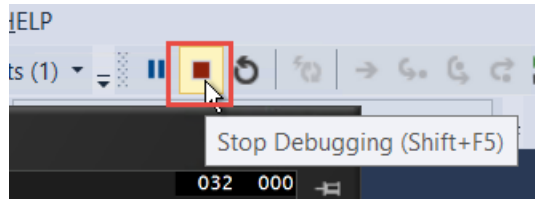


Figure 7: Stopping the Debug session

## Exercise 2: Create new Azure Mobile Service

In this exercise, you will create an Azure Mobile Service that the Windows Store application will use. You will need your Azure account for this as well.

1. Go to the **Microsoft Azure Management Console** by opening up **Internet Explorer** from the Taskbar and go to <https://manage.windowsazure.com>
2. You will need to enter your **Microsoft account credentials** to access your Azure Subscription and click on **Sign In**.

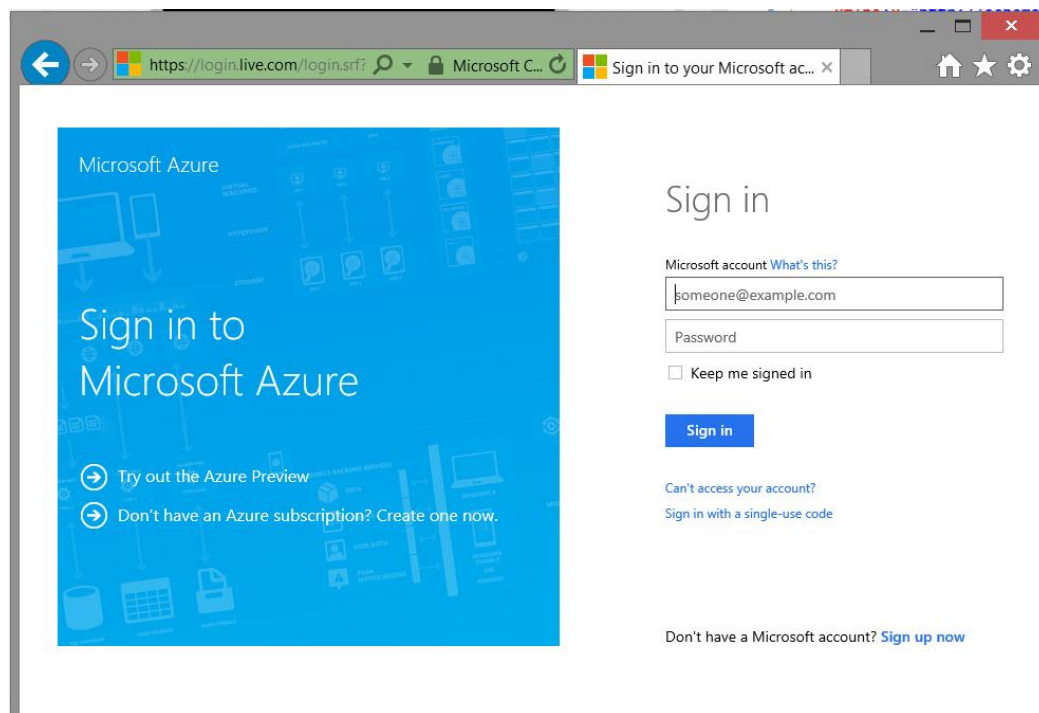


Figure 8: Microsoft Azure Sign In Page

3. Once logged in, select **Mobile Services** in the left-hand Azure Services menu.

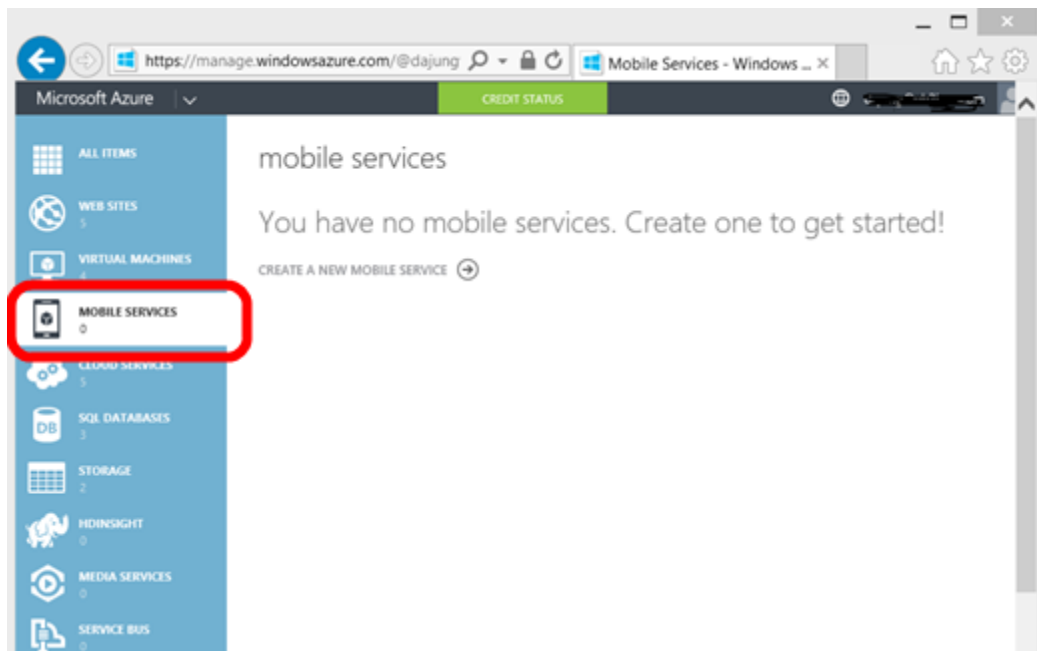


Figure 9: Microsoft Azure Mobile Services page

4. Select **New** in the lower left-hand corner to create the new Mobile Service.

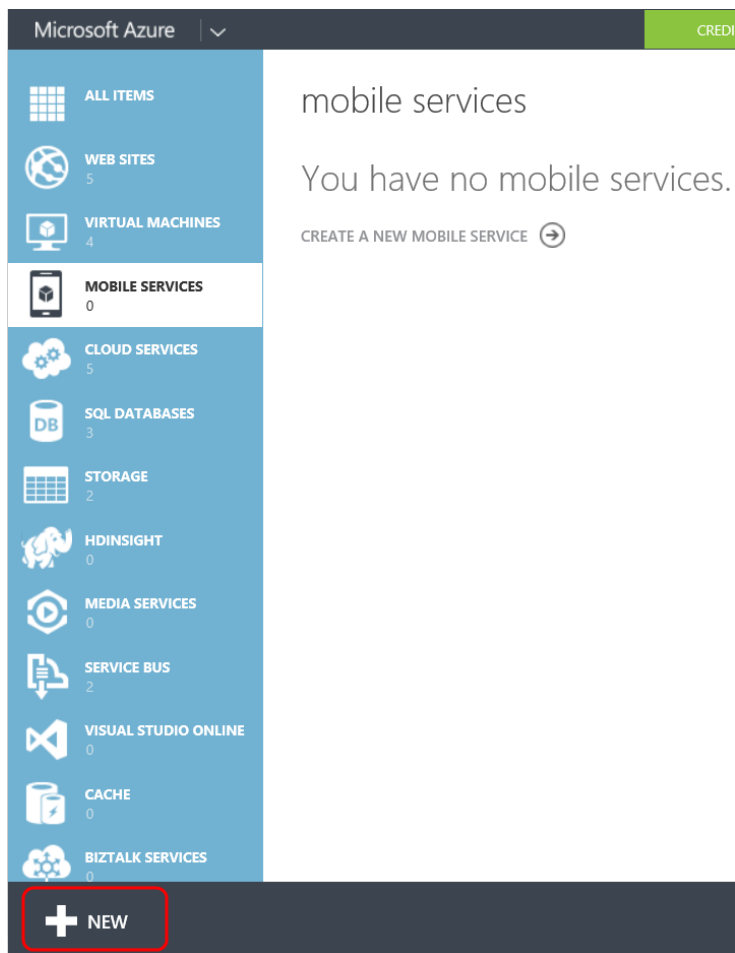


Figure 10: Click **New** to create a new Mobile Service

5. This will bring up the **wizard** and click **Create** to create the Mobile Service.

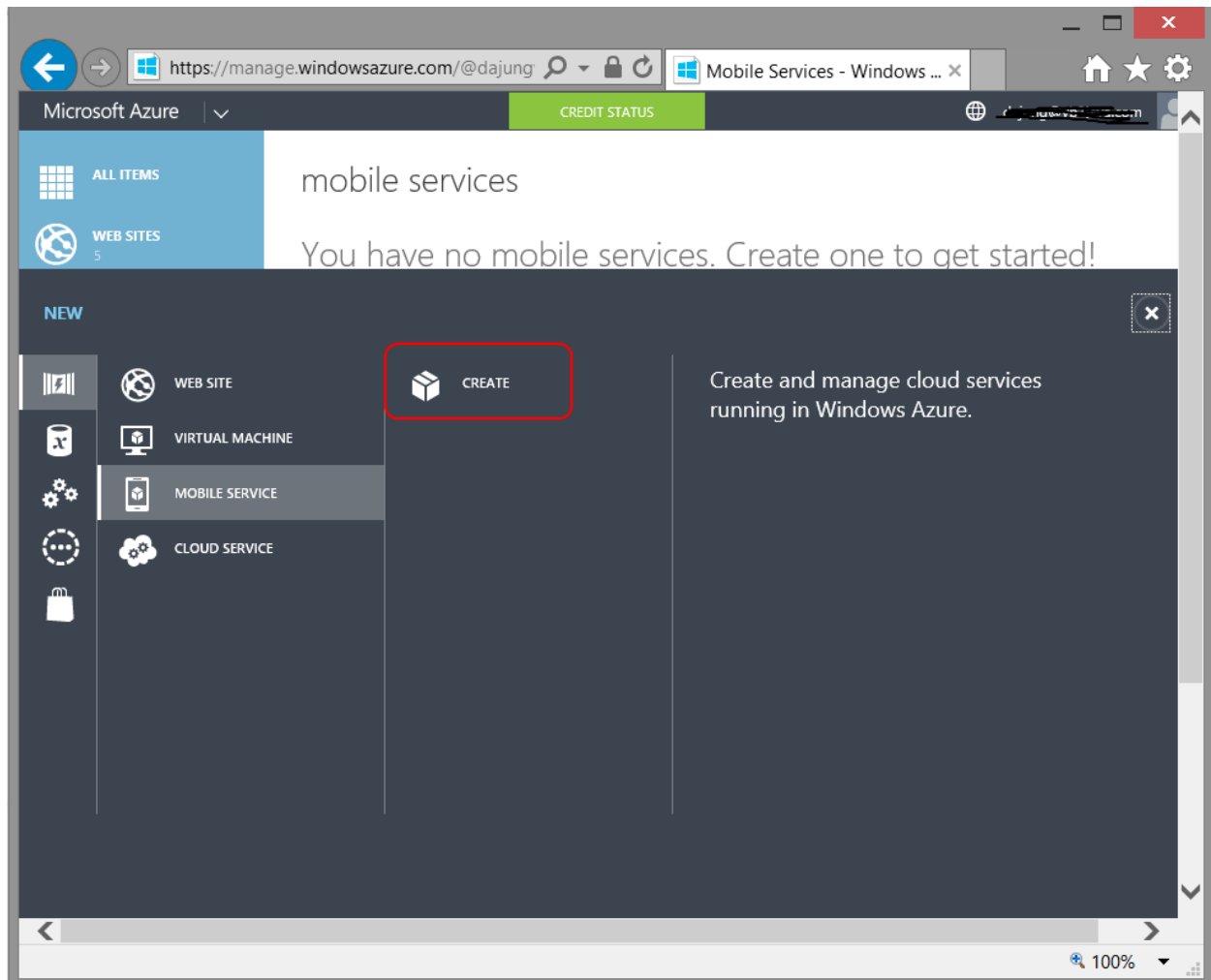


Figure 11: Create an Azure Mobile Service

6. To create a new Mobile Service instance, you need to enter in a unique Mobile Service URL. For this exercise, use **AzureMobileHOL**. Part of the Mobile Service is a SQL Server Database. Select **Create a new SQL database instance** in the Database drop-down list. Select the region you would like the SQL Server instance to be in. For this exercise, use **West Region**. Select **JavaScript** for the backend. Then **click** the **Right Arrow** button on the lower right hand corner of the dialog.



# Create a Mobile Service

URL  
AzureMobileHOL  
.azure-mobile.net

DATABASE  
Create a new SQL database instance

REGION  
West US

BACKEND  
JavaScript

☐ ADVANCED PUSH NOTIFICATION SETTINGS ?

→ 2

Figure 12: Creating a new Mobile Service

- On the **Specify database settings** screen, select **New SQL database server** in the **Server** dropdown. This will create a new Azure SQL Database instance for you. For the **Server Login Name**, use **AzureMobileAdmin**. For the **Server Login Password**, use **P2ssw0rd**. You will need to enter it twice to make sure you entered it in correctly. Once you have done that, **press the Check mark** in the lower right-hand corner to accept the information. Azure will provision the database for you.

NEW MOBILE SERVICE

Specify database settings

NAME

AzureMobileHOL\_db

SERVER

New SQL database server

SERVER LOGIN NAME

AzureMobileAdmin

SERVER LOGIN PASSWORD

CONFIRM PASSWORD

REGION

West US

☐ CONFIGURE ADVANCED DATABASE SETTINGS

1

←

✓

Figure 13: Specify database settings

- This will take the system a few minutes to provision. It is creating an Azure web site that will host our REST end-points and creates a SQL Server instance.

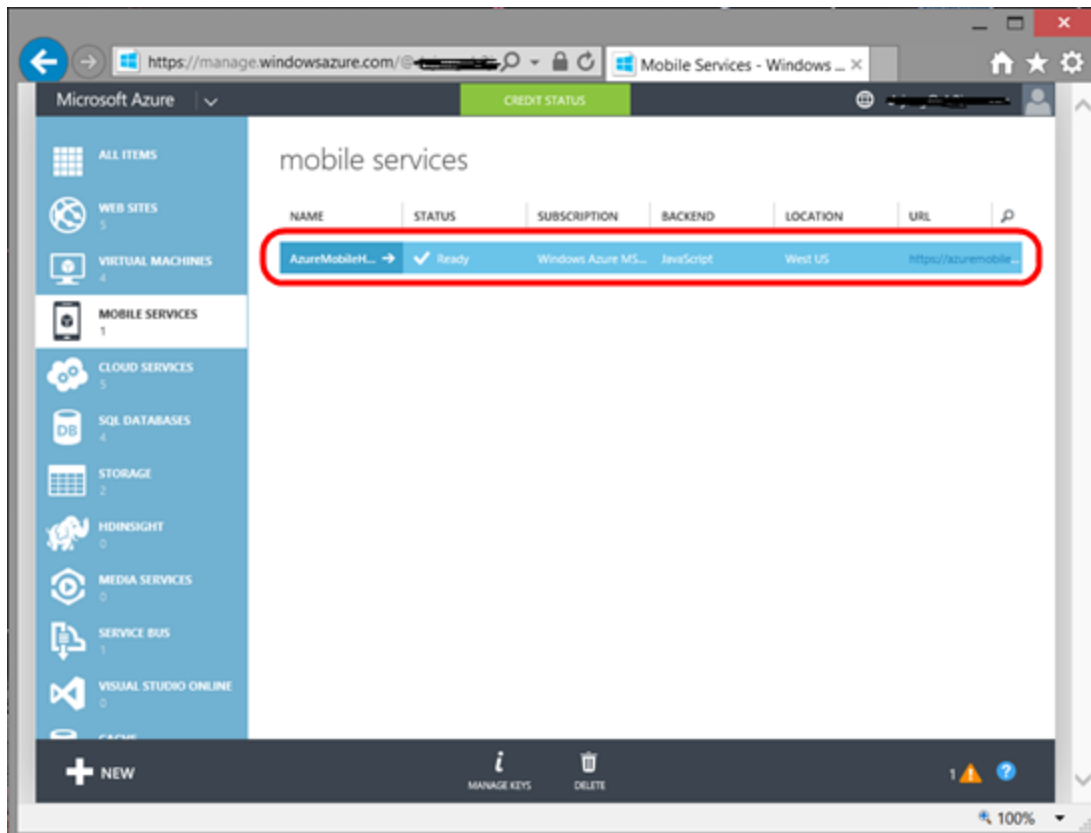


Figure 14: Newly created Mobile Service

- Click on the **right arrow** next to the name of your Mobile Service to bring up the **Dashboard/Quick Start** page.

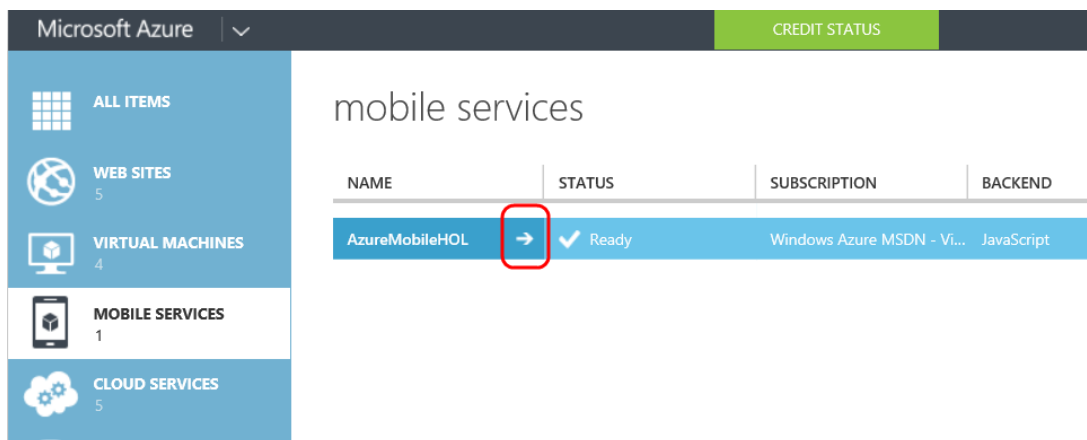


Figure 15: Select the AzureMobileHOL Mobile Service to bring up its Dashboard.

- On the **Dashboard/Quick start** page, under **Choose a Platform**, select the **Windows** platform. This will provide you with information on how to connect this Azure Mobile Service to our Windows Store application.

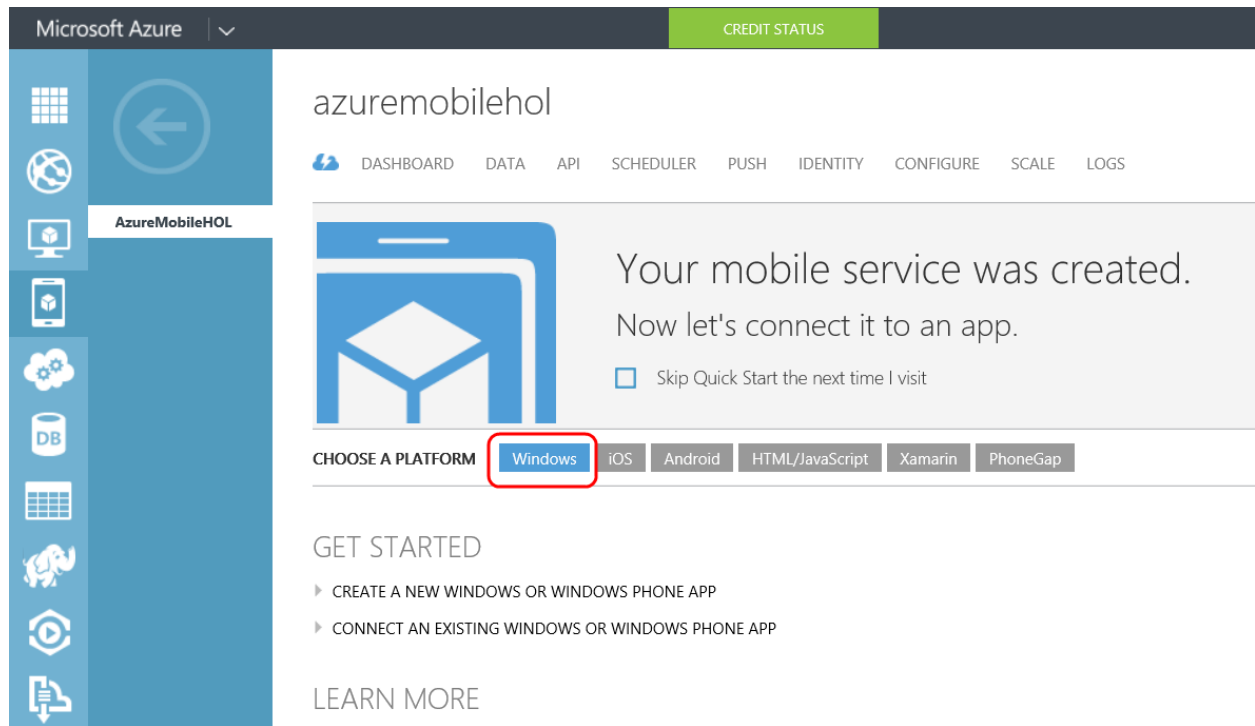


Figure 16: Select the Windows Platform

11. This lab is using a Windows Store application; therefore, select the **Connect an Existing Windows or Windows Phone App** link. The application is written in C#, so select **C#** in the Language drop-down. This will display **AppURL** and **AppKey** you will need to connect your application to this Mobile Service.

The screenshot shows the Azure Mobile Services portal for a service named 'azuremobilehol'. The left sidebar contains various Azure service icons. The main content area has a navigation bar with links: DASHBOARD, DATA, API, SCHEDULER, PUSH, IDENTITY, CONFIGURE, SCALE, and LOGS. A message states: 'Your mobile service was created. Now let's connect it to an app.' Below this is a 'CHOOSE A PLATFORM' section with buttons for Windows, iOS, Android, HTML/JavaScript, Xamarin, and PhoneGap. The 'GET STARTED' section lists two options: 'CREATE A NEW WINDOWS OR WINDOWS PHONE APP' and 'CONNECT AN EXISTING WINDOWS OR WINDOWS PHONE APP'. The second option is selected, and the steps are: 1. Connect your app, 2. Store data in your mobile service. Step 1 includes a 'LANGUAGE' dropdown set to 'C#' and a code snippet for connecting the app to the mobile service.

Microsoft Azure CREDIT STATUS dajung@vb2jav

azuremobilehol

DASHBOARD DATA API SCHEDULER PUSH IDENTITY CONFIGURE SCALE LOGS

Your mobile service was created.  
Now let's connect it to an app.

☐ Skip Quick Start the next time I visit

CHOOSE A PLATFORM Windows iOS Android HTML/JavaScript Xamarin PhoneGap

GET STARTED

► CREATE A NEW WINDOWS OR WINDOWS PHONE APP

◄ CONNECT AN EXISTING WINDOWS OR WINDOWS PHONE APP

Follow these steps to connect an existing application to your mobile service.

1 Connect your app

LANGUAGE: C#

Right-click your client project, select **Manage NuGet Packages**, search for the **WindowsAzure.MobileServices** package and add a reference to it. "using Microsoft.WindowsAzure.MobileServices;" then copy and paste the following code into your **App.xaml.cs** file:

```
public static MobileServiceClient MobileService = new MobileServiceClient(
    "https://azuremobilehol.azure-mobile.net/",
    "ofhQIadjpPoDy1RVoDsAAcPUUngLjY24"
);
```

2 Store data in your mobile service

To store data in your mobile service, you need a table. Click the button below to create an Item table for your application. You can add and remove

Figure 17: Information to connect your Windows Store App to your Mobile Service

**Note:** The parameters generated for the Mobile Service are called the **AppURL** and **AppKey** and they are unique for each Azure Mobile Service you create. We will come back to this page later in the Lab.

## Exercise 3: Leverage Azure Mobile Services for Data

Now that we have created the Azure Mobile Service, this exercise will create a database table that we will use to store the mobile application data.

1. To create a table that the Windows Store application will use, scroll back up to the top of the page and click on **Data**. You will see that there are not tables defined yet. Now click on the **Create** button at the bottom of the screen.

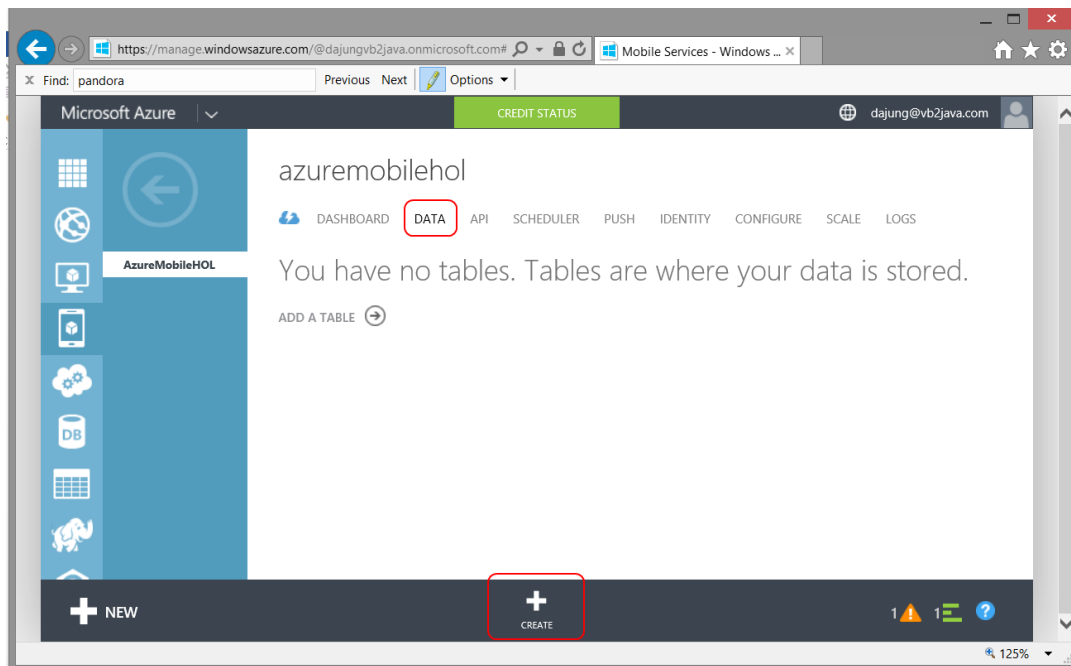


Figure 18: Create a new Table

2. In the **Create New Table** dialog, use **ToDoItem** as the table name. As you can see, you set different permissions for different Create, Read, Update, And Delete (CRUD) rules. For this lab, leave them at the default and the OK check button.

A screenshot of the 'Create New Table' dialog in the Azure Mobile Services portal. The title is 'MOBILE SERVICES: DATA' and 'Create New Table'. The 'TABLE NAME' field is highlighted with a red box and contains the text 'ToDoItem'. Below this, there is a message: 'Tables have changed; your existing client code may need to be modified. [Learn more](#)'. A section titled 'You can set a permission level against each operation for your table.' contains four dropdown menus for 'INSERT PERMISSION', 'UPDATE PERMISSION', 'DELETE PERMISSION', and 'READ PERMISSION', all set to 'Anybody with the Application Key'. A checkmark button is at the bottom right.

Figure 19: Create New Table dialog

3. With the table created, we are ready to connect our Windows Store application to our Azure Mobile Service.

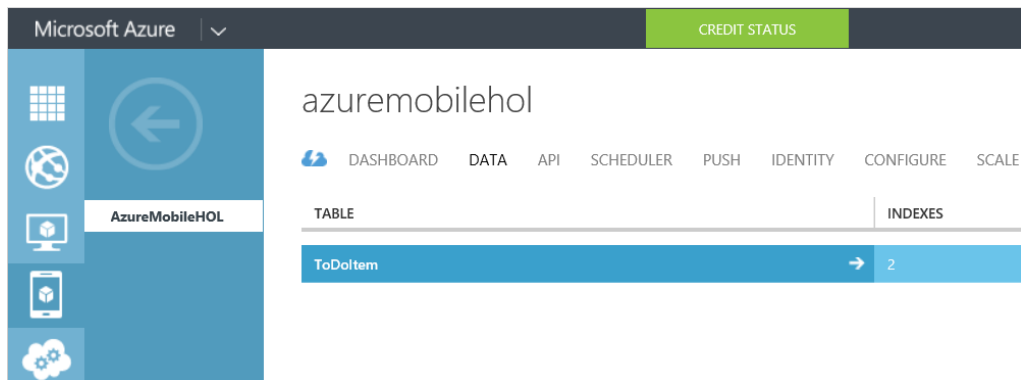


Figure 20: Newly created ToDoltem table

**Note: Do not close Internet Explorer.** We will come back to the Azure Portal in Exercise 4.

## Exercise 4: Update Store App to use Azure Mobile Service

In this exercise, you will add the components needed for the Windows Store application used in Exercise 1 to consume Azure Mobile Services to store the data.

1. If you closed Visual Studio at the end of Exercise 1, launch Visual Studio and open the **AzureMobileHOL** project.
2. Now let's take a look at how we can take advantage of **Azure Mobile Services** to store data. The Windows Azure Mobile Service code base is installed via NuGet. Connect the application to the Mobile Service by selecting **Manage NuGet Packages** in Solution Explorer through the context menu, search for the **WindowsAzure.MobileServices** package, and install the reference.

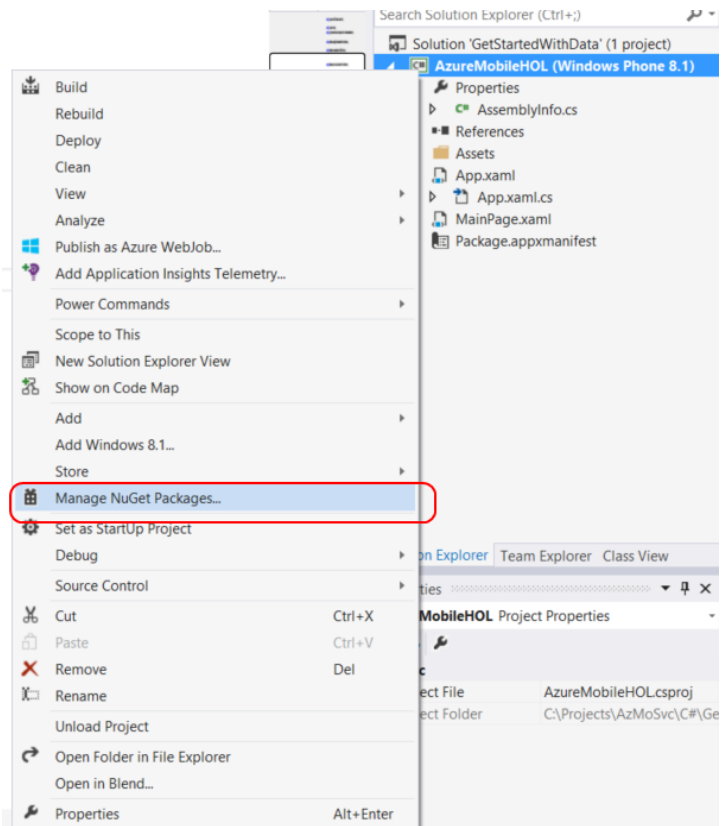


Figure 21: Manage NuGet Packages in the Context Menu



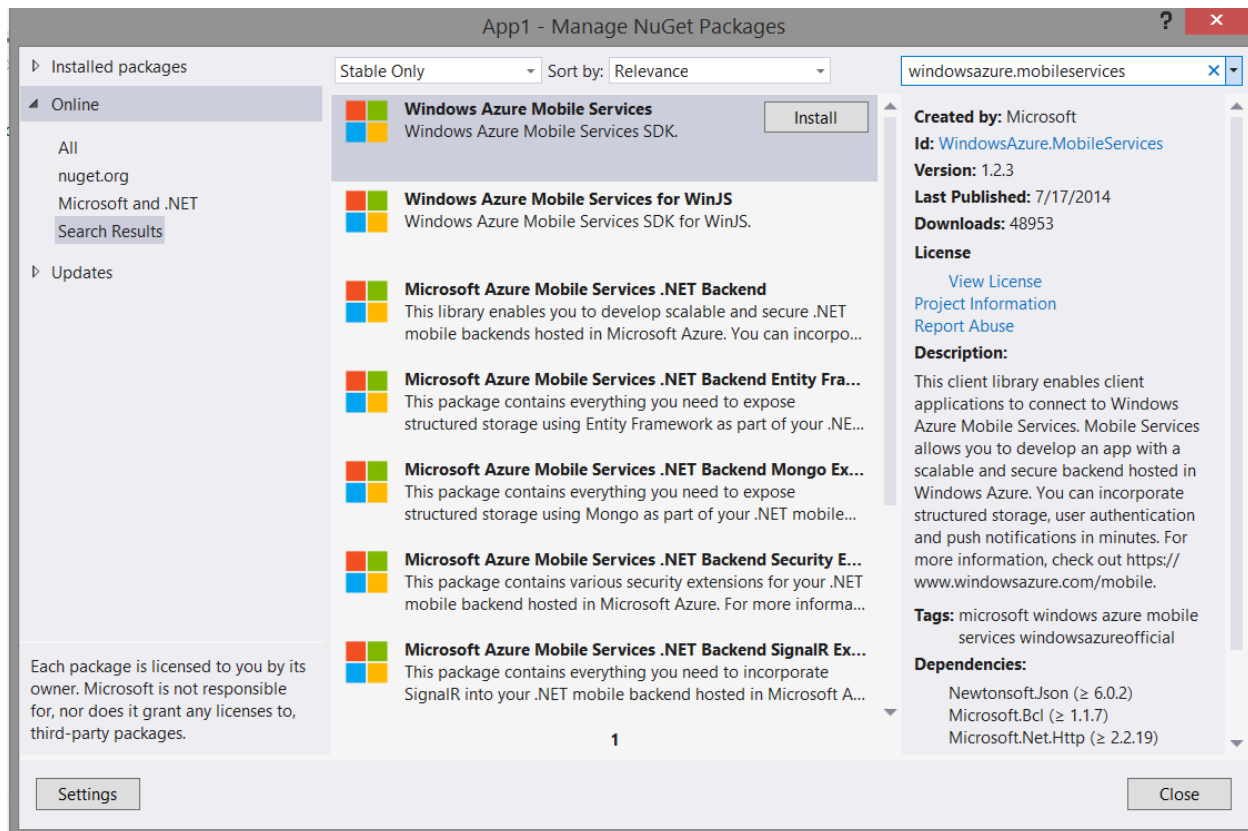


Figure 22: Searching for Windows Azure Mobile Services in the Manage NuGet Packages dialog

3. You may be prompted with a **License Acceptance** dialog for a number of libraries that are going to be added to your project. If you do, select **I Accept**.

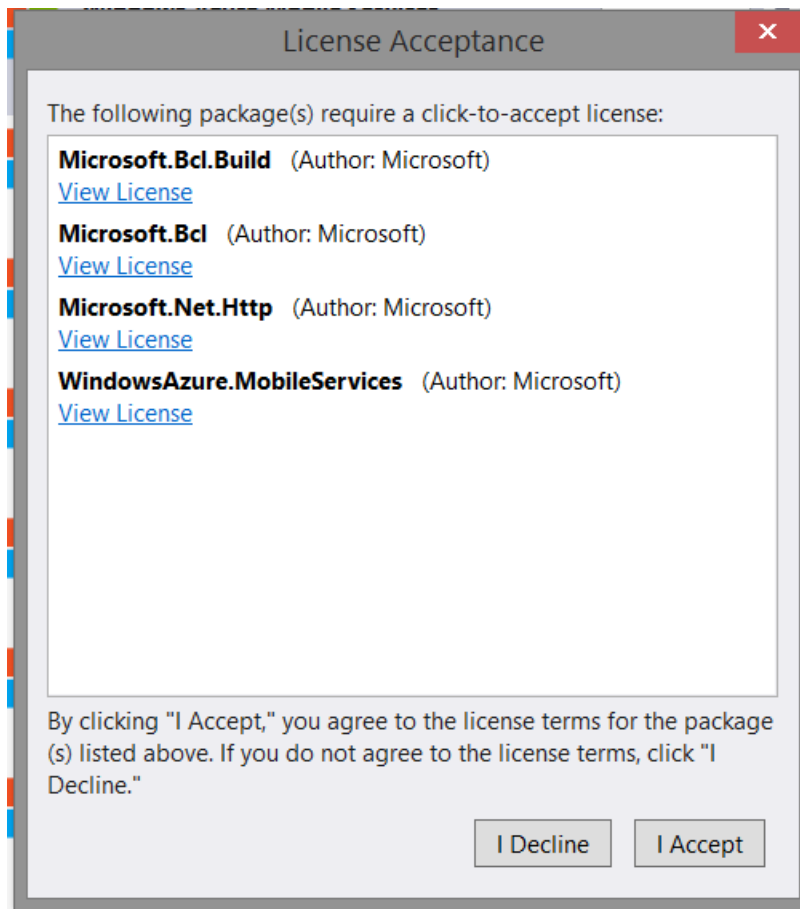
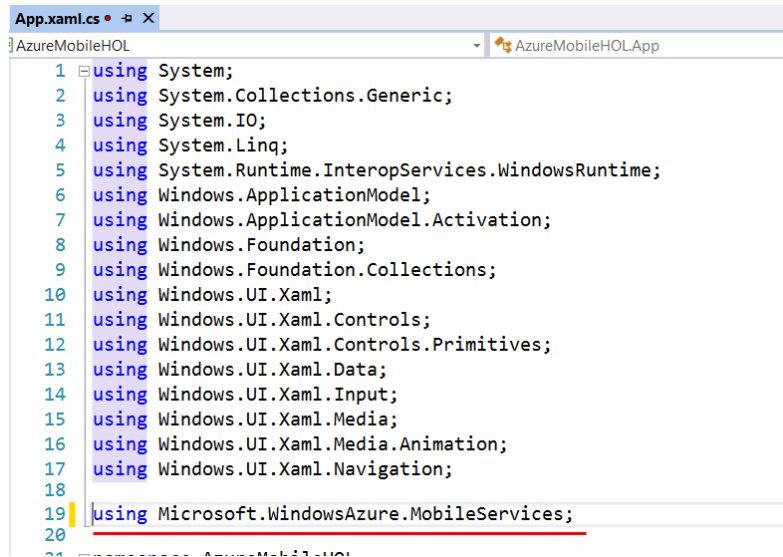


Figure 23: License Acceptance dialog

4. After the **Windows Azure Mobile Services** is installed in your project, click **Close** to close the **Manage NuGet Packages** dialog.
5. From the **Solution Explorer**, expand **App.xaml** so you can see the **App.xaml.cs** file. Open the file and add the following **using** statement reference. This lets our application know we are now going to use **Azure Mobile Services**.



```
1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using System.Linq;
5 using System.Runtime.InteropServices.WindowsRuntime;
6 using Windows.ApplicationModel;
7 using Windows.ApplicationModel.Activation;
8 using Windows.Foundation;
9 using Windows.Foundation.Collections;
10 using Windows.UI.Xaml;
11 using Windows.UI.Xaml.Controls;
12 using Windows.UI.Xaml.Controls.Primitives;
13 using Windows.UI.Xaml.Data;
14 using Windows.UI.Xaml.Input;
15 using Windows.UI.Xaml.Media;
16 using Windows.UI.Xaml.Media.Animation;
17 using Windows.UI.Xaml.Navigation;
18
19 using Microsoft.WindowsAzure.MobileServices;
20
```

Figure 24: Added Azure Mobile Service reference to our code

6. It's time to add the **AppURL** and **AppKey** that were generated for us earlier in the Lab (Exercise 2 Step 11). Assuming you did not close Internet Explorer at the end of Exercise 3, switch back to Internet Explorer. Click on the **Cloud with a Lightning Bolt icon** to get back to the **Quick Start** page.

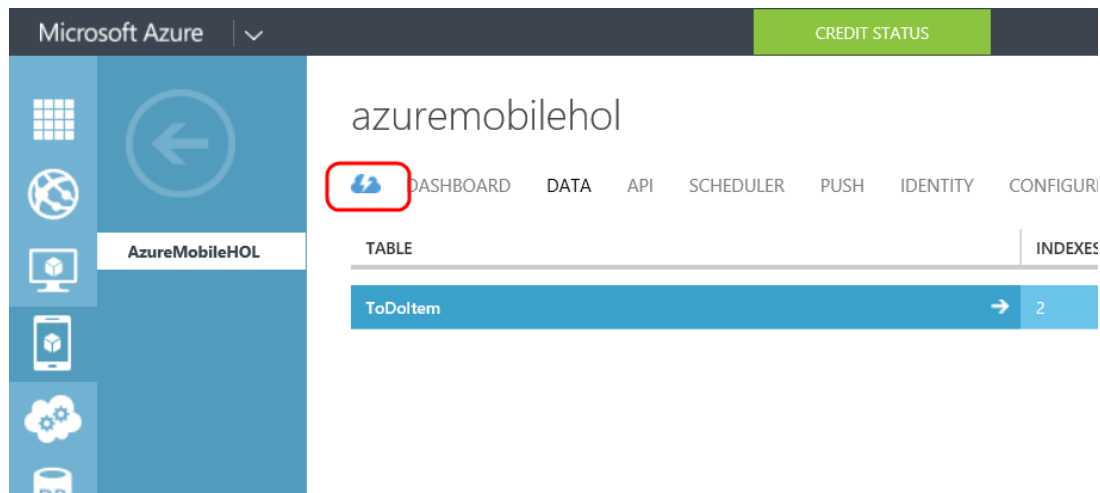


Figure 25: Link to take you back to the Quick Start page of our Mobile Service

7. To add the **AppURL** and **AppKey** to our project, let us copy and paste it from the Mobile Service's Quick Start page. On the **Quick Start** page, click on **Windows** for the **Choose a Platform**. Select **C#** for the **Language**. Click on the **Copy** icon in the upper right hand corner of the code snippet.

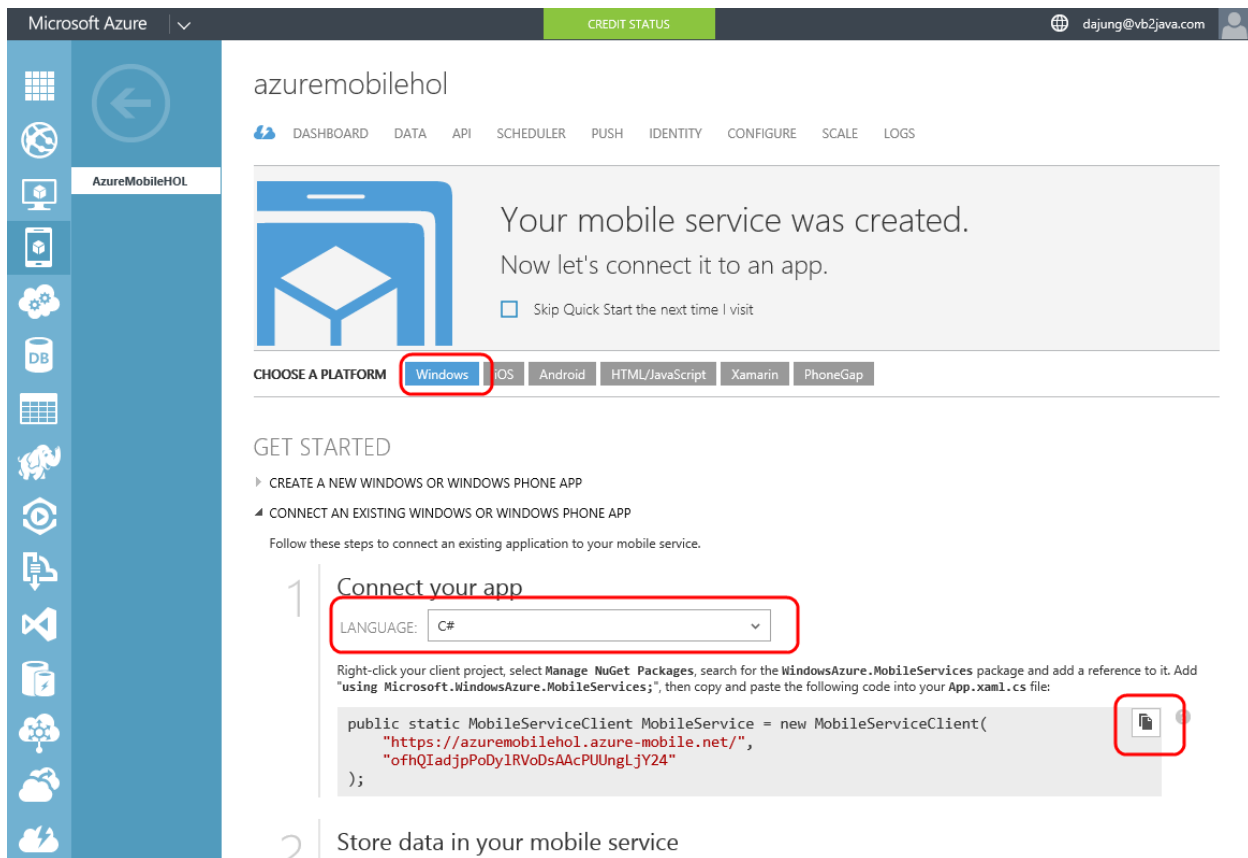


Figure 26: Code snippet that contains the AppURL and AppKey for the Mobile Service

- Paste the **AppURL** and **AppKey** code snippet into the **App.xaml.cs** file inside the **App** partial class.

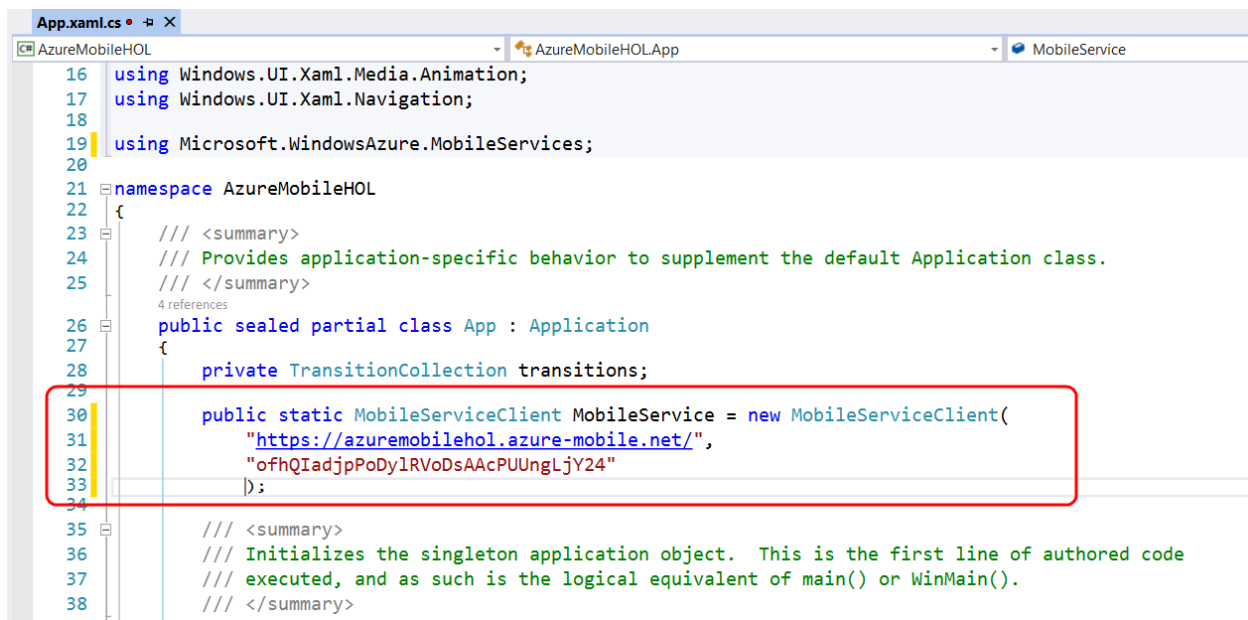


Figure 27: Added MobileService AppURL and AppKey to your code

9. Save the changes you have made by pressing the **Save** icon in the Toolbar.

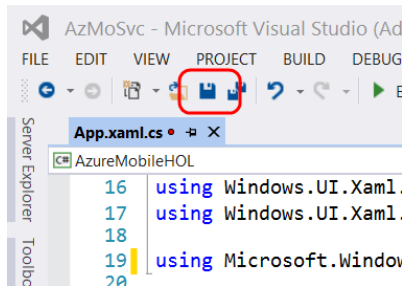


Figure 28: Save your App.xaml.cs changes

10. In the Solution Explorer, expand MainPage.xaml so you can the **MainPage.xaml.cs** file and **double-click** on it to open it. Add the references to both **Microsoft.WindowsAzure.MobileServices** and **Newtonsoft.Json** to **MainPage.xaml.cs**.

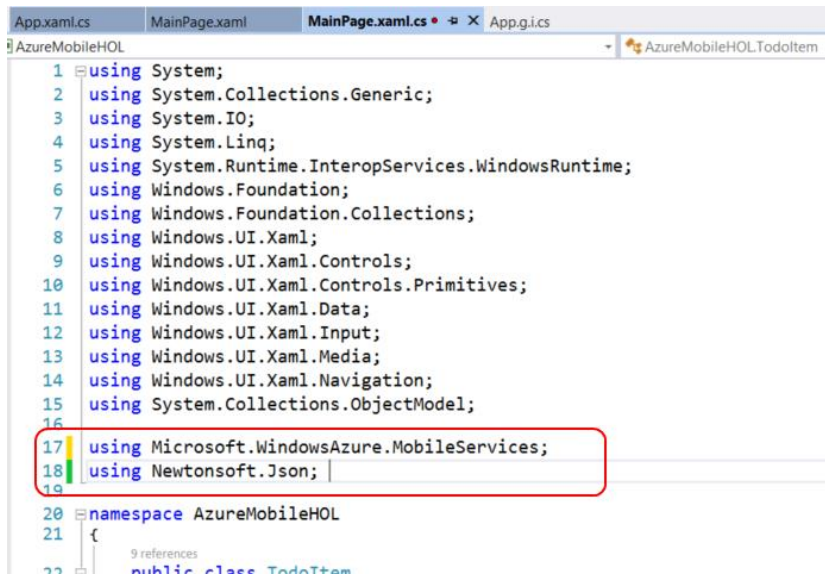


Figure 29: Adding WindowsAzure.MobileServices and Newtonsoft.Json references to project

11. Add the following **Json** serialization attributes to the **ToDoItem** class object in the **MainPage.xaml.cs** file.

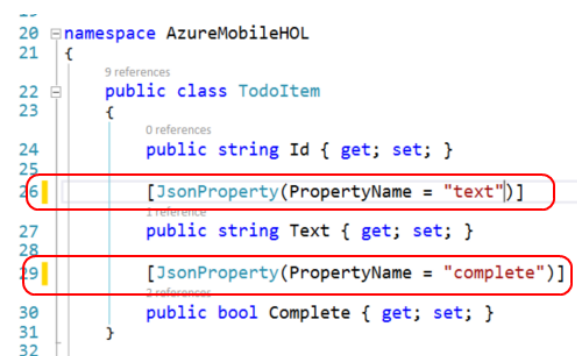


Figure 30: Added two Json Properties that will be used to store data values

12. In the **MainPage** method of **MainPage.xaml.cs**, remove the private **ObservableCollection** because it stored the collection in memory. Add the following two **MobileService** references to establish the connection to the Azure Mobile Service Table we created in the previous exercise.

```
36 6 references
37 public sealed partial class MainPage : Page
38 {
39     //private ObservableCollection<TodoItem> items = new ObservableCollection<TodoItem>();
40     private MobileServiceCollection<TodoItem, TodoItem> items;
41     private IMobileServiceTable<TodoItem> todoTable = App.MobileService.GetTable<TodoItem>();
42 }
```

Figure 31: Added code to access the **MobileService** rather than local memory

13. In the **InsertToDoItem** method of the **MainPage.xaml.cs** file, change the method to be **Async**, add the **todoTable** statement and remove the **todoItem.ID** statement because the Mobile Service will generate the ID for us.

```
1 reference
private async void InsertToDoItem(TodoItem todoItem)
{
    //todoItem.Id = Guid.NewGuid().ToString();
    /// This code inserts a new TodoItem into the database. When the operation completes
    /// and Mobile Services has assigned an Id, the item is added to the CollectionView
    await todoTable.InsertAsync(todoItem);
    items.Add(todoItem);
}
```

Figure 32: Added code to make method Async and insert data into Azure Mobile Service data table

14. In the **RefreshToDoItem** method of the **MainPage.xaml.cs** file, make the method **Async** and add the following two statements. The first one defines a simple query to retrieve all the items in the table. The other filters out all the completed items.

```
2 references
private async void RefreshToDoItems()
{
    items = await todoTable.ToCollectionAsync();
    items = await todoTable
        .Where(todoItem => todoItem.Complete == false)
        .ToCollectionAsync();
    ListItems.ItemsSource = items;
}
```

Figure 32: Updating method to be Async and refresh data from Azure Mobile Service data table

15. In the **UpdateCheckedToDoItem** method of the **MainPage.xaml.cs** file, make the method **Async** and add the following statement, which will update the database when a ToDo is marked complete and when the Mobile Service responds, the item will be removed from the list.

```
1 reference
75 private async void UpdateCheckedToDoItem(TodoItem item)
76 {
77     await todoTable.UpdateAsync(item);
78     items.Remove(item);
79 }
80
```

Figure 33: Updating method to be Async and access the Azure Mobile Service data table

16. Save the changes you have made to the **MainPage.xaml.cs** file by pressing the **Save** icon in the Toolbar

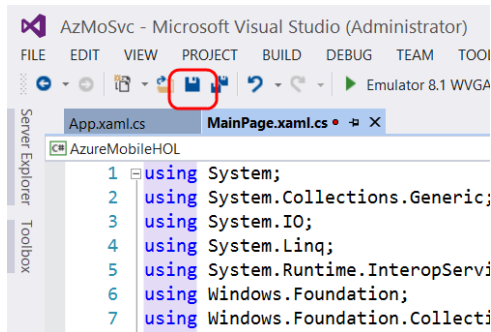


Figure 34: Save your MainPage.xaml.cs changes

17. **Right-click** on the **AzureMobileHOL** project in Solution Explorer and select **Build** to build the solution. Correct any errors you might encounter.

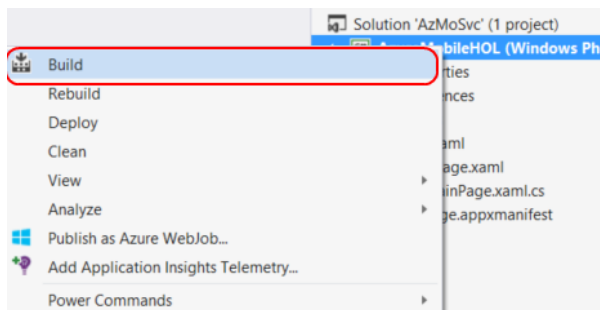


Figure 35: Building the application

18. Click on the **Simulator** button to see the application run through the simulator.

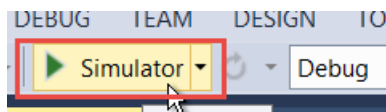


Figure 36: Click on Simulator to debug application

19. The application will launch in the Simulator and you will be able to enter data into text box like you did in Exercise 1. This time the data will be sent to the Azure Mobile database.

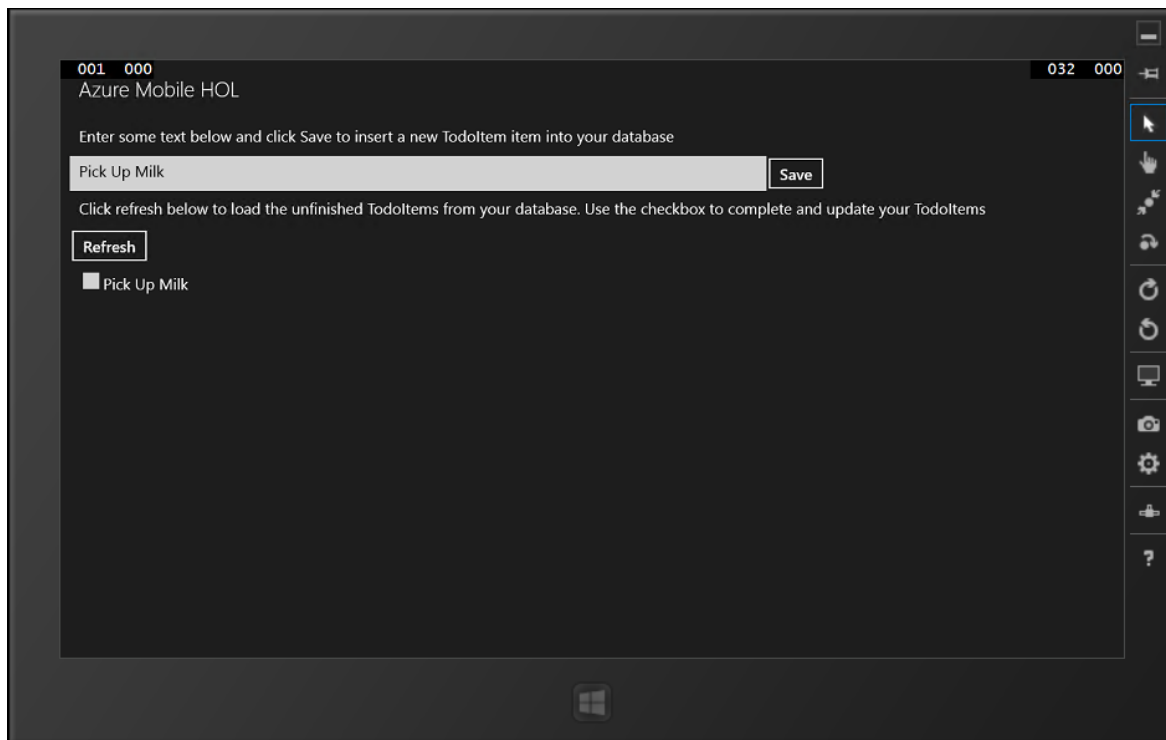


Figure 37: Windows Store application saving data against the Azure Mobile Service Data store

20. Go back to the Azure Portal (<http://manage.windowsazure.com>) to see the data in the Azure Mobile data and log in with the credentials you used earlier. Click on **Mobile Services** and select **AzureMobileHOL**.

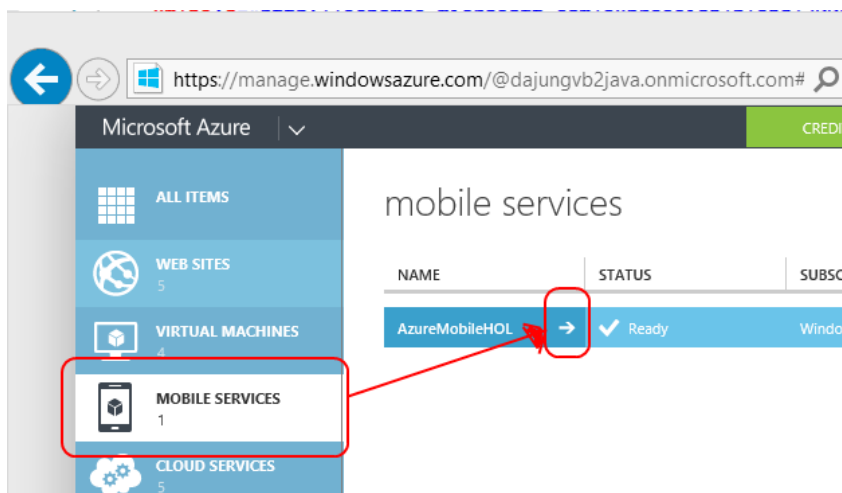


Figure 38: Going into Azure Mobile Services portal

21. Click on the **Data** link along the top and then select the **ToDoItem** table we created earlier.



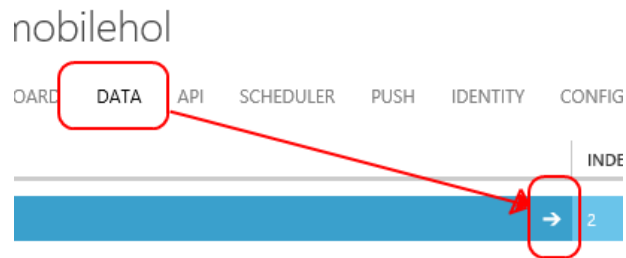






Figure 39: Going to view the data stored in the Azure Mobile SQL Database


22. You will see that the data you entered through the Windows Store app in the **ToDoItem** table.

Microsoft Azure

▼

CREDIT STATUS





ToDoItem

todoitem

BROWSE

SCRIPT

COLUMNS

PERMISSIONS

id	text	complete	__createdAt
8493E531-3378-426F-9318...	Pick up Milk	false	2014-08-21T15:54

Figure 40: Data entered in Windows Store application store in Azure SQL Table

23. Once you are done with the Lab, it is a good idea to delete both the Mobile Service and it created. Currently (as of 8/26/2014), Mobile Service is a free service, but the SQL Database it uses is not. If you leave them online, you might incur some unwanted Azure Database changes.
24. To remove the Mobile Service, click on the **Mobile Service** tab along the left hand side of the Azure Management Portal, select the **AzureMobileHOL** mobile service, and **click** the **Delete** icon along the bottom.

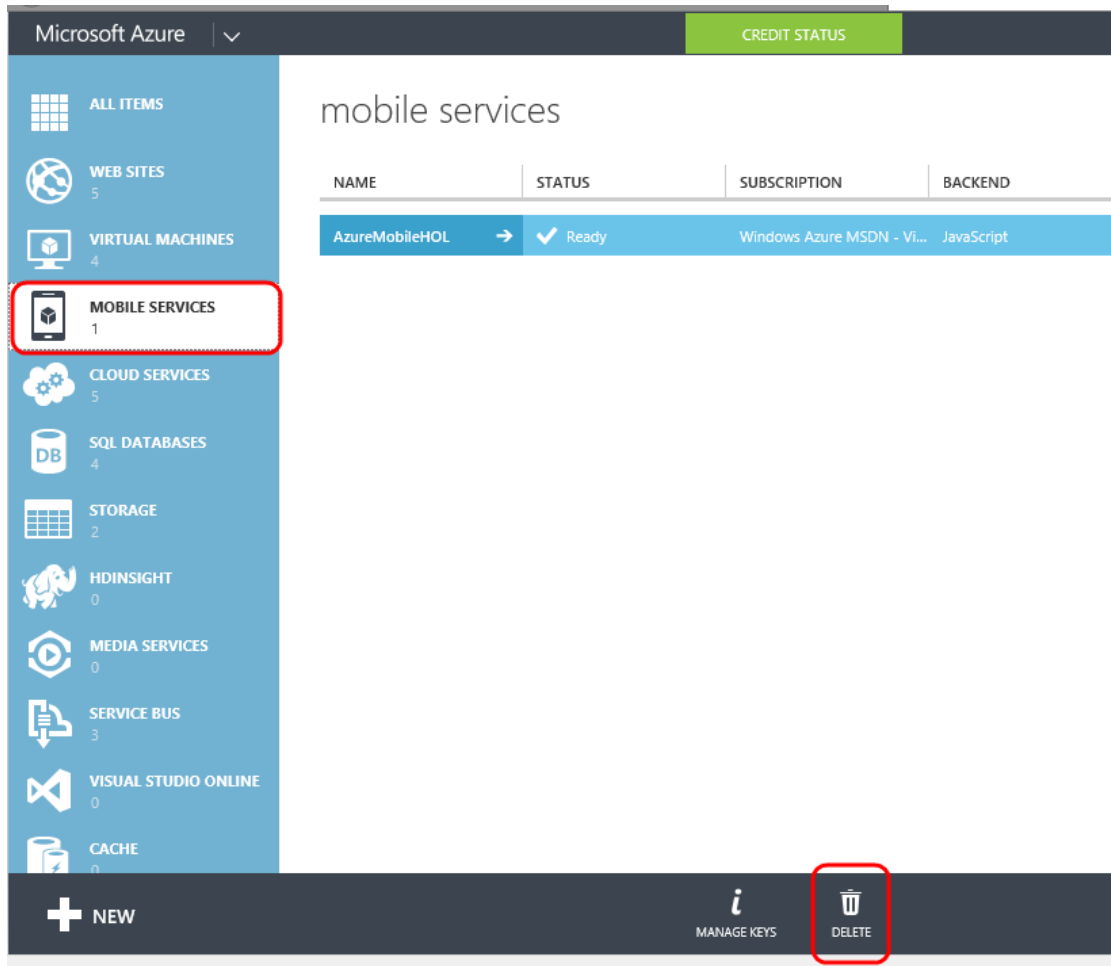


Figure 41: Deleting the AzureMobileHOL Mobile Service

25. You will be prompted to confirm if you want really delete the mobile service. **Mark the check box** to delete the data from the Mobile Service's database and press the **Check Mark** to proceed.

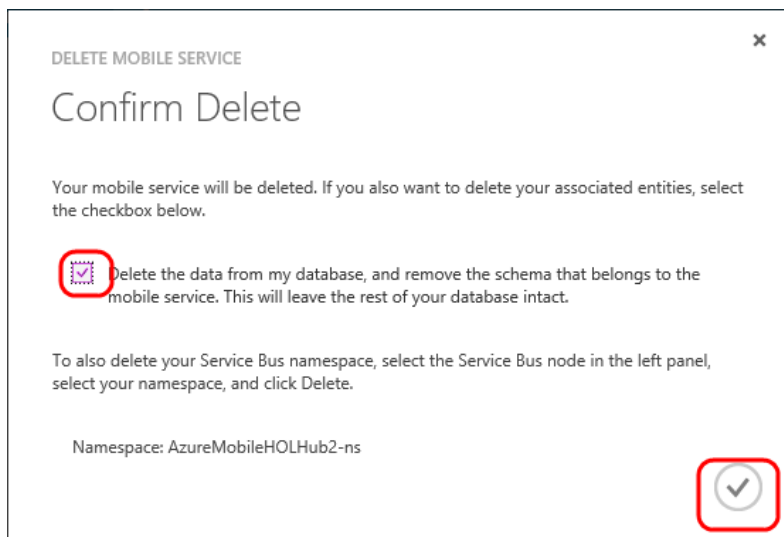


Figure 42: Confirm Delete pop-up for the Mobile Service

26. To remove the SQL Database that was created with the AzureMobileHOL Mobile Service, select the **SQL Databases** in the left hand menu of the Azure Portal. Select AzureMobileHOL\_db database and **press the Delete** button in the bottom menu.

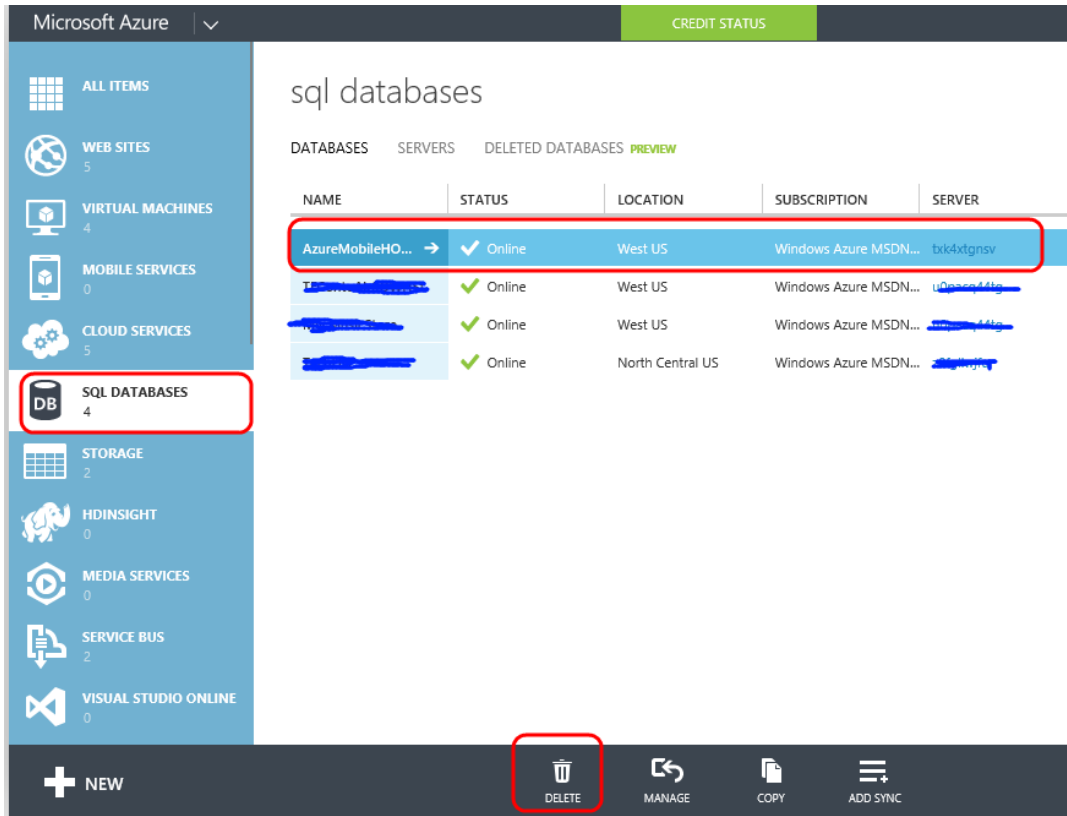


Figure 43: Delete the AzureMobileHOL\_db database

27. You will be prompted to confirm if you want to delete the database. **Press the Yes, Delete** button to commit the database deletion.

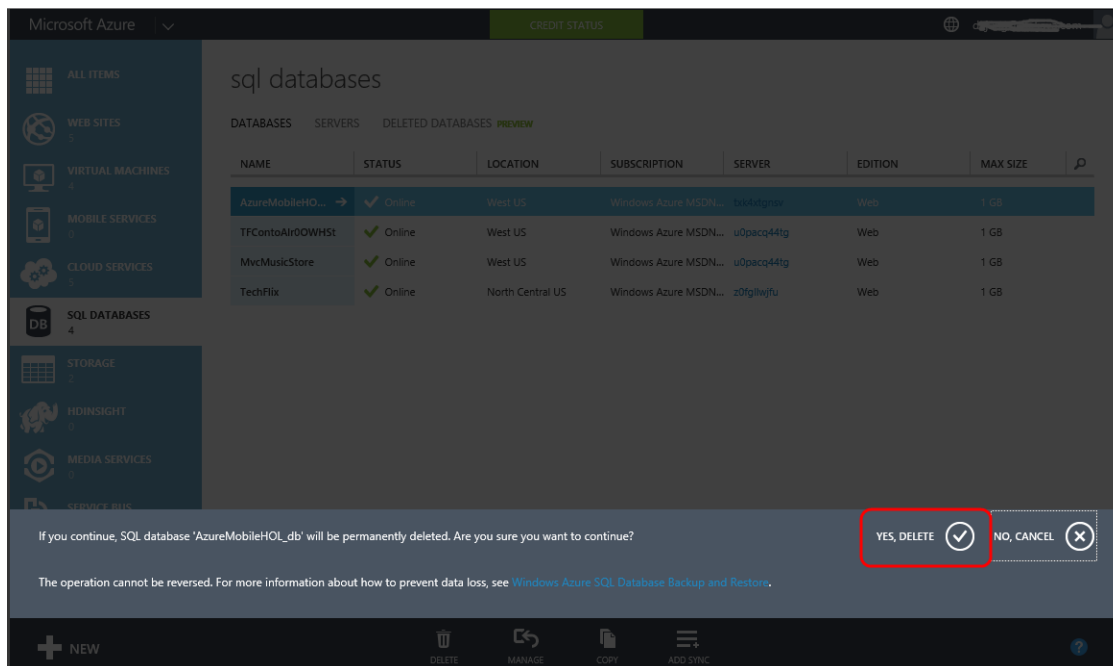


Figure 44: Confirm you want to delete the database

28. The Azure Mobile Service also creates a **Service Bus** that we will need to remove. Select the **Service Bus** from the left hand menu and select the **AzureMobileHOLHub-ns**. Click on the **Delete** button along the bottom menu.

Microsoft Azure | CREDIT STATUS

service bus

NAMESPACE NAME	STATUS	LOCATION	SUBS
SomethingHOLHub-ns	✓ Active	West US	Wind
AzureMobileHOLHub2-ns →	✓ Active	West US	V Inc

NEW CREATE CONNECTION INFORMATION DELETE

Figure 45: Delete the Service Bus created by the Mobile Service

29. You will be prompted to confirm you want to delete the Service Bus. You will need to enter the Service Bus name that is provided for you at the end of the warning paragraph. Type the namespace, it should be something like **AzureMobileHOLHub-ns**, into the **Confirm Namespace Name** field and press the **Check Mark** to process the request.

DELETE SERVICE BUS NAMESPACE

×

Delete Namespace Confirmation

If you continue, your namespace (and its entities) will be permanently deleted, and you won't be able to ~~recover them~~. If you still want to delete the namespace, type its name ('AzureMobileHOLHub2-ns').

CONFIRM NAMESPACE NAME

AzureMobileHOLHub2-ns

×

ENTITIES	COUNT	
Queues	0	
Topics	0	
Relays	0	
Event Hubs	0	
Notification Hubs	1	

✓

Figure 46: Confirm you want to delete the Service Bus