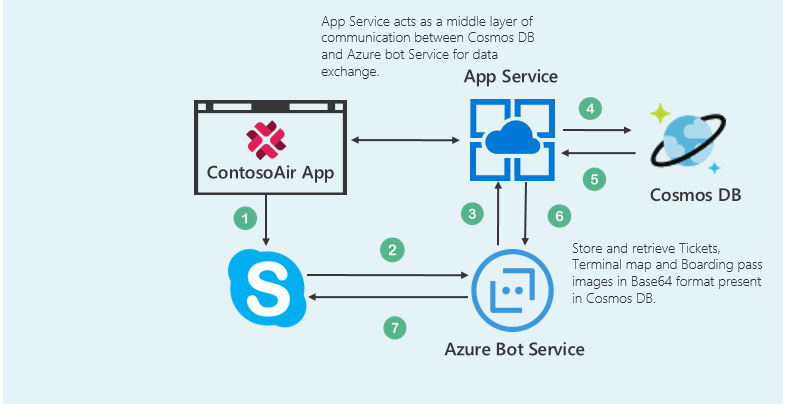
# Azure Bot Service using Skype Bot

## User Story

Consider the user scenario, Kevin is driving towards the airport and due to some reasons, he would get late to catch up his flight. He wants immediate help regarding **flight** **schedule**, **alternate** **flight** **options** and **E** **check-in** to get **boarding** **pass**. He also need **airport** **terminal** **map**, so that he can save his time to find out the route of boarding gate.

Using **ContosoAir** application you will come across **Azure Bot Service**, **Cosmos DB** database to get flight assistance. Azure Bot Service creates an intelligent Bot that interacts with user naturally wherever they are.

## Components Used in This Scenario



* The **Azure Bot Service** provides an integrated environment that is purpose-built for bot development, enabling you to build, connect, test, deploy, and manage intelligent bots, all from one place. You can write your bot in C# or Node.js directly in the browser using the Azure editor. [(Azure Bot Service)](https://docs.microsoft.com/en-us/bot-framework/azure-bot-service-quickstart)
* **Azure Cosmos DB** is Microsoft's globally distributed, multi-model database. With the click of a button, Azure Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure's geographic regions. It offers throughput, latency, availability, and consistency guarantees with comprehensive [service level agreements](https://aka.ms/acdbsla) (SLAs), something no other database service can offer. ([Azure Cosmos DB](https://docs.microsoft.com/en-us/azure/cosmos-db/))

## What you will learn from this lab

* Creating a skype Bot using **Azure** **Bot Service**.
* Create collections/insert data in **Cosmos** **DB** using **Data** **Migration** **Tool**.
* Playing around code in **Visual Studio 2017** IDE to add some options in the menu list in skype bot.

***Ready? Let’s get started!***

## Scenario 1 - Creating Azure Bot

*We’ll start with accessing* ***Azure Portal.*** *For creating resources, you need to* ***Sign in*** *to* ***Azure Portal.*** *The steps to do the same are given below:*

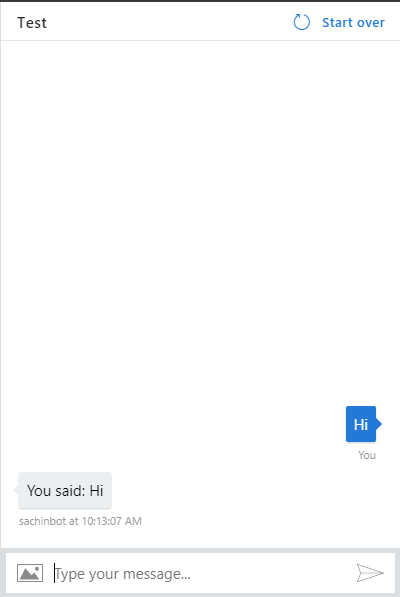
1. Click on the link [www.portal.azure.com](www.portal.azure.com%20) to open **Azure Portal** and maximize the browser window.
2. In the **Email** or **Phone** field, enter the given subscribed user Email id or phone number “[user208240@cloudplatimmersionlabs.onmicrosoft.com](mailto:user208240@cloudplatimmersionlabs.onmicrosoft.com)**”**.
3. In the **Password** field, enter the given password “**JuXe9g:\*4jk[”**
4. Click on **Sign in** button.
5. You may encounter a popup entitled **Welcome to Microsoft Azure** with options to **Start Tour** and **Maybe Later** – **Choose Maybe Later**. [Ignore the step no. 5 if didn’t get the popup message]

Great! You are now logged in into the Azure Portal.

1. Click on the **Resource groups** option present under icon  and click on the Resource group named as “**BotResourceGroup**” under **Resource groups blade**.
2. Now click on  present in right side panel with **Type** as App Service.
3. Select **C#** from **choose a template** option then click on **Basic** template and click on **Next** button.
4. Click on **Create Microsoft App ID and password** button.  It will redirect you to another tab for generating **App** **ID** and **Password**.
5. In the **Email** or **Phone** field, enter the given subscribed user Email id or phone number [user208240@cloudplatimmersionlabs.onmicrosoft.com](mailto:user208240@cloudplatimmersionlabs.onmicrosoft.com) and click on **Next** button.
6. In the **Password** field, enter the password **“JuXe9g:\*4jk[”**.
7. Click on **Sign in** button to register the bot on **Application** **Registration** **Portal**.
8. Copy **App** **ID** and paste it into the **Notepad** file named as **Credentials** present on the **Desktop** and save the changes.
9. Now again come back to the browser and click on  button to generate the **App** **Password**. It will generate the **pop-up** window containing **App** **password**.
10. Copy generated Password into the **Notepad** file named as **Credentials** present on the **Desktop** and save the file. Again, come back on the browser and click on **Ok** button.
11. Click on  button and paste the **App** **Password** into field **Paste** **password from the Microsoft App registration portal**, select **both** the checkboxes  for accepting T**erms** and **Privacy** **Conditions**.
12. Finally click on **Create bot** button to deploy the **Bot** on Azure.

**Note:** Please wait, it takes some time to deployment Bot template.

1. After successful completion of the deployment process you will be redirected to the **BUILD** tab .
2. Click on **Download zip file** button present under **Download source code** section to download the source code of the created Bot.
3. To test your Bot, click on the **Test** button provided on right side of the page. 
4. Type your text message in the provided text field and press **Enter** button. The Bot will reply you intelligently.



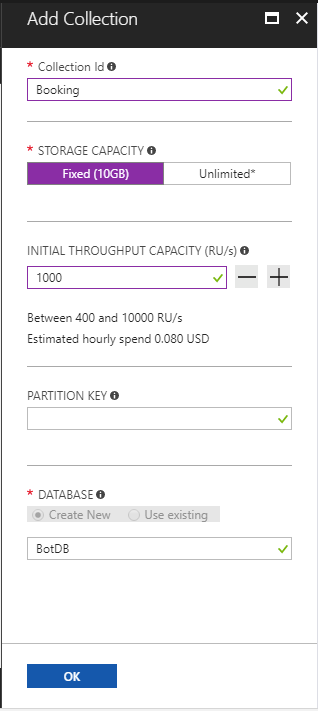
*Nice work! You have successfully created Azure Bot.*

## Scenario 2 - Creating Cosmos DB collection

### Part A - Adding collection into Cosmos DB

*Let’s create database into Cosmos DB(SQL) with Bookings as collection and store data into it.*

1. Switch to Azure Portal by clicking on the link [poral.azure.com](http://portal.azure.com) as launched in **Scenario 1**.
2. Click on the **Resource groups** option present under  icon and click on the Resource group named as “**BotResourceGroup**” under **Resource groups blade**.
3. Now click on  present in right side panel with **Type** as **Azure Cosmos DB account**.
4. Click on option present on the menu bar.
5. Enter Collection Id as **Bookings** in the **Collection** **Id** field present under Add Collection blade. Select the **STORAGE CAPACITY** as **Fixed** and enter **DATABASE** as **BotDB** in the **DATABASE** field and finally click on **OK** button.



1. Now click on **Keys** option present under **Azure** **Cosmos** **DB** **account** blade in the left panel and copy the **URI**, **PRIMARY** **KEY** and **PRIMARY** **CONNECTION** **STRING** values and paste it into the **Notepad** file named as **Credentials** present on **Desktop** and save the file.



### Part B - Inserting data in Cosmos DB using Data Migration Tool

*But, what you will do with Cosmos DB without data? So, let’s insert some data into Cosmos DB using Microsoft’s Data Migration Tool.*

For more details about the migration tool, click on the given link: [Data Migration tool.](https://docs.microsoft.com/en-us/azure/cosmos-db/import-data)

1. Launch the **Data Migration Tool** by double clicking on **dtui.exe** file present on **Desktop**.
2. After the tool is launched, click on **Source Information** menupresent on the left panel**.**
3. Select **JSON file(s)** option from **Import from** dropdown.
4. Click on **Add Files** button.
5. Select first JSON file named **Flights.json** at the location **“C:\Source\CosmosDBCollections”** and click on **Open** button.

**Note:**

Here there are three Json file, please select one file at a time and follow the below steps for each file one by one.

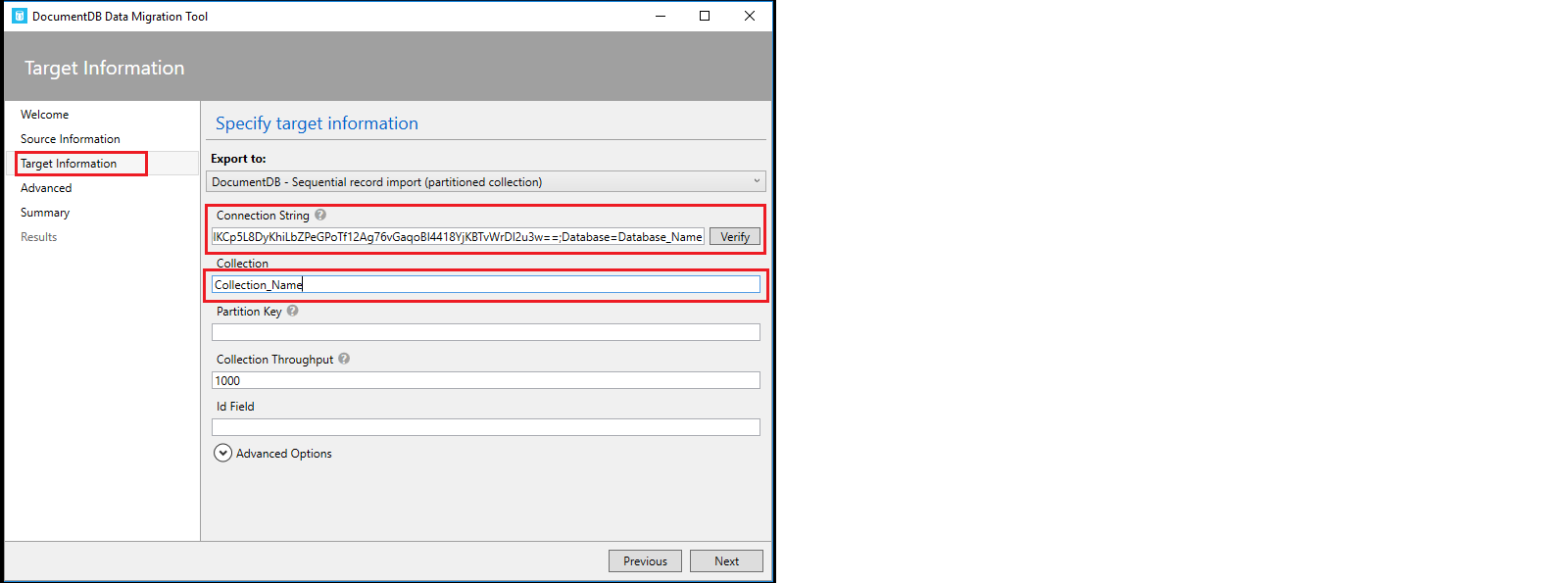
1. Then click **Next button** to redirect to **Target information** menu.
2. Now copy the **PRIMARY CONNECTION STRING** from the notepad file present on **Desktop**.
3. Switch back to **Data Migration** **Tool** already launched in **Step** **1** of **Part B**.
4. Paste the copied **PRIMARY** **CONNECTION** **STRING** in **Connection String** text box and appended it withthe **Database=”Your Database Name”** at the end of the string in **Target information** option.

**For ex.**

**AccountEndpoint=https://botdbsachin.documents.azure.com:443/;AccountKey=A2PtWXbTdNwCvdFD30zfwIFHsvZx4jHfhSBBTur0EUYWZiD5BiQDhLfbU1j5JTion569Ze4PtiKXUMPTI62lvw==;**Database=BotDB

**Note:**

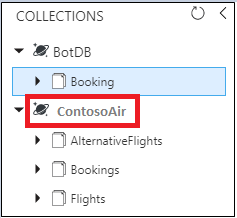
In our case **Your Cosmos DB Account Name** is **BotDB.** so, append the string **Database= BotDB** at the end of copied **PRIMARY** **CONNECTION** **STRING**.

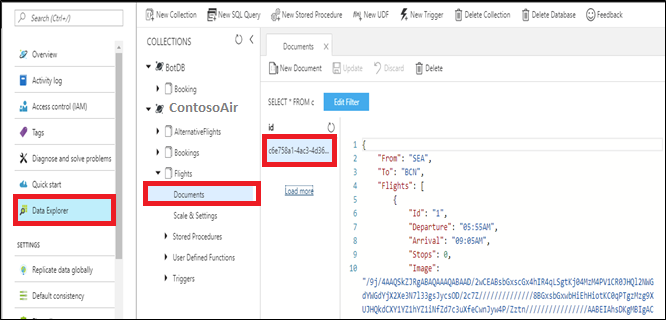


1. Enter the **Collection Name** same as the name of the **Json** file mentioned in **step 5** in the **Collection** field. **(Flights, Bookings, AlternateFlights)**

**Note**: The Collection name must be Flights, Bookings, AlternativeFlights only for the respective three Json file mentioned under folder “C:\Source\CosmosDBCollections”

1. Now click on **Verify** button next to **Connection String** field to verify the connection string.
2. On successful verification of Connection string, success pop-up window will display, then click on **OK** button.
3. On verification failure of connection string, a pop-up window will appear prompting the **failure message**, then click on **Ok** button and again verify the connection string till successful verification.
4. Click on **Next -> Next ->** **Import** to import the data present in Json file into **Cosmos DB collection.**
5. This will start the data uploading process. Wait for some time to complete the process and once the process is completed successfully **Close** the **migration tool**.
6. Repeat the steps **1 to 6** and **10 to 18** to import other two files **AlternateFlights.json** and **Bookings.json** present at the location **“C:\Source\Json”** into **Cosmos** **DB**.
7. Now, switch to **Azure Portal** in which you are already logged in from **Scenario 1**.
8. Click on Azure Portal’s **Resource Group** option present in the favourites blade in the left side panel and click on **“BotResourceGroup”.**
9. Click on **“cosmosdbbot”** which is your **Cosmos DB Account.** Then click on **Data Explorer** option present under **Azure Cosmos DB account** blade to view the imported **collections.**
10. Then click on created collection **“Flights”** to expand. Then, click on **Documents** option under the **COLLECTIONS** to view the data imported through **Migration tool** into **Cosmos DB.**





*Nice work! You have successfully imported data in your Cosmos DB collection through Data Migration Tool.*

## Scenario 3 – Playing around the Visual Studio code

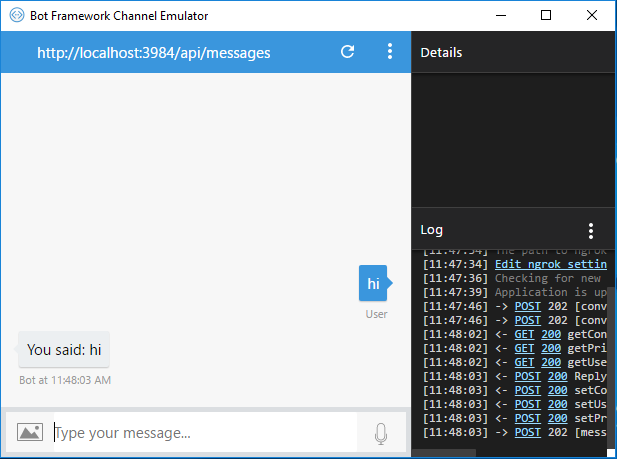
### Part A - Importing the projects into Visual Studio 2017 IDE

*Let’s play with the bot code locally, downloaded from the Azure portal*

1. Go to **Downloads** section. Right click on the file **ContosoAirBot.zip** and click on the option **Extract to ContosoAirBot** option for extracting the zip file into the folder and double click on that folder to open.
2. Double click on **Microsoft.Bot.Sample.SimpleEchoBot.sln** file to load the project into **Visual Studio 2017** IDE.
3. Now click on the  button to run the project.
4. It redirects to the browser, copy the **URL** present on browser and paste it into the botframework-emulator present on the **Desktop** by double clicking on the botframework-emulator icon and append **/api/messages** text in the URL, click on **Connect** button.

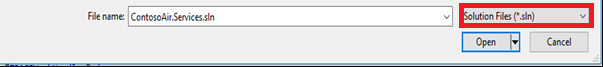
**For eg:**

<http://localhost:3984/api/messages>



*Here you are successfully connected the bot with botframework-emulator to see the changes locally. Type the texts in the provided text box, The Bot will reply you intelligently.*

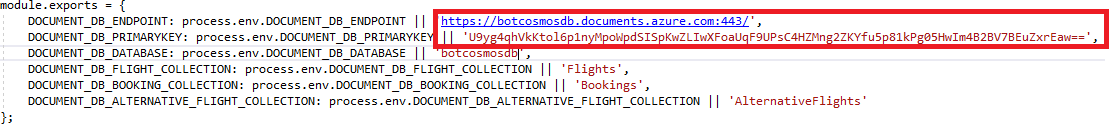
1. Go to the **Visual Studio 2017** IDE and click on  button to stop debugging. Now right click on the **Solution** window-> **Add** and then click on **Existing Project**.
2. Right click on the Solution **Microsoft.Bot.Sample.SimpleEchoBot** and click on **Add**->**Existing** **Project** and select the file type as **Solution** **Files** (\*.sln) browse the file **ContosoAir.Services.sln** present at location **C:\Source\ContosoAir** and click on **Open** button.



1. Again Right click on the Solution **Microsoft.Bot.Sample.SimpleEchoBot** and click on **Add**->**Existing** **Project** and select the file type as **Solution** **Files** (\*.sln) browse the file **ContosoAir.Xamarin.sln** present at location **C:\Source\ContosoAir** and click on **Open** button.



1. Double Click on **config.js** file from **ContosoAir.Services** project to open in the code editor window.
2. Replace the values for parameters **DOCUMENT\_DB\_ENDPOINT**, **DOCUMENT\_DB\_PRIMARYKEY** with **URI** and **PRIMARY KEY** present in **Credentials** file on the **Desktop** and save the file.



1. Right click on project **ContosoAir.Services**, click **on Open Command Prompt Here** option. It will open the command prompt screen. Now write command **npm install** and press enter button to install all the dependencies.

13. Select the **Release** option and click on **Internet** **Explorer** to run the **ContosoAir.Service**

project in the release mode.



Note:

Here it deployes the services on localhost.

It will redirect you to browser, Please ignore this step.

14. Switch to **Azure Portal** by clicking on the link [portal.azure.com](http://portal.azure.com) as launched in **Scenario 1**.

15. Click on the **Resource groups** option present under  icon and click on the Resource group named as “**BotResourceGrp**” under **Resource groups blade**.

16. Now click on the option present in the right-side panel with **Type** as **App Service.**

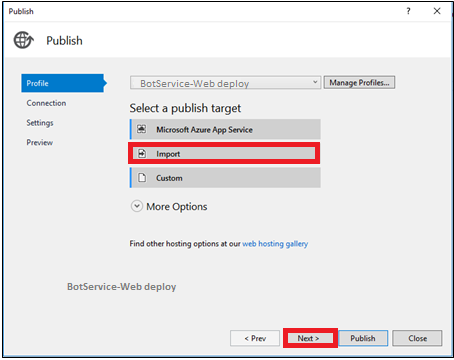
17. Click on  option present on Menu bar. It will **download** one file.

18. Now go to **Visual Studio 2017** IDE and right click on the project **ContosoAir.Services** from

the **Solution** **Explorer** window and click on **Publish** option.

19. Click on the **Import** option present under **Profile** Tab. Click on the **Browse** button and select

the downloaded file from **Downloads** section and click on **Open**->**OK** button.



20. Click on  button to validate the connection then click on **Next**->**Next**-

>**Publish** button. After successful publish it will redirect you to the **Browser** and copy and

paste the **URL** into the txt file present on **Desktop**.

### Part B – Publishing the Bot

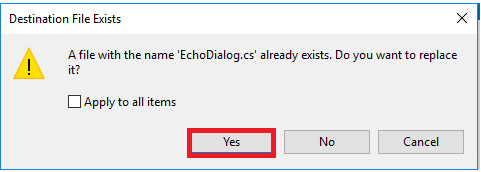
*Lets modify and publish your bot with some changes*

1. Now go to the **Visual Studio 2017** IDE
2. Create two folders with the following name at the root level of the project named **Microsoft.Bot.Sample.SimpleEchoBot.**

* Forms
* Model

1. Right click on the **Dialogs** folder and click on the **Add** button. Select **Existing** **Item**.

Go to the **Documents/AzureBotCode/Dialogs** under **This** **PC** section, select all the file and click on **Add** button. It will show one pop-up window click on **Yes** button.



1. Right click on the **Forms** folder and click on **Add** button. Select **Existing** **Item**.

Go to the **Documents/AzureBotCode/Forms** under **This PC** section, select all the files and click on **Add** button.

1. Right click on the **Models** folder and click on **Add** button. Select **Existing** **Item**.

Go to the **Documents/AzureBotCode/Models** under **This** **PC** section, select all the files and click on **Add** button.

1. Now right click on the Project **Microsoft.Bot.Sample.SimpleEchoBot** and select **Add** button, click on **Existing** **Item** and browse **Locale.cs** and **Constants.cs** files from the **Documents/AzureBotCode** under **This** **PC** and click on **Add** button.
2. Add **using SimpleEchoBot.Dialogs; namespace** in the **MessagesController.cs** file present under **Controller** folder and save the file.
3. Now go to the project **Microsoft.Bot.Sample.SimpleEchoBot** and double click on **Global.asax.cs** file add the following namespaces in the namespace section.

**Note:**

using Autofac;

using System.Configuration;

using Microsoft.Bot.Connector;

using Microsoft.Bot.Builder.Azure;

using Microsoft.Bot.Builder.Dialogs;

using Microsoft.Bot.Builder.Dialogs.Internals;

using Microsoft.Bot.Builder.Internals.Fibers;

**Note:**

Add below code in the application **start()** function block in **Global.asax.cs** file.

var uri = new Uri(ConfigurationManager.AppSettings["DocumentDbUrl"]);

var key = ConfigurationManager.AppSettings["DocumentDbKey"];

var store = new DocumentDbBotDataStore(uri, key);

Conversation.UpdateContainer(

builder =>

{

builder.Register(c => store)

.Keyed<IBotDataStore<BotData>>(AzureModule.Key\_DataStore)

.AsSelf()

.SingleInstance();

builder.Register(c => new CachingBotDataStore(store, CachingBotDataStoreConsistencyPolicy.ETagBasedConsistency))

.As<IBotDataStore<BotData>>()

.AsSelf()

.InstancePerLifetimeScope();

builder.RegisterModule(new ReflectionSurrogateModule());

});

1. Copy below three lines of code under the **appSettings** tag in **Web.config** file, replace the **URL**, **KEY**, **APIEndopoint** with the values present in **Credentials** file present on the **Desktop** make sure that **/api/bot** text is appended after the **APIEndpoint**.

**Note:**

<add key="DocumentDbUrl" value="https://bot.documents.azure.com:443/"/>

<add key="DocumentDbKey" value="fKNcATLq4vu74YxVXyFDW3rEUdv17l8nKhEyH0xA9ArXUs1wVfqvSm61GirqS0pjMRRmQRpSU8wqvSpInhdpiA=="/>

<add key="APIEndpint" value="http://botapi.azurewebsites.net/api/bot" />

1. Click on  to run the application locally.

**Note:** It will take some time to run the application. After completion of process it redirects to the browser.

**For ex:** [**http://localhost:3984**](http://localhost:3984)

copy the **URL**, append text **/api/messages** and paste in the botframework-emulator  and click on Connect button. Here you will see 3 options in the skype bot.

**Note:**

Now add your own option into bot and see some changes.

1. Adding 4 th option in the existing Bot

Add below line at Line number 40

MenuOption.Options.Add(new Options(4, "Flight Status"));

Add below at line number 82

else if (number == 4)

context.Wait(CheckFlightStatus);

public async Task CheckFlightStatus(IDialogContext context, IAwaitable<IMessageActivity> result)

{

var res = await result;

var PNRCode = res.Text;

var message = context.MakeMessage();

if (res.Text != "")

{

try

{

IEnumerable<FlightStatus> flightstatusdata = null;

FlightStatus flightstatus = null;

using (var httpClient = new HttpClient())

{

var response = await httpClient.GetAsync(ConfigurationManager.AppSettings[Constants.KeyApi] + "/flightstatus/" + PNRCode);

object DeserializeResult = JsonConvert.DeserializeObject(await response.Content.ReadAsStringAsync());

flightstatusdata = JsonConvert.DeserializeObject<IEnumerable<FlightStatus>>(DeserializeResult.ToString());

}

foreach (FlightStatus flttatus in flightstatusdata)

{

flightstatus = flttatus;

}

if (flightstatus != null)

{

var sb = new StringBuilder();

sb.AppendLine(" Flight No : " + flightstatus.Flight);

sb.AppendLine(Constants.EmptyLine);

sb.AppendLine(" Departure Time : " + flightstatus.departTime);

sb.AppendLine(Constants.EmptyLine);

sb.AppendLine(" Terminal No : " + flightstatus.FromTerminalNo);

sb.AppendLine(Constants.EmptyLine);

sb.AppendLine(" Gate No : " + flightstatus.FromGateNo);

sb.AppendLine(Constants.EmptyLine);

await context.PostAsync(sb.ToString());

context.Wait(OnComplete);

}

else

{

await context.PostAsync(Locale.FlightNotFound);

context.Wait(OnComplete);

}

}

catch (Exception ex)

{

await context.PostAsync($"Failed with message: {ex.Message}");

}

}

}

Note:

1. Click on  to run the application locally.

**Note:** It will take some time to run the application. After completion of process it will redirect to browser.

**For ex:** [**http://localhost:3984**](http://localhost:3984)

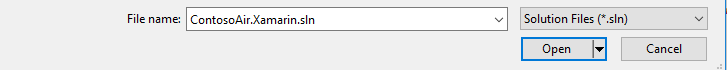
copy the **URL**, append text **/api/messages** and paste it in the botframework-emulator  present on the **Desktop** and click on Connect button. Here you will see 4 options in the skype bot.

13. Now go to **Visual Studio 2017** IDE and right click on the project **Microsoft.Bot.Sample.SimpleEchoBot** from the **Solution** **Explorer** window and click on **Publish** option.

Awesome, here you have successfully added options for the skype bot.

**Here mention the directory**

1. Right click on the Solution **Microsoft.Bot.Sample.SimpleEchoBot** and click on Add->Existing Project and select the file type as Solution Files (\*.sln) and select ContosoAir.Xamarin.sln and click on **Open** button.



Expand App folder from soluton expand ContosoAir.Clients.Core and click on GlobalSettings.cs file

Copy APP ID from the credentials file present on the Desktop and past it after 28:

As

Example

public const string SkypeBotAccount = "28:114eaf90-29cf-4095-90ab-bc42dfdaa773";

Now go to App-> ContosoAir.Clients.UWP and right click on it, click on Set as Set as StartUp Project.



Click on local machine.

It will take some time to build the whole thing

## Conclusion

In this experience, you explored the configuration of **Azure Bot Service, Cosmos DB.** You also played around the code using **Visual Studio – 2017 IDE** to change the Bot source code with the help of demo **ContosoAir App.**

Now you know how to

* Create **Bot** using **Azure** **Bot** **Services**
* Create collections/insert data in **Cosmos** **DB** using **Data Migration Tool**.
* Playing around the code using **Visual** **Studio** **2017** IDE.