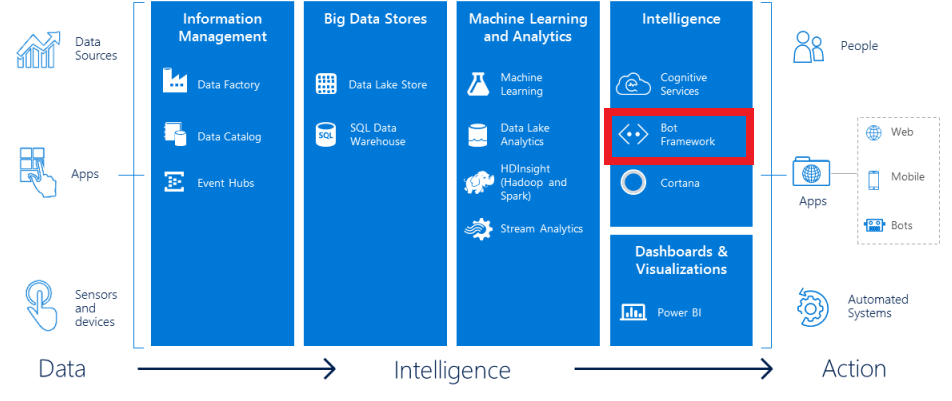
Azure Bot Service

LAB to create a simple bot, hosted in Azure using an App Service Plan

# Summary

This lab explores how to build and host a bot using the Azure Bot Service



In this lab, you will create a bot to interact with users. Several bot features will be used including dialogs, form flows, prompt dialogs, and bot state.

## Business Case

An example of a business case would be to provide support for internal or external users. This is key to maintaining their satisfaction. Ideally, this support is provided in a timely manner and without using more resources than required.

## Learning Objectives

Upon completing this lab, you will have hands-on experience with the following functions and concepts related to Microsoft’s Bot Framework:

* Creating a bot using the Azure Bot Service
* Testing a bot inside the Azure Bot Service
* Updating the bot using Visual Studio 2017
* Using form flow to create a bot form
* Creating a bot dialog
* Interacting with a user through the dialog

# Create a bot with the Azure Bot Service

The Azure Bot Service accelerates the process of developing a bot by provisioning a web host with one of five bot templates you can modify in an integrated environment that is purpose-built for bot development.

## Prerequisites

You must have a Microsoft Azure subscription before you can use the Azure Bot Service.

## Step 1 – Create your bot

1. On the Azure portal select **New** in the menu blade.
2. In the **New** blade, navigate to the Data + Analytics category, and select **Bot Service**.
3. In the Bot Service blade, provide the requested information, and click **Create** to create the bot service and deploy it to the cloud.

* Set **App name** to your bot’s name. The name is used as the subdomain when your bot is deployed to the cloud (for example, mybasicbot.azurewebsites.net).
* Select the subscription to use (your subscription should be visible in the drop-down box).
* Select the [resource group](https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/), [hosting plan](https://docs.microsoft.com/en-us/bot-framework/azure-bot-service-hosting-plan), and [location](https://azure.microsoft.com/en-us/regions/).
  + For this Lab, **create** a **new** resource group and use a new **App Service Plan**. You can use a Free App Service Plan (F1) for this lab.

1. Click on **Create** to deploy the bot service

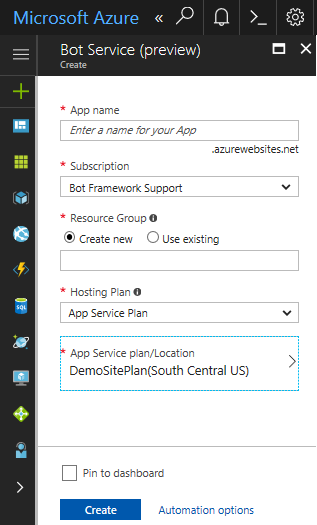


Figure 1 - Create a bot service to host your bot

1. Confirm that the bot service has been deployed.
   * Click **Notifications** (the bell icon that is located along the top edge of the Azure portal). The notification will change from **Deployment started** to **Deployment succeeded**.
   * After the notification changes to **Deployment succeeded**, click **Go to resource** on that notification.

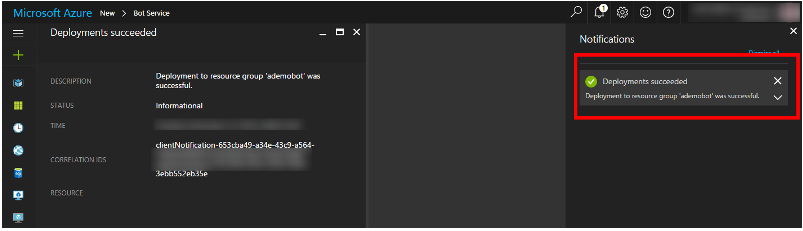


Figure 2 - The bot service has been desployed succesfully

## Step 2 – Select the programming language and template for your bot and create it

1. Choose the programming language that you want to use to develop your bot. For this lab we will select **C#**.

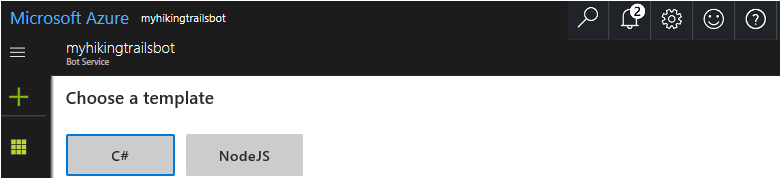


Figure 3 - Selecting the programming langage for the bot

1. Select the template to use as the starting point for developing your bot. For this lab choose the **Basic** template.
2. Then click **Next** to create the bot based on the programming language and template that you have chosen.

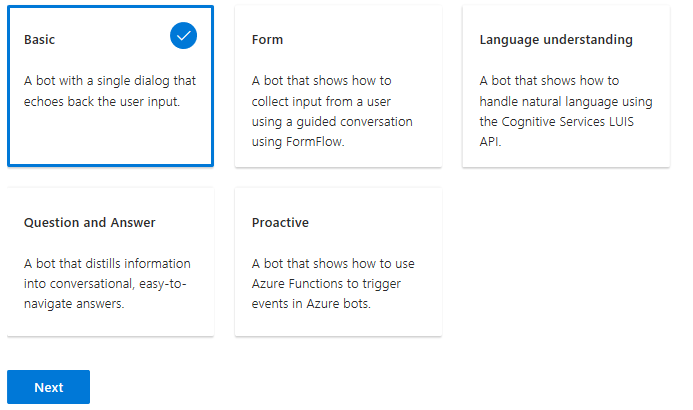


Figure 4 - Check the template and create the bot

## Step 3 - Create App ID and password

Next, create an app ID and password for your bot, so that it will be able to authenticate with the Bot Framework.

1. Click **Create Microsoft App ID and password**.
2. On the page that opens in a new browser tab, click **Generate an app password to continue**.
3. Copy and securely store the password that is shown, and then click **Ok**. It might be a good idea to save both the App name, the App ID and the password into notepad.
4. Click **Finish and go back to Bot Framework**.
5. Back in the Azure Portal, assure the **app ID** field is auto-populated for you, and paste the password that you copied (in step 3 above) into the password field.
6. Agree to terms, and click **Create bot**.

When the bot service finishes generating your bot, the Azure editor will contain the bot's source files. At this point, the bot has been created, registered with the Bot Framework, deployed to the cloud, and is fully functional.

## Step 4 - Test your bot

Now that your bot is running in the cloud, try it out by typing a few messages into the built-in chat control that's located to the right of the code editor in Azure. You should see that the bot responds to each message you send by echoing back your message prefixed with the text You said.

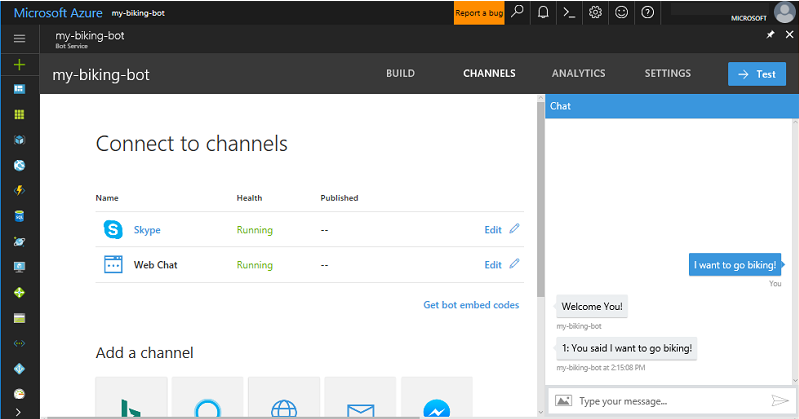


Figure 5 - Testing your bot with the built-in chat control

## Step 5 - Deploy changes to your web app bot

Using the App Service plan, it is very easy to modify your bot source and re-deploy your changes.

1. In Azure, click your bot's **BUILD** tab, and click **Open online code editor**.
2. Open the Dialogs folder, and click EchoDialog.cs.
3. Change text in line 22 from You said to You just said.
4. To deploy your changed source, click the Open Console icon.
5. In the Console window, type **build.cmd**, and press the enter key. The console window shows the deployment's progress until it's complete
6. Test your bot again by typing a few messages into the built-in chat control. Make sure to click the **Start over** button in the chat control before testing the bot again.

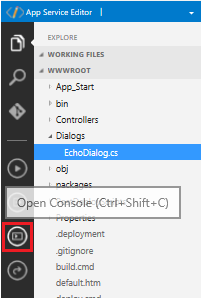


Figure 6 - Changing and re-building your bot

## Step 6 - Adding more functionality to your bot

For small changes to your bot, it is adequate to use the online editor as you have seen in the previous section. However, it is also possible to make changes to your bot using Visual Studio 2017. In this section we are taking the existing bot and add a more dialog based interface to it. You will download the bot’s solution to your development machine and open it in Visual Studio 2017. After that, you will add additional code, test the bot locally and finally upload the modified bot to your App Service Plan in Azure.

1. In Azure, click your bot's **BUILD** tab, and click **Download zip file**.
2. Store the zip file to a location on your local hard drive.
3. Unzip the file and navigate to the unzipped folder that was just created.
4. Open the solution that contains all your bot’s source code in Visual Studio 2017 by clicking **File**, **Open** and navigating to the solution file that you just downloaded and unzipped.
5. Click **OK** on the Security Warning window.
6. Build the entire solution to make sure that all NuGet packages are downloaded to your development machine by selecting **Build** – **Rebuild Solution** from the Visual Studio Menu.

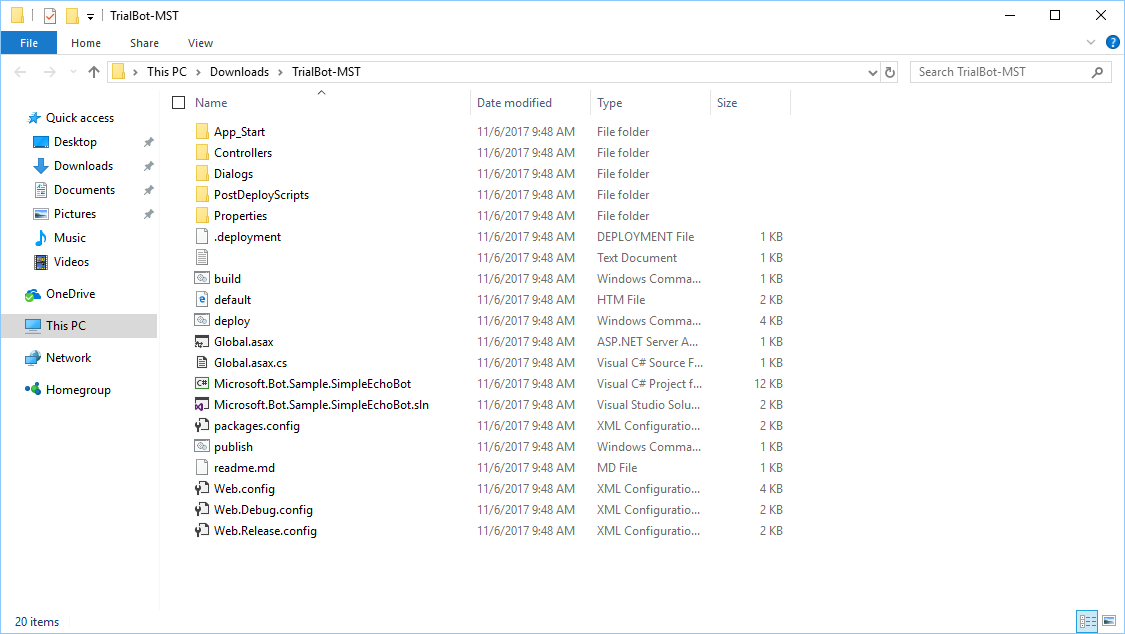


Figure 7 – Downloaded solution containing your bot, including a solution and project file.

## Step 7 – Test your bot locally using the Bot Framework Channel Emulator

The Bot Framework Emulator is a desktop application that allows bot developers to test and debug their bots, either locally or remotely. Using the emulator, you can chat with your bot and inspect the messages that your bot sends and receives. The emulator displays messages as they would appear in a web chat UI and logs JSON requests and responses as you exchange messages with your bot.

1. If you haven’t done so yet, you need to download the Bot Framework Emulator.
2. Download packages for Mac, Windows, and Linux are available via the [GitHub releases page](https://github.com/Microsoft/BotFramework-Emulator/releases). The latest Windows installer is available via the [emulator download page](https://emulator.botframework.com/) (download starts immediately). If you are running Windows, select the \*.exe Setup file.
3. Start your bot from inside Visual Studio 2017 by clicking on the start button. NOTE: If you get an error message, most likely you are using an outdated version of IISExpress 10.0. The easiest way to fix this is by reinstalling IISExpress.

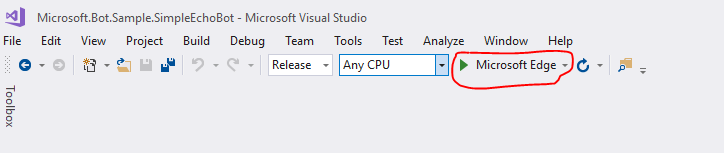


Figure 8 - Starting your bot from inside Visual Studio

1. Use the endpoint that you will see in the address bar of your browser to connect to the Bot Framework Emulator

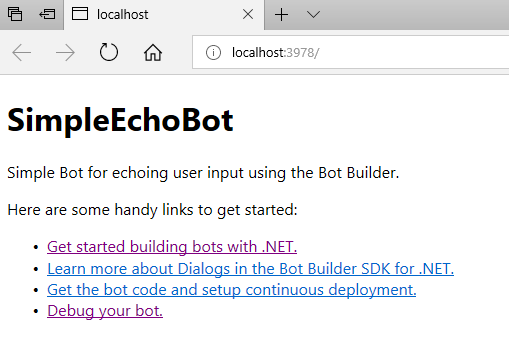


Figure 9 – Your bot’s endpoint

1. Launch the Bot Framework Emulator and enter your bot’s endpoint into the emulator’s address bar (in this example: <http://localhost:3978/api/messages>.
2. Enter your bot’s App Service credentials to the Bot Framework Emulator as shown in Figure 10.
3. Click on the Connect button.

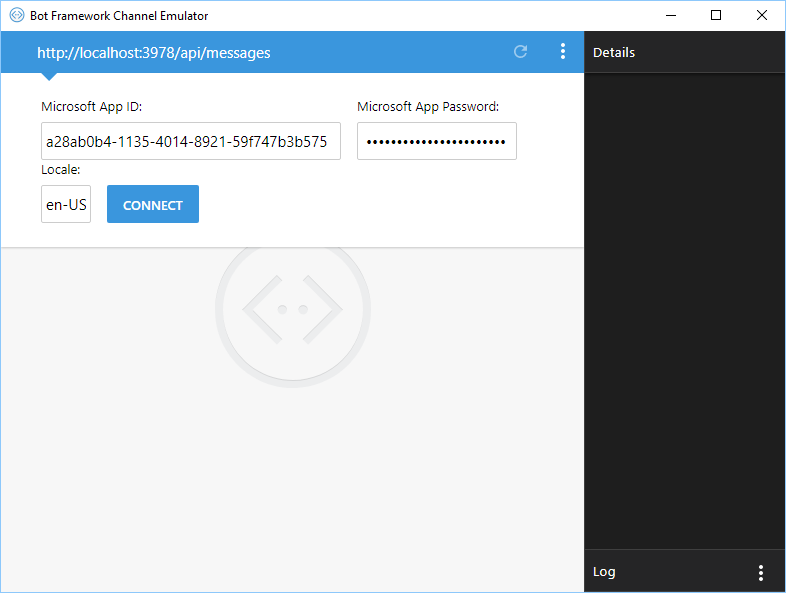


Figure 10 - Connecting the Bot Framework Channel Application

1. Stop debugging in Visual Studio 2017.
2. Inside Visual Studio 2017, open the **Web.Config** file and add the same credentials on line 9 and line 10 in the file and save it.
3. Start debugging the bot again inside Visual Studio 2017 and use the Bot Framework Channel Emulator to debug your bot locally.

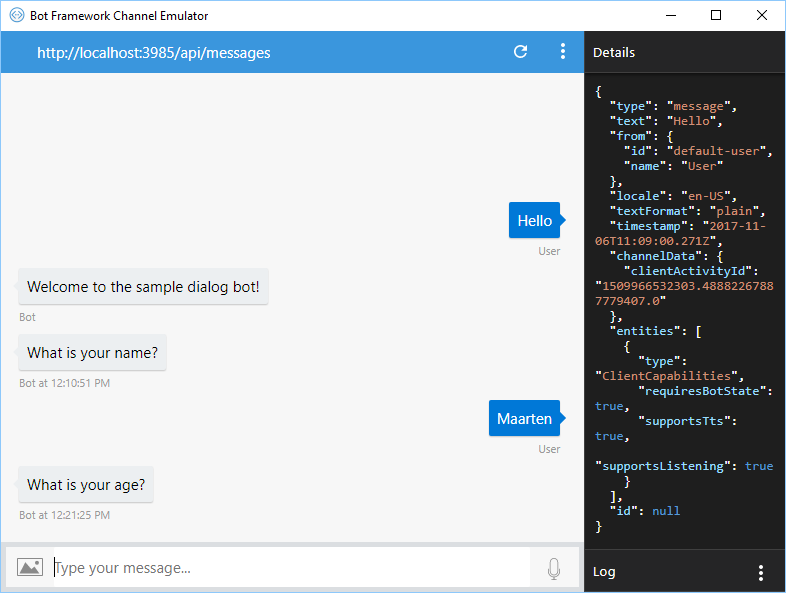


Figure 11 - Our bot running locally inside the Bot Framework Channel Emulator

## Step 8 – Add a dialog based conversation to your bot

Now we will add a dialog style conversation where the bot will ask for your (first) name and age.

1. Add a new class to your solution by navigating to the **Dialog** folder in your solution, **right-clicking** in that folder and select **Add New Item – Class**.
2. Call your new class **NameDialog.cs**.

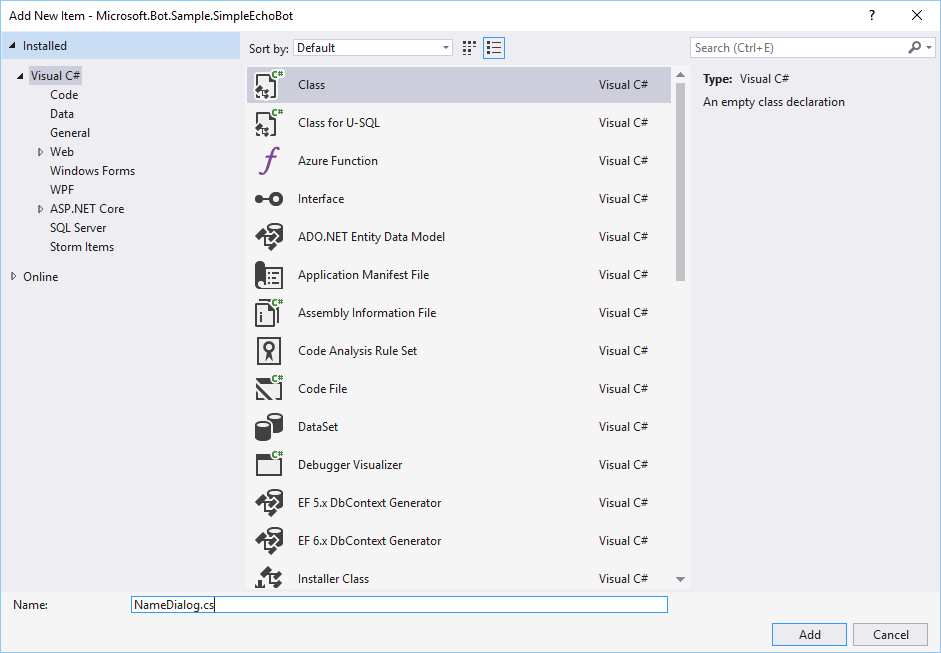


Figure 12 - Adding a new class NameDialog.cs to your solution.

1. Replace the contents of the newly created **NameDialog** class by the following:



1. Add another new class to your solution by navigating to the **Dialog** folder in your solution, **right-clicking** in that folder and select **Add New Item – Class**.
2. Call your new class **AgeDialog.cs**.
3. Replace the contents of the newly created **AgeDialog** class by the following:



1. Add another new class to your solution by navigating to the **Dialog** folder in your solution, **right-clicking** in that folder and select **Add New Item – Class**.
2. Call your new class **RootDialog.cs**. This class will determine the navigation between the different dialogs in your bot/
3. Replace the contents of the newly created **RootDialog** class by the following:



1. Finally, open the exiting class MessagesController.cs that you can find in the folder Controllers inside your Visual Studio solution.
2. Replace the following line of code (should be at line 25):

await Conversation.SendAsync(activity, () => new EchoDialog());

with the following

await Conversation.SendAsync(activity, () => new RootDialog());

1. Start debugging the bot again inside Visual Studio 2017 and use the Bot Framework Channel Emulator to debug your bot locally. You should now be able to have a simple conversation with your bot. It will ask you for your name and age. After entering, it will show you a message and starts all over from the beginning.

## Step 9 – Publish your modified back in Azure

To set up publishing from Visual Studio using the .PublishSettings file, perform the following steps:

1. In Visual Studio, click **View**, and click **Solution Explorer**.
2. In the Solution Explorer pane, right-click your project, and click **Publish...** The Publish window opens.
3. In the Publish window, click **Create new profile**, click **Import profile**, and click **OK**.
4. Navigate to your project folder, navigate to the **PostDeployScripts** folder, select the file that ends in .PublishSettings, and click **Open**.

You have now configured publishing for this project. To publish your local source code to Azure Bot Service, right-click your project, click **Publish...**, and click the **Publish** button.

Now you can go back to your Azure portal Now that your bot is running in the cloud, try it out by typing a few messages into the built-in chat control that's located to the right of the code editor in Azure. You can now have the same conversation with your bot, but this time it is running inside Azure.

This concludes the lab on bots. Of course, this was just an initial exploration of the Microsoft Bot Framework. We encourage you to continue experimenting with the Microsoft Bot Framework and also to look at integrating Microsoft Cognitive Services into your bot.