BOT FRAMEWORK

Contents

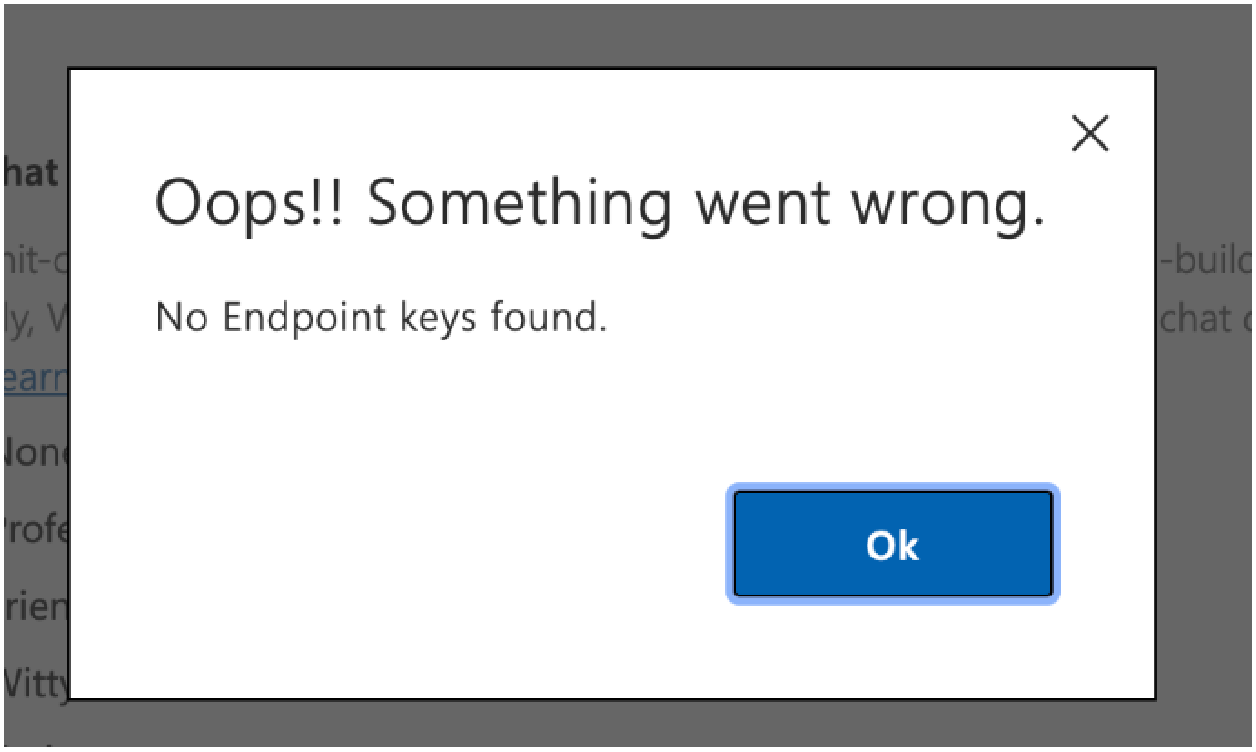
Create an Azure resource Group

https://docs.microsoft.com/en-us/azure/azure-resource-manager/manage-resource-groups-portal#create-resource-groups

Create a QnA maker bot

a. Create a knowledge base (KB) in the QnAMaker portal https://www.qnamaker.ai/
   You can use your Azure account in the QnA maker portal.
   This explains how to create the KB:
   https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/tutorials/create-publish-query-in-portal

   Note: To avoid this error when you create the KB:


Oops!! Something went wrong.

No Endpoint keys found.

Make sure you create all the services in the same region as the resource group.

For the "questions – answers" pairs to populate your KB, you may use this sample URL
https://www.microsoft.com/en-us/software-download/faq
Upload this document Sample multi-turn.docx
And choose a chat mode (enthusiastic for example).

After you created the KB, test it, save & train and publish.
Test it again.

b. Create the bot after you created the knowledge base:
https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/tutorials/create-qna-bot

Download the source code from the Azure Portal + Test it locally in the emulator

https://docs.microsoft.com/en-us/azure/bot-service/abs-quickstart?view=azure-bot-service-4.0#download-code

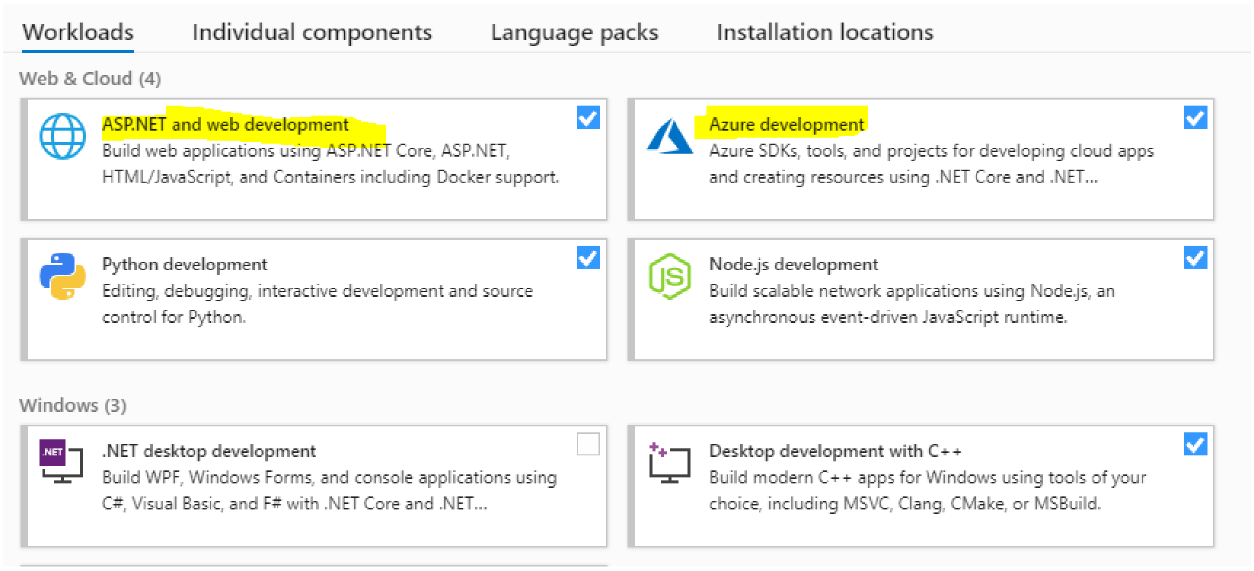
Open the project in Visual Studio and inspect it, to check the structure.

Download Visual Studio https://visualstudio.microsoft.com/downloads/ (free download)

You might have errors because the DLLs are missing.

If you have errors:

a) Install .NETCore SDK from Visual Studio: go to Tools → Get tools and features ASP.NET + Azure development



[✓] ASP.NET and web development
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

[✓] Azure development
Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET Core and .NET...

[✓] Python development
Editing, debugging, interactive development and source control for Python.

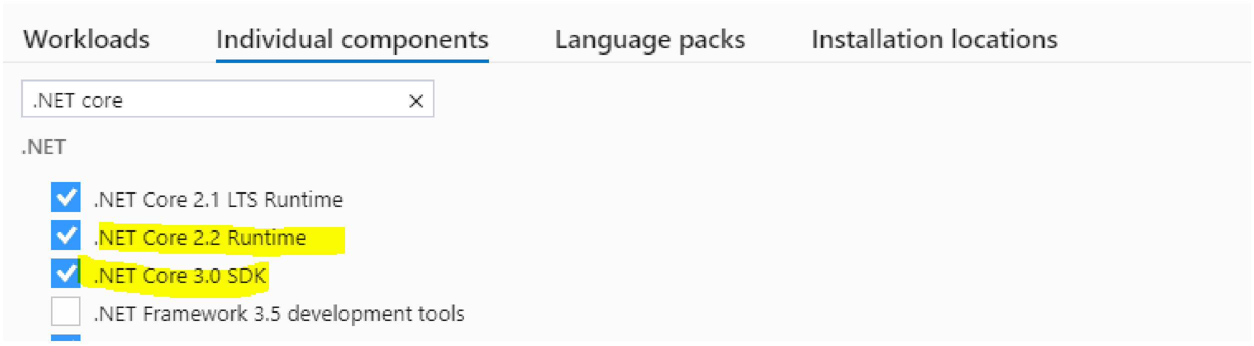
[✓] Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Windows (3)

[ ] .NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET Core and .NET...

[✓] Desktop development with C++
Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

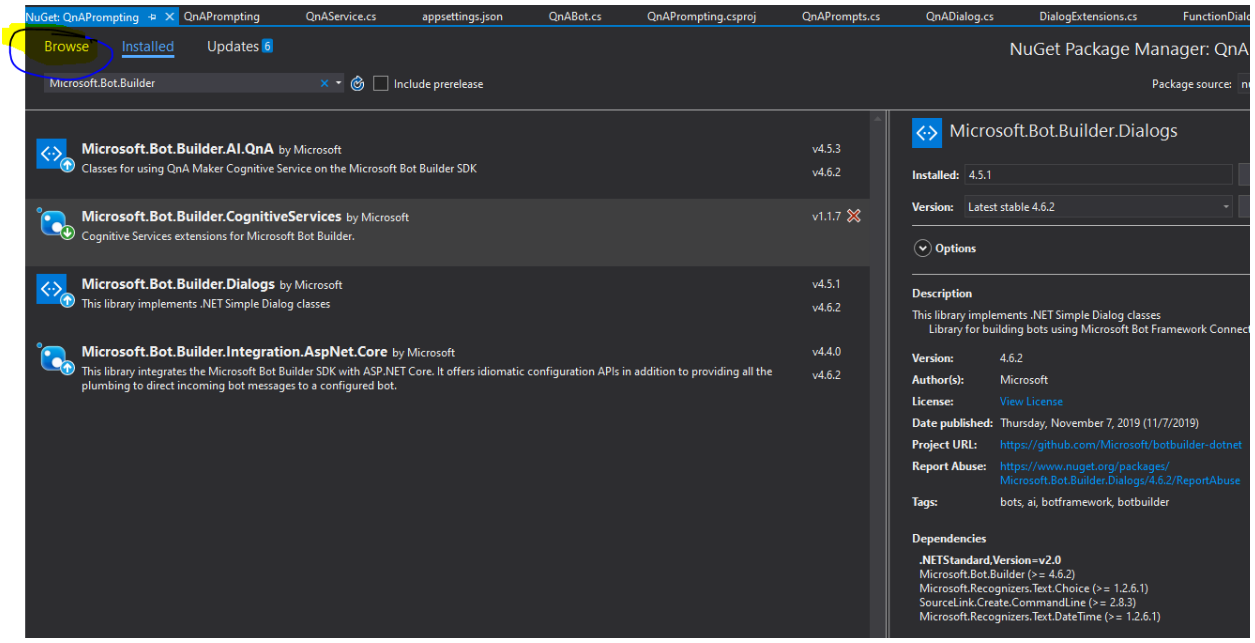
b) Then on individual components: make sure you have .NET 2.2 runtime and .NET Core 3.0 SDK



[✓] .NET Core 2.1 LTS Runtime
[✓] .NET Core 2.2 Runtime
[✓] .NET Core 3.0 SDK
[ ] .NET Framework 3.5 development tools

Press OK to install the features and restart Visual Studio.

c) press on Manage NuGet packages for the project to add the necessary libraries



Microsoft.Bot.Builder.AI.QnA by Microsoft — v4.5.3 / v4.6.2
Classes for using QnA Maker Cognitive Service on the Microsoft Bot Builder SDK

Microsoft.Bot.Builder.CognitiveServices by Microsoft — v1.1.7
Cognitive Services extensions for Microsoft Bot Builder.

Microsoft.Bot.Builder.Dialogs by Microsoft — v4.5.1 / v4.6.2
This library implements .NET Simple Dialog classes

Microsoft.Bot.Builder.Integration.AspNet.Core by Microsoft — v4.4.0 / v4.6.2
This library integrates the Microsoft Bot Builder SDK with ASP.NET Core. It offers idiomatic configuration APIs in addition to providing all the plumbing to direct incoming bot messages to a configured bot.

Microsoft.Bot.Builder.Dialogs

Installed: 4.5.1
Version: Latest stable 4.6.2

Description
This library implements .NET Simple Dialog classes
Library for building bots using Microsoft Bot Framework Connect

Version:	4.6.2
Author(s):	Microsoft
License:	View License
Date published:	Thursday, November 7, 2019 (11/7/2019)
Project URL:	https://github.com/Microsoft/botbuilder-dotnet
Report Abuse:	https://www.nuget.org/packages/Microsoft.Bot.Builder.Dialogs/4.6.2/ReportAbuse
Tags:	bots, ai, botframework, botbuilder

Dependencies
.NETStandard,Version=v2.0
Microsoft.Bot.Builder (>= 4.6.2)
Microsoft.Recognizers.Text.Choice (>= 1.2.6.1)
SourceLink.Create.CommandLine (>= 2.8.3)
Microsoft.Recognizers.Text.DateTime (>= 1.2.6.1)

d) Search for Microsoft.Bot.Builder under Browse section.

https://www.nuget.org/packages/Microsoft.Bot.Builder/

We also need these packages.

Microsoft.Bot.Builder.Integration.AspNet.Core

Microsoft.Bot.Builder.AI.QnA

Microsoft.Bot.Builder.Dialogs

e) Install the packages.

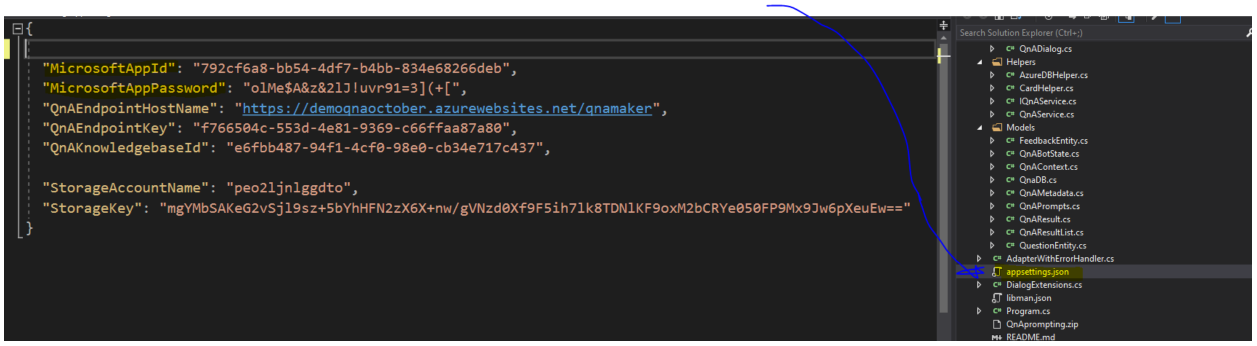
Test the bot into the emulator

a. Download the bot framework emulator https://github.com/Microsoft/BotFramework-Emulator/releases

b. Connect the emulator to the bot.

https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=csharp#connect-to-a-bot-running-on-localhost

B1. In the appSettings.json, you will find the App ID and password.



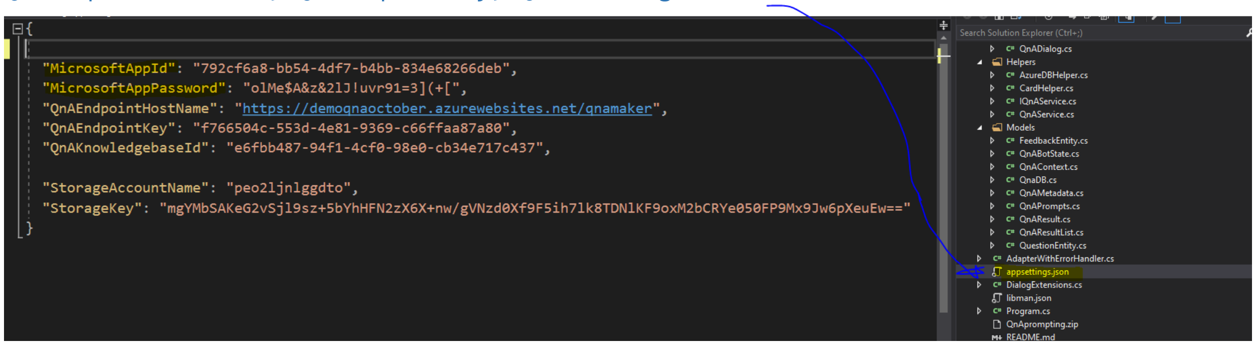
```
{

  "MicrosoftAppId": "792cf6a8-bb54-4df7-b4bb-834e68266deb",
  "MicrosoftAppPassword": "olMe$A&z&21J!uvr91=3](+[",
  "QnAEndpointHostName": "https://demoqnaoctober.azurewebsites.net/qnamaker",
  "QnAEndpointKey": "f766504c-553d-4e81-9369-c66ffaa87a80",
  "QnAKnowledgebaseId": "e6fbb487-94f1-4cf0-98e0-cb34e717c437",

  "StorageAccountName": "peo2ljnlggdto",
  "StorageKey": "mgYMbSAKeG2vSjl9sz+5bYhHFN2zX6X+nw/gVNzd0Xf9F5ih7lk8TDNlKF9oxM2bCRYe050FP9Mx9Jw6pXeuEw=="
}
```

B2. Copy them and place them into the emulator
B3. Then press Connect

Make some code changes to add prompts support

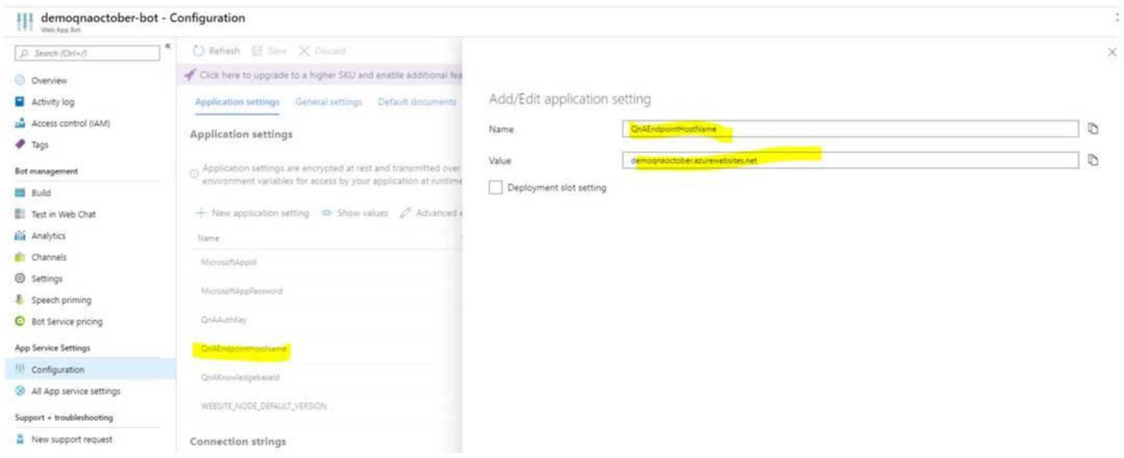
By default, in the QnA maker portal, prompts are shown.
In the bot framework, we do not have a template including prompts.
However, we have this sample project from GitHub.
https://github.com/microsoft/BotBuilder-Samples/tree/master/experimental/qnamaker-prompting/csharp_dotnetcore

a. Download the sample from GitHub by pressing Clone or download. Use the Download zip
b. Open the sln or csproj in Visual Studio.
c. Find the file appSettings.json.
d. Place the corect values in the fields `MicrosoftAppId, MicrosoftAppPassword, QnAEndpointHostName, QnAEndpointKey, QnAKnowledgebaseId`



```
{

  "MicrosoftAppId": "792cf6a8-bb54-4df7-b4bb-834e68266deb",
  "MicrosoftAppPassword": "olMe$A&z&21J!uvr91=3](+[",
  "QnAEndpointHostName": "https://demoqnaoctober.azurewebsites.net/qnamaker",
  "QnAEndpointKey": "f766504c-553d-4e81-9369-c66ffaa87a80",
  "QnAKnowledgebaseId": "e6fbb487-94f1-4cf0-98e0-cb34e717c437",

  "StorageAccountName": "peo2ljnlggdto",
  "StorageKey": "mgYMbSAKeG2vSjl9sz+5bYhHFN2zX6X+nw/gVNzd0Xf9F5ih7lk8TDNlKF9oxM2bCRYe050FP9Mx9Jw6pXeuEw=="
}
```

You find these values in the bot configuration:



Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application settings are exposed as environment variables for access by your application at runtime

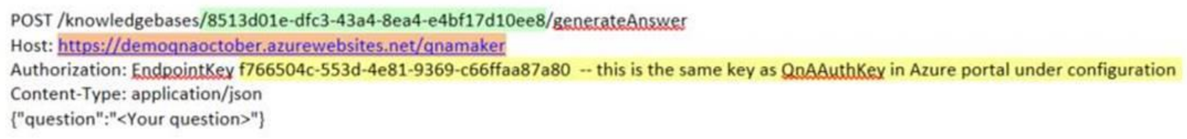
Name
MicrosoftAppId
MicrosoftAppPassword
QnAAuthKey
QnAEndpointHostName
QnAKnowledgebaseId
WEBSITE_NODE_DEFAULT_VERSION

Add/Edit application setting

Name	QnAEndpointHostName
Value	demoqnaoctober.azurewebsites.net

☐ Deployment slot setting

In the QnA Maker portal, when you edit the KB, you see the details are identical as in the bot Configuration:



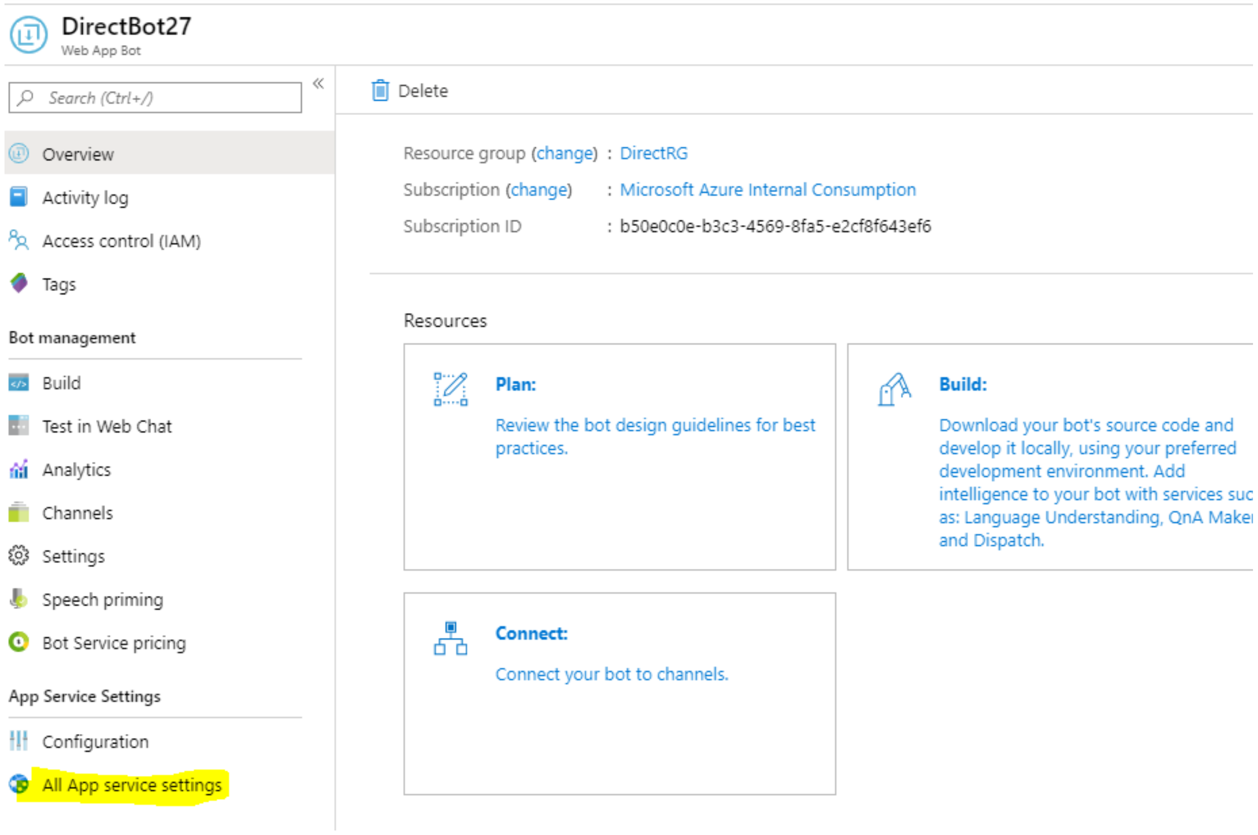
```
POST /knowledgebases/8513d01e-dfc3-43a4-8ea4-e4bf17d10ee8/generateAnswer
Host: https://demoqnaoctober.azurewebsites.net/qnamaker
Authorization: EndpointKey f766504c-553d-4e81-9369-c66ffaa87a80  -- this is the same key as QnAAuthKey in Azure portal under configuration
Content-Type: application/json
{"question":"<Your question>"}
```

e. Save the file.
f. Build the project and run it.
g. Connect the emulator to the bot.
   https://docs.microsoft.com/en-us/azure/bot-service/bot-service-debug-emulator?view=azure-bot-service-4.0&tabs=csharp#connect-to-a-bot-running-on-localhost
h. Test it into the emulator by sending a message.

Publish the bot back to Azure from Visual Studio

a. Download Publish profile

A1) In the Azure portal https://ms.portal.azure.com/ open the bot → go to All app service settings



DirectBot27
Web App Bot

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags

Bot management
Build
Test in Web Chat
Analytics
Channels
Settings
Speech priming
Bot Service pricing

App Service Settings
Configuration
All App service settings

Delete

Resource group (change) : DirectRG
Subscription (change) : Microsoft Azure Internal Consumption
Subscription ID : b50e0c0e-b3c3-4569-8fa5-e2cf8f643ef6

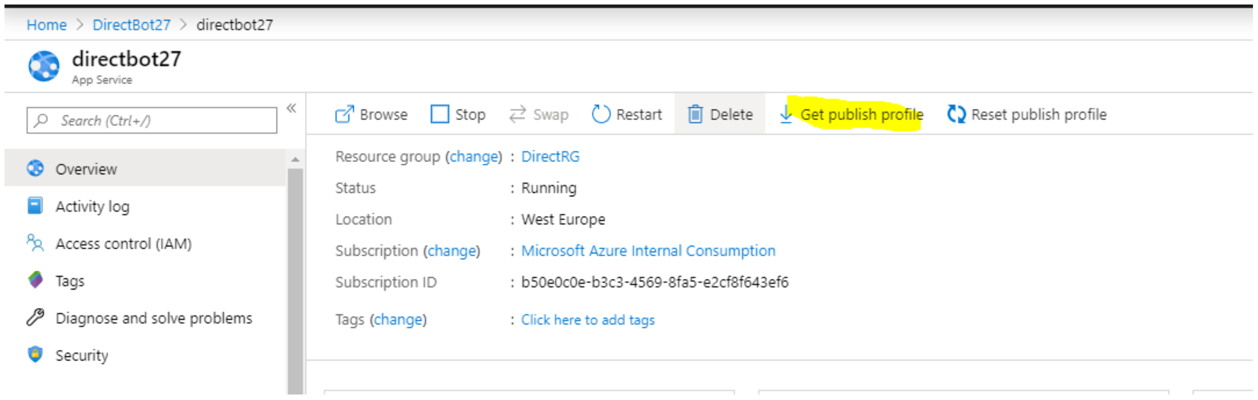
Resources

Plan:
Review the bot design guidelines for best practices.

Build:
Download your bot's source code and develop it locally, using your preferred development environment. Add intelligence to your bot with services suc as: Language Understanding, QnA Make and Dispatch.

Connect:
Connect your bot to channels.

A2) Press on the button Get publish profile



Home > DirectBot27 > directbot27

directbot27
App Service

Search (Ctrl+/)

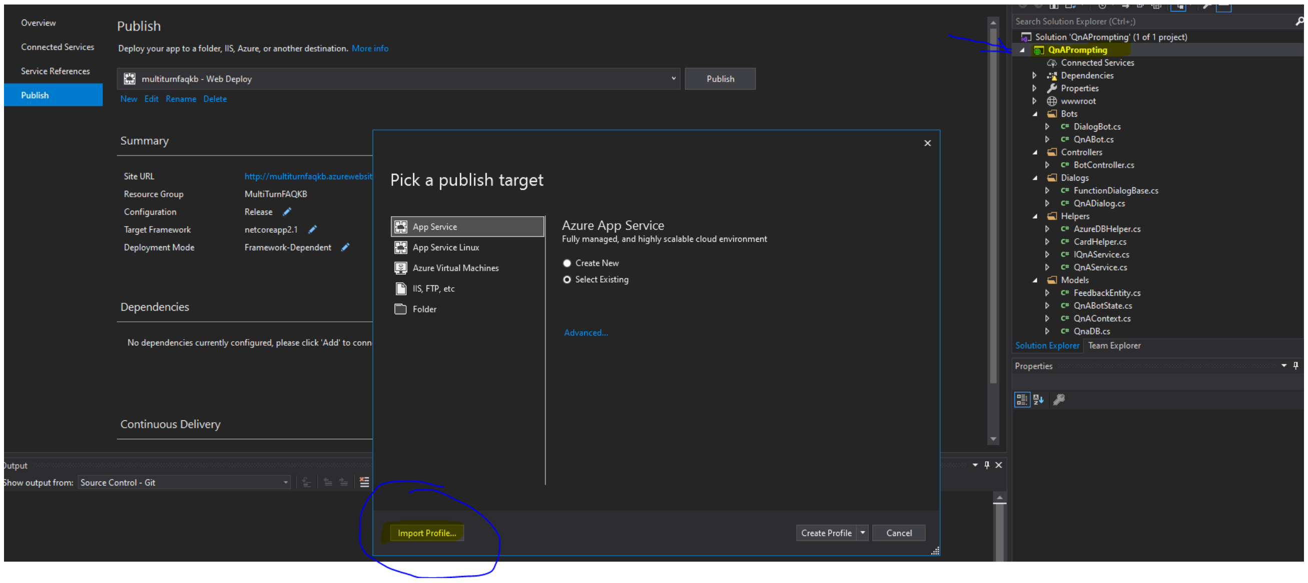
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Security

Browse | Stop | Swap | Restart | Delete | Get publish profile | Reset publish profile

Resource group (change) : DirectRG
Status : Running
Location : West Europe
Subscription (change) : Microsoft Azure Internal Consumption
Subscription ID : b50e0c0e-b3c3-4569-8fa5-e2cf8f643ef6
Tags (change) : Click here to add tags

b. Connect visual studio to Azure account https://docs.microsoft.com/en-us/azure-stack/user/azure-stack-install-visual-studio?view=azs-1908#connect-to-azure-stack-with-azure-ad

c. Right click on the solution → Publish and import the profile



Publish

Deploy your app to a folder, IIS, Azure, or another destination. More info

multiturnfaqkb - Web Deploy | Publish

Site URL	http://multiturnfaqkb.azurewebsit
Resource Group	MultiTurnFAQKB
Configuration	Release
Target Framework	netcoreapp2.1
Deployment Mode	Framework-Dependent

No dependencies currently configured, please click 'Add' to conn

Show output from: Source Control - Git

Azure App Service
Fully managed, and highly scalable cloud environment

○ Create New
● Select Existing

Solution 'QnAPrompting' (1 of 1 project)
QnAPrompting
DialogBot.cs
QnABot.cs
BotController.cs
FunctionDialogBase.cs
QnADialog.cs
AzureDBHelper.cs
CardHelper.cs
IQnAService.cs
QnAService.cs
FeedbackEntity.cs
QnABotState.cs
QnAContext.cs
QnaDB.cs

d. Test the bot in webchat after the publish is successful. It should reply in the chat window.