

# How to Build Your Bot



Russ Williams & Charlene Wahl

Microsoft

7/4/2019

## Table of Contents

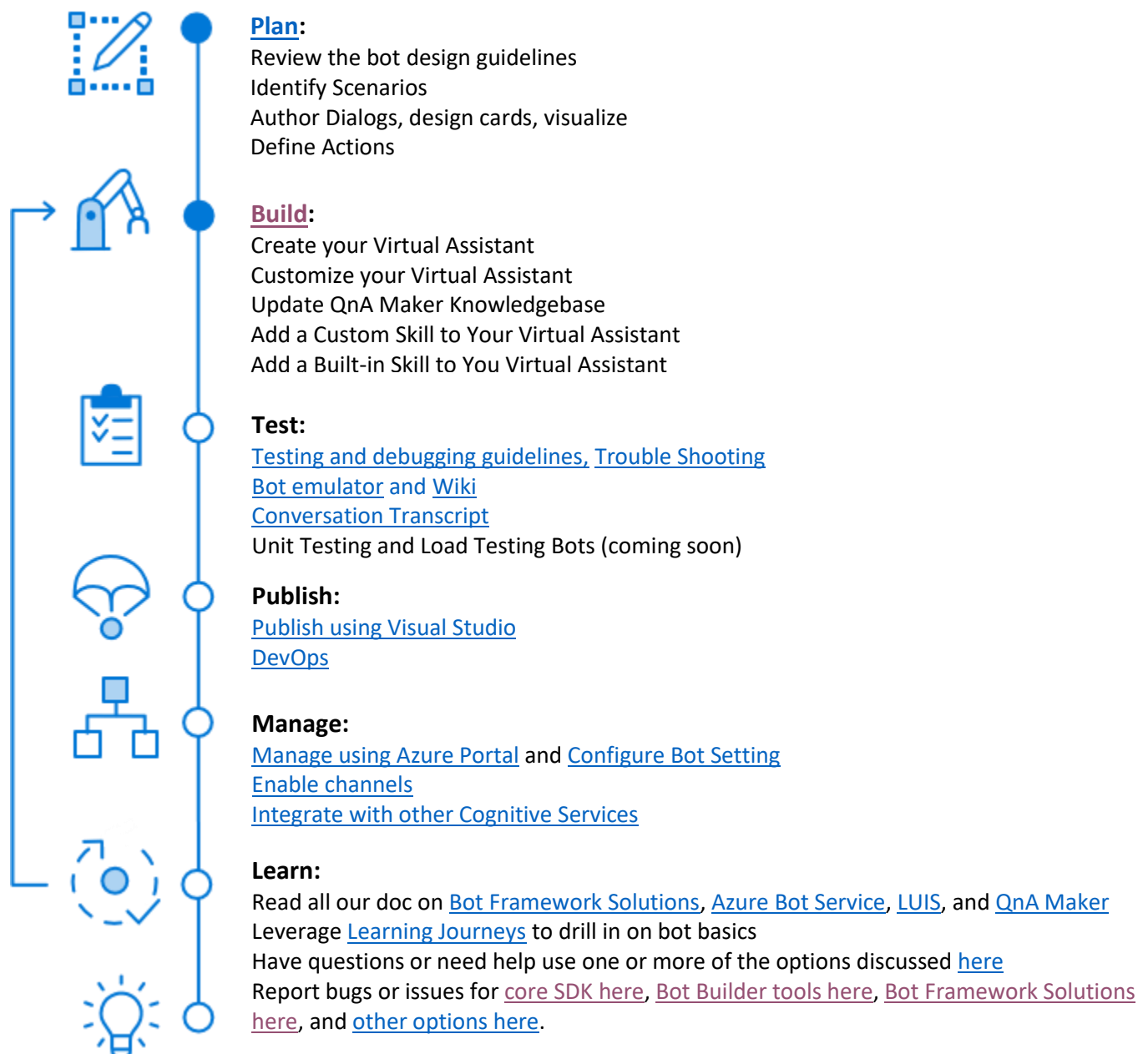
Overview .....	2
Plan .....	3
Review Design Guidelines .....	3
Identify Scenarios.....	3
Author Dialogs, Design Cards, and Visualize .....	3
Define Actions and Supporting Activities.....	4
Workshop Prerequisites.....	4
Build .....	5
Conversational AI fundamentals and motivation .....	5
Create, Deploy, and Run Your Virtual Assistant.....	5
Update NuGet Packages .....	5
Customize your Virtual Assistant .....	6
Update QnA Maker Knowledgebase.....	6
Collaborate.....	6
Add a Custom Skill.....	6
Update a Custom Skill .....	6
Add Multiturn QnA Prompts to Your Assistant.....	6
Add Multiturn QnA to Your New Virtual Assistant .....	7
Add a Built-in Skill to Your Virtual Assistant .....	7
What do these parameters mean? .....	8
Wrapping Up Build Stage .....	8
Changing endpoint .....	8
Adding secure web chat control .....	9
Configure DevOps .....	9
Unit Testing Bots (to complete CI/CD pipeline).....	9
Analyzing Bot Usage.....	10
Appendix – Important Links .....	11
Appendix – Publish Virtual Assistant or Skill using Visual Studio .....	12

## Overview

This workshop **IS NOT** about how to build **A BOT**, it's about how to build **YOUR BOT**! Think of it as a "one-stop shop" for building your first enterprise bot. Its part recipe, part master plan, which you can use to accelerate your bot journey from conception to production deployment. You'll use the **Plan** and **Build** links from the master plan to complete this workshop and then use the other links in the master plan as you continue to extend and refine your bot all the way through to production deployment.

# How to Build Your Bot

*Master Plan for Building a Virtual Assistant using the Microsoft Bot Framework*



## Plan

1. [Review Design Guidelines](#)
2. [Identify Scenarios](#)
3. [Author Dialogs, Design Cards, Visualize](#)
4. [Define Actions and Supporting Activities](#)
5. [Workshop prerequisites](#)

In the planning stage you'll take steps to prepare for an effective workshop that's been designed to accelerate the development of an enterprise grade bot. These steps do not require participants to have any prior experience with the Microsoft Bot Framework. Spending quality time preparing for the workshop is the best way to insure a successful outcome.

### Review Design Guidelines

Reviewing the bot design guidelines [here](#) will help beginners learn how to design bots that align with best practices and lessons learned by the Bot Framework team over the past several years.

### Identify Scenarios

Knowing what scenarios the bot will be expected to handle is as important to the bot as knowing a person's job responsibilities are for a human. Actually, thinking about the bot as a real person can be very helpful in discovering scenarios. When you hire a new employee, you tell them what you expect of them and what job duties they are expected to perform. Think of the bot as a human performing a specific role and job duties. What kinds of specific job duties do you want your bot to perform? The answer to that question will be your scenarios.

If your bot were performing the role of a bank teller, some of the specific job duties might be:

- Open bank account
- Close bank account
- Make deposit
- Make withdrawal
- Provide account balance

### Author Dialogs, Design Cards, and Visualize

Continuing with this idea of thinking about the bot as a person, and having identified one or two key scenarios, it would be only natural to now wonder how each scenario should be handled. What would a conversation for each scenario look like? What kind of back and forth would be necessary for someone to gather enough information to carry out the goal of the scenario? In Conversational AI parlance, we call that back and forth a "dialog". Dialog in this context means the conversation between the user and the bot, not a rectangle form on a computer screen.

The easiest way to model and design a dialog for a scenario is to simply jot it down and label each sentence with the name of who spoke those words. So, it might be something like:

- Bot: Hello, I'm Angie, your Contoso Bank teller
- Bot: I can help with various things like opening or closing an account, making deposits or withdrawals, or getting an account balance.

- Bot: What can I help you with today?
- User: I'd like to open an account
- Bot: Super, let's do that! I'll need to ask a few questions first, but it won't take long.
- Bot: What is your name?
- User: Russ Williams
- Bot: Russ, what is your email?
- etc., etc., etc.
- Bot: Alright, that's all we need to setup your account Russ. You'll receive a confirmation email shortly to keep for your records
- Bot: What else can I help you with?

The Microsoft Bot Framework include a set of [bot-builder tools](#) that make various aspects of developing bots easier. One of those is the [Chatdown tool](#) which can be used to create a rich mockup of a dialog. You can use this tool in cases where you'd like to share a realistic "design comp" to internal business owners and stakeholders to get feedback and approval before spending the effort to build it. For dialogs that only require simple answers (e.g. text answers) for each step in the conversation, the Chatdown tool might not provide any more value than the jot-down technique described previously. But for right dialogs that will flow things like dropdown controls, radio boxes, buttons, and carousel cards, the Chatdown tool is very helpful in capturing that. Note: Not all channels support rich dialogs (i.e. dialogs with non-text conversation) so you'll want to consider that when designing your dialogs.

### Define Actions and Supporting Activities

Actions carry out the intent of the scenarios. The goal of the dialog for a scenario is to gather enough information to be able to carry out its intent which you can think of as an action. These actions will be integration points between the conversational AI of your bot and the application backend that will execute that task.

Some scenarios will also require support activities that use the current context of the conversation to gather intermediate information that's required by the next step in the conversation. These supporting activities are also integration points between the conversation AI of your bot and the application backend that will provide that intermediate information.

Early on in the development of your bot its wise to simply mock these integration points so that you can take an agile approach and quickly iterate over your scenarios as you tune the conversation. Integration points are always challenging and generally requires a lot of effort to implement so mocking them will allow you to concentrate on getting the conversation right without requiring the integration points to be constantly reworked.

### Workshop Prerequisites

A critical step in preparing for the workshop is installing the bot framework SDK and tools. There are two sets of prerequisites that have to be installed and those instructions have to be followed exactly or you will experience strange errors later when you try to build your bot and it won't be obvious that the errors you are seeing are a result of improper installation.

The prerequisites that need to be installed for the Virtual Assistant can be found [here](#) and the second set of installation steps for Skills can be found [here](#) but, for Skills, you only need to do step 1 where you

install the Skills template since the rest of those installation steps are redundant. If you find yourself unsure of what to do in any of the installation steps, you can look at [this short screen capture](#) that shows the whole process end-to-end.

Although not exactly necessary, it's never a bad idea to update Visual Studio which you can do by launching it and choosing **Help | Check for Updates**.

## Build

1. [Conversational AI fundamentals and motivation](#)
2. [Create, deploy, and run your Virtual Assistant](#)
3. [Customize your Virtual Assistant](#)
4. [Update QnA Maker Knowledgebase](#)
5. [Add a Custom Skill](#)
6. [Update a Custom Skill](#)
7. [Add Multiturn QnA Prompts to Your Assistant](#)
8. [Add a Built-in Skill to Your Virtual Assistant](#)
9. Add Personal Assistants (Cortana, Alexa, Google Home) – Coming soon
10. [Wrapping Up Build Stage](#)

In this stage of bot lifecycle, you'll build a custom enterprise baseline for your new Virtual Assistant. When you finish the steps below, you'll have a core scenario working and all the development skills you'll need to complete your bot. Simply rinse and repeat to incrementally add remaining bot functionality.

### Conversational AI fundamentals and motivation

Before jumping into details of building your bot, it's important to first have an understanding of the big picture. Your workshop guide/instructor will take a few minutes to go over the fundamentals of Conversational AI and the Microsoft Bot Framework. You can also take a look at overviews of [Conversation AI](#), [Conversational AI Tools](#), [Azure Bot Service](#), [Virtual Assistant](#), and [Microsoft Bot Framework](#).

### Create, Deploy, and Run Your Virtual Assistant

Follow the instructions [here](#) and although not exactly necessary, it's a good idea to update the NuGet packages for each project in the solution (including the Skills projects you'll create later). You can update the NuGet packages with these steps:

#### Update NuGet Packages

1. Open NuGet Manager for Assistant project by right-clicking the project and choosing **Manage NuGet packages...**
2. Select the **Updates** tab and then type in "Microsoft.bot" in the **Search** field to filter out all non Bot Framework packages
3. Update these packages first and in this order to avoid import errors: Schema, Connector, Builder, Configuration
4. Click the Select

**Note:**

PowerShell Core get installed to \$env:ProgramFiles\PowerShell\<version>\pwsh.exe but you can run it from old PowerShell command line by entering pwsh.exe. If you run it once it will then show up in Cortana search term completion from then on.

## Customize your Virtual Assistant

Follow the instructions [here](#)

## Update QnA Maker Knowledgebase

Follow the instructions [here](#)

**Note:** Over time, as you develop your assistant, you might decide to add additional QnA Knowledgebases (or refactor your current Knowledgebase into multiple Knowledgebases) and the instructions for how to do that are located [here](#)

~~Note: There is a bug in the current documentation so follow these steps instead.~~

- ~~1. Open PowerShell Core 6~~
- ~~2. Change directory to the project directory of your Virtual Assistant~~
- ~~3. Run this command:~~  
~~**dispatch refresh**~~
- ~~4. Run the app and use Bot Emulator to verify new QnA pairs are working~~

## Collaborate

You can setup QnA Maker so that several people can collaborate on a single QnA Maker KB by following the instructions [here](#).

## Add a Custom Skill

Follow the instructions [here](#)

For detailed instruction on adding a skill review [this](#)

## Update a Custom Skill

Follow the instructions [here](#)

Detailed

## Add Multiturn QnA Prompts to Your Assistant

**[Important!!!!]** *The Virtual Assistant Template and bot framework tools have not yet been updated to respect multiturn follow-on prompts so the following is a temporary workaround and some follow-on prompts will not work properly. You can use this to explore follow-on prompts in your assistant, but the SDK will eventually be updated and obviate this workaround. Bottom line – this code should be removed before the assistant is deployed in production or once the SDK has been updated. When the SDK is updated, this section will be updated with the proper instructions on how to incorporate follow-on prompts]*

In this next topic we'll use follow-up prompts to create a multiturn QnA conversation as described [here](#).

## Add Multiturn QnA to Your New Virtual Assistant

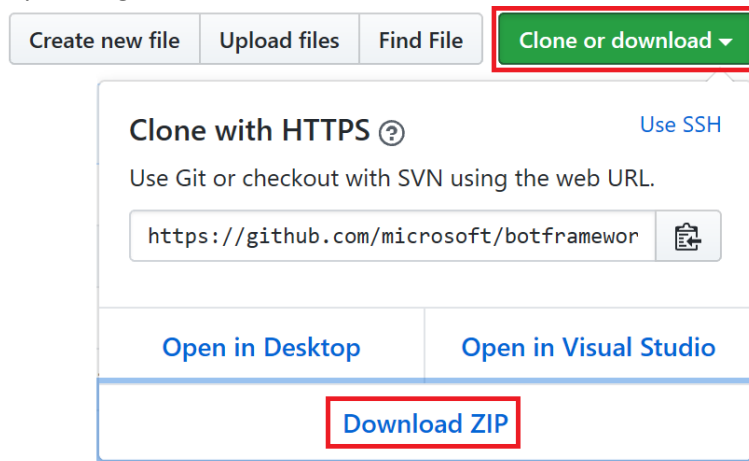
The new QnA Maker Follow-on prompts do not automatically appear in bot clients, you must add code to make that happen. This topic will show you how to add the ability to show the new QnA Maker follow-on prompts in your new assistant. Follow the steps described [here](#) skipping the first step related to creating and deploying your assistant since you've already done that.

## Add a Built-in Skill to Your Virtual Assistant

The Microsoft Bot Framework includes several built-in skills that can be added to your bot which can be found [here](#). This is an optional task but if you'd like to get a feel for how this works you might consider adding the To Do Skill since it's useful and straight forward.

Follow these steps to install the To Do Skill:

1. Browse to the Bot Framework Solutions Repository [here](#) and clone it to your development PC by clicking the **Clone or download** button and then choose **Download ZIP**.



2. Extract the ZIP to your local hard drive and then copy the **botframework-solutions-master\skills\src\csharp\todoskill** folder and paste it into the root folder of your Virtual Assistant solution (i.e. the folder that holds the **.sln** file)
3. Open your Virtual Assistant solution in Visual Studio and right-click the solution in the Solution Explorer and choose **Add | Existing Project...** and add **todoskill\todoskill\ToDoSkill.csproj**
4. Add the **todoskill\todoskilltest\ToDoSkillTest.csproj** the same way you did in the last step
5. Now we'll deploy the todoskill.  
Open PowerShell Core 6 and run the following command to temporarily set the execution policy to **Bypass** for the current PowerShell session. If this is not done, you'll get an error for attempting to run a script that is not digitally signed. When the PowerShell session ends the setting reverts.

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

6. Deploy the todoskill by opening PowerShell Core 6 and change directory to todoskill project folder (**todoskill\todoskill**) and run the following command:

```
.\Deployment\Scripts\deploy.ps1
```



What do these parameters mean?

Parameter	Description	Required
name	<b>Unique</b> name for your bot. By default this name will be used as the base name for all your Azure Resources and must be unique across Azure so ensure you prefix with something unique and <b>not</b> <i>MyAssistant</i>	<b>Yes</b>
location	The region for your Azure Resources. By default, this will be the location for all your Azure Resources	<b>Yes</b>
appPassword	The password for the <a href="#">Azure Active Directory App</a> that will be used by your bot. It must be at least 16 characters long, contain at least 1 special character, and contain at least 1 numeric character. If using an existing app, this must be the existing password.	<b>Yes</b>
luisAuthoringKey	The authoring key for your LUIS account. It can be found at <a href="https://www.luis.ai/user/settings">https://www.luis.ai/user/settings</a> or <a href="https://eu.luis.ai/user/settings">https://eu.luis.ai/user/settings</a>	<b>Yes</b>

7. Now follow same instructions for adding a custom skill skipping to the Test Your Skill step found [here](#)

## Wrapping Up Build Stage

### Changing endpoint

You will want to keep the [key limits of the Authoring Key](#) in mind when using it in development. When you use the Authoring Key to access the LUIS prediction endpoint you are limited to 1,000 calls per month. This has bit me more than once where I [ran out of monthly quota](#) when using the Authoring Key for development. Once I ran out of quota right in the middle of a customer demo!

Remember, [LUIS Authoring keys and not the same thing as LUIS prediction endpoint keys](#). The LUIS Authoring Key is meant for authoring scenarios like creating and modifying your LUIS app whereas the LUIS prediction endpoint is used for querying your LUIS app for Intent recognition. So two totally separate keys.

The problem here is that the Authoring Key (with its 1,000 query limit) is assigned to the LUIS prediction endpoint by default when a LUIS app is created so Authoring quota is used on every call to the endpoint during development. For the uninitiated bot developer, they are unaware that calls to their newly

created bots from the Bot Emulator are actually hitting against this quota right from the get go. On an active project it's not actually hard to hit the 1,000 query Authoring limit (as I mentioned I had at the worst possible time).

To make matter worse, because of the defaulted Authoring Key, developers are able to build bots connected to LUIS apps without ever understanding this Authoring Key dynamic so the first time they see the "out of quota error" in the form of an HTTP 403 or 429, they have no clue what has happened or how to fix it. Have you ever tried to Bing for a solution where you weren't even sure of what the problem is? Not pretty and certainly not a quick resolution.

IMHO, understanding the roles of the Authoring Key and the Endpoint Key and the import of how they are initially assigned is very difficult since its spread over several different document pages and the most important doc that shows how to actually create an endpoint and assign the key is buried in a ["Quick Start Deployment" focused page](#).

The key thing to remember here is that **LUIS prediction endpoints are created in the Azure portal** and their **endpoint keys must then be assigned to the LUIS app in the LUIS portal**.

I am no longer using authoring/starter key for projects that will last longer than a single demo. It's a [bit of a pain to have add a subscription key](#) but well worth it and DEFINITELY something that needs to be communicated to customers.

Adding secure web chat control

Coming soon

Configure DevOps

Setting up DevOps as early as possible in development is critical to modern development and you can establish a core DevOps Continuous Deployment baseline by following the instructions [here](#). As your bot development progresses, you'll need to mature your DevOps pipeline to target new deployment environments that might not exist early on. Having a core DevOps baseline will allow you to incrementally automate bot specific processes and discipline in a very pragmatic fashion.

<https://channel9.msdn.com/Series/DevOps-for-the-Bot-Framework> which has pointers to all stages.

Unit Testing Bots (to complete CI/CD pipeline)

<https://www.microsoft.com/developerblog/2017/01/20/unit-testing-for-bot-applications/>

<https://blogs.msdn.microsoft.com/kenakamu/2017/06/20/devops-with-bot-framework-chat-for-business-application-1-en/>

<https://blogs.msdn.microsoft.com/jamiedalton/2017/03/08/devops-with-bots-cicd-pipeline-with-the-botframework-and-azure-end-to-end-walkthrough/>

## Analyzing Bot Usage

One of the most critical phases in the life of a bot is the Evaluation stage and a key part of that stage is analyzing bot usage, but because this workshop ends at the Build stage and because analyzing bot usage can be a useful tool during Usability Testing (i.e. testing with representative users or internal project stakeholders and business owners) we include a discussion of it here.

- [Bot Analytics](#)
- [Overview and usage](#) of the [Bot Framework Power BI Template](#)
- [Custom Telemetry](#)
- [Advanced Custom Insights](#)
- Active Learning for [QnA Maker](#) and [LUIS](#)

## Appendix – Important Links

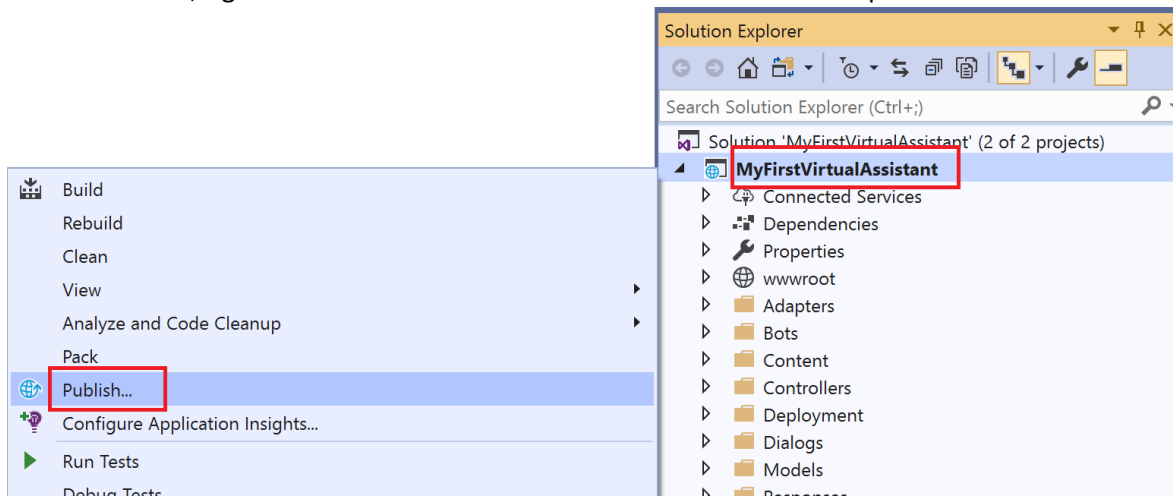
Bot Framework Solutions Repository for Conversational Assistant

- <https://github.com/Microsoft/botframework-solutions>

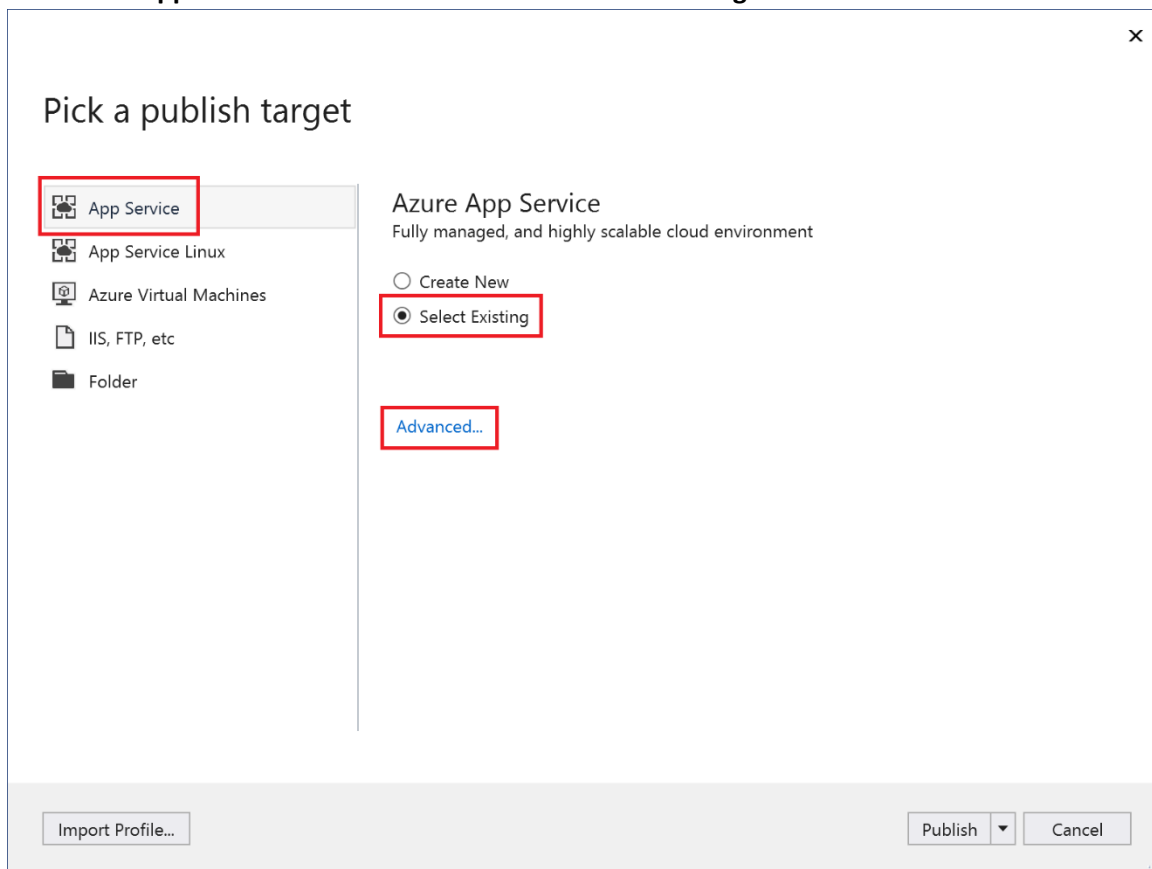
## Appendix – Publish Virtual Assistant or Skill using Visual Studio

Publishing a Virtual Assistant using Visual Studio is slightly different than what most developers are used to doing when they use the Visual Studio Publishing Wizard since the App Service has already been created. This means the flow changes to allow for the existing App Service to be selected in the publishing wizard rather than created as is normally the case. Here are the steps.

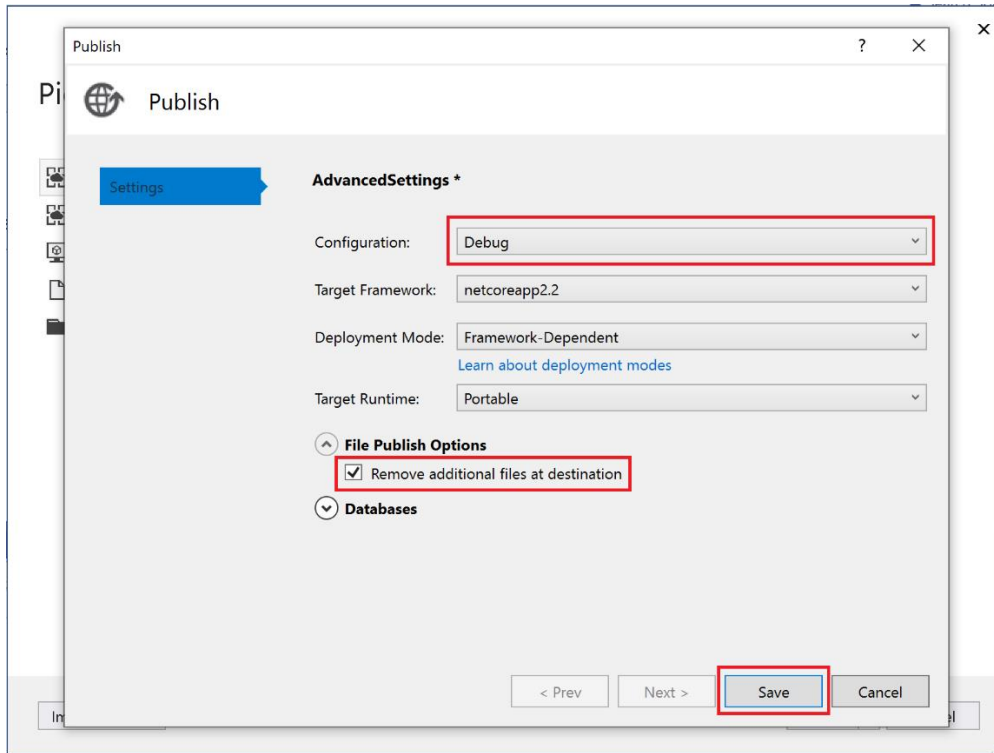
1. In Visual Studio, right-click the Virtual Assistant or Skill in the Solution Explorer and choose **Publish**



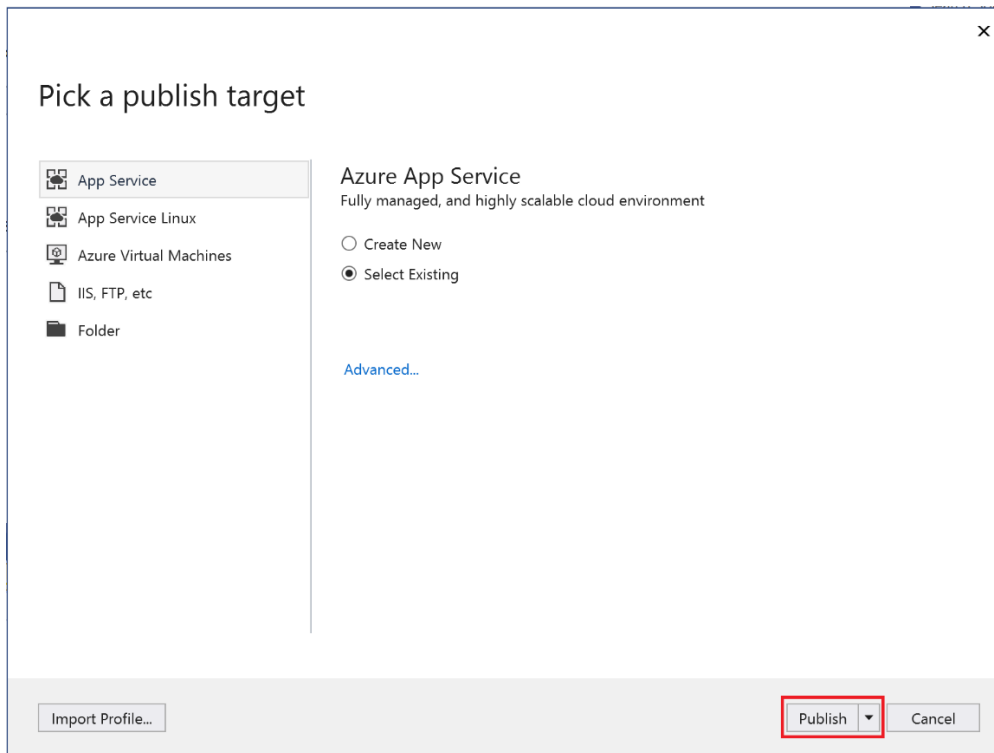
2. Make sure **App Service** is selected then choose **Select Existing** and then click the **Advanced...** link



3. Select **Debug** for the *Configuration* setting and expand *File Publish Options* and check **Remove additional files at destination** to insure there are no pesky leftover after any future deployment and then click **Save**.



4. Now click **Publish**



5. Select your Azure account and subscription then expand the App Service folder that contains the Virtual Assistant App Service you want to publish to and then click **OK** and the publishing process will begin. You can open the View | Output window to watch the progress and check to make sure it completes successfully.

