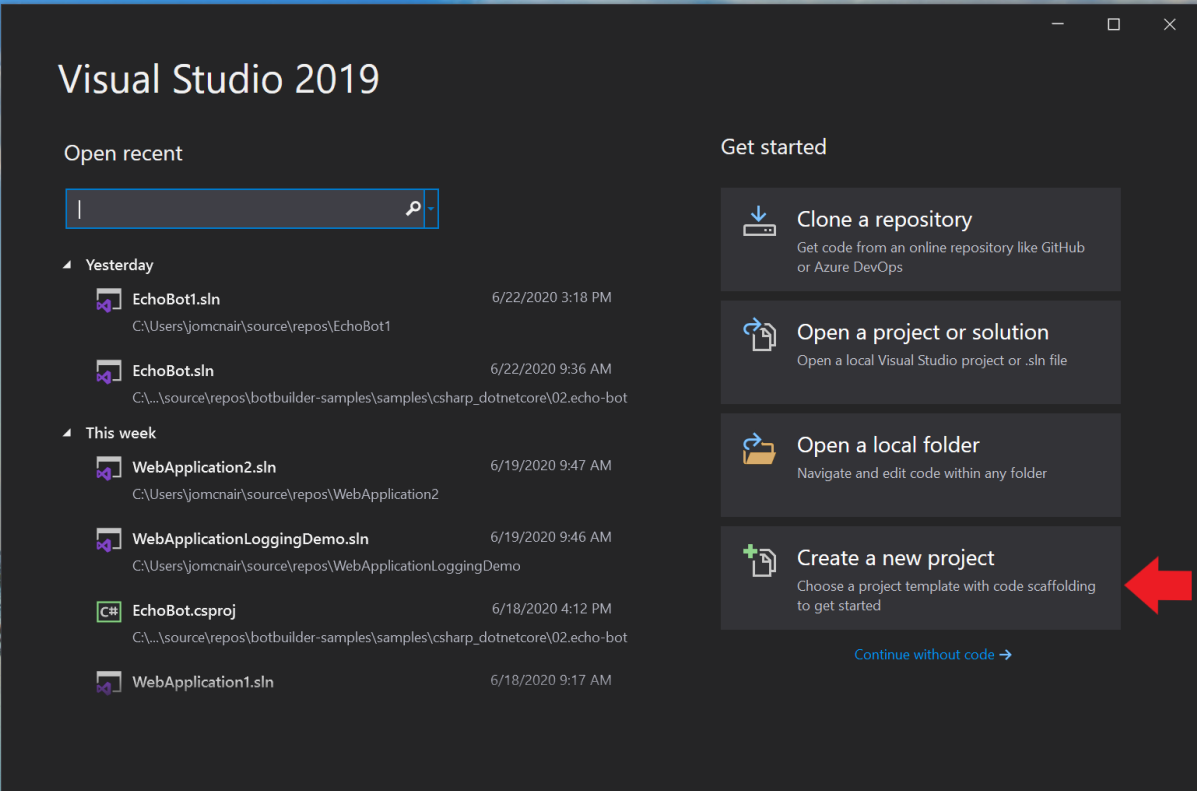
**Create an Azure Echo Bot in Visual Studio.**

## Prerequisites

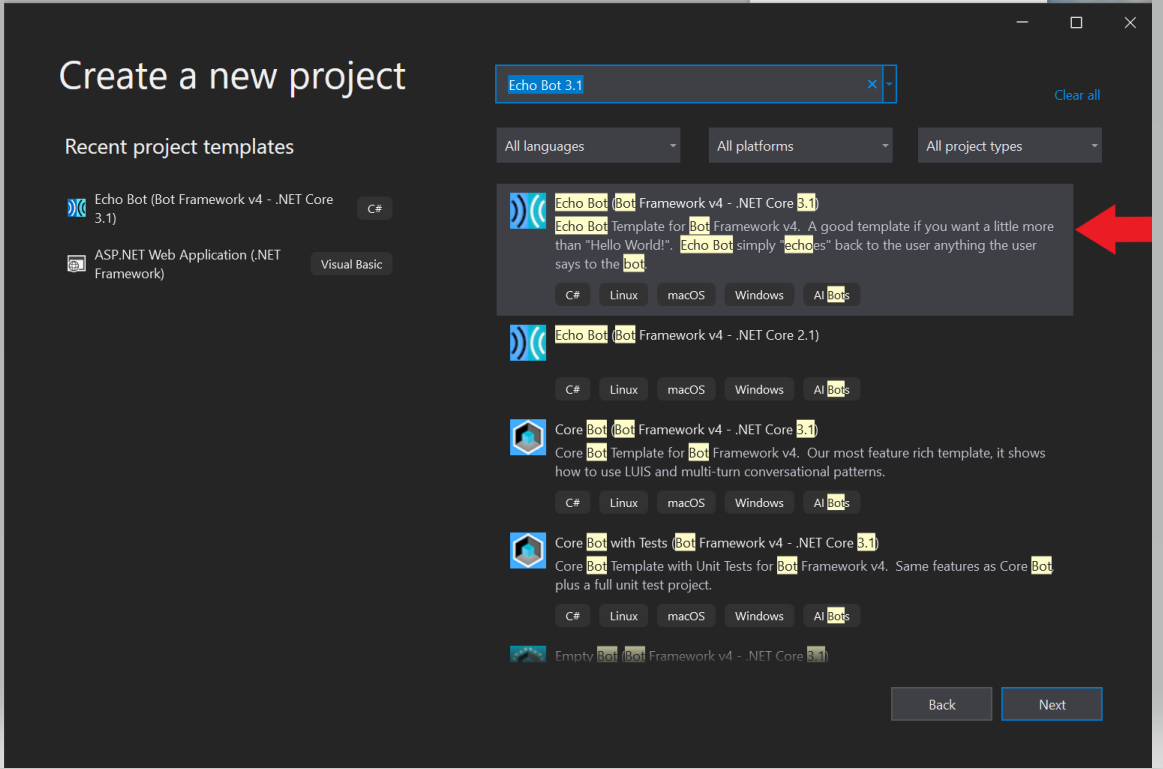
* [Visual Studio 2019 or later](https://www.visualstudio.com/downloads)
* [Bot Framework SDK v4 template for C#](https://aka.ms/bot-vsix)
* [Bot Framework Emulator](https://aka.ms/bot-framework-emulator-readme)
* Knowledge of [ASP.Net Core](https://docs.microsoft.com/en-us/aspnet/core/) and [asynchronous programming in C#](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/)

Install [BotBuilderVSIX.vsix template](https://aka.ms/bot-vsix) that you downloaded in the prerequisites section.

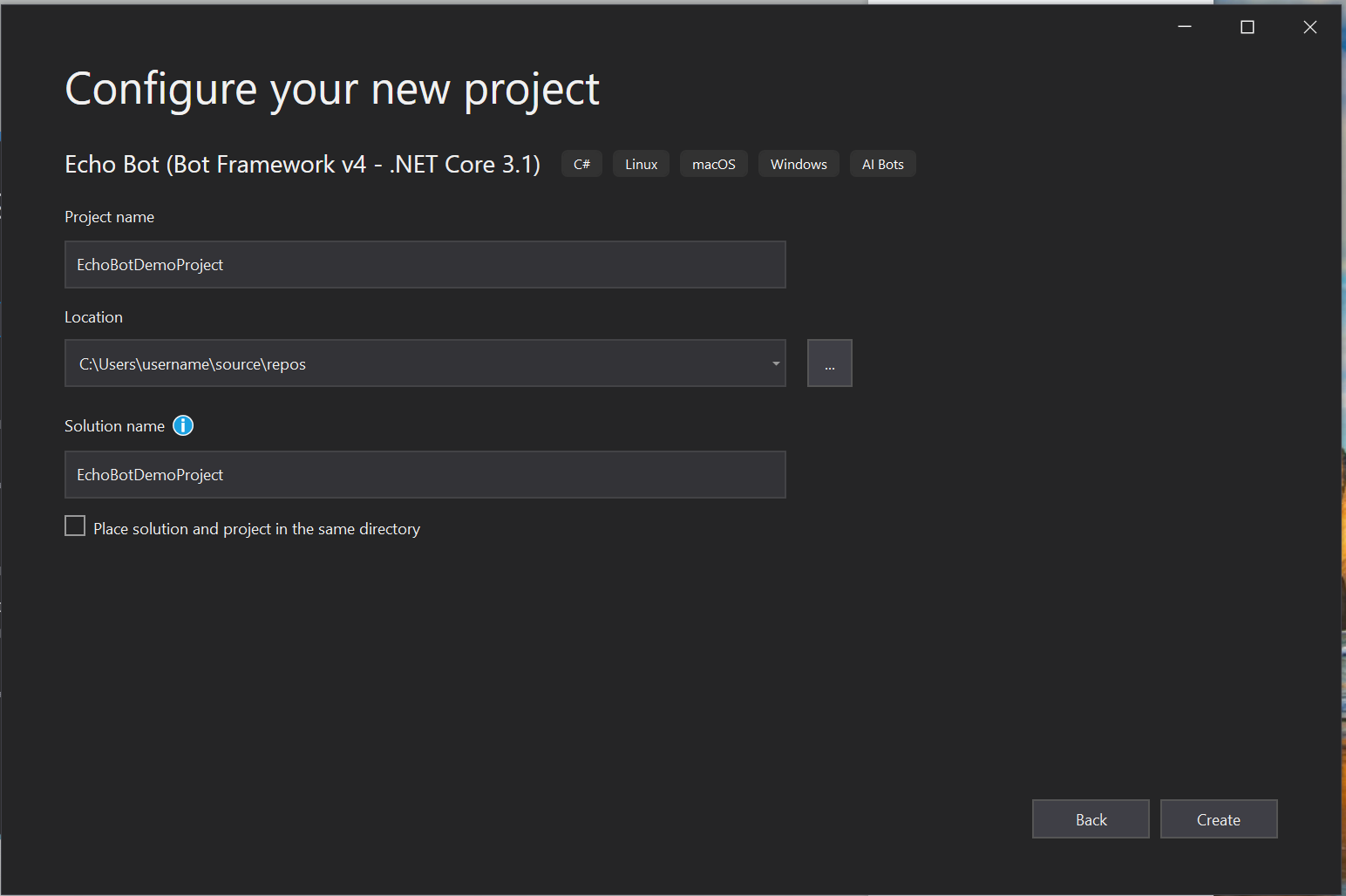
In Visual Studio, create a new bot project using the **Echo Bot (Bot Framework v4 - .NET Core 3.1)** template. Choose **AI Bots** from the project types to show only bot templates or use the search feature and type “**Echo Bot 3.1**”.



Select the appropriate project template and click “**Next**”

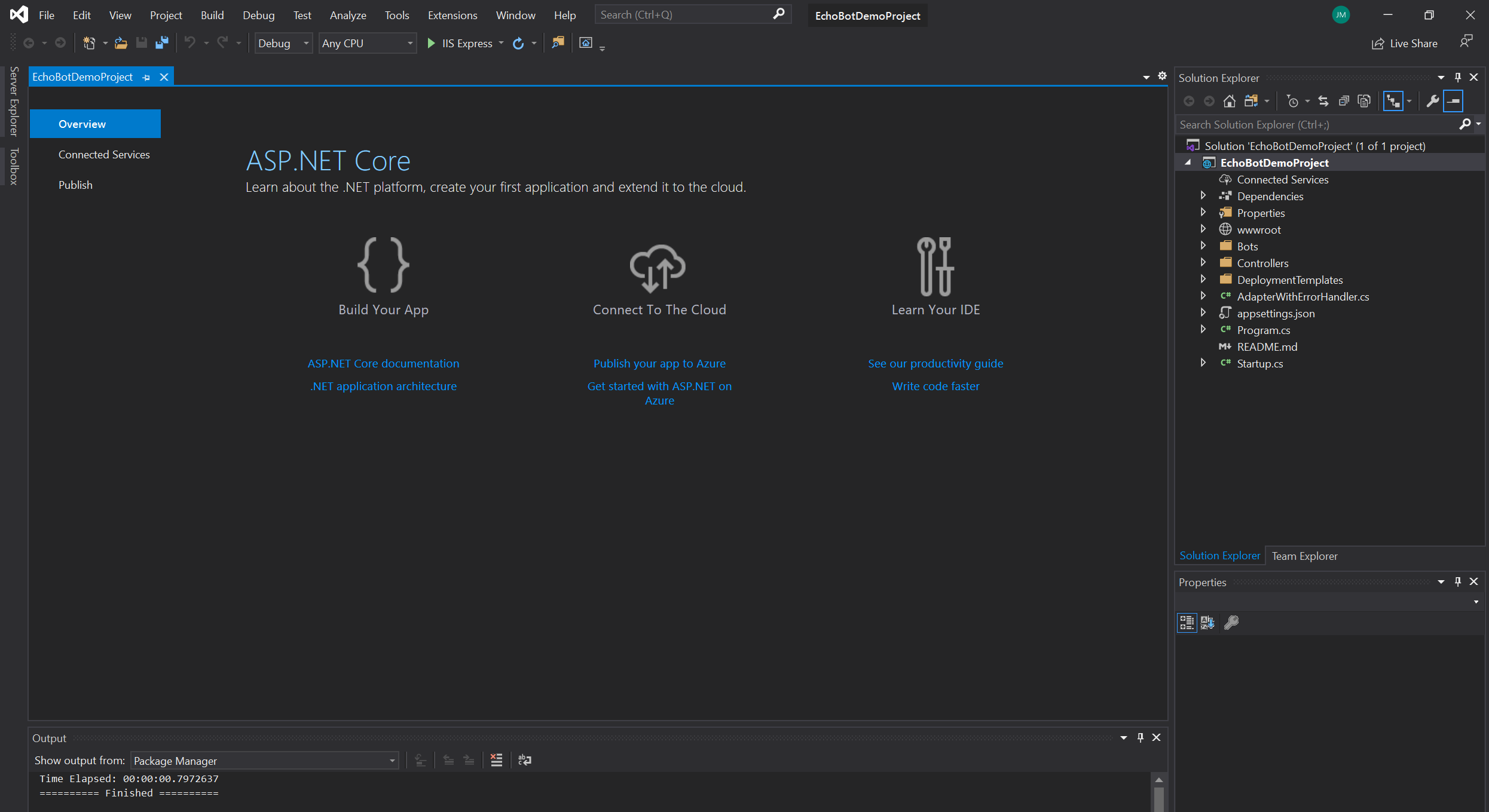


Name the project and click “**Create**”

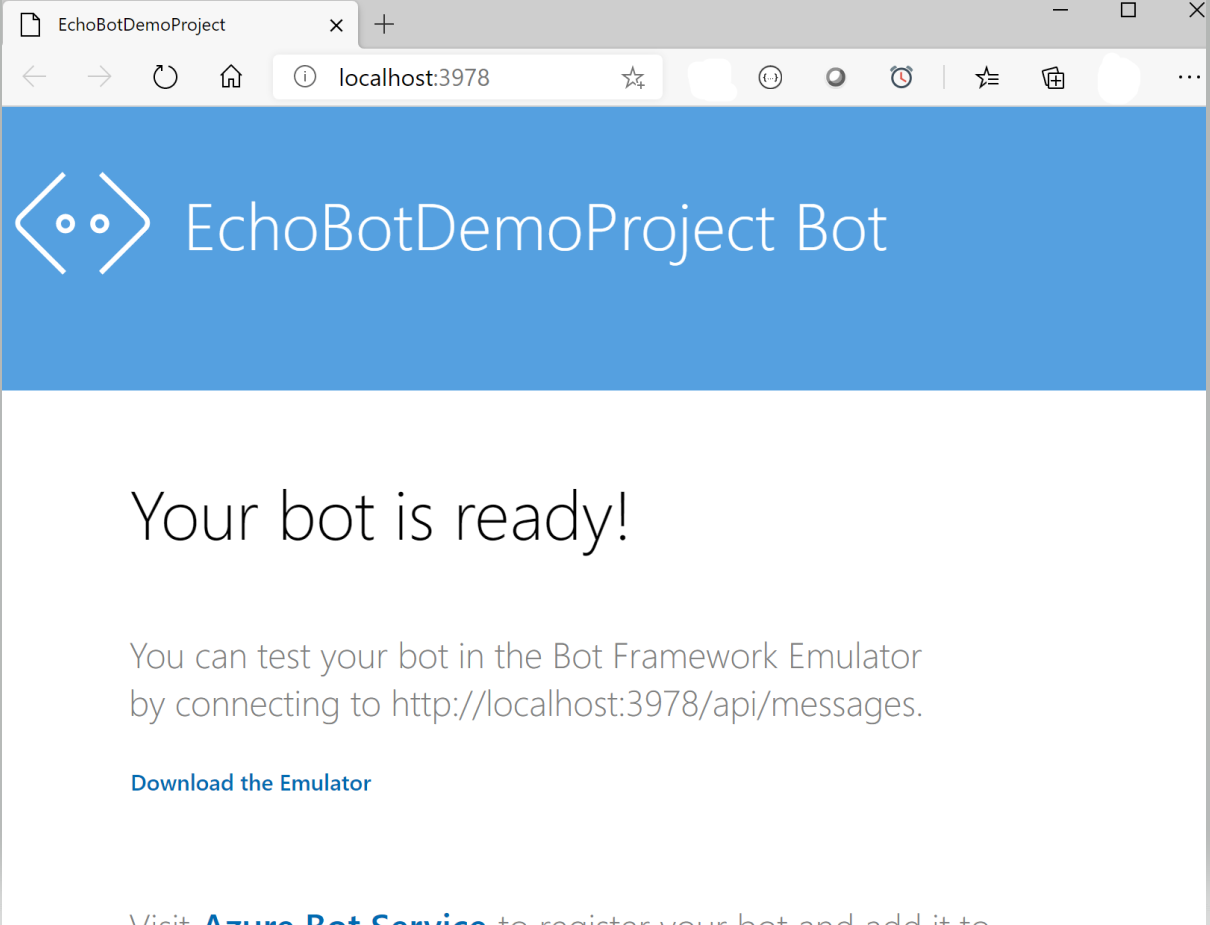


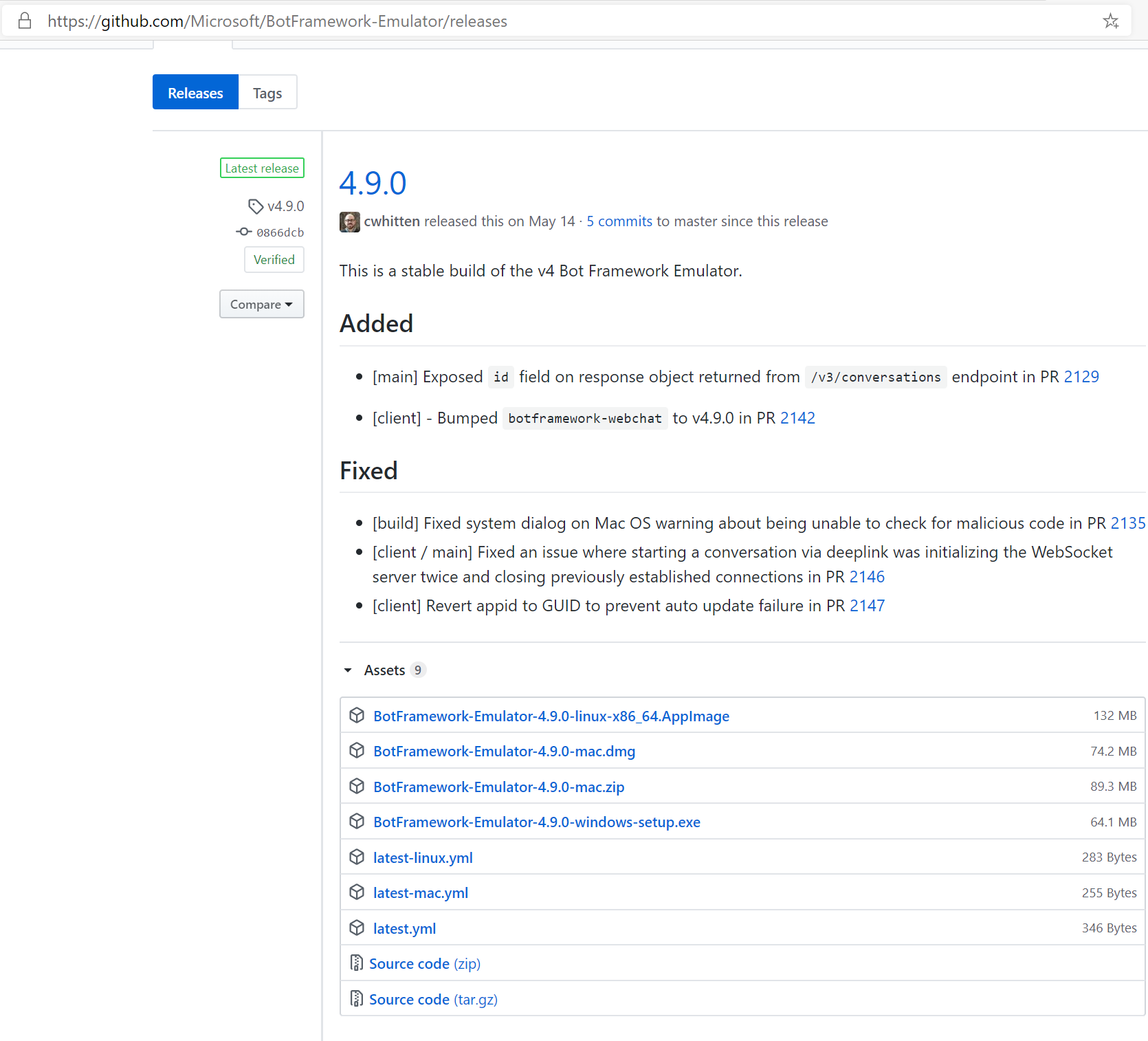
Now that our project is loaded in Visual Studio, we can begin testing our bot locally first.

Click on “**IIS Express**” or click “**F5**” in Visual Studio to start debug mode locally.

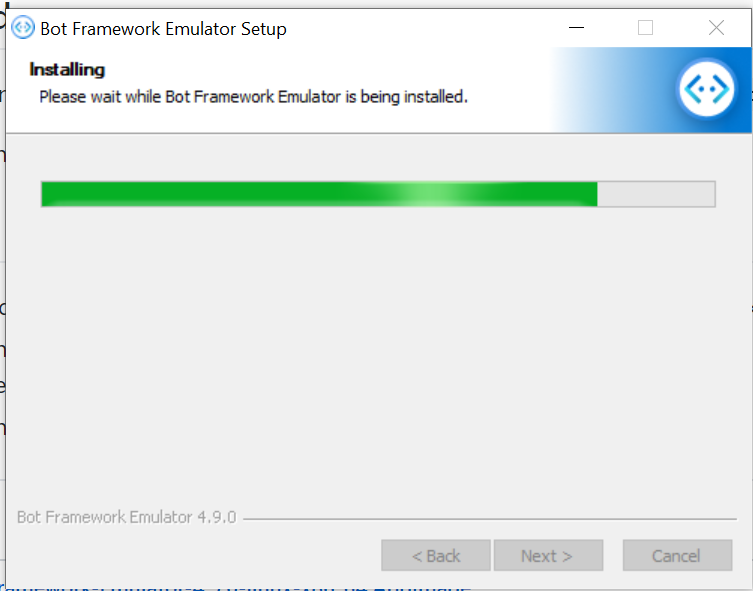


This should result in the program running a browser session on port 3978 in a new window.

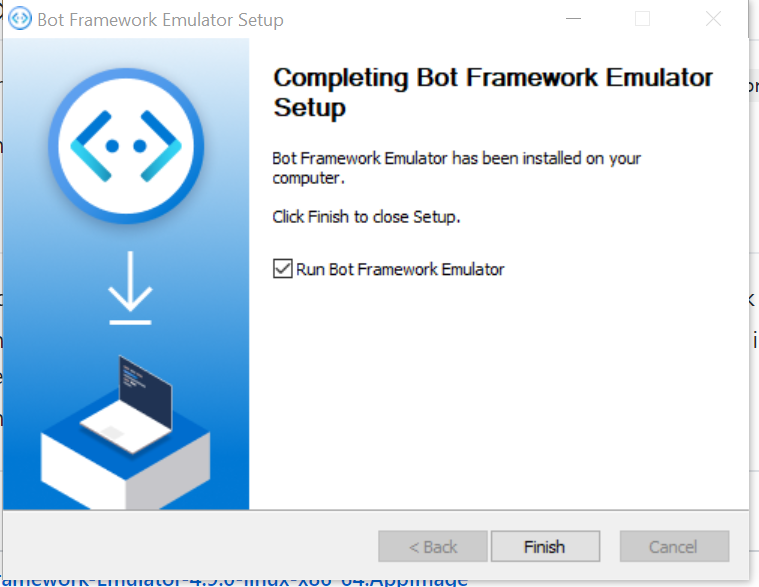


If you have not done so already download the [Bot Emulator Framework](https://github.com/Microsoft/BotFramework-Emulator/releases) from GitHub for your specific release.

Download the appropriate version and go through the installation.



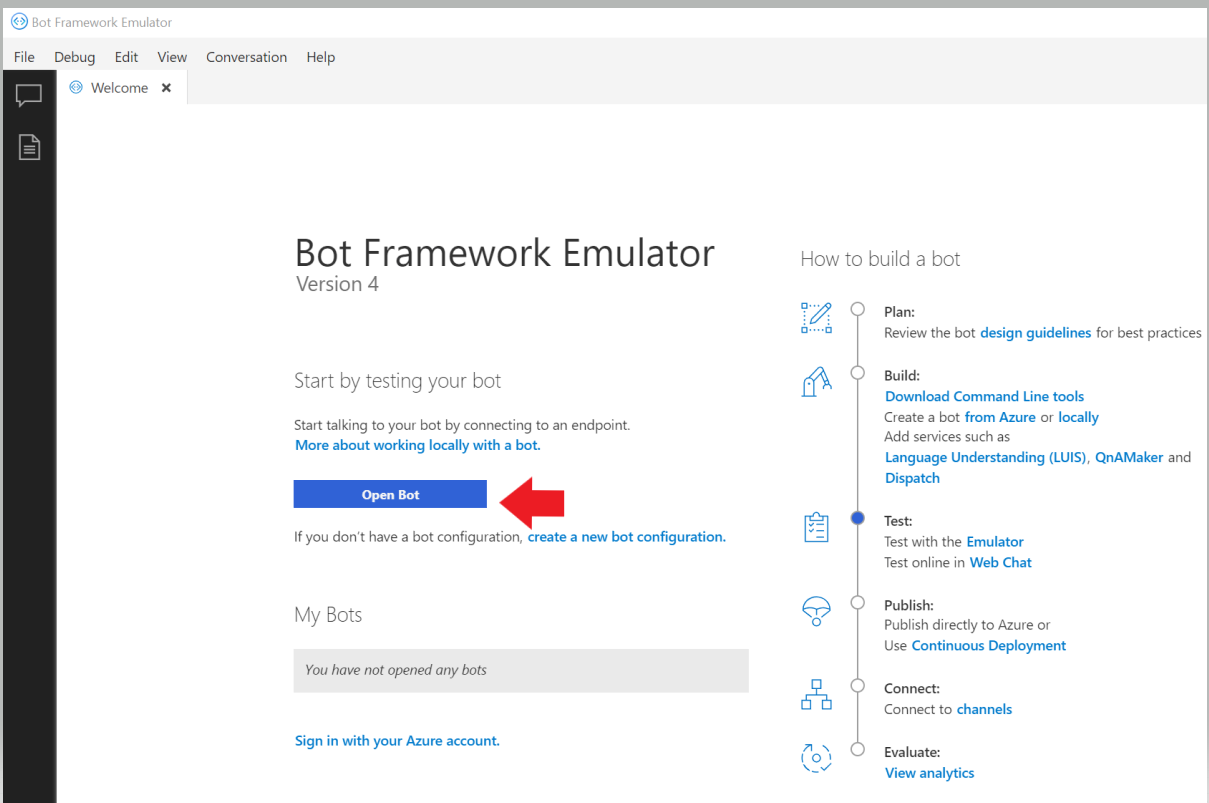
Once installation is complete click “**Finish**”allow bot emulator to run.

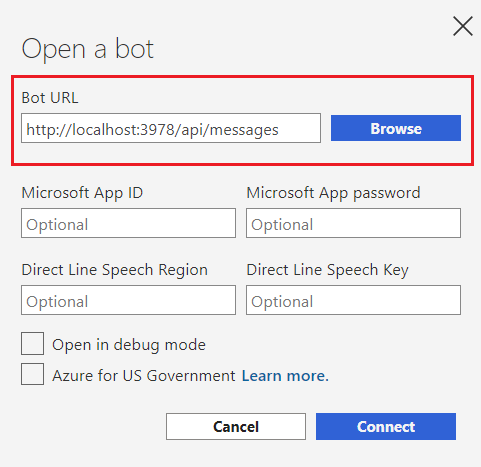


## Start the Emulator and connect to your bot

Next, start the emulator and then connect to your bot in the emulator:

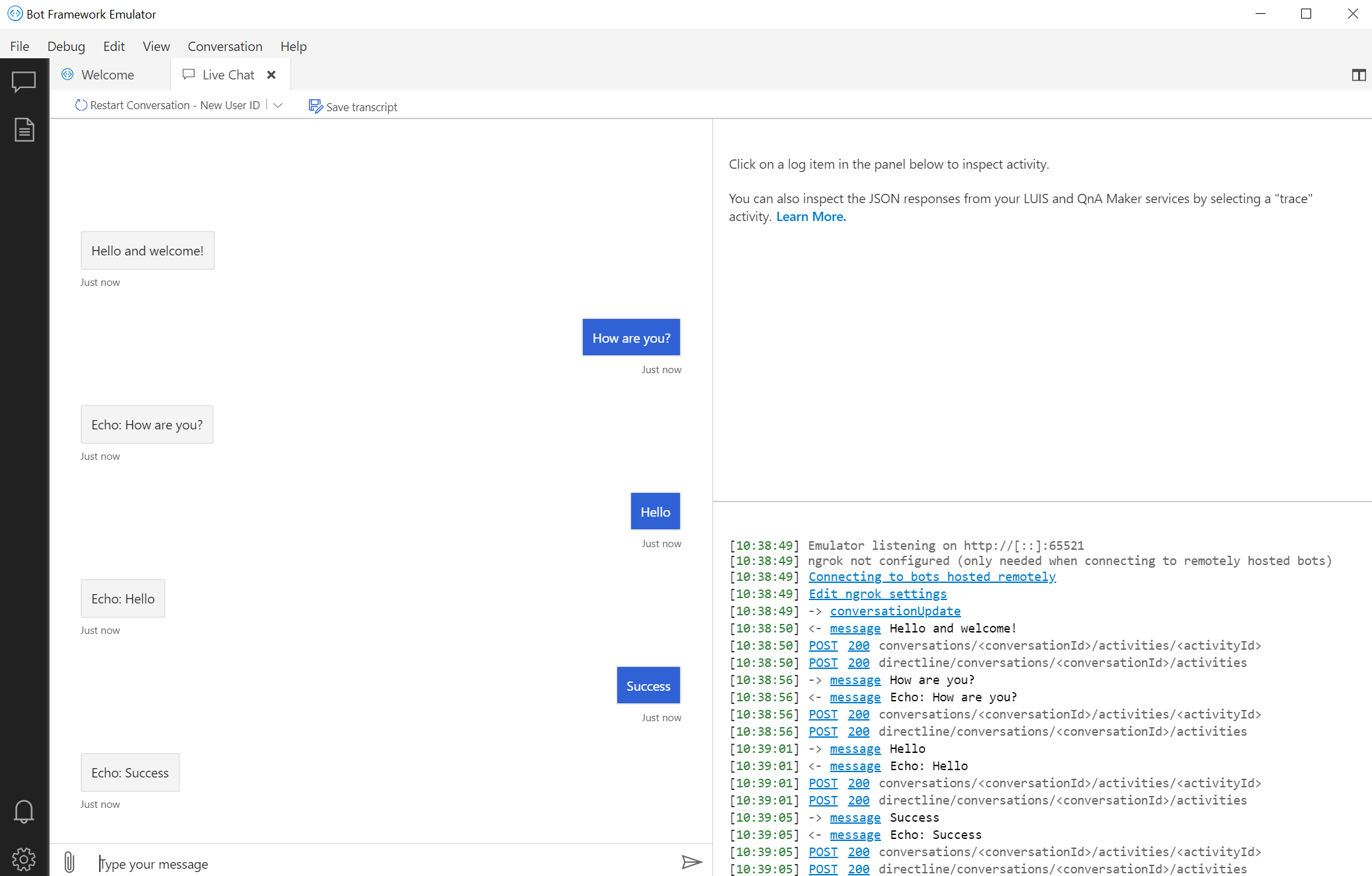
1. Start the Bot Framework Emulator.
2. Click **Open Bot** on the Emulator's **Welcome** tab.



1. Enter your bot's URL, which is the URL of the local port, with /api/messages added to the path, typically http://localhost:3978/api/messages.
2. Then click **Connect**.

Send a message to your bot, and the bot will respond back with a message.

\*If the bot does not respond please be sure your project is still running locally within Visual Studio.



## Deploy your bot

**Prerequisites**

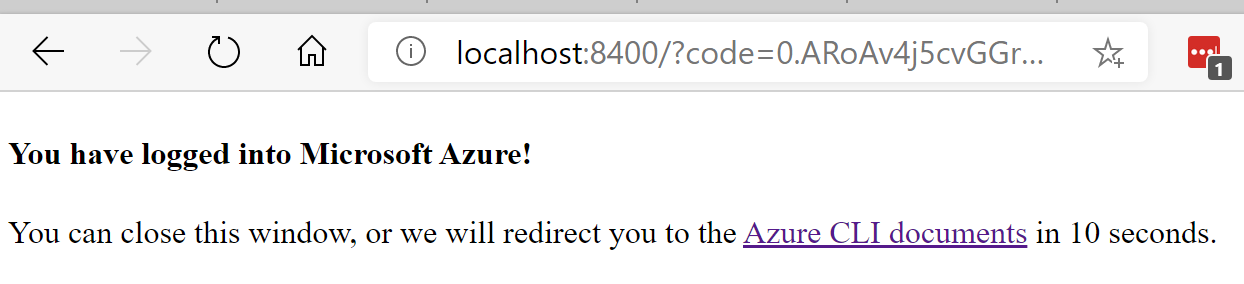
* A subscription to [Microsoft Azure](https://azure.microsoft.com/free/).
* A C#, JavaScript, TypeScript, or Python bot that you have developed on your local machine.
* Latest version of the [Azure CLI](https://docs.microsoft.com/en-us/cli/azure/?view=azure-cli-latest).
* Familiarity with [Azure CLI and ARM templates](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview).

Log in to Azure via Azure CLI from your local machine by typing “**az login**” and following the prompts.

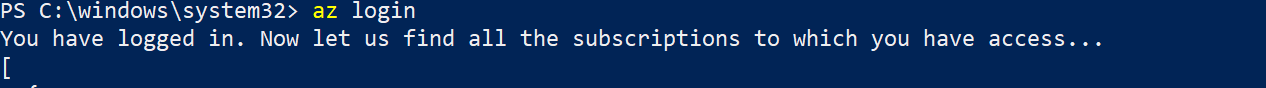
**Screenshot 1**



**Screenshot 2**



**Screenshot 3**

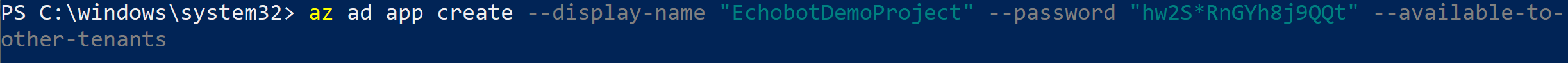


Set your default subscription by using the command **az account set --subscription "<azure-subscription>"**



**Create application registration**

* **Command**: az ad app create --display-name "displayName" --password "AtLeastSixteenCharacters\_0" --available-to-other-tenants
* **Actual Command:** az ad app create --display-name "EchobotDemoProject" --password " hw2S\*RnGYh8j9QQt" --available-to-other-tenants



#### **Record the appId and appSecret values**

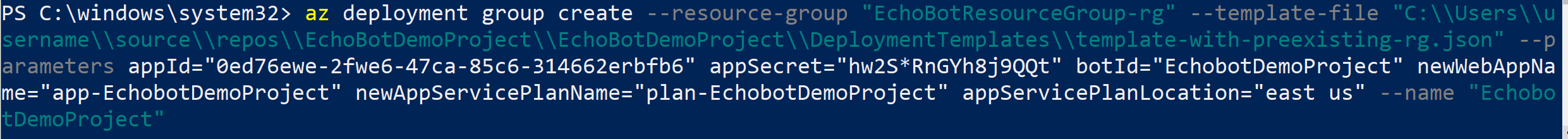
You will need to record two values after executing the command in the previous step:

* The password you create in the previous step.
* The appId which you can find in the output JSON.

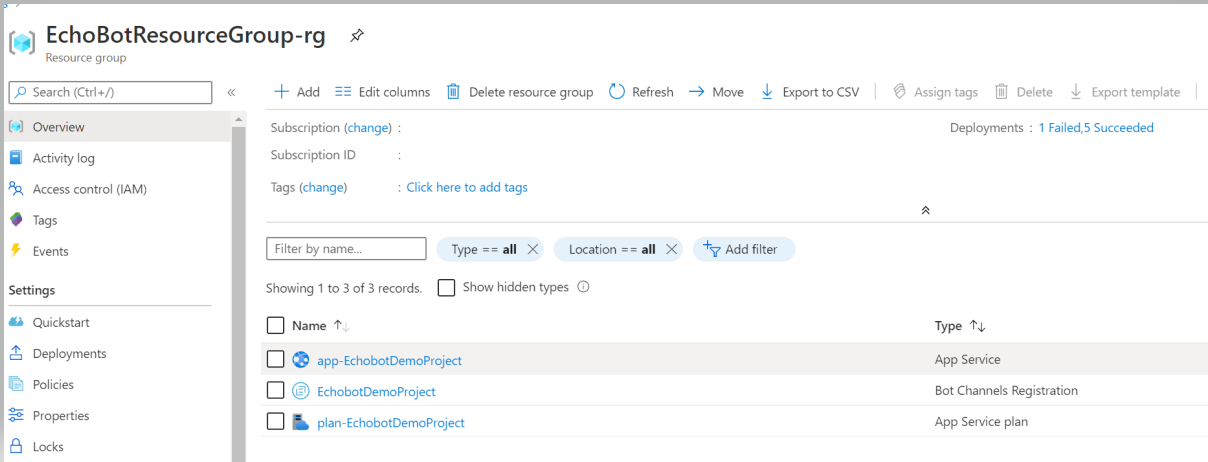
Copy and save the appId and password values. You will need to use them in the ARM deployment step, to assign values to the appId and the appSecret parameters, respectively.

**Deploy via ARM Template**

1. **Deploy using an existing resource group to a new app service plan**:
   * + **Note**: Intended Resource group will need to be present with proper permissions prior to running this command.
   1. **Command**: az deployment group create --resource-group "<name-of-resource-group>" --template-file "<path-to-template-with-preexisting-rg.json>" --parameters appId="<app-id-from-previous-step>" appSecret="<password-from-previous-step>" botId="<id or bot-app-service-name>" newWebAppName="<bot-app-service-name>" newAppServicePlanName="<name-of-app-service-plan>" appServicePlanLocation="<region-location-name>" --name "<bot-app-service-name>"
   2. **Actual Command**: az deployment group create --resource-group "EchoBotResourceGroup-rg" --template-file "C:\\Users\\username\\source\\repos\\EchoBotDemoProject\\EchoBotDemoProject\\DeploymentTemplates\\template-with-preexisting-rg.json" --parameters appId="0ed76v3se-2f16-47ca-85w6-314662feweb6" appSecret="hw2S\*RnGYh8j9QQt" botId="EchobotDemoProject" newWebAppName="app-EchobotDemoProject" newAppServicePlanName="plan-EchobotDemoProject" appServicePlanLocation="east us" --name "EchobotDemoProject"

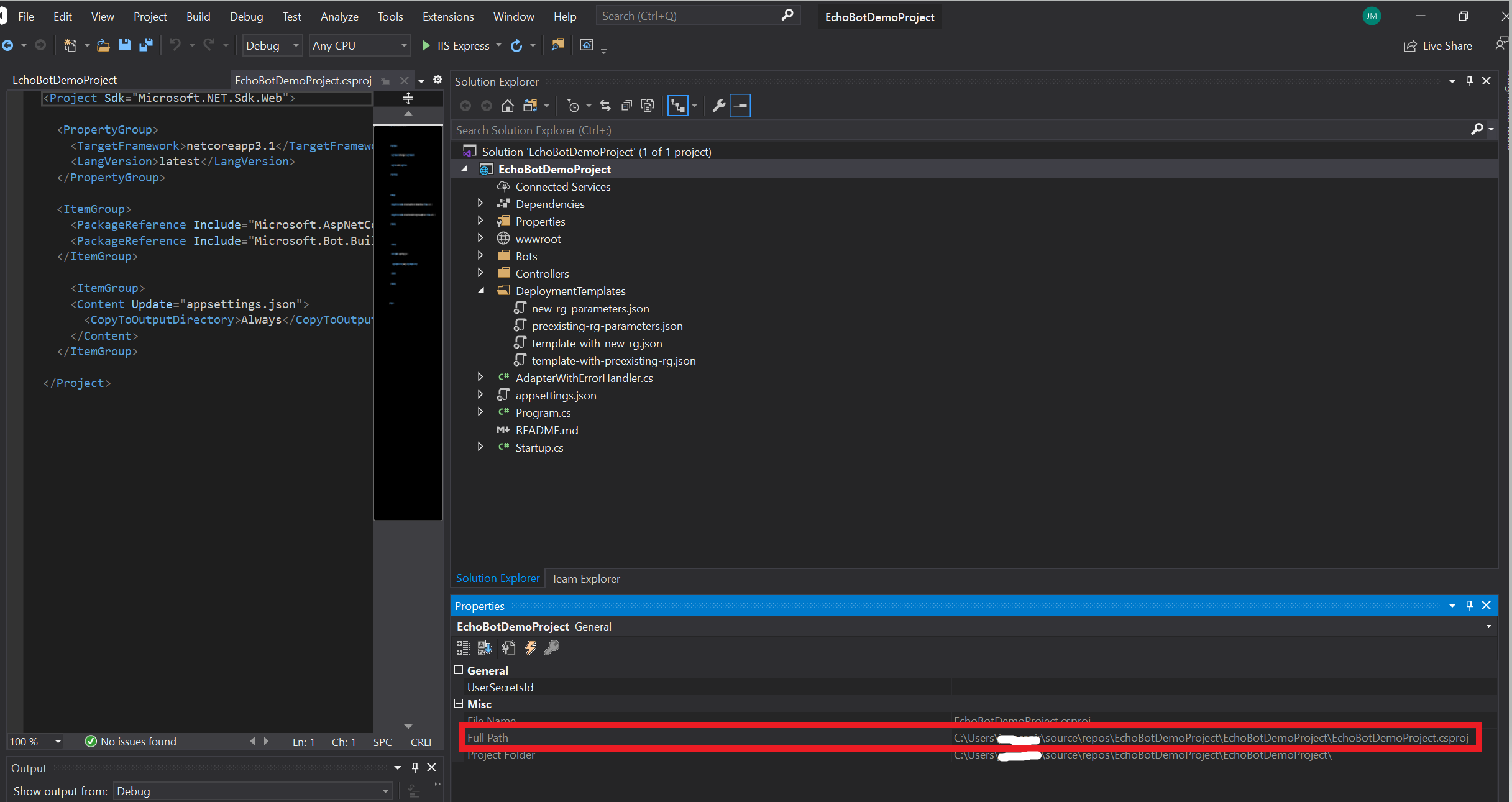


Successful deployment of resources into Azure.



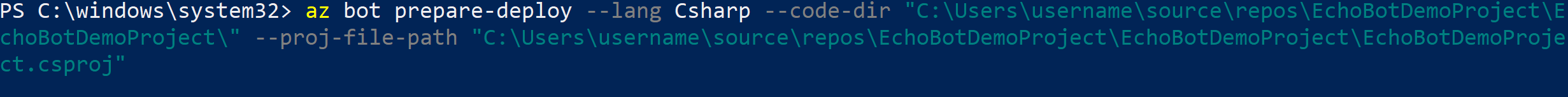
**Prepare Code for Deployment**

Navigate to you project within Visual Studio and locate the full project path under solution explorer.

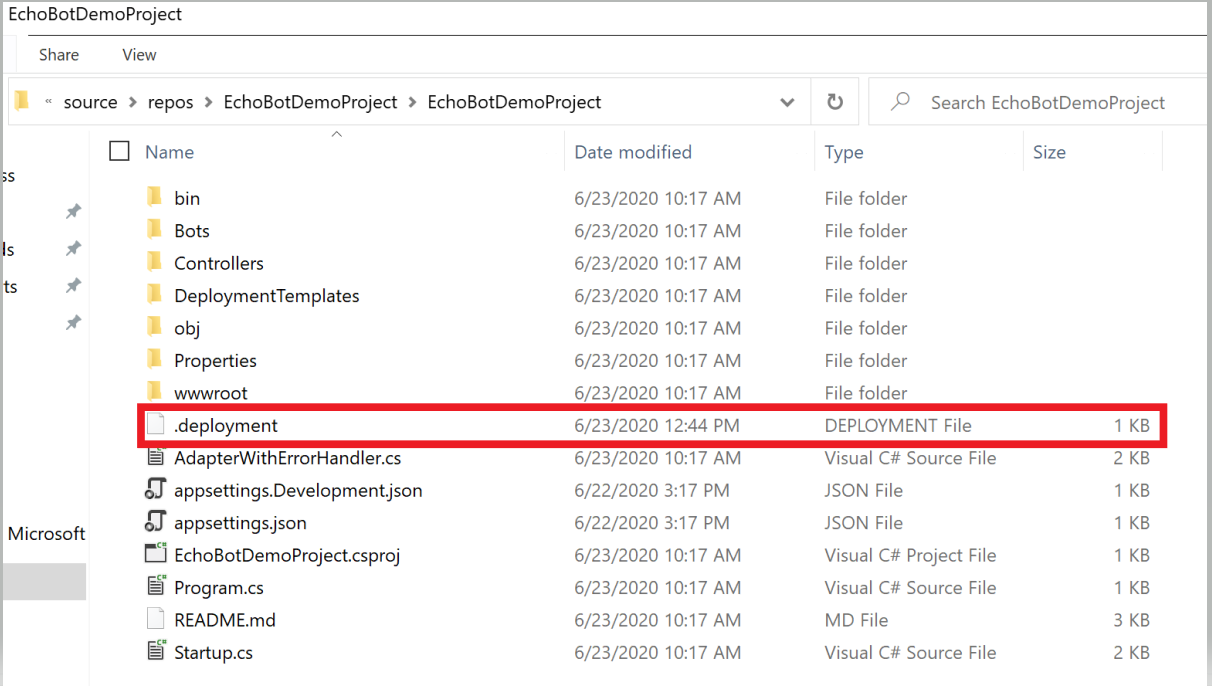


From Azure CLI run the command: az bot prepare-deploy --lang Csharp --code-dir "." --proj-file-path "MyBot.csproj"

**Actual Command**: az bot prepare-deploy --lang Csharp --code-dir “C:\Users\username\source\repos\EchoBotDemoProject\EchoBotDemoProject\” --proj-file-path “C:\Users\username\source\repos\EchoBotDemoProject\EchoBotDemoProject\EchoBotDemoProject.csproj”

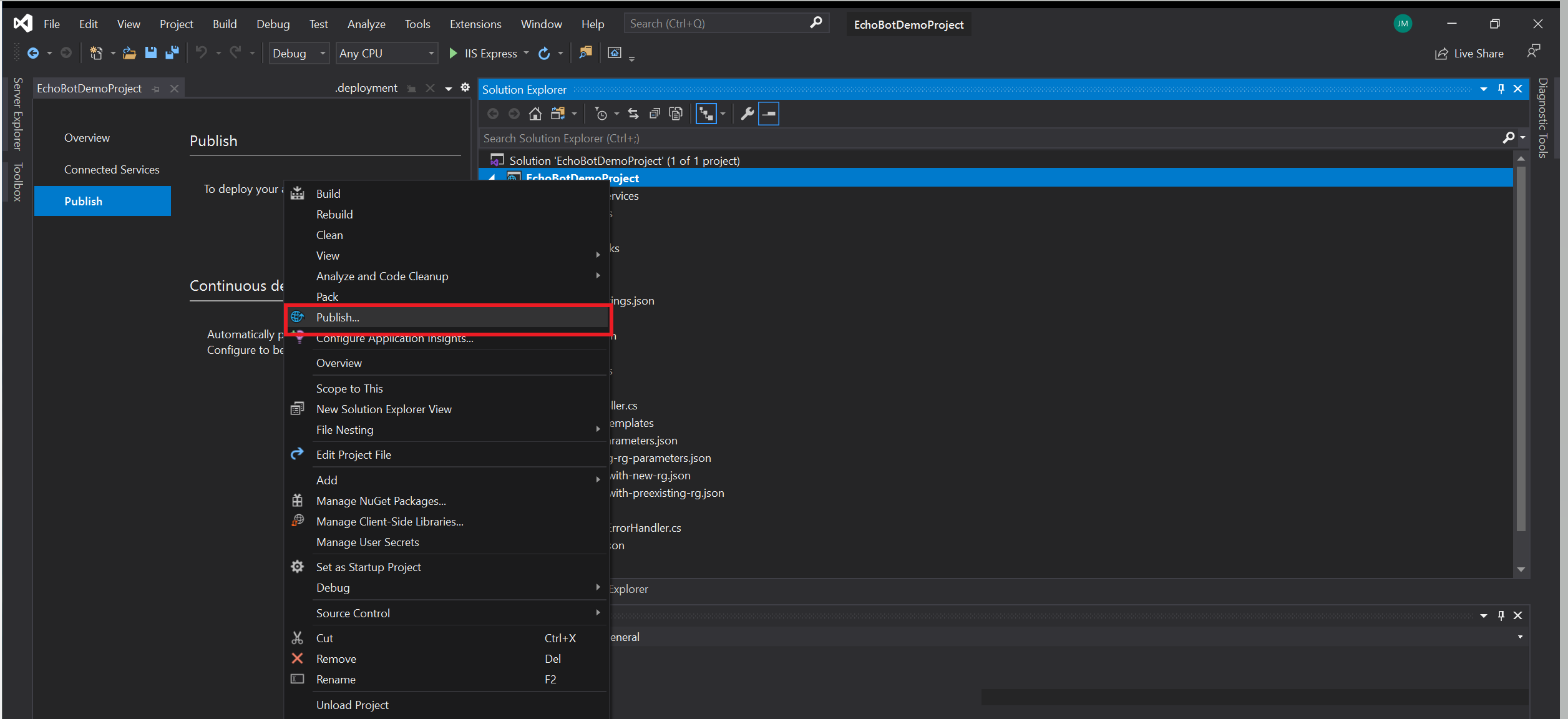


This will generate a .deployment file in the code directory path specified as shown below.

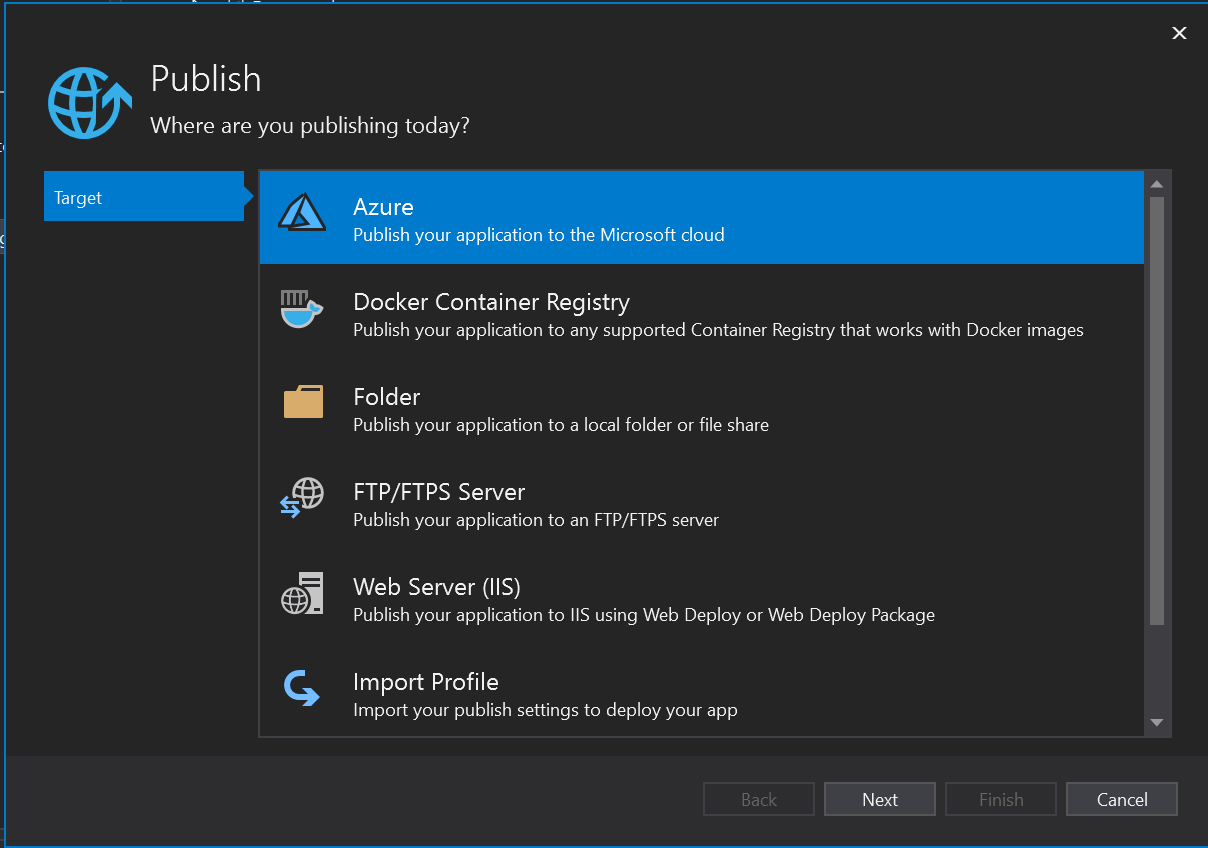


**Deploy code to Azure**

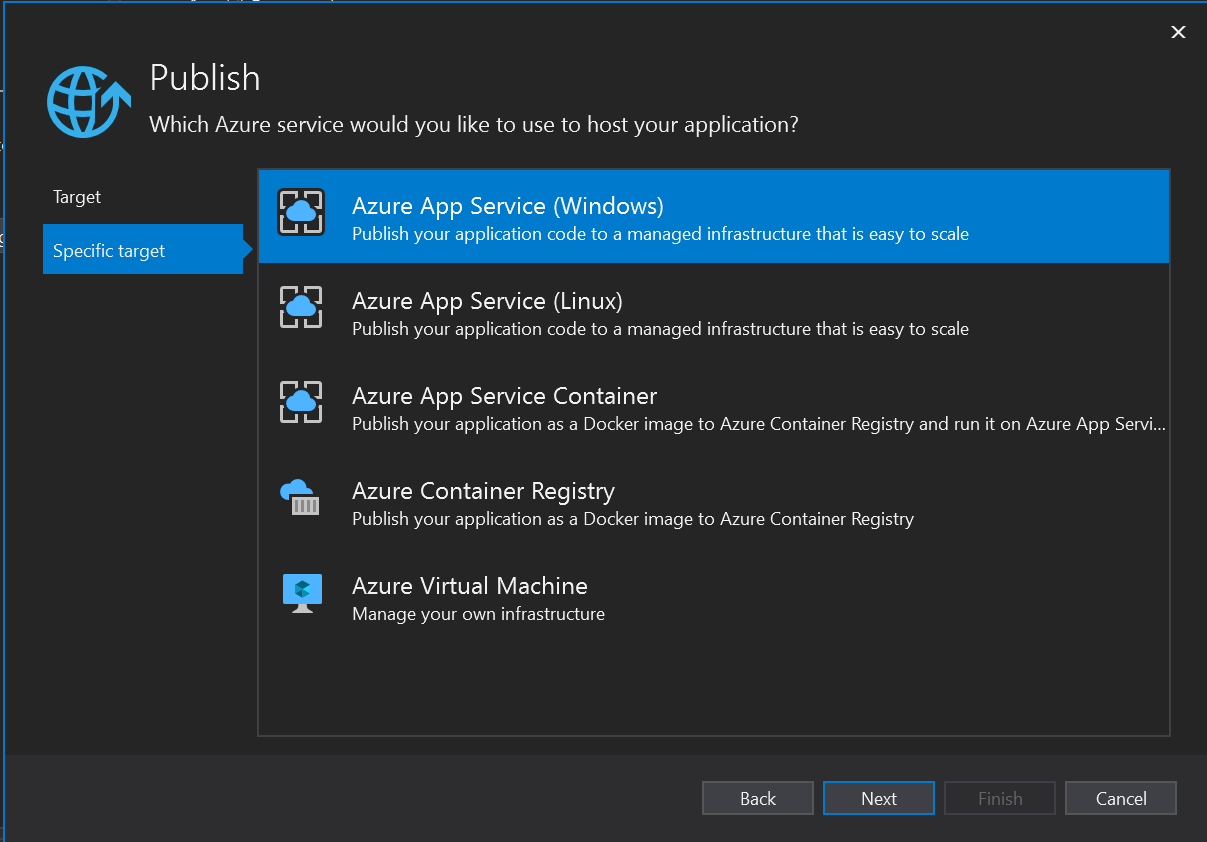
From Visual Studio we want to highlight our solution and right click to find “**Publish**”



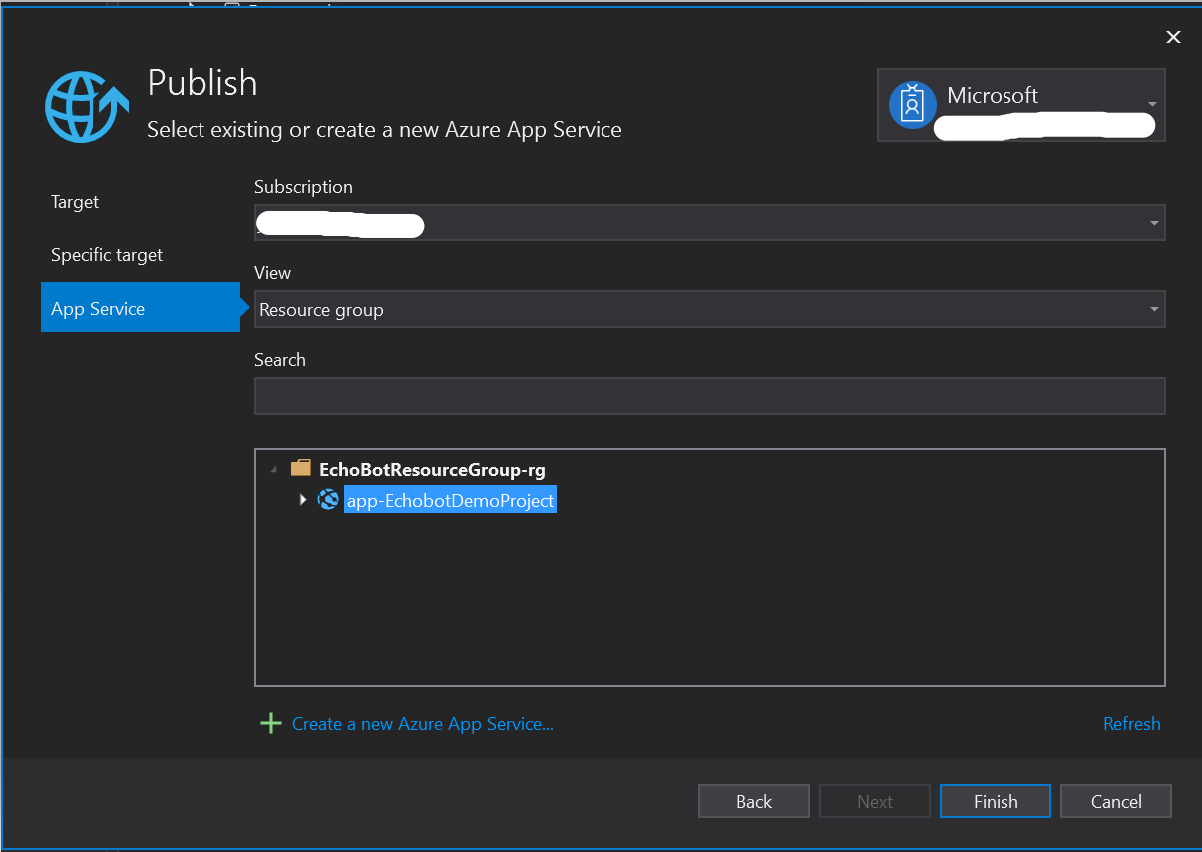
Choose “**Azure**” as your target location.



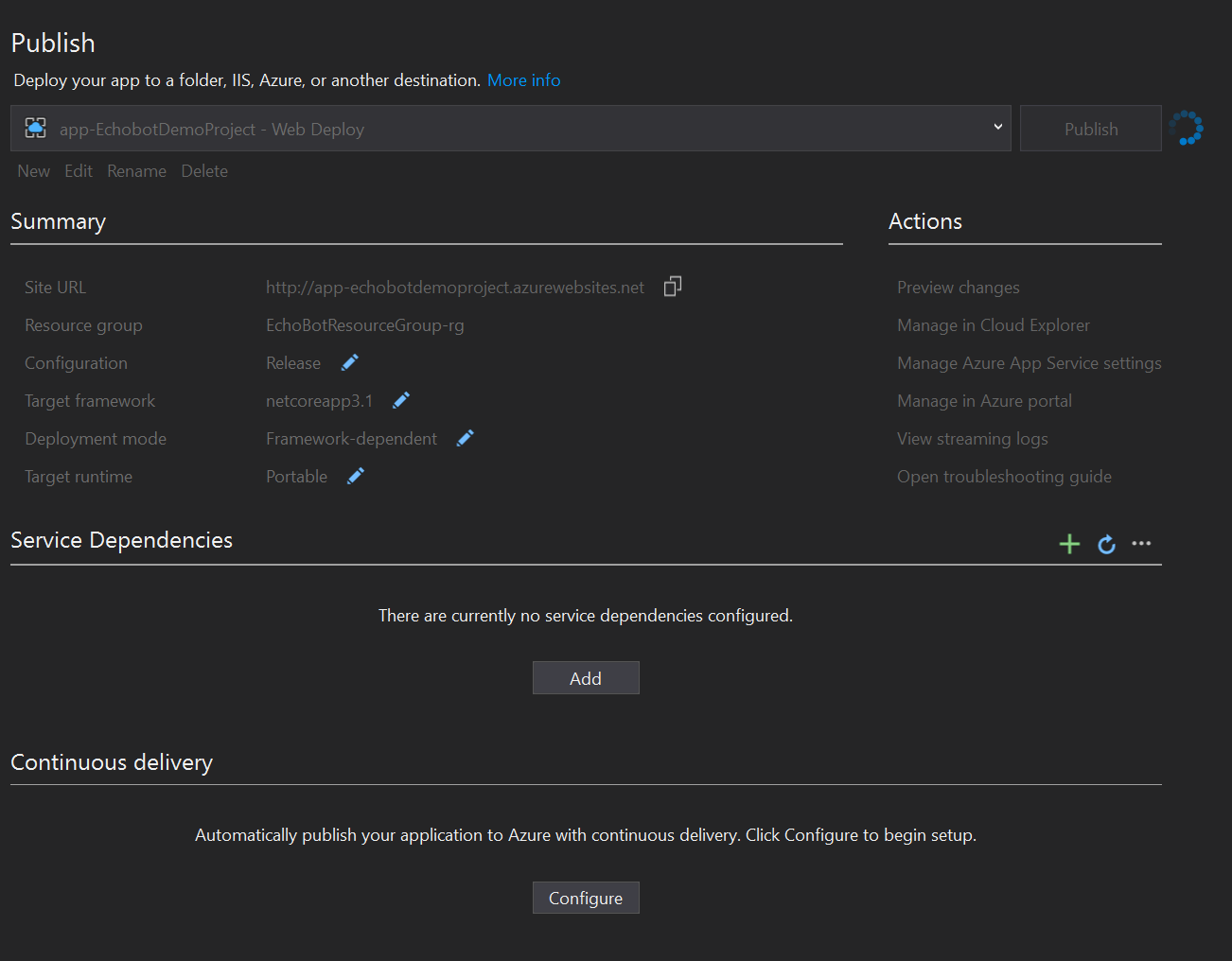
Choose “**Azure App Service**” as your specific target.



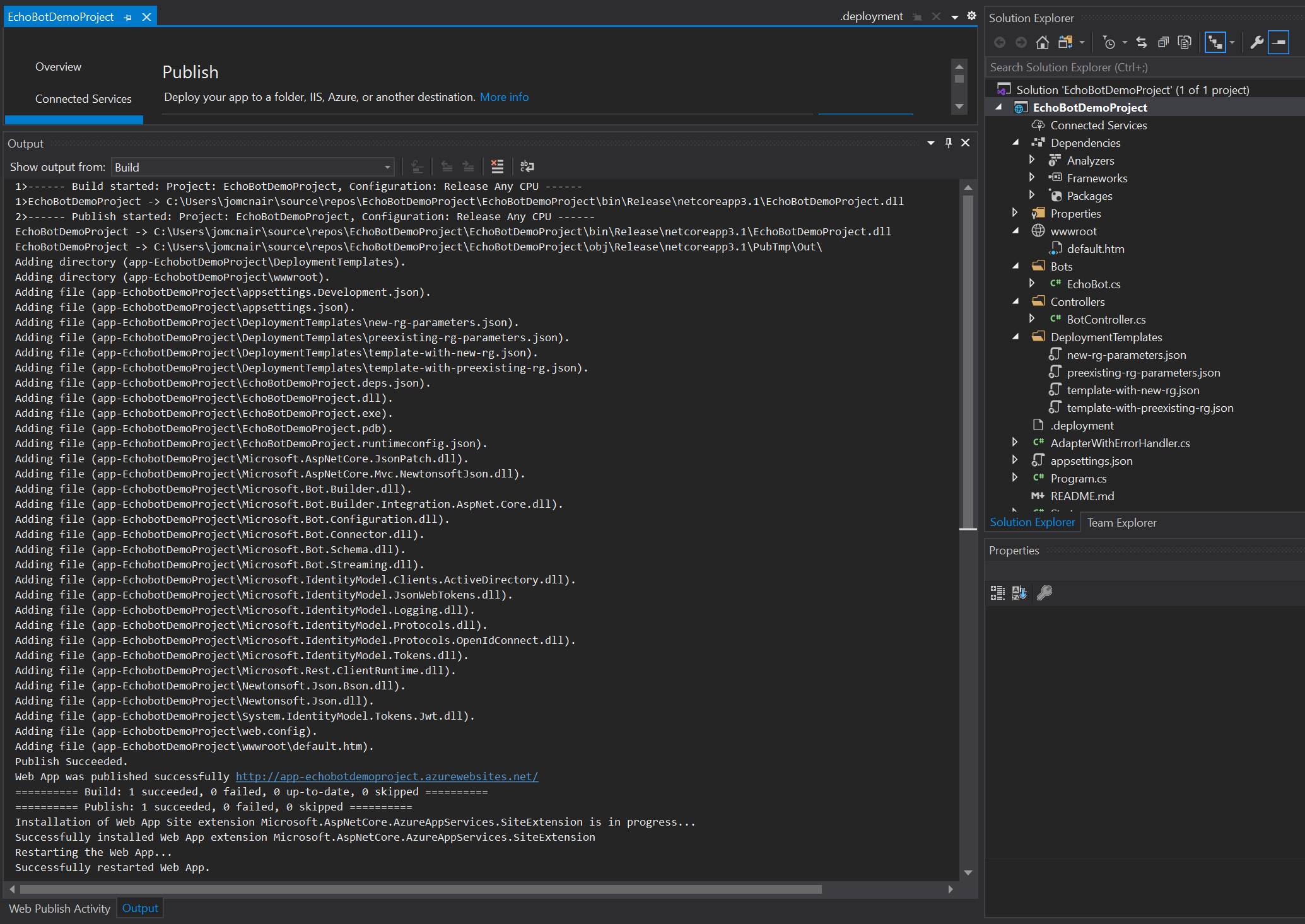
When prompted make the appropriate selections for the Subscription, Resource Group, and existing Web App that was previously built. Click Finish



Click Publish once all selections are confirmed.



Review your Output log in Visual Studio to see whether the deployment succeeded.



Go into Azure portal and test using the “Test in Web Chat” feature to confirm functionality.

