

Lab 05. Build and integrate a Bot with the search API

Objective

In this lab you are going to use a bot sample code and integrate it with the search API.

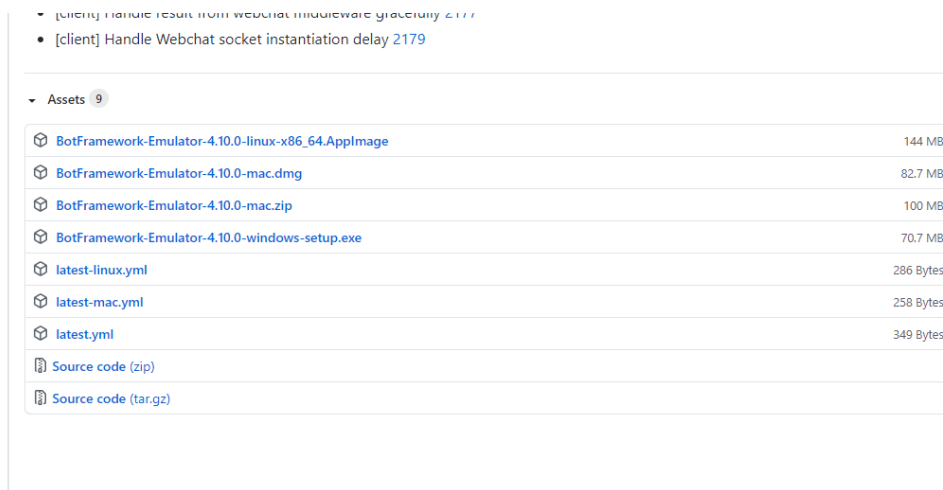
Prerequisites

- Complete **Lab 2**.
- Visual Studio Code 2017 or later.

Steps

Download bot framework emulator

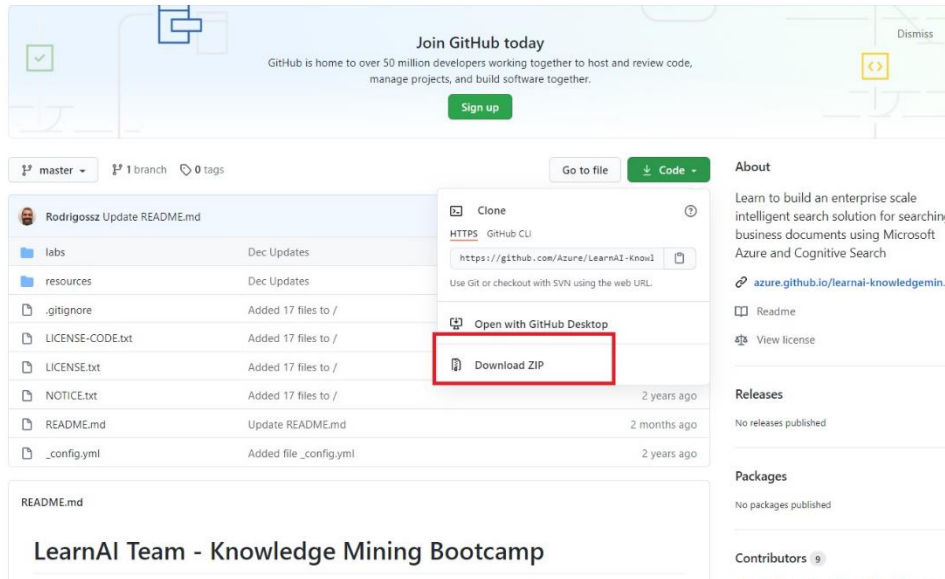
1. Open a browser and navigate to the following link <https://github.com/Microsoft/BotFramework-Emulator/releases> , Download the most recent version of the bot emulator, (for example if you are using windows download the BotFramework-Emulaotr-4.XX.X-windows-setup.exe)



2. Proceed to install the Bot Framework Emulator. NOTE: If you already have a previous version installed, this wizard will perform an upgrade.

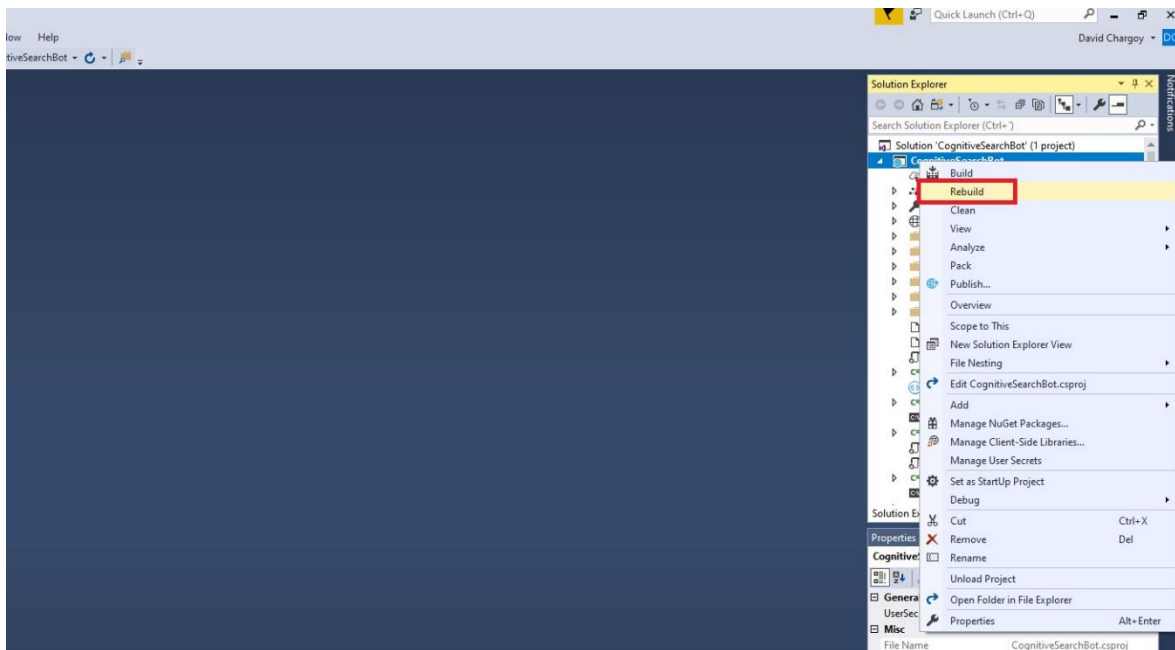
Clone the Repo

- 1.- Open a new browser window and navigate to <https://github.com/Azure/LearnAI-KnowledgeMiningBootcamp.git>
- 2.- Select Clone or download.
- 3.- Extract the zip file to your local machine.



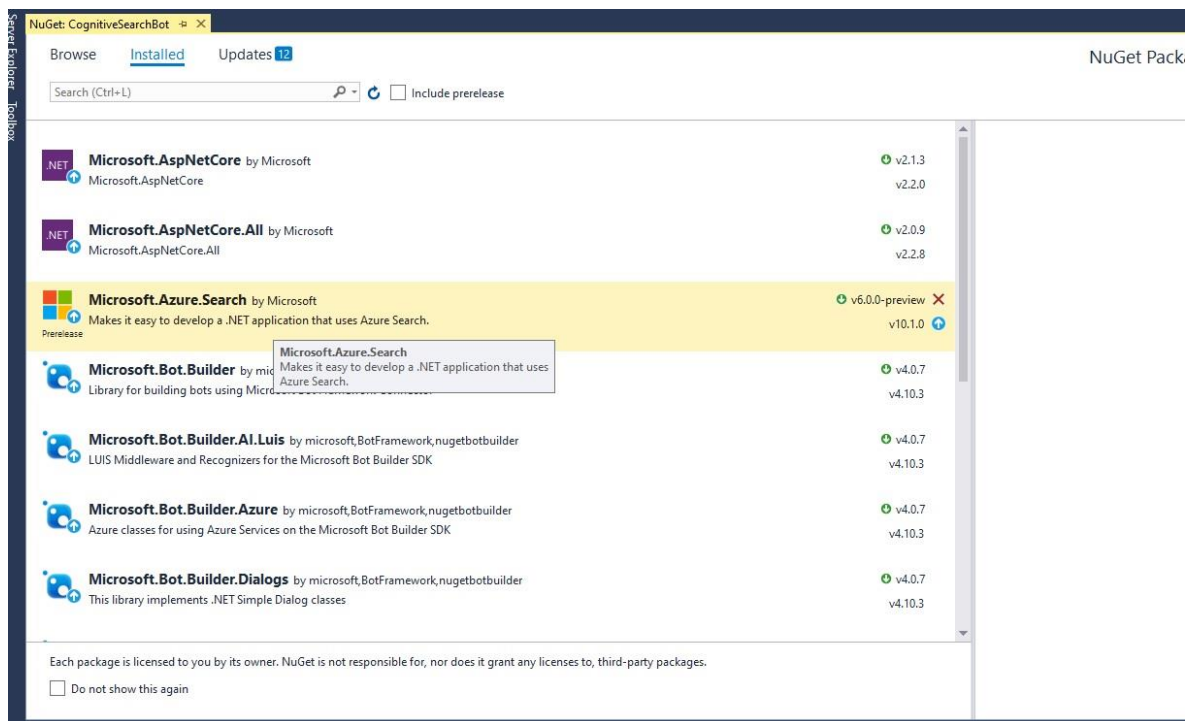
4- Navigate to your extracted folder and go to **resources>code-bot** and double click on the file **CognitiveSearchBot.sln** to open the visual studio solution

5.- in Solution Explorer, right-click on the **CognitiveSearchBot** solutions and select **Rebuild**



6.- Right-click on the solution and select **Manage NuGet Packages for Solution**

7.- Review that under **Installed** you see **Microsoft.Azure.Search** listed.



8.- Open the **Constants.cs** file

9.- Locate the Strings **SearchServiceName**, **SearchServiceApiKey** and **TargetIndexName** and replace them with your values. NOTE: These values are from the Lab02, in the case you do not have them you can go to the Azure Portal and obtain the values

```

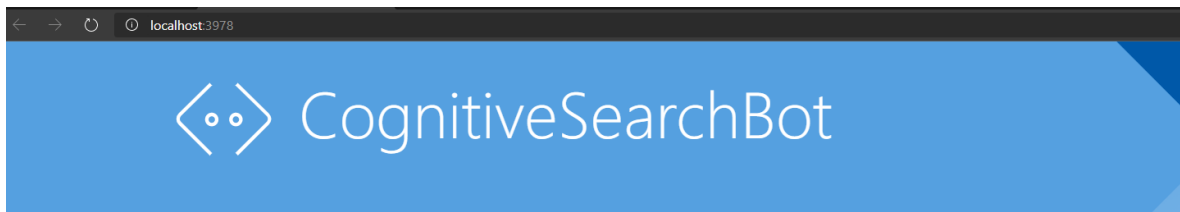
5
6
7 namespace CognitiveSearchBot
8 {
9     public static class Constants
10    {
11        // replace "Your-Azure-Search-Service-Name", "Your-Azure-Search-Service-key", and "Your-Azure-Search-I
12        public static string SearchServiceName = "Your-Azure-Search-Service-Name";
13        public static string SearchServiceApiKey = "Your-Azure-Search-Service-Key";
14        public static string TargetIndexName = "Your-Azure-Search-Index-name";
15    }
16 }
17

```

10.- Save the file

Run the project

1.- On Visual Studio press **F5** to build and run your bot locally and will open a browser window



Your bot is ready!

You can test your bot in the Bot Framework Emulator by opening the .bot file in the project folder.

[Download the Emulator](#)

Visit [Azure Bot Service](#) to register your bot and add it to various channels. The bot's endpoint URL typically looks like this:

`https://your_bots_hostname/api/messages`

HOW TO BUILD A BOT



Plan:

Review the bot [design guidelines](#)



Build:

Create a bot from [Azure](#) or [locally](#)
Download [Command-line tools](#)
Add services such as [Language Understanding](#),
[QnA Maker](#) and [Dispatch](#)



Test:

Test using the [Emulator](#)
Test online in [Web Chat](#)



Publish:

Publish [directly to Azure](#) or
Use [Continuous Deployment](#)

- 2.- Copy the URL from the browser
- 3.- Open the **Bot Framework Emulator**
- 4.- Click on **File** menu and select **Open Bot**
- 5.- On the **Bot URL** textbox, paste the URL and append **/api/messages**

×

Open a bot

Bot URL

Browse

Microsoft App ID

Optional

Microsoft App password

Optional

Direct Line Speech Region

Optional

Direct Line Speech Key

Optional

☐ Open in debug mode

☐ Azure for US Government [Learn more.](#)

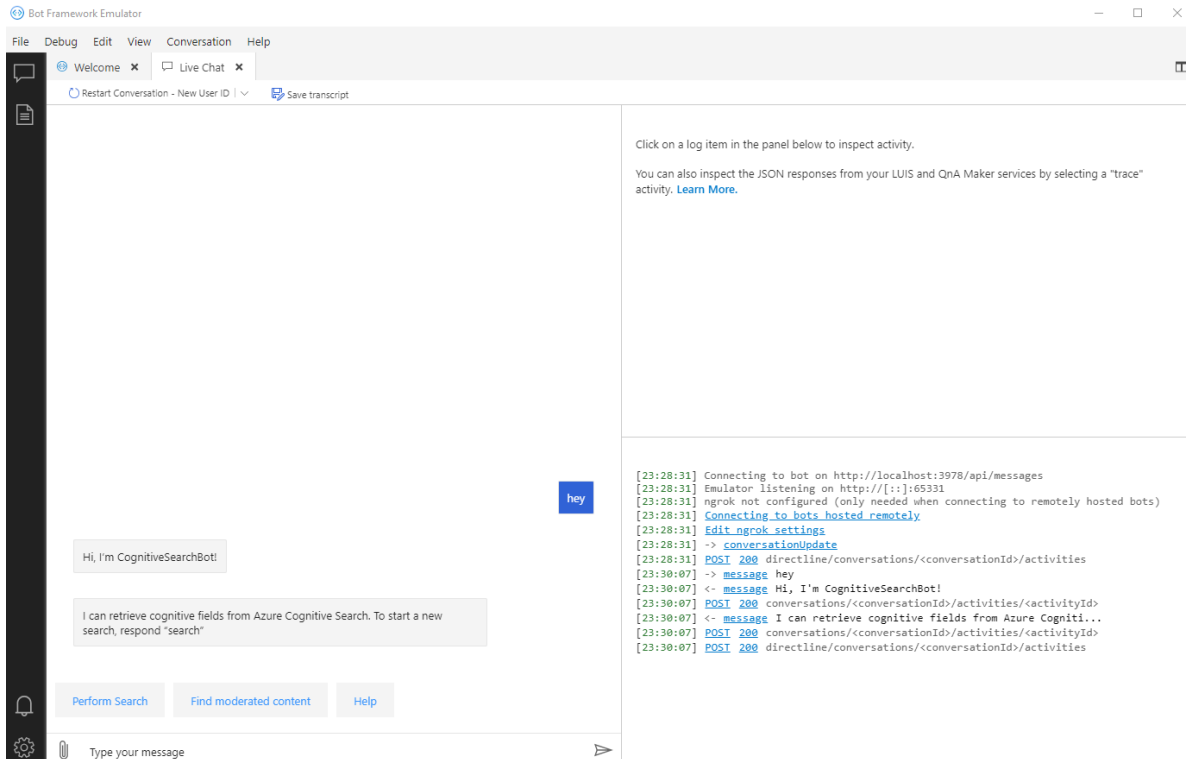
Cancel

Connect

6.- Click on **Connect**

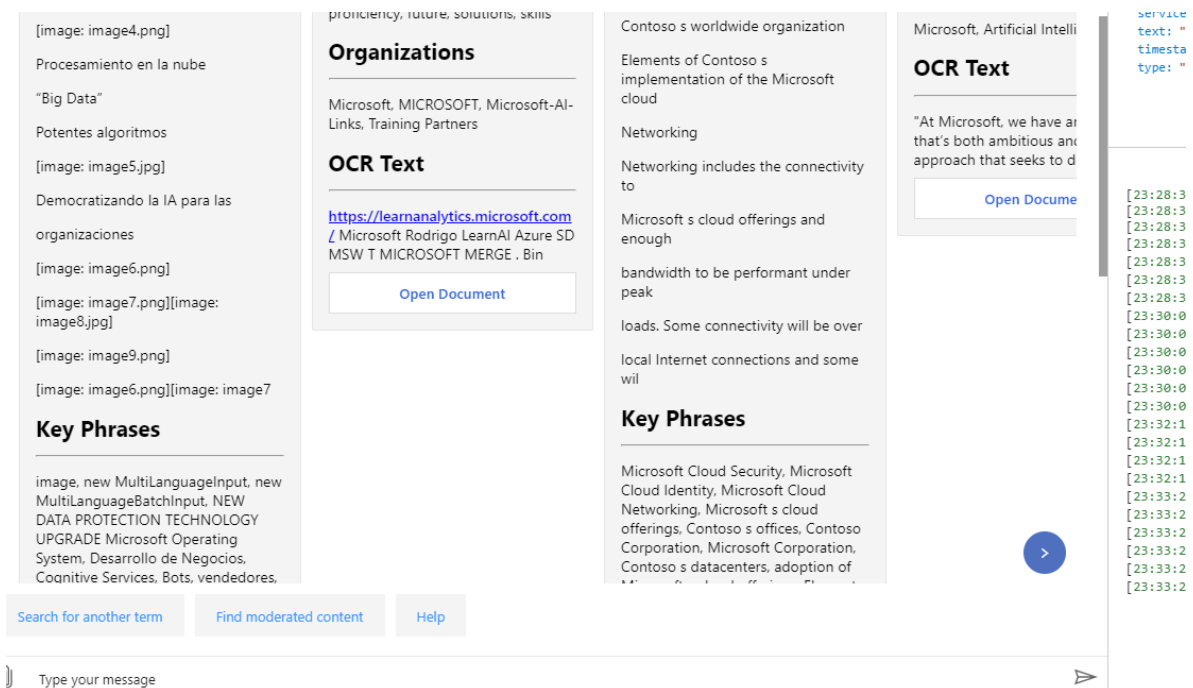
7.- This will open a chat windows with your bot, Start by saying something like Hello, Hi or Hey

8.- The bot will respond with a greeting message followed by the actions that the Bot can do for you




9.- Select the **Perform Search** option, and submit some searches to the bot, for example “Microsoft”

10.- Look on the results, you will see some cards with the information obtained about your search, also you could use the navigation arrows to review all the information presented



11/1/2014



"At Microsoft, we have an approach that's both ambitious and broad, an approach that seeks to democratize Artificial Intelligence (AI), to take it from the ivory towers and make it accessible for all."

satyanadellalinux.jpg

Key Phrases

approach, Artificial Intelligence, ivory towers, AI, Microsoft


Organizations

Microsoft, Artificial Intelligence

OCR Text

"At Microsoft, we have an approach that's both ambitious and broad, an approach that seeks to democratize Artificial Intelligence (AI), to take it from the ivory towers and make it accessible for all."

Open Document



satyanadellalinux.jpg

Key Phrases

Microsoft Linux

Organizations

Microsoft Linux

OCR Text

Microsoft Linux

Open Document

MSFT_FY17_10K.docx

Content Preview

[bookmark: _GoBack] UNITED STATES SECURITIES AND EXCHANGE COMMISSION Washington, D.C. 20549 FORM 10-K ANNUAL REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934 For the Fiscal Year Ended June 30, 2017 OR TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934 For the Transition Period From to Commission File Number 001-37845 MICROSOFT CORPORATION WASHINGTON

91-1144442

(STAT

Key Phrases

check mark, Securities Act, SECURITIES EXCHANGE ACT, SECTION, large accelerated filer, non-accelerated filer, Non-

12.- Type **Help** and send it to the bot, the bot will answer you with the things that the bot can do for you

14.- Return to Visual Studio and press **Shift + F5** to stop your debugging

Create a bot framework registration

- 1.- Open a browser and navigate to <https://ngrok.com/download>

2.- Download the ngrok tool and extract the file **ngrok.exe** to your bot's folder

CognitiveSearchBot.cs	29/07/2020 05:25 p. m.	Visual C# Source F...	6 KB
CognitiveSearchBot.csproj	29/07/2020 05:25 p. m.	Visual C# Project f...	2 KB
CognitiveSearchBot.deps.json	29/07/2020 05:25 p. m.	JSON File	274 KB
CognitiveSearchBot.runtimeconfig.json	29/07/2020 05:25 p. m.	JSON File	1 KB
CognitiveSearchBot.sln	29/07/2020 05:25 p. m.	Visual Studio Solu...	2 KB
Constants.cs	09/10/2020 10:46 a. m.	Visual C# Source F...	1 KB
deploy.cmd	29/07/2020 05:25 p. m.	Script de comand...	3 KB
ngrok.exe	08/10/2019 02:53 p. m.	Aplicación	26,131 KB
ngrok-stable-windows-amd64.zip	09/10/2020 10:39 a. m.	Archivo WinRAR Z...	13,496 KB
Program.cs	29/07/2020 05:25 p. m.	Visual C# Source F...	2 KB
readme.md	29/07/2020 05:25 p. m.	Markdown File	1 KB
Startup.cs	29/07/2020 05:25 p. m.	Visual C# Source F...	9 KB
web.config	29/07/2020 05:25 p. m.	XML Configuratio...	1 KB

3.- Open a PowerShell Windows with administrative privileges and navigate to your bots folder

4.- execute the following command

`./ngrok http 3978`

You will be presented with a windows like this

```
ngrok by @inconshreveable

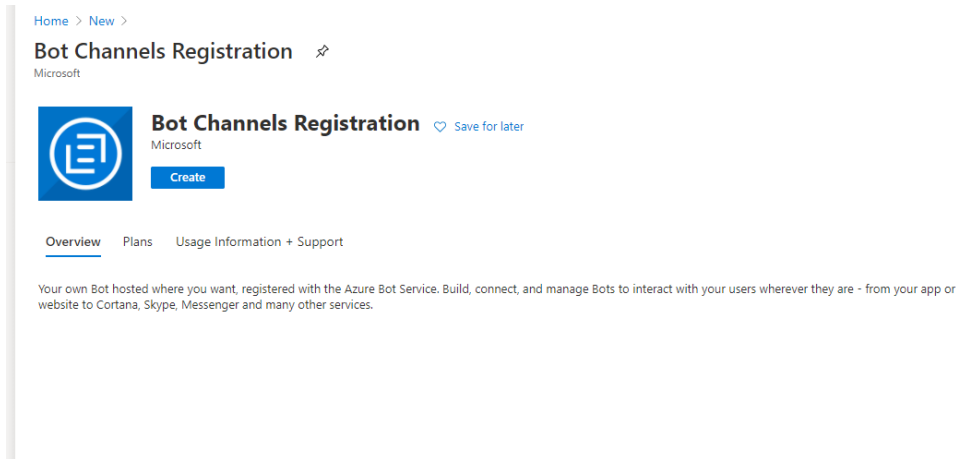
Session Status      online
Session Expires    7 hours, 59 minutes
Version            2.3.35
Region             United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding          http://d0aad89e20a2.ngrok.io -> http://localhost:3978
                   https://d0aad89e20a2.ngrok.io -> http://localhost:3978

Connections        ttl    opn    rt1    rt5    p50    p90
                   0      0      0.00   0.00   0.00   0.00
```

5.- Sign in into the Azure portal at <https://portal.azure.com>

6.- Select **Create a resource** from the favourites menu

7.- Search for Bot Channel Registration and select it and then click on Create



8.- Enter the information bellow and then select create

- **Bot handle:** type a name for your bot. The bot name must be unique and will give you a message when it's not.
- **Subscription:** ensure your Azure subscription is selected.
- **Resource group:** select **Create new** and give it a name such as **WebAppBotResource**.
- **Location,** will default to your current tenant location. You can leave this as the default or select a different location.
- **Pricing tier:** select **F01 (10K Premium Messages)**
- **Messaging Endpoint:** type the https url that ngrok gave you and append /api/messages
- **Application Insights:** Off

9.- Once your deployment has completed navigate to the Resource Group **WebAppBotResource**, and inside Settings, select **Deployments**

WebAppBotResource
Resource group

Search (Ctrl+/) << + Add Edit columns Delete resource group Refresh Export to CSV Open q

Overview

- Activity log
- Access control (IAM)
- Tags

Settings

- Quickstart
- Deployments**
- Policies
- Properties
- Locks

Cost Management

- Cost analysis

Essentials

Subscription (change) : MSDN Platforms Subscription Deploymer

Subscription ID : 8f1eeacc-ecf4-4a1b-b731-f494376aaaf4

Tags (change) : Click here to add tags

Filter by name... Type == all X Location == all X Add filter

Showing 1 to 1 of 1 records. ☐ Show hidden types ⓘ

<input type="checkbox"/> Name ↑↓	Type ↑↓
<input type="checkbox"/> exabottesting	Bot Channels

10.- Select the deployment

11.- From the deployment blade, select Inputs

Home > Resource groups > WebAppBotResource >

exabottesting_878 | Overview
Deployment

Search (Ctrl+/) << Delete Cancel Redeploy Refresh

Overview

- Inputs**
- Outputs
- Template

Your deployment is complete

Deployment name: exabottesting_878 Start time: 1
Subscription: MSDN Platforms Subscription Correlation ID:
Resource group: WebAppBotResource

Deployment details (Download)

Next steps

[Go to resource](#)

12.- Look for the appId and app Secret fields and copy them

13.- Go Back To Visual Studio and open the file **BotConfiguration.bot**

14.- Change the values for the appId and appPassword with the appId and appSecret and save the project

15.- Press **F5** to run your project.

16.- Go Back to the Azure Portal and, select **Resource groups** then select the groups called **WebAppBotResource**

17.- Then select the from the overview blade the Bot Channel Registration

18.- Under **Bot management** select **Test in Web Chat**

19.- Say Hi to your bot and now you will be using your local bot within the Azure portal

The screenshot shows the Azure Portal interface for testing a bot. The breadcrumb navigation at the top reads: Home > Resource groups > WebAppBotResource > exabottesting. The main heading is 'exabottesting | Test in Web Chat' with a sub-label 'Bot Channels Registration'. Below this is a search bar with the placeholder 'Search (Ctrl+ /)' and a 'Test' button. A 'Start over' link is in the top right. A left-hand navigation pane lists various options: Overview, Activity log, Access control (IAM), Tags, Bot management (highlighted), Test in Web Chat (selected), Analytics, Channels, Settings, Speech priming, Bot Service pricing, Automation, Tasks, Support + troubleshooting, and New support request. The chat area shows a conversation: a user message 'hi' (3 minutes ago) and a bot response 'Hi, I'm CognitiveSearchBot!' (3 minutes ago). The bot's response includes a text block: 'I can retrieve cognitive fields from Azure Cognitive Search. To start a new search, respond "search"'. Below the text are three buttons: 'Perform Search', 'Find moderated content', and 'Help'. At the bottom is a text input field with the placeholder 'Type your message' and a send button.