Create a Bot with the Bot Framework SDK

Bots are software agents that can participate in conversational dialogs with human users. The Microsoft Bot Framework provides a comprehensive platform for building bots that can be delivered as cloud services through the Azure Bot Service.

In this exercise, you'll use the Microsoft Bot Framework SDK to create and deploy a bot.

Before you start

Let's start by preparing the environment for bot development.

Update the Bot Framework Emulator

You're going to use the Bot Framework SDK to create your bot, and the Bot Framework Emulator to test it. The Bot Framework Emulator is updated regularly, so let's make sure you have the latest version installed.

> **Note**: Updates may include changes to the user interface that affect the instructions in this exercise.

1. Start the **Bot Framework Emulator**, and if you are prompted to install an update, do so for the currently logged in user. If you are not prompted automatically, use the **Check for update** option on the **Help** menu to check for updates.
2. After installing any available update, close the Bot Framework Emulator until you need it again later.

Clone the repository for this course

If you have not already cloned **AI-102-AIEngineer** code repository to the environment where you're working on this lab, follow these steps to do so. Otherwise, open the cloned folder in Visual Studio Code.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/AI-102-AIEngineer` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.

   > **Note**: If you are prompted to add required assets to build and debug, select **Not Now**.

Create a bot

You can use the Bot Framework SDK to create a bot based on a template, and then customize the code to meet your specific requirements.

> **Note**: In this exercise, you can choose to use either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **13-bot-framework** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.
2. Right-click the folder for your chosen language and open an integrated terminal.
3. In the terminal, run the following commands to install the bot templates and packages you need:

C#

Code Copy

```
dotnet new -i Microsoft.Bot.Framework.CSharp.EchoBot
dotnet new -i Microsoft.Bot.Framework.CSharp.CoreBot
dotnet new -i Microsoft.Bot.Framework.CSharp.EmptyBot
```

Python

Code Copy

```
pip install botbuilder-core
pip install asyncio
pip install aiohttp
pip install cookiecutter==1.7.0
```

1. After the templates and packages have been installed, run the following command to create a bot based on the *EchoBot* template:

C#

Code Copy

```
dotnet new echobot -n TimeBot
```

Python

Code Copy

```
cookiecutter https://github.com/microsoft/botbuilder-python/releases/download/Templates/echo.zip
```

If you're using Python, when prompted by cookiecutter, enter the following details:

- **bot_name**: TimeBot
- **bot_description**: A bot for our times

1. In the terminal pane, enter the following commands to change the current directory to the **TimeBot** folder list the code files that have been generated for your bot:

   Code Copy

   ```
   cd TimeBot
   dir
   ```

Test the bot in the Bot Framework Emulator

You've created a bot based on the *EchoBot* template. Now you can run it locally and test it by using the Bot Framework Emulator (which should be installed on your system).

1. In the terminal pane, ensure that the current directory is the **TimeBot** folder containing your bot code files, and then enter the following command to start your bot running locally.

C#

Code Copy

```
dotnet run
```

Python

Code Copy

```
python app.py
```

When the bot starts, note the endpoint at which it is running is shown. This should be similar to **http://localhost:3978**.

1. Start the Bot Framework Emulator, and open your bot by specifying the endpoint with the **/api/messages** path appended, like this:

   `http://localhost:3978/api/messages`
2. After the conversation is opened in a **Live chat** pane, wait for the message *Hello and welcome!*.
3. Enter a message such as *Hello* and view the response from the bot, which should echo back the message you entered.
4. Close the Bot Framework Emulator and return to Visual Studio Code, then in the terminal window, enter **CTRL+C** to stop the bot.

Modify the bot code

You've created a bot that echoes the user's input back to them. It's not particularly useful, but serves to illustrate the basic flow of a conversational dialog. A conversation with a bot consists of a sequence of *activities*, in which text, graphics, or user interface *cards* are used to exchange information. The bot begins the conversation with a greeting, which is the result of a *conversation update* activity that is triggered when a user initializes a chat session with the bot. Then the conversation consists of a sequence of further activities in which the user and bot take it in turns to send *messages*.

1. In Visual Studio Code, open the following code file for your bot:

   - **C#**: TimeBot/Bots/EchoBot.cs
   - **Python**: TimeBot/bot.py

   Note that the code in this file consists of *activity handler* functions; one for the *Member Added* conversation update activity (when someone joins the chat session) and another for the *Message* activity (when a message is received). The conversation is based on the concept of *turns*, in which each turn represents an interaction in which the bot receives, processes, and responds to an activity. The *turn context* is used to track information about the activity being processed in the current turn.
2. At the top of the code file, add the following namespace import statement:

C#

Code Copy

```
using System;
```

Python

Code Copy

```
from datetime import datetime
```

1. Modify the activity handler function for the *Message* activity to match the following code:

C#

Code Copy

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationToken cancellationToken)
{
    string inputMessage = turnContext.Activity.Text;
    string responseMessage = "Ask me what the time is.";
    if (inputMessage.ToLower().StartsWith("what") && inputMessage.ToLower().Contains("time"))
    {
        var now = DateTime.Now;
        responseMessage = "The time is " + now.Hour.ToString() + ":" + now.Minute.ToString("D2");
    }
    await turnContext.SendActivityAsync(MessageFactory.Text(responseMessage, responseMessage),
cancellationToken);
}
```

Python

Code Copy

```
async def on_message_activity(self, turn_context: TurnContext):
    input_message = turn_context.activity.text
    response_message = 'Ask me what the time is.'
    if (input_message.lower().startswith('what') and 'time' in input_message.lower()):
        now = datetime.now()
        response_message = 'The time is {}:{:02d}.'.format(now.hour,now.minute)
    await turn_context.send_activity(response_message)
```

1. Save your changes, and then in the terminal pane, ensure that the current directory is the **TimeBot** folder containing your bot code files, and then enter the following command to start your bot running locally.

C#

Code Copy

```
dotnet run
```

Python

Code Copy

```
python app.py
```

As before, when the bot starts, note the endpoint at which it is running is shown.

1. Start the Bot Framework Emulator, and open your bot by specifying the endpoint with the **/api/messages** path appended, like this:

   `http://localhost:3978/api/messages`
2. After the conversation is opened in a **Live chat** pane, wait for the message *Hello and welcome!*.
3. Enter a message such as *Hello* and view the response from the bot, which should be *Ask me what the time is*.
4. Enter *What is the time?* and view the response.

   The bot now responds to the query “What is the time?” by displaying the local time where the bot is running. For any other query, it prompts the user to ask it what the time is. This is a very limited bot, which could be improved through integration with the Language Understanding service and additional custom code, but it serves as a working example of how you can build a solution with the Bot Framework SDK by extending a bot created from a template.

5. Close the Bot Framework Emulator and return to Visual Studio Code, then in the terminal window, enter **CTRL+C** to stop the bot.

More information

To learn more about the Bot Framework, view the Bot Framework documentation.