Create a bot with Bot Service

Bot Service provides the core components for creating bots, including the Bot Builder SDK for developing bots and the Bot Framework for connecting bots to channels. Bot Service provides five templates you can choose from when creating your bots with support for .NET and Node.js. In this topic, learn how to use Bot Service to create a new bot that uses the Bot Builder SDK.

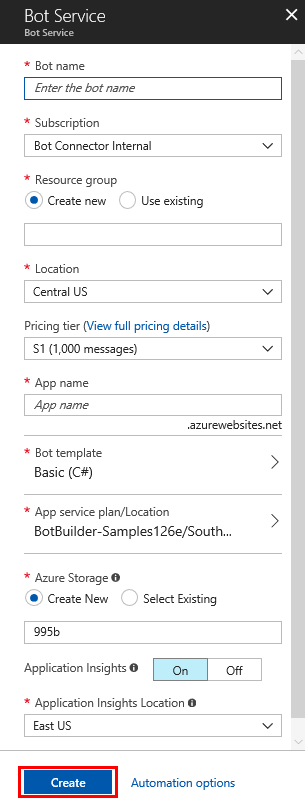
## **Log in to Azure**

Log in to the [Azure portal](http://portal.azure.com/).

Tip

If you do not already have a subscription, you can register for a [free account](https://azure.microsoft.com/en-us/free/).

## **Create a new bot service**

1. Click the **New** button found on the upper left-hand corner of the Azure portal, then select **AI + Cognitive Services > Web App bot**.
2. A new blade will open with information about the **Web App Bot**. Click the **Create**button to start the bot creation process.
3. In the **Bot Service** blade, provide the requested information about your bot as specified in the table below the image.   
   

| Setting | Suggested value | Description |
| --- | --- | --- |
| **Bot name** | Your bot's display name | The display name for the bot that appears in channels and directories. This name can be changed at anytime. |
| **Subscription** | Your subscription | Select the Azure subscription you want to use. |
| **Resource Group** | myResourceGroup | You can create a new [resource group](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview#resource-groups) or choose from an existing one. |
| **Location** | The default location | Select the geographic location for your resource group. Your location choice can be any location listed, though it's often best to choose a location closest to your customer. The location cannot be changed once the bot is created. |
| **Pricing tier** | F0 | Select a pricing tier. You may update the pricing tier at any time. For more information, see [Bot Service pricing](https://azure.microsoft.com/en-us/pricing/details/bot-service/). |
| **App name** | A unique name | The unique URL name of the bot. For example, if you name your bot myawesomebot, then your bot's URL will be http://myawesomebot.azurewebsites.net. The name must use alphanumeric and underscore characters only. There is a 35 character limit to this field. The App name cannot be changed once the bot is created. |
| **Bot template** | Basic | Choose either **C#** or **Node.js** and select the **Basic**template for this quickstart, then click **Select**. The Basic template creates an echo bot. [Learn more](https://docs.microsoft.com/en-us/bot-framework/bot-service-concept-templates) about the templates. |
| **App service plan/Location** | Your app service plan | Select an [app service plan](https://azure.microsoft.com/en-us/pricing/details/app-service/plans/) location. Your location choice can be any location listed, though it's often best to choose a location closest to your customer. (Not available for Functions Bot.) |
| **Azure Storage** | Your Azure storage account | You can create a new data storage account or use an existing one. By default, the bot will use [Table Storage](https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction#table-storage). |
| **Application Insights** | On | Decide if you want to turn [Application Insights](https://docs.microsoft.com/en-us/bot-framework/bot-service-manage-analytics) **On** or **Off**. If you select **On**, you must also specify a regional location. Your location choice can be any location listed, though it's often best to choose a location closest to your customer. |

1. Note
2. If you are creating a **Functions Bot**, you will not see an **App service plan/Location** field. Instead, you will see a Hosting plan field. In that case, choose a [Hosting plan](https://docs.microsoft.com/en-us/bot-framework/bot-service-overview-readme#hosting-plans).
3. Click **Create** to create the service and deploy the bot to the cloud. This process may take several minutes.

Confirm that the bot has been deployed by checking the **Notifications**. The notifications will change from **Deployment in progress...** to **Deployment succeeded**. Click **Go to resource**button to open the bot's resources blade.

Tip

For performance reasons, **Functions Bot** running Node.js templates have already been packaged using the Azure Functions Pack tool. The Azure Functions Pack tool takes all the Node.js modules and combined them into one \*.js file. For more information, see [Azure Functions Pack](https://github.com/Azure/azure-functions-pack).

## **Test the bot**

Now that your bot is created, [test it in Web Chat](https://docs.microsoft.com/en-us/bot-framework/bot-service-manage-test-webchat). Enter a message and your bot should respond

# Create a bot with the Bot Builder SDK for .NET

The [Bot Builder SDK for .NET](https://github.com/Microsoft/BotBuilder" \t "_blank) is an easy-to-use framework for developing bots using Visual Studio and Windows. The SDK leverages C# to provide a familiar way for .NET developers to create powerful bots.

This tutorial walks you through building a bot by using the Bot Application template and the Bot Builder SDK for .NET, and then testing it with the Bot Framework Emulator.

Important

The Bot Builder SDK for .NET currently supports C#. Visual Studio for Mac is not supported.

## **Prerequisites**

Get started by completing the following prerequisite tasks:

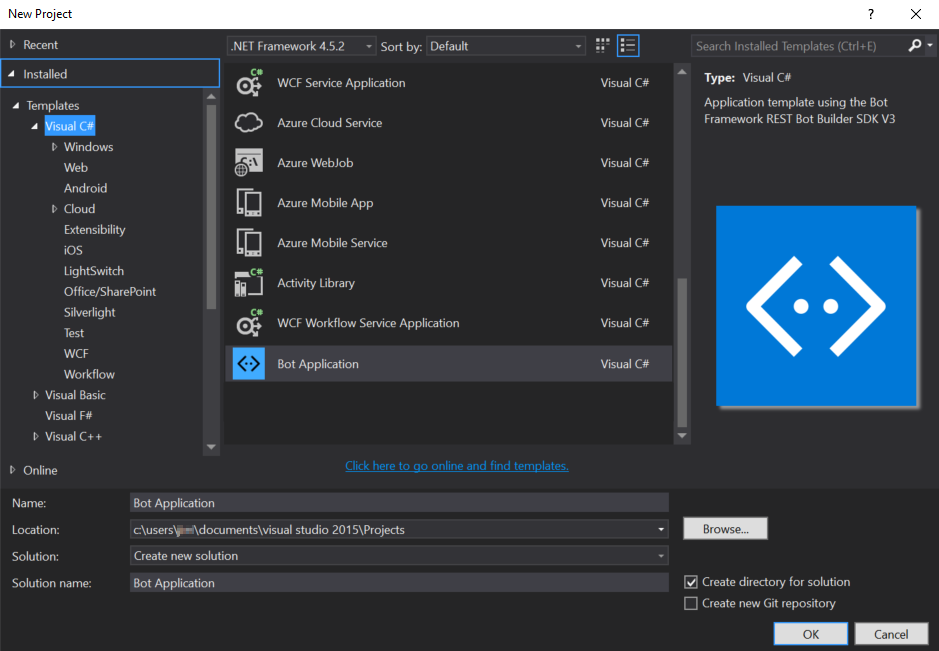
1. Install [Visual Studio 2017](https://www.visualstudio.com/downloads/) for Windows.
2. In Visual Studio, [update all extensions](https://docs.microsoft.com/en-us/visualstudio/extensibility/how-to-update-a-visual-studio-extension) to their latest versions.
3. Download the [Bot Application](http://aka.ms/bf-bc-vstemplate), [Bot Controller](http://aka.ms/bf-bc-vscontrollertemplate), and [Bot Dialog](http://aka.ms/bf-bc-vsdialogtemplate) .zip files. Install the project template by copying Bot Application.zip to your Visual Studio 2017 project templates directory. Install the item templates by copying Bot Controller.zip and Bot Dialog.zip to your Visual Studio 2017 item templates directory.

Tip

The Visual Studio 2017 project templates directory is typically located at %USERPROFILE%\Documents\Visual Studio 2017\Templates\ProjectTemplates\Visual C#\ and the item templates directory is at %USERPROFILE%\Documents\Visual Studio 2017\Templates\ItemTemplates\Visual C#\5

## **Create your bot**

Next, open Visual Studio and create a new C# project. Choose the Bot Application template for your new project.



Note

Visual Studio might say you need to [download and install IIS Express](https://www.microsoft.com/en-us/download/details.aspx?id=48264).

By using the Bot Application template, you're creating a project that already contains all of the components that are required to build a simple bot, including a reference to the Bot Builder SDK for .NET, Microsoft.Bot.Builder. Verify that your project references the latest version of the SDK:

1. Right-click on the project and select **Manage NuGet Packages**.
2. In the **Browse** tab, type "Microsoft.Bot.Builder".
3. Locate the Microsoft.Bot.Builder package in the list of search results, and click the **Update** button for that package.
4. Follow the prompts to accept the changes and update the package.

Thanks to the Bot Application template, your project contains all of the code that's necessary to create the bot in this tutorial. You won't actually need to write any additional code. However, before we move on to testing your bot, take a quick look at some of the code that the Bot Application template provided.

## **Explore the code**

First, the Post method within **Controllers\MessagesController.cs** receives the message from the user and invokes the root dialog.

C#Copy

[BotAuthentication]

public class MessagesController : ApiController

{

/// <summary>

/// POST: api/Messages

/// Receive a message from a user and reply to it

/// </summary>

public async Task<HttpResponseMessage> Post([FromBody]Activity activity)

{

if (activity.Type == ActivityTypes.Message)

{

await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());

}

else

{

HandleSystemMessage(activity);

}

var response = Request.CreateResponse(HttpStatusCode.OK);

return response;

}

...

}

The root dialog processes the message and generates a response. The MessageReceivedAsyncmethod within **Dialogs\RootDialog.cs** sends a reply that echos back the user's message, prefixed with the text 'You sent' and ending in the text 'which was ## characters', where ##represents the number of characters in the user's message.

C#Copy

[Serializable]

public class RootDialog : IDialog<object>

{

public Task StartAsync(IDialogContext context)

{

context.Wait(MessageReceivedAsync);

return Task.CompletedTask;

}

private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<object> result)

{

var activity = await result as Activity;

// calculate something for us to return

int length = (activity.Text ?? string.Empty).Length;

// return our reply to the user

await context.PostAsync($"You sent {activity.Text} which was {length} characters");

context.Wait(MessageReceivedAsync);

}

}

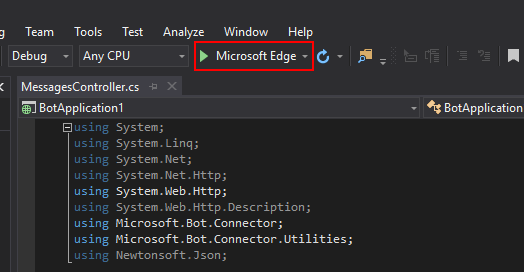
## **Test your bot**

Next, test your bot by using the [Bot Framework Emulator](https://docs.microsoft.com/en-us/bot-framework/bot-service-debug-emulator) to see it in action. The emulator is a desktop application that lets you test and debug your bot on localhost or running remotely through a tunnel.

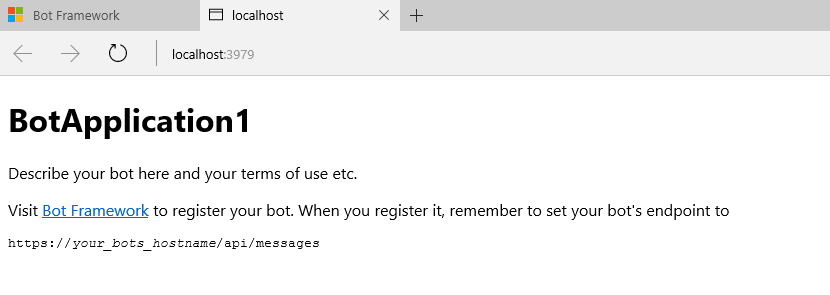
First, you'll need to download and install the emulator. Click [here](https://emulator.botframework.com/) to download the emulator. After the download completes, launch the executable and complete the installation process.

### **Start your bot**

After installing the emulator, start your bot in Visual Studio by using a browser as the application host. This Visual Studio screenshot shows that the bot will launch in Microsoft Edge when the run button is clicked.



When you click the run button, Visual Studio will build the application, deploy it to localhost, and launch the web browser to display the application's **default.htm** page. For example, here's the application's **default.htm** page shown in Microsoft Edge:



Note

You can modify the **default.htm** file within your project to specify the name and description of your bot application.1

### **Start the emulator and connect your bot**

At this point, your bot is running locally. Next, start the emulator and then connect to your bot in the emulator:

1. Type http://localhost:port-number/api/messages into the address bar, where **port-number** matches the port number shown in the browser where your application is running.
2. Click **Connect**. You won't need to specify **Microsoft App ID** and **Microsoft App Password**. You can leave these fields blank for now. You'll get this information later when you [register your bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-quickstart-registration).

1

Tip

In the example shown above, the application is running on port number **3979**, so the emulator address would be set to: http://localhost:3979/api/messages.

### **Test your bot**

Now that your bot is running locally and is connected to the emulator, test your bot by typing a few messages in the emulator. You should see that the bot responds to each message you send by echoing back your message prefixed with the text 'You sent' and ending with the text 'which was ## characters', where ## is the total number of characters in the message that you sent.

Tip3

In the emulator, click on any speech bubble in your conversation. Details about the message will appear in the Details pane, in JSON format.

You've successfully created a bot by using the Bot Application template Bot Builder SDK for .NET!

# Create a bot with the Bot Builder SDK for Node.js

The Bot Builder SDK for Node.js is a framework for developing bots. It is easy to use and models frameworks like Express & restify to provide a familiar way for JavaScript developers to write bots.

This tutorial walks you through building a bot by using the Bot Builder SDK for Node.js. You can test the bot in a console window and with the Bot Framework Emulator.

## **Prerequisites**

Get started by completing the following prerequisite tasks:

1. Install [Node.js](https://nodejs.org/).
2. Create a folder for your bot.
3. From a command prompt or terminal, navigate to the folder you just created.
4. Run the following **npm** command:

1

NodeJSCopy

npm init

Follow the prompt on the screen to enter information about your bot and npm will create a **package.json** file that contains the information you provided.3

## **Install the SDK**

Next, install the Bot Builder SDK for Node.js by running the following **npm** command:

NodeJSCopy

npm install --save botbuilder

Once you have the SDK installed, you are ready to write your first bot.

For your first bot, you will create a bot that simply echoes back any user input. To create your bot, follow these steps:

1. In the folder that you created earlier for your bot, create a new file named **app.js**.
2. Open **app.js** in a text editor or an IDE of your choice. Add the following code to the file:

JavaScriptCopy

var builder = require('botbuilder');

var connector = new builder.ConsoleConnector().listen();

var bot = new builder.UniversalBot(connector, function (session) {

session.send("You said: %s", session.message.text);

});

1. Save the file. Now you are ready to run and test out your bot.

### **Start your bot**

Navigate to your bot's directory in a console window and start your bot:2

NodeJSCopy

node app.js

Your bot is now running locally. Try out your bot by typing a few messages in the console window. You should see that the bot responds to each message you send by echoing back your message prefixed with the text "You said:".

## **Install Restify**

Console bots are good text-based clients, but in order to use any of the Bot Framework channels (or run your bot in the emulator), your bot will need to run on an API endpoint. Install [restify](http://restify.com/" \t "_blank) by running the following **npm** command:

NodeJSCopy

npm install --save restify

Once you have Restify in place, you're ready to make some changes to your bot.

## **Edit your bot**

You will need to make some changes to your **app.js** file.

1. Add a line to require the restify module.
2. Change the ConsoleConnector to a ChatConnector.
3. Include your Microsoft App ID and App Password.
4. Have the connector listen on an API endpoint.

JavaScriptCopy

var restify = require('restify');

var builder = require('botbuilder');

// Setup Restify Server

var server = restify.createServer();

server.listen(process.env.port || process.env.PORT || 3978, function () {

console.log('%s listening to %s', server.name, server.url);

});

// Create chat connector for communicating with the Bot Framework Service

var connector = new builder.ChatConnector({

appId: process.env.MICROSOFT\_APP\_ID,

appPassword: process.env.MICROSOFT\_APP\_PASSWORD

});

// Listen for messages from users

server.post('/api/messages', connector.listen());

// Receive messages from the user and respond by echoing each message back (prefixed with 'You said:')

var bot = new builder.UniversalBot(connector, function (session) {

session.send("You said: %s", session.message.text);

});

1. Save the file. Now you are ready to run and test out your bot in the emulator.

Note

You do not need a **Microsoft App ID** or **Microsoft App Password** to run your bot in the Bot Framework Emulator.

## **Test your bot**

Next, test your bot by using the [Bot Framework Emulator](https://docs.microsoft.com/en-us/bot-framework/bot-service-debug-emulator) to see it in action. The emulator is a desktop application that lets you test and debug your bot on localhost or running remotely through a tunnel.1

First, you'll need to [download](https://emulator.botframework.com/) and install the emulator. After the download completes, launch the executable and complete the installation process.1

### **Start your bot**

After installing the emulator, navigate to your bot's directory in a console window and start your bot:

NodeJSCopy

node app.js

Your bot is now running locally.

### **Start the emulator and connect your bot**

After you start your bot, connect to your bot in the emulator:

1. Type http://localhost:3978/api/messages into the address bar. (This is the default endpoint that your bot listens to when hosted locally.)
2. Click **Connect**. You won't need to specify **Microsoft App ID** and **Microsoft App Password**. You can leave these fields blank for now. You'll get this information later when you [register your bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-quickstart-registration).

### **Try out your bot**

Now that your bot is running locally and is connected to the emulator, try out your bot by typing a few messages in the emulator. You should see that the bot responds to each message you send by echoing back your message prefixed with the text "You said:".

You've successfully created your first bot using the Bot Builder SDK for Node.js!

# Register a bot with Bot Service

If you already have a bot hosted elsewhere and you would like to use the Bot Service to connect it to other channels, you will need to register your bot with the Bot Service. In this topic, learn how to register your bot with the Bot Service by creating a **Bot Channels Registration** bot service.

Important

You only need to register your bot if it is not hosted in Azure. If you [created a bot](https://docs.microsoft.com/en-us/bot-framework/bot-service-quickstart) through the Azure portal then your bot is already registered with the Bot Service.

## **Log in to Azure**

Log in to the [Azure portal](http://portal.azure.com/).

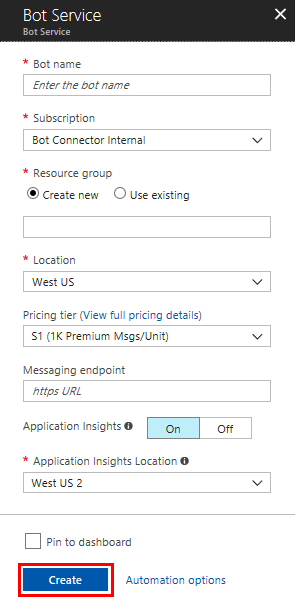
Tip

If you do not already have a subscription, you can register for a [free account](https://azure.microsoft.com/en-us/free/).

## **Create a Bot Channels Registration**

You need a **Bot Channels Registration** bot service to be able to use Bot Service functionality. A registration bot lets you connect your bot to channels.

To create a **Bot Channels Registration**, do the following:

1. Click the **New** button found on the upper left-hand corner of the Azure portal, then select **AI + Cognitive Services > Bot Channels Registration**.
2. A new blade will open with information about the **Bot Channels Registration**. Click the **Create** button to start the creation process.
3. In the **Bot Service** blade, provide the requested information about your bot as specified in the table below the image.   
   

| Setting | Suggested value | Description |
| --- | --- | --- |
| **Bot name** | Your bot's display name | The display name for the bot that appears in channels and directories. This name can be changed at anytime. |
| **Subscription** | Your subscription | Select the Azure subscription you want to use. |
| **Resource Group** | myResourceGroup | You can create a new [resource group](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview#resource-groups) or choose from an existing one. |
| Location | West US | Choose a location near where your bot is deployed or near other services your bot will access. |
| **Pricing tier** | F0 | Select a pricing tier. You may update the pricing tier at any time. For more information, see [Bot Service pricing](https://azure.microsoft.com/en-us/pricing/details/bot-service/). |
| **Messaging endpoint** | URL | Enter the URL for your bot's messaging endpoint. |
| **Application Insights** | On | Decide if you want to turn [Application Insights](https://docs.microsoft.com/en-us/bot-framework/bot-service-manage-analytics) **On** or **Off**. If you select **On**, you must also specify a regional location. |

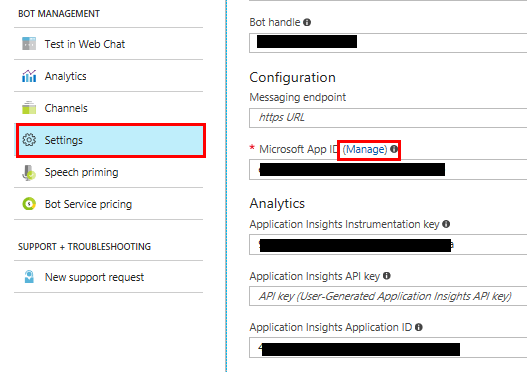
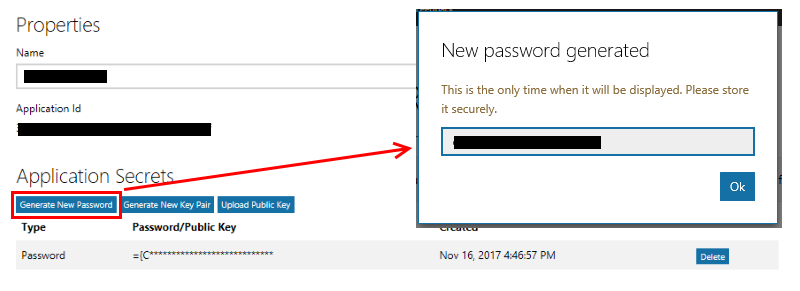
1. Click **Create** to create the service and register your bot's messaging end point.

Confirm that the registration has been created by checking the **Notifications**. The notifications will change from **Deployment in progress...** to **Deployment succeeded**. Click **Go to resource**button to open the bot's resources blade.

## **Bot Channels Registration password**

**Bot Channels Registration** bot service does not have an app service associated with it. Because of that, this bot service only has a MicrosoftAppID. You need to generate the password manually and save it yourself. You will need this password if you want to test your bot using the [emulator](https://docs.microsoft.com/en-us/bot-framework/bot-service-debug-emulator).

To generate a MicrosoftAppPassword, do the following:

1. From the **Settings** blade, click **Manage**. This is the link appearing by the **Microsoft App ID**. This link will open a window where you can generate a new password.   
   
2. Click **Generate New Password**. This will generate a new password for your bot. Copy this password and save it to a file. This is the only time you will see this password. If you do not have the full password saved, you will need to repeat the process to create a new password should you need it later.   
   

## **Test the bot**

Now that your bot service is created, [test it in Web Chat](https://docs.microsoft.com/en-us/bot-framework/bot-service-manage-test-webchat). Enter a message and your bot should respond.