Create a Bot, check-in to Team Services (creates GitHub repository)

# Create a Bot

1. <https://portal.azure.com>
2. Sign in with your MSDN account
3. Click New > search for bot > Click Bot Service (preview) > Create
4. Fill out the form
   1. App name: {MyBot}
   2. Subscription: {Visual Studio Ultimate with MSDN}
   3. Resource Group: {create or use existing}
   4. Hosting Plan: {App Service Plan}
   5. App Service plan/Location: {location}
5. Choose a template
   1. C#
   2. Language understanding
   3. Next
6. Create a Microsoft App ID
   1. Click Create a Microsoft App ID
   2. Sign in with your MSDN account
   3. Record the App name and App ID
   4. Click Generate an app password to continue
   5. Record the App password
   6. Click Finish and go back to Bot Framework
      1. If you actually get redirected back to the Bot Framework, instead go back to your portal.azure.com tab
7. Paste the App password
8. Check the two terms of use check boxes
   1. Be aware the terms are the same as Cognitive Services as mentioned above
9. Click Create Bot
10. Follow the on-screen instruction to create, register, and deploy the bot
    1. It will take a few minutes
11. Choose how to work with your code
    1. Continuous deployment from source control
    2. Fill out the form
       1. Source control provider: Visual Studio Team Services
       2. Url to your Visual Studio Team Services site
          1. Click View Visual Studio Profile to open your VSTS site, so you can copy the url for the account you want to use
          2. Ex. https://something.visualstudio.com
       3. Access Token
          1. [Create an access token](https://docs.microsoft.com/en-us/vsts/accounts/use-personal-access-tokens-to-authenticate) and assign it "Project and team (read, write, and manage" and "Code (full)" scope.
          2. Record the token
       4. Click Enable
       5. Click Got it
12. Continuous deployment from source control
    1. Click Configure continuous deployment
       1. Doesn't look like you need to do anything.
13. We'll download the source locally and open in VS2017 later.

# Verify your bot was registered

1. <https://dev.botframework.com/>
2. Sign in with your MSDN account > Click My bots
3. Your bot, {MyBot}, should be in the list

# Troubleshooting – 401 access denied error

The bot service is creating the Luis.ai service for you, so these steps are done for you automatically now. While you can still create your own Luis.ai service with these steps, you would need to update the bot application settings with the new LuisAppId and LuisAPIKey. However, I had nothing but problems with this as I got 401 access denied errors.

Create a Luis.ai Service

<http://aihelpwebsite.com/Blog/EntryId/4/Creating-Intelligent-Web-Applications-With-LUIS>

You need to create a Luis.ai Service with the free pricing tier (F0) if you want 5 calls per second and 10K calls per month. The Luis.ai app defaults to a starter key of 1K calls per month.

# Create an Azure Luis Cognitive Service

You need to create the Luis service to host your bot.

1. <https://portal.azure.com>
   1. If you don't have an account, then sign up for a free MSDN account
2. Click New > search: luis > click Language Understanding Intelligent Service (preview)
3. Read the Legal Notice, because the standard terms and conditions do not apply and they will use your data to improve the service and expect you to get approval from users.
4. Click Create
5. Fill out the form (examples below)
   1. Name: {MyLuis}
   2. Subscription: Visual Studio Ultimate with MSDN
   3. Location: East US 2
   4. Pricing tier: F0 (free: 5 calls per second; 10K calls per month; limit of 1 free account)
   5. Resource group: {create one if it doesn't already exist}
   6. Legal: check
   7. Pin to dashboard: check
6. Click Create
7. Wait for the service to be deployed

If the Luis.ai is properly bound to your MSDN account, you should just have to click the dropdowns now as noted later in the Luis.ai app Publish steps.

# ~~Get an Access Key to your Luis service~~

~~You need an access key to grant the Luis.ai access to your Azure Luis service.~~

1. [~~https://portal.azure.com~~](https://portal.azure.com)
2. ~~On the dashboard click on your Luis Cognitive Service~~
3. ~~Under the Essentials section > Manage keys > click Show access keys~~
4. ~~Record the guid for Key 1~~

# Create a Luis.ai account

You need to create an account and grant permissions.

1. <https://www.luis.ai>
2. Click Sign in
3. Sign in with your MSDN account
4. Grant permissions as requested

# ~~Create a new Luis.ai app~~

1. [~~https://www.luis.ai~~](https://www.luis.ai)
2. ~~Read/skip overview~~
3. ~~Create New App~~
4. ~~Fill out the form (examples below)~~
   1. ~~Name: MyLuis~~
   2. ~~Culture: English~~
   3. ~~Description: Test~~
5. ~~Click Create~~
6. ~~Record the App Id on the Dashboard~~

# Create a Luis.ai account

You need to create an account and grant permissions.

1. <https://www.luis.ai>
2. Click Sign in
3. Sign in with your MSDN account
4. Grant permissions as requested

# Luis Overview

1. Dashboard: Other than stats of your bots, this is where you find the App Id
2. Intent: An action you want to process
   1. Ex. Search, Define, Get status, Change status, etc.
3. Utterance: A phrase associated with the intent and a variable
4. Entities: Variables
5. Prebuilt domains (preview): Sample intents, utterances, and entities for common domains such as calendars, entertainments, gaming, reminders, places, web navigation, etc.
6. Features: Fine tune the decisions for choosing the correct intent and entities by giving specific examples of utterances.
7. Train & Test: Enter some utterances to verify you have trained it properly
8. Publish App: Publish your LUIS service to Staging or Production

# Create an intent, utterance, and entity

1. Click Intents > Add Intent > Intent name: "Greeting" > Save
2. Type the following utterances and press enter after each one
   1. Hi
3. Click Save
4. Click Intents > Add Intent > Intent name: "Define" > Save
5. Type the following utterances and press enter after each one
   1. Define DRY
   2. Definition of DRY
   3. Explain DRY
   4. What is DRY?
   5. Describe DRY
6. Click Save
7. Click Entities > Add custom entity > Entity name: "term" > Save
8. Click on Intents > click your intent: "Define"
9. For each utterance, select "dry" and then click "term", so they look as follows.
   1. Define [$term]
   2. Definition of [$term]
   3. Explain [$term]
   4. What is [$term]?
   5. Describe [$term]
   6. You can select multiple words by clicking the first word and then the last word. If you have multiple words selected and you want to unselect, then click an empty spot on the page.
10. Click Save

# Train and Test

1. Click Train & Test > Train Application
2. Type "hi" and verify the Intent is "Greeting"
3. Try out each utterance to see if the intent is "Define" and entity is [$term]
   1. Define DRY
   2. Definition of DRY
   3. Explain DRY
   4. What is DRY?
   5. Describe DRY
   6. Define test
4. What about some phrases we didn't add as utterances?
   1. Tell me about DRY
   2. Speak on DRY
   3. I want to know about DRY
   4. Help me understand DRY
5. All of these phrases work just as well. Keep in mind we only have one intent at this time, so after adding more intents you may need to fine toon the process under Features.

# Publish App and Test

1. Click Publish App > Click Assigned endpoint key (Change)
   1. If any of the information isn't loaded in the drop downs, try signing out and signing back into Luis.ai
   2. The 3 dropdown options only had one option for each
2. Fill out the form (examples below)
   1. Tenant Id: {[your organization id associated with your Azure account](https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-resources-faq)}
   2. Subscription Name: Visual Studio Ultimate with MSDN
   3. Key: MyLuis {name of your Azure Luis service}
3. Click Save
4. Endpoint Slot: Production
   1. For now choose Production, later publish everything in Staging until it is ready for production.
5. Click Train
6. Click Publish
7. Adjust the Timezone to your desired timezone
8. Click Publish
9. Copy the Endpoint url to notepad and add "Define DRY" to the end
   1. … q=Define DRY
10. Copy the update url to a browser and run it. You'll get a response like the following.

Download the source and update settings

# Get the Credentials

1. <https://portal.azure.com>
2. On the dashboard click on your {MyBot} > Click on Settings > Application settings: click Open > scroll down to App settings
3. Record the following credentials
   1. **MicrosoftAppId**: {guid}
   2. **MicrosoftAppPassword**: {password}
   3. **LuisAppId**: {guid}
   4. **LuisAPIKey**: {password}

# Get the Bot's solution and open in Visual Studio 2017

Assumption: You've already connected to your visualstudio.com site in VS2017. <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/luis-csharp-tutorial-build-bot-framework-sample>

1. Open VS2017 > Team Explorer > Manage Connections > Manage Connections > Connect to a Project > Expand the list to your Bot project > Click Connect
2. In the Solution Explorer double-click your .sln file

# Update the web.config with your credentials



# Update the BasicLuisDialog.cs



Bind to the Luis.ai service

# Get the localhost url

1. Open Visual Studio > connect to the GitHub repository
2. Run the Microsoft.Bot.Sample.LuisBot.sln
3. Record the localhost url noting the port number
   1. Ex. <http://localhost:3979/api/messages>

# Connect the Microsoft.Bot.Sample.LuisBot to your Luis.ai web service

1. Update the web.config



# Setup the BotFramework Emulator

1. <https://github.com/Microsoft/BotFramework-Emulator/>
2. Download
   1. <https://github.com/Microsoft/BotFramework-Emulator/releases>
      1. [botframework-emulator-Setup-3.5.31.exe](https://github.com/Microsoft/BotFramework-Emulator/releases/download/v3.5.31/botframework-emulator-Setup-3.5.31.exe)
3. Install the EXE
4. Run the Emulator from the shortcut added to your desktop
5. [Fill out the form](https://github.com/Microsoft/BotFramework-Emulator/wiki/Getting-Started#connect-to-an-azure-bot-service)
   1. Uri: {See previous sections}
      1. Ex. http://localhost:3979/api/messages
      2. Enter your endpoint URL from your localhost with the correct port
   2. Microsoft App ID: {See previous sections}
   3. Microsoft App Password: {See previous sections}

Debug

# Local

1. In VS2017 add breakpoints in your NoneIntent method and DefineIntent method
2. Run the application in debug mode (F5)
3. Run the Bot Emulator
4. Type: hi
   1. Response: Query: hi; Intent: Greeting; Entity: ;
5. Type: define dry
   1. Response: Query: define dry; Intent: Define; Entity: dry;

# Commit and Sync code

1. If everything is working up to this point, then commit your changes and sync the code
2. If the continuous deployment is setup correctly to your TFS GitHub repository, then the bot should have also been deployed automatically.

# Production

1. <https://portal.azure.com>
2. On the dashboard click on your {MyBot} > Click on Channels > Click Get bot embed codes > Click the Web Chat icon > Click "Click here to open the Web Chat configuration page" > Copy the Url from the Embed code iframe src
   1. Ex. https://webchat.botframework.com/embed/{MyBot}?s=YOUR\_SECRET\_HERE
3. Click Show for the first Secret key > Replace YOUR\_SECRET\_HERE with the secret key
4. Paste that url into a browser and the web chat should open
5. Type: hi
   1. Response: Query: hi; Intent: Greeting; Entity: ;
6. Type: define dry
   1. Response: Query: define dry; Intent: Define; Entity: dry;

# Resources

1. <https://docs.microsoft.com/en-us/bot-framework/azure-bot-service-template-language-understanding>