<https://github.com/annedroid/WTHAdvancedDatabricks/blob/master/README.md>

https://github.com/annedroid/WTHAdvancedDatabricks/AdvDatabricksChallenges.zip

DTA 2018 Hackathon: ML Challenge

Welcome to DTA 2018's ML/AI Challenge! We will focus on hands-on activities that develop proficiency in Azure Databricks ML-oriented libraries like MMLSpark and pySpark. These challenges assume an introductory to intermediate knowledge of these services, and if this is not the case, please spend time working through related resources available as part 1: Intro to Azure Databricks Workshop.

**Goals**

Most challenges observed by customers in these realms include stitching multiple services together across an end-to-end ML pipeline. As such, we have tried to place key concepts in the context of a broader example.

Once all hackathon challenges (Part 1 and 2) are completed, you should be able to:

* Provision an Azure Databricks cluster
* Build an application that calls various Cognitive Services APIs (specifically Computer Vision)
* Effectively leverage the custom vision service to create image classification services that can then be leveraged by an application
* Implement Azure Search features to provide a positive search experience inside applications
* Configure an Azure Search service to extend your data to enable full-text, language-aware search
* Build, train, and publish a LUIS model to help your bot communicate effectively
* Build an intelligent bot using Microsoft Bot Framework that leverages LUIS and Azure Search
* Effectively log chat conversations in your bot
* Perform rapid development/testing with Ngrok and test your bots with unit tests and direct bot communication

**Getting Started**

Background Knowledge

This workshop is meant for an AI Developer on Azure. Since our time today is limited, there are certain things you will need to read or setup after you arrive. If you do not have this background knowledge, please work closely with your team to learn from others or use the links below.

* **Visual Studio**   
  Previous exposure to Visual Studio will be helpful. Your team will be using it for everything we are building today, so you should be familiar with [how to use it](https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide) to create applications. We assume each team will have some familiarity with C# (intermediate level - you can learn [here](https://mva.microsoft.com/en-us/training-courses/c-fundamentals-for-absolute-beginners-16169?l=Lvld4EQIC_2706218949)), but you do not know how to implement solutions with Cognitive Services.
* **Bot Framework**   
  You should have some experience developing bots with Microsoft's Bot Framework ([https://dev.botframework.com](https://dev.botframework.com/)). We won't spend a lot of time discussing how to design them or how dialogs work.
* **Azure Portal**   
  You should have experience with the Azure portal ([https://portal.azure.com](https://portal.azure.com/)) and understand how to create resource groups and configure individual services.

Pre-requisites

This is a list of pre-requisites needed to successfully complete the challenges.  Some of these are items to deploy to your development machine.  Some are decisions you should discuss and define as a team, like the language to use for development.

1. **Azure Account**   
   You must have an Azure account to complete the hackathon. Either use your existing subscription or setup a free trial to complete today’s challenges. We will not be providing Azure passes for this workshop.

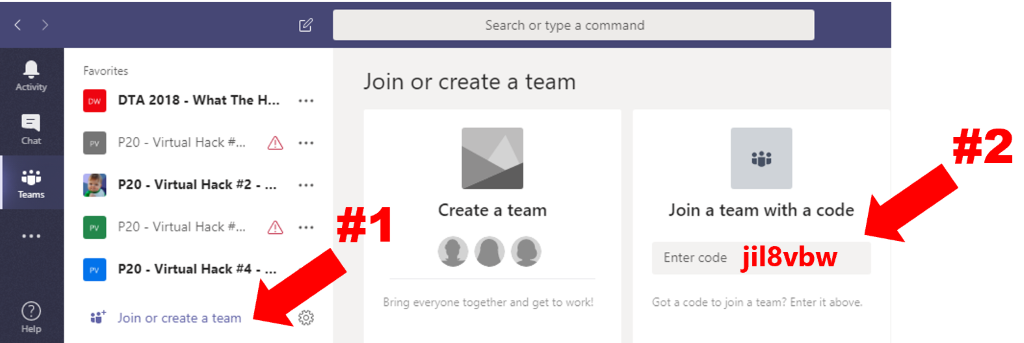
1. **Custom Vision Training Key**   
   The training API key allows you to create, manage and train Custom Vision project programmatically.  You can obtain a key by creating a new project at [https://customvision.ai](https://customvision.ai/) and then clicking on the “setting” gear in the top right.

Working together

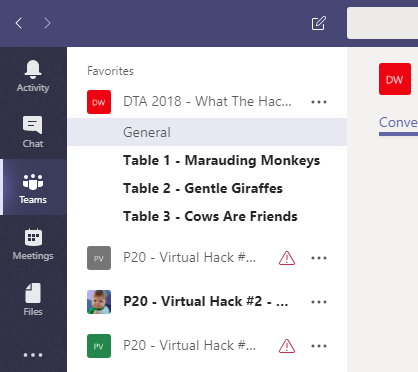
Before you get started on today’s challenges, please take a few minutes to get to know your team.

* Understand each team members AI background and experience
* Will your team work together or individually?
* Determine Azure Subscription you will use for deployment
* Optional: Git for Windows (<https://gitscm.com/downloads>) – allows for working with Git repos locally on your machine as well as in VSTS. This may be helpful, if you are creating team code that you want to access after the hack event is over.

Please run Teams and join the [DTA 2018 – What the Hack – AI](https://teams.microsoft.com/l/team/19%3a9ed23ced7d994ee89e1da444ba3a19a0%40thread.skype/conversations?groupId=4381f3f4-b3ce-4e03-a451-1dca28e69c51&tenantId=72f988bf-86f1-41af-91ab-2d7cd011db47) team site using code: **jil8vbw**…



Once you have joined the team, get together as a team and think of a team name. Create a new channel with that name prefixed with your table number, so you have a place to collaborate, share helpful notes, and store files during today’s hack.

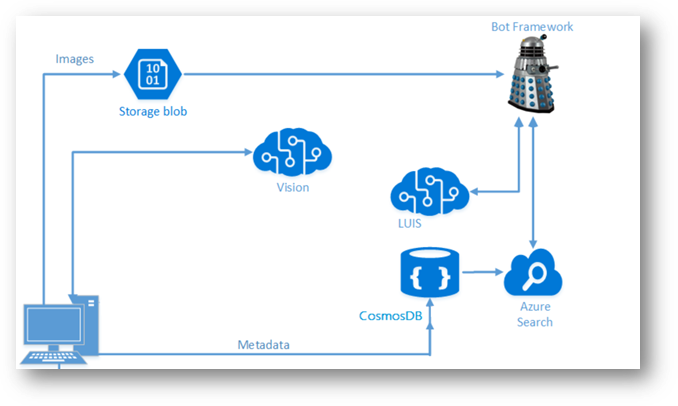


**Challenges**

Your team’s mission today is to learn more about AI capabilities through hands-on practice by completing challenges in two key areas: Cognitive Services and Bots.

Your team will start by building a simple C# application that allows you to ingest pictures from your local drive, then invoke the [Computer Vision API](https://www.microsoft.com/cognitive-services/en-us/computer-vision-api) to analyze the images and obtain tags and a description. Once you have this data, you will process it to pull out the details we need, and store it all into [Cosmos DB](https://azure.microsoft.com/en-us/services/cosmos-db).

You'll continue by build an [Azure Search](https://azure.microsoft.com/en-us/services/search/) Index (Azure Search is our PaaS offering for faceted, fault-tolerant search) on top of Cosmos DB, then you’ll build a [Bot Framework](https://dev.botframework.com/) bot to query it. Finally, you'll extend this bot with [Language Understanding (LUIS)](https://www.microsoft.com/cognitive-services/en-us/language-understanding-intelligent-service-luis) to automatically derive intent from your queries and use those to direct your searches intelligently.

During the challenges below, you will find it helpful to keep track of your various keys in a text file.

Cognitive Services Keys

* Computer Vision API:

Storage Keys

* Azure Blob Storage Connection String:
* Cosmos DB URI:
* Cosmos DB key:

Custom Vision

* Training Key
* Prediction Key

Language Understanding

* LUIS API:

Azure Search

* Azure Search Name:
* Azure Search Key:

Bot App

* Bot App Name:
* Bot App ID:
* Bot App Password:

We understand time today is very limited, so just complete what you can. Remember to work closely with your team and share any tips to the “Shared Tips” area under the General channel in the [DTA 2018 – What the Hack – AI](https://teams.microsoft.com/l/team/19%3a9ed23ced7d994ee89e1da444ba3a19a0%40thread.skype/conversations?groupId=4381f3f4-b3ce-4e03-a451-1dca28e69c51&tenantId=72f988bf-86f1-41af-91ab-2d7cd011db47) team site.

Part 1: Cognitive Services

This challenge involves building an end-to-end application that allows you to pull in your own pictures, use Cognitive Services to obtain a caption and some tags about the images, and then store that information in Cosmos DB. Build on your work by using the NoSQL store (Cosmos DB) to populate an Azure Search index, and then build a Bot Framework bot using LUIS to allow easy, targeted querying.

**Challenge 1.1: Create a Cognitive Services App using Portable Class Libraries**

Build an application that allows you to:

1. Pull in your own pictures and store them into Blob Storage.  Sample pictures can be found in the Team site under the General channel files in the challenge1.1-computer\_vision\sample\_images folder.
2. Use the Cognitive Services Computer Vision API to obtain a caption and some tags about the images, and then store that information in Cosmos DB.   
   *Note: In later challenges, we will use the NoSQL Store (Cosmos DB) to populate an Azure Search index, and then build a Bot Framework bot using LUIS to allow easy, targeted querying.*

*Challenge considerations:*

* A Data Science Virtual Machine may reduce setup.
* Refer to the documentation to understand your implementation options: <https://docs.microsoft.com/en-us/azure/cognitive-services/>
* You will need to create a Computer Vision API key.
* Microsoft Azure Storage Explorer ([http://storageexplorer.com](http://storageexplorer.com/)) be useful for validating content stored in Blob Storage and Cosmos DB.
* *Need help? Since time is limited, if your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge1.1-computer\_vision folder. See the 0\_README.md file for guidance.*

**Challenge 1.2: Create an Image Classification Application using the Custom Vision Service - Trees**

Create a basic application that uses the Custom Vision API to:

1. Create a console application and prepare the training key and sample images.   
   Hint: Include two helper methods:

* A method called `GetTrainingKey` to prepare the training key.
* A method called `LoadImagesFromDisk` to load two sets of images that will be used to train the project and one test image that the example loads to demonstrate the use of the default prediction endpoint.

1. Create a Custom Vision Service project
2. Add custom tags "Hemlock" and "Japanese Cherry".
3. Create code that will upload images from the local disk. Images can be found in the Team site under the General channel files in the challenge1.2-customvision01\images folder.
4. Train the project using the sample tree images.
5. Use the model for prediction:

* Obtain the default prediction endpoint URL for the project.
* Use the endpoint to programmatically test an image.

*Challenge considerations:*

* In the Images folder are three folders:
* Hemlock
* Japanese Cherry
* Test

The Hemlock and Japanese Cherry folders contain images of these types of trees that will be trained and tagged. The Test folder contains an image that will be used to perform the test prediction.

* You can obtain a key by creating a new project at <https://customvision.ai> and then clicking on the "setting" gear in the top right.
* Refer to documentation:
* C#: Tutorial using Custom Vision SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/csharp-tutorial>
* Python: Tutorial using the Custom Vision Service SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/python-tutorial>
* *Need help? If your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge1.2-customvision01 folder. See the 0\_README.md file for guidance.*

**Challenge 1.3: Create an Image Classification Application using the Custom Vision Service – Vehicle Damage**

Create a basic application that uses the Custom Vision API to:

1. Create a console application and prepare the training key and the sample images.
2. Create a Custom Vision Service project.
3. Add custom tags “Dent” and “WriteOff”.
4. Add code to get and manage the training key

Hint: Create a method `GetTrainingKey` with two parameters of `trainingKey` with a data type of string, and a second parameter of args with the data type of string, using the value from the trainingkey variable. The code can include control of flow logic to either use the key if it already defined, or to prompt for the key should it be missing.

1. Add code to get and manage the prediction key

Hint: create a method `GetPredictionKey` with two parameters of `predictionKey` with a data type of string, and a second parameter of args with the data type of string, using the value from the predictionkey variable. The code can include control of flow logic to either use the key if it already defined, or to prompt for the key should it be missing.

1. Create code that will upload images from the local disk. Images can be found in the Team site under the General channel files in the challenge1.3-customvision02\images folder.
2. Train using the sample Dent and WriteOff images.
3. Use the model for prediction:

* Obtain the default prediction endpoint URL for the project.
* Use the endpoint to programmatically test an image.

*Challenge considerations:*

* In the Resources\Images folder are three folders:
* Dent
* writeoff
* test

The Dent and writeoff folders contain images of these types of damages to vehicles that will be trained and tagged. The Test folder contains an image that will be used to perform the test prediction.

* You can obtain a key by creating a new project at <https://customvision.ai> and then clicking on the "setting" gear in the top right.
* Refer to documentation:
* C#: Tutorial using Custom Vision SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/csharp-tutorial>
* Python: Tutorial using the Custom Vision Service SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/python-tutorial>
* *Need help? If your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge1.3-customvision02 folder. See the 0\_README.md file for guidance.*

**Challenge 1.4: Create an Image Classification Application using the Custom Vision Service – Bikes**

Create a basic application that uses the Custom Vision API to:

1. Create a console application and prepare the training key and the images needed for the example.
2. Add code to validate and manage the training key
3. Add code to validate and manage the prediction key
4. Create a Custom Vision Service project.
5. Add custom tags “Mountain” and “Racing”.
6. Create code that will upload sample bicycle images from the local disk. Images can be found in the Team site under the General channel files in the challenge1.4-customvision03\images folder.
7. Train using the sample images.
8. Use the model for prediction:

* Obtain the default prediction endpoint URL for the project.
* Use the endpoint to programmatically test an image.

Hint: Build and run the solution. You will be required to input your training API key into the console app when running the solution so have this at the ready. The training and prediction of the images can take 2 minutes. The prediction results appear on the console.

*Challenge considerations:*

* In the Resources\Images folder are three folders:
* Mountain
* Racing
* test

The Mountain and Racing folders contain images of these types of bicycles that will be trained and tagged. The Test folder contains an image that will be used to perform the test prediction.

* You can obtain a key by creating a new project at <https://customvision.ai> and then clicking on the "setting" gear in the top right.
* Refer to documentation:
* C#: Tutorial using Custom Vision SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/csharp-tutorial>
* Python: Tutorial using the Custom Vision Service SDK: <https://docs.microsoft.com/en-us/azure/cognitive-services/Custom-Vision-Service/python-tutorial>
* *Need help? If your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge1.4-customvision03 folder. See the 0\_README.md file for guidance.*

**Challenge 1.5: Develop Intelligent Applications with Language Understanding (LUIS)**

Create a simple intelligent application with Language Understanding (LUIS).

1. Create the Language Understanding service in the Azure portal
2. Create a new LUIS app

* Navigate to [https://www.luis.ai](https://www.luis.ai/) and sign in using the same Microsoft account as above.
* LUIS allows you to map natural language utterances (words/phrases/sentences the user might say when talking to the bot) to intents (tasks or actions the user wants to perform).  For your application, you might have several intents: finding pictures, sharing pictures, and ordering prints of pictures, for example.  Give a few example utterances as ways to ask for each of these things, and LUIS will map additional new utterances to each intent based on what it has learned.

1. Add the following functionality to your app:

* Search/find pictures
* Share pictures on social media
* Order prints of pictures
* Greet the user

*Challenge considerations:*

* Internet Explorer is not recommended when working with LUIS
* Below are the steps you will generally take when creating LUIS applications:

1. Add Intents: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/add-intents>
2. Add utterances: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/add-example-utterances>
3. Add entities: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/add-entities>
4. Improve performance using features: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/add-features>)
5. Train and test: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/train-test>
6. Use active learning: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/label-suggested-utterances>
7. Publish: <https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/publishapp>

* *Need help? If your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge1.5-luis folder. See the 0\_README.md file for guidance. If you fall behind and want to cheat, you can click the "Import App" button within the LUIS website and import the LUIS model (./resources/code/LUIS/PictureBotLuisModel.json).*

Finishing up Part 1: Cognitive Services Challenge

Your team has prepared a Cognitive Services proof of concept. Take a few minutes to discuss the following questions:

1. What are some potential ideas for how Custom Vision could be used to bring value?
2. Are there any additional Cognitive Services that you think could be used to help bring value to businesses?
3. What might your main intents in LUIS be?

Part 2: Bots

You are working with Contoso LLC, which sells bicycles and bicycle equipment to its customers. Contoso currently processes new product orders and queries through human operators and is starting to devise a plan to implement your proposed solution using bots. The solution will provide an automated approach that allows Contoso to seamlessly scale up to handle a large call volume while maintaining zero wait times and freeing up staff to manage other tasks.

**Challenge 2.1: Developing Intelligent Applications with Azure Search**

The example above illustrates some of the components’ users are expecting in their search experience. Azure Search can accomplish these user experience features, along with giving you monitoring and reporting, simple scoring, and tools for prototyping and inspection.  The workflow steps involved include the following:

1. Provision / Create a service (*hint*: portal or PowerShell can be used).
2. Create an index (*hint*: think of an ‘index’ like a container for data or ‘table’. It has a schema, CORS options and search options.  --> Can be done in the Portal or during app initialization.
3. Index your data (*hint:* Two ways to populate an index with data: Azure Search API or via a supported Data Source which you point your index to).
4. Search an Index using Search Options (including filter, sort, project and page over results).

*Challenge considerations:*

* F0 tier is sufficient when completing step (a) above – Keep in mind:

1. Only one free Azure Search service is allowed per subscription
2. Use one of the following regions: West US 2, East US, West Europe, Southeast Asia.

* *Need help? Since time is limited, if your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge2.1-azure\_search folder. See the 0\_README.md file for guidance.*

**Challenge 2.2: Building Intelligent Bots**

*Pre-requisite: 1.5 and 2.1 - Note: This challenge combines some of the results obtained from various challenges (Computer Vision, Azure Search, and LUIS) from earlier in this hackathon. If you did not complete the above listed challenges, you will need to complete the Azure Search and LUIS challenges before moving forwards. Alternatively, you can request to use a neighbor's keys from their Azure Search/LUIS challenges.*

This challenge involves creating an intelligent bot from end-to-end using the Microsoft Bot Framework, Azure Search, and Microsoft's Language Understanding Intelligent Service (LUIS).

1. Build an intelligent application named “PictureBot”.
2. Add Regular Expressions and Scorable Groups:    
   Add code to your bot that will match on expressions from the user that start with "hi", "hello", and "help".  When the user asks for help, present him/her with a simple menu of buttons on the three core things your bot can do: search pictures, share pictures, and order prints.
3. Configure your bot to connect to the Azure Search service from the previous challenge.   
   Hint: In the Bot Emulator, try searching for something and ensure that you are seeing results when tags from your pictures are requested.
4. Incorporate your LUIS model into your bot, so that you call LUIS when Regex does not recognize a user's intent.
5. Publish and register your bot.

*Challenge considerations:*

* The Microsoft Bot Framework (<http://botframework.com/>) provides what you need to build and connect intelligent bots that interact naturally wherever your users are talking. You may choose to utilize v3 SDK or v4 SDK (<https://github.com/Microsoft/botbuilder-dotnet>).
* *Need help? Since time is limited, if your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge2.2-building\_bots folder. See the 0\_README.md file for guidance.*

**Challenge 2.3: Log Chat Conversations in your Bot**

Unless your bot is logging the conversation data somewhere, the bot framework will not perform any logging for you automatically. This has privacy implications, and many bots simply can't allow that in their scenarios.

For this challenge, you will demonstrate how to perform logging using Microsoft Bot Framework to store chat conversations. More specifically, the aim of this challenge is to:

* Understand how to intercept and log message activities between bots and users.
* Log conversations to a file using global events and activity logger.
* Extend the logging to SQL DB using global events and activity logger.

To complete this challenge:

1. Activity Logger: Implement the IActivityLogger interface to write message activities when running in debug.
2. File Logger: Demonstrate how you can log conversations to a file using global events.
3. SQL Logger: Extend your file logger code from the previous task to log conversations in SQL.

*Challenge considerations:*

* *Need help? Since time is limited, if your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge2.3-logging\_chat\_conversations folder. See the 0\_README.md file for guidance.*

**Challenge 2.4: Testing your Bot**

Chatbots bring their own set of challenges to testing including testing across environments, integrating third party APIs, etc.

1. Use ngrok to perform rapid development/testing.    
   With Microsoft Bot Framework, to configure the bot to be available to a particular channel, you will need to host the Bot service on a public URL endpoint. The channel will not be able to access your bot service if it is on a local server port hidden behind a NAT or firewall. When designing / building / testing your code, you do not always want to have to keep redeploying. This will result in additional hosting costs. This is where ngrok can really help in speeding up the development/testing phases of bots. Use ngrok to expose your bot to public internet and use the public endpoints to test your bots in the emulator.
2. Perform mocking to unit test your bot code. Unit tests can help:

- Verify functionality as you add it

- Validate your components in isolation

- People unfamiliar with your code verify they haven't broken it when they are working with it   
Explore some of these scenarios.

1. Demonstrates how to communicate directly with your bot from a custom client using Direct Line.

*Challenge considerations:*

* The Bot Framework Emulator may be helpful for this challenge. To install the Bot Framework Emulator, download it from <https://emulator.botframework.com/>. Please refer to [[this documentation article](https://github.com/microsoft/botframework-emulator/wiki/Getting-Started) to learn more about the Bot Framework Emulator.
* If you do not have ngrok installed, download ngrok from <https://ngrok.com/download> and install for your OS.
* For information on using each tool, see the challenge2.4-testing\_bots/docs folder.
* *Need more help? Since time is limited, if your team has done its research but is still struggling with this challenge, extract the zip file in the Team’s General channel files in the challenge2.4-testing\_bots folder. See the 0\_README.md file for guidance.*

Finishing up Part 2: Bots Challenge

As a group, consider the following questions:

1. Which features from Azure Search should you take advantage of?
2. How can bot logging be used to benefit Contoso?
3. What are some tasks you'll have to complete to create an efficient and functional calling bot?

**Resources**

Helpful Links

**Intelligent Kiosk Sample Application**   
<https://github.com/Microsoft/Cognitive-Samples-IntelligentKiosk/tree/master/Kiosk/ServiceHelpers>Utilizing these resources makes it easy to add and remove the service helpers in your future projects as needed.

**Cognitive Services**   
<https://www.microsoft.com/cognitive-services>

**Cosmos DB**   
<https://docs.microsoft.com/en-us/azure/cosmos-db/>

**Azure Search**   
<https://azure.microsoft.com/en-us/services/search/>

**Bot Developer Portal**   
[http://dev.botframework.com](http://dev.botframework.com/)

**Natural Language Understanding (LUIS)**   
<https://azure.microsoft.com/en-us/services/cognitive-services/language-understanding-intelligent-service/>

**Understanding LUIS**   
<https://docs.microsoft.com/en-us/azure/cognitive-services/LUIS/Home>

**Bot Framework Emulator**   
<https://emulator.botframework.com/>

**Bot Framework Emulator documentation**   
<https://github.com/microsoft/botframework-emulator/wiki/Getting-Started>