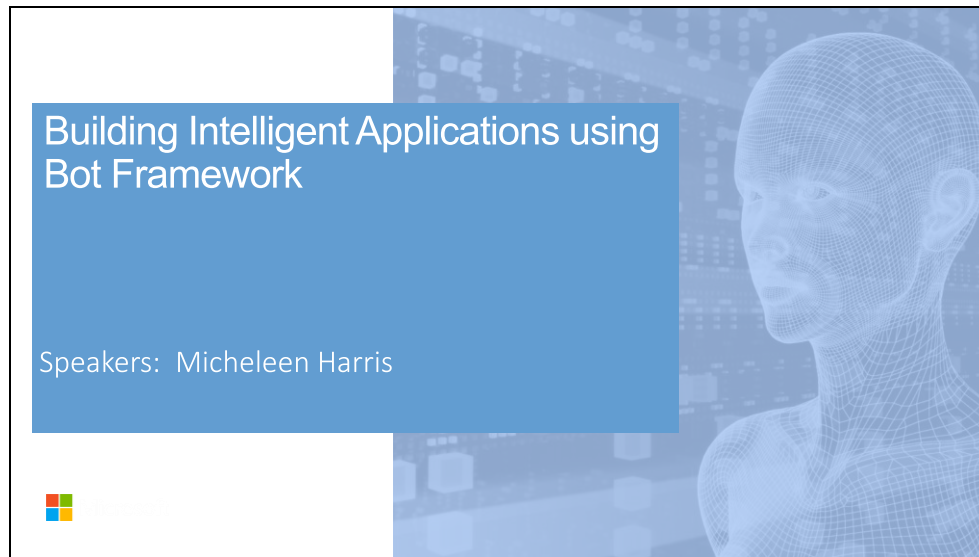


Slide 1



If the links in this deck are broken please let us know ([mailto: michhar<at>Microsoft.com](mailto:michhar@Microsoft.com)). Thanks in advance and enjoy learning about bots and the Microsoft Bot Framework.

Slide 2

Main site: aka.ms/odscbots
Chat room: aka.ms/botedu-discuss

This link contains additional resources on the bot framework and related topics. [mailto: michhar](mailto:michhar) for questions/comments.
[Show site.](#)

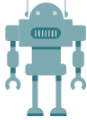
Learning objectives

What You'll Know at the End of this Session


1. What **a bot is** and is **not**
2. The **major components** of the Bot Framework
3. Deploying and working with **channels**
4. Your **arsenal** or **toolbox**





Learning objectives for this overview module on the Bot Framework

What is a bot? 

What a bot is not

It's not AI 

It's not natural language processing only 

It's not text interfaces only 

Not AI:

- Bots can be simple task automation utilities.
- Example: Password reset bot. There's no AI here. Just ask a couple of security validation questions, then reset the password.
- They may have AI as well, if the scenario applies

Not only NLP:

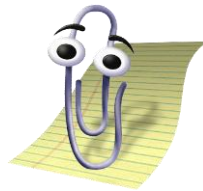
- Natural language processing has limitations, still. The more your bot depends on NLP, the worse the experience gets. Hint: Typing isn't always the best option.
- Move away from NLP as quickly as possible
- "Drive" the user as much as you can (menus, choices, etc). Less typing = better

Not only text interfaces:

- Bot channels are evolving quickly to support richer experiences: Media, buttons, custom controls. These are here or on their way. Text is not known to be the best experience for everything.
- Examples:
 - Skype allows audio and 3D bots as well.
 - Slack, Facebook and Skype have buttons/custom UIs

What is a bot?

Simply put, a bot is an **application** that performs an automated task. That's it.



Siri, Cortana, the old-school MS Clippy and even AOL's SmarterChild are some examples. Essentially, bots perform automated tasks that are generally **REPETITIVE** for humans to do. We want to make life easier for the end user of the bot.

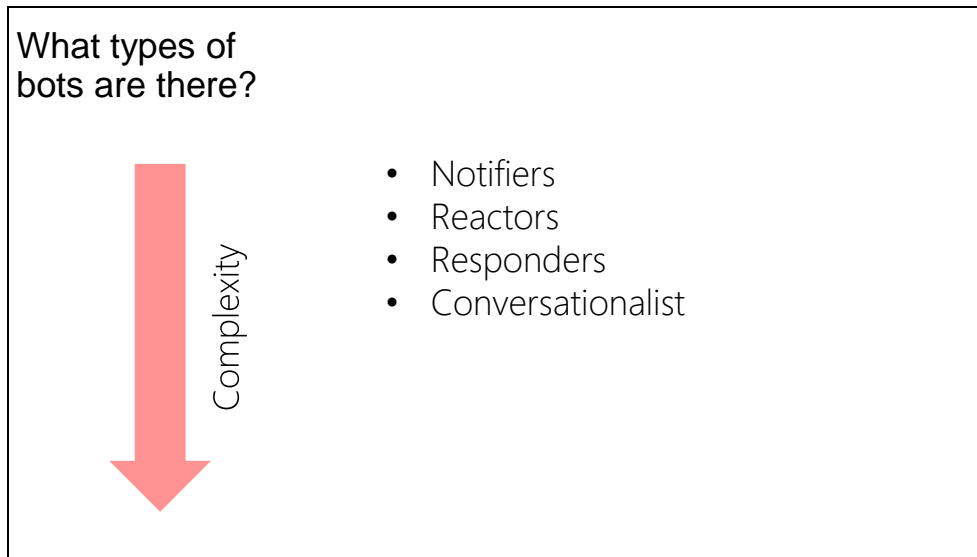
Bots are apps.

They can:

- Exist in different channels and across platforms.
- Do anything from simple task automation like taking food orders to leveraging sophisticated deep learning algos as is used by CaptionBot (<https://www.captionbot.ai/> which describes the contents of an image how a human would) and other AI-esque capabilities.

What a bot can do is only limited to the APIs your bot uses.

Bots don't have to leverage the MS Bot Framework (e.g. MimikerAlarm <https://www.microsoft.com/cognitive-services/en-us/mimickeralarm>, an app for waking you up), but the Framework makes dev and deploy much simpler and faster for.



Based on this blog post: http://willschenk.com/bot-design-patterns/?imm_mid=0e50a2&cmp=em-data-na-na-newsltr_20160622 about different bot types and the definitions of these.

- Notifier - simply broadcast messages aka push bot e.g. ping me when there's a interesting tweet about Hadley Wickam
- Reactor - reacts to messages on service, but doesn't persist anything (message, user state, location) e.g. send me the stock price for a stock I specify, but don't remember me or what I say
- Responder - reacts to messages on service, persists message and knows who I am e.g. send me today's weather forecast for a city, use my user name on this channel, and remember what cities I choose
- Conversationalist - reacts to messages, persists messages, knows who I am, knows about the "place" I'm at (channel, room,...), knows the state of the conversation e.g. send me today's weather forecast for a city, use my user name on this channel, remember what cities I choose, format it nicely for this channel, and if the conversation is old, archive it and send as email.

From "Bot Design Patterns": Questions that help us formulate what kind of bot we might want or need:

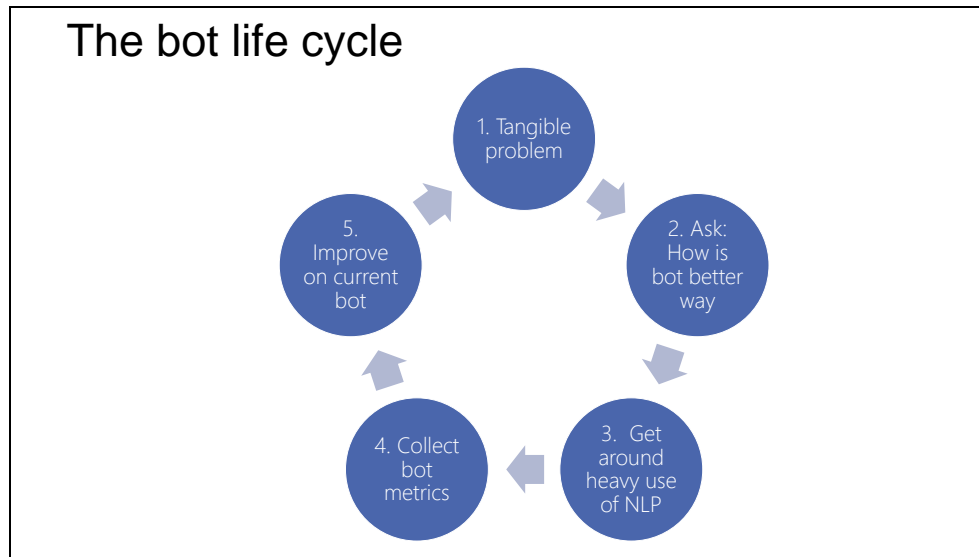
Do they react to messages?

Do they know who they are talking to?

Can they learn from what was said?

Do they know where the conversation is taking place?

Do they remember the overall conversation?



- Start by asking what problem are we trying to solve. Refine until it looks like a tangible problem and not “magic”
- Ask how a bot will be a better experience. User experience is EVERYTHING
- Avoid too much natural language. Careful with unrealistic expectations. Natural language recognition is limited. Menus work great. Commands work great. Buttons, etc.
- You can only analyze and improve your bot if you’re collecting metrics for it
- Iterate, improve

Consideration when going the route of a bot

“When developing A.I. we must guard **against** bias, ensuring proper, and representative research so that the wrong heuristics **cannot be used to discriminate.**” – Satya Nadella

This can be applied to bots. Even simple bots.

Additionally,

- Program the bot to **identify itself as a bot**

This may have been in your mind before this tutorial. Ethical and societal considerations taken directly from an article by Satya Nadella: <https://www.linkedin.com/pulse/partnership-future-how-humans-ai-can-work-together-solve-nadella>

Here enters the Bot
Framework

The Benefits of the Bot Framework

- **For developers**

- Bots are more capable nowadays so more functions
- Bot Builder SDKs or custom code – you have choices
- Faster testing, dev and deployment
- Easy integration with the cloud
- Growing community

- **For end users**

- User choice of channels
- Users have trust and control of their data
- New experiences

- **For businesses**

- Broad access to their customers and new experience
- Reduced cost of development
- High quality bots

For developers

Bots are more capable because of supporting services i.e. MS Cognitive Services and BF State Service

Bring your own bot or build your own bot with the Bot Builder SDKs

With SDKs and sample code on github

Smooth cloud deployment and integration

Big community (open source community, issues, gitters, stackoverflow – active and responsive)

For end users

Users can choose from a variety of conversation channels

Users have trust and control of their data – encrypted and accessible anytime

Frictionless fun or at least makes something much easier

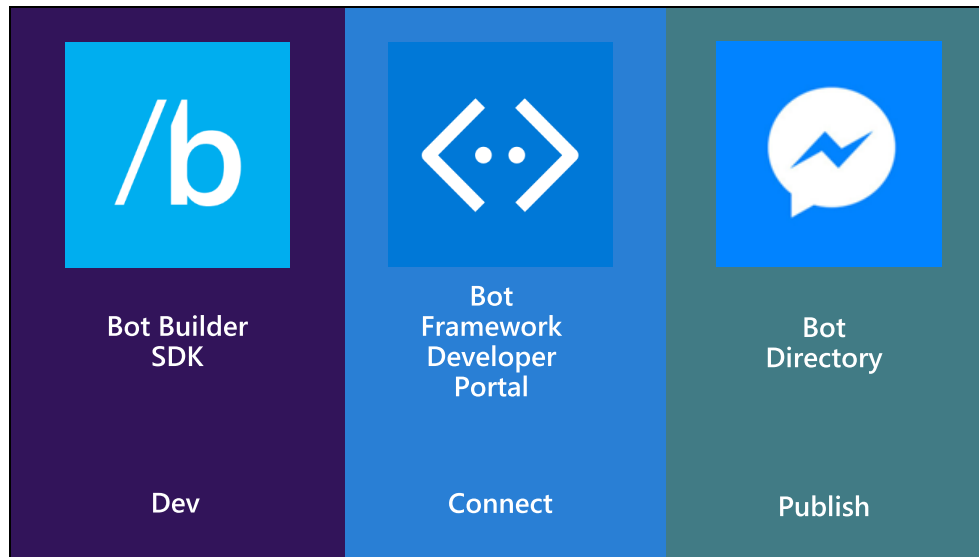
For businesses

Broad access to their customers where they already are conversing giving them new experiences

Reduced cost of development – just faster with SDKs and builtin functionality like dialog handling and language understanding

High quality bots (big support and dev community) as well as, bots are reviewed after publishing and surfaced on Bot Directory

Slide 13



Bot Builder is itself a framework for building conversational applications ("Bots").

The Bot Builder SDK is [an open source SDK hosted on GitHub](#) that provides everything you need to build great dialogs within your Node.js-, .NET- or REST API-based bot.

The Bot Framework Developer Portal lets you connect your bot(s) seamlessly text/sms to Skype, Slack, Facebook Messenger, Kik, Office 365 mail and other popular services. Register, configure and publish.

The Bot Directory is a public directory of all reviewed bots registered through the Developer Portal.

NB: Bot builder and bot connector SDK now one in V3 of framework: <http://docs.botframework.com/en-us/support/upgrade-to-v3/#botbuilder-and-connector-are-now-one-sdk>

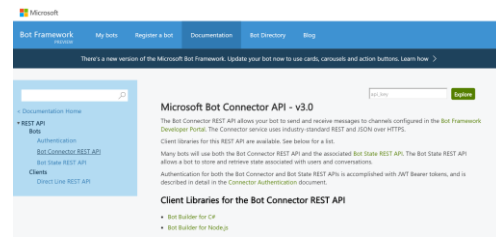
Bot Builder: Development Kits and REST

Bot Builder SDKs for:

- .NET framework for C#
- Node.js

And there's also

- REST and REST State APIs



SDKs infographic: http://docs.botframework.com/en-us/images/faq-overview/bot_builder_sdk_july.png

Developer Portal: what registration does for you

Your bot's web
service in the
cloud



MS Bot
Framework
Connector service



Expose a Microsoft Bot Framework-compatible API on the Internet, then the Bot Framework Connector service will forward messages from your Bot to a user, and will send user messages back to your Bot.

State Service: types of bot data stored for us

User data

Conversation
data

User-
conversation
data

This data is currently stored for free for you within the Bot Framework State Service.

However, you may bring in your own data source (format: key-value store)

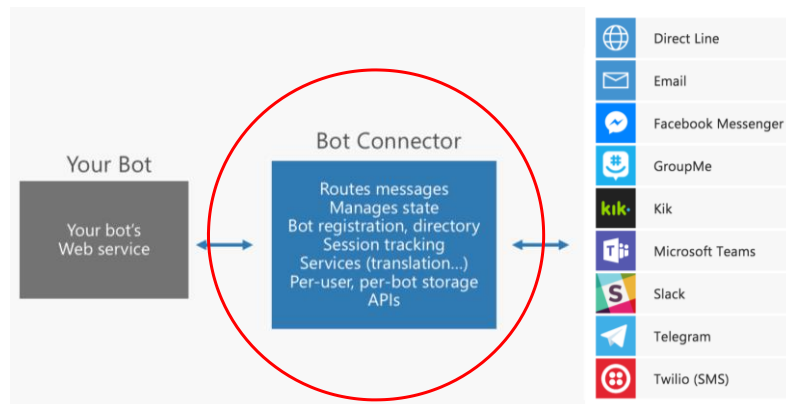
User data – globally available for user across all conversations

conversation data – stores globally for a single conversation (many users could be involved)

User-conversation data – stores globally conversation data for a user (But private to just that user)

Dialog data as well – persists for a single dialog (helpful for temp data in a waterfall set of steps)

What is the Bot Connector?



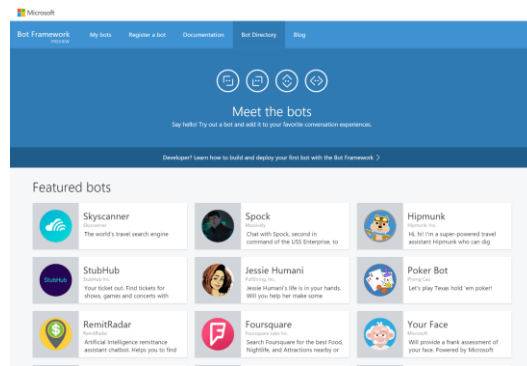
It's part of the Bot Builder SDK

<http://docs.botframework.com/en-us/csharp/builder/sdkreference/gettingstarted.html#channels>

Bot Directory

Public Directory of Bot Framework Bots

- Discover, try, and add bots from here with no added configuration
- Bots are public at developer discretion; must be reviewed
- Searchable here



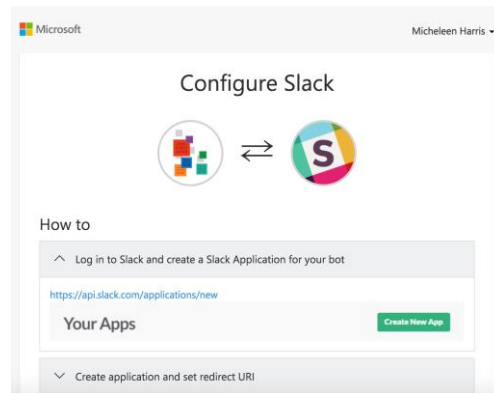
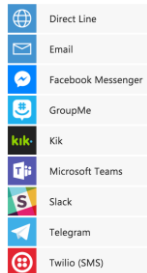
Bots must be submitted for review and approved in order to appear in the directory

Working with channels

Adding a channel

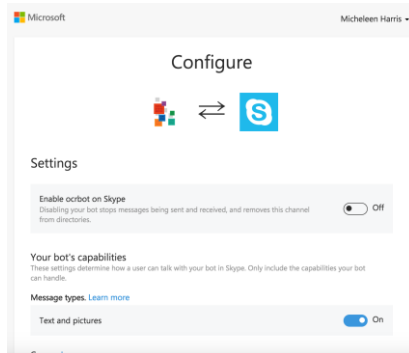
- Skype is added by default
- Instructions are laid out

Add another channel



Often, the most time will be spent configuring your credentials as a developer on the target service, registering your app, and getting a set of OAuth keys that Microsoft Bot Framework can use on your behalf

Editing a channel



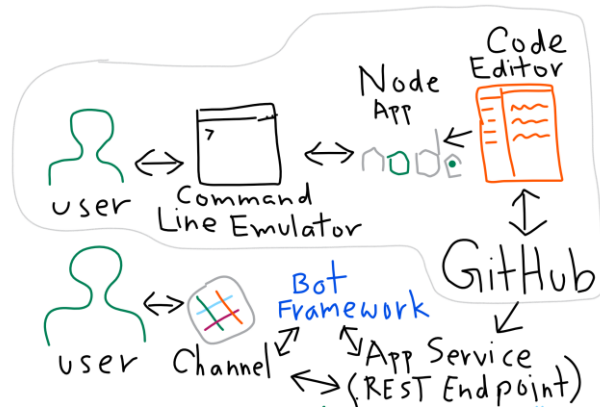
Skype for instance, through configuring we can:

- Disable/enable globally
- Turn on/off group messaging
- and more

Toolbox

Let's see some tech stuff and code!

Dev process



I should not draw or learn to draw someday. The "local dev" we will do is in the grey circled in part

Node App: get the Bot Builder module from command line

As easy as:

```
npm install --save botbuilder
```

Or:

```
git clone https://github.com/Microsoft/BotBuilder.git  
cd BotBuilder/Node  
npm install
```

We will set up after the Cognitive Services Overview
<https://docs.botframework.com/en-us/node/builder/guides/core-concepts>

In code editor: the Connector in Node.js Bot Builder SDK

UniversalBot – a simplified way to connect bots to dialogs

```
var connector = new builder.ChatConnector({  
  appId: process.env.MICROSOFT_APP_ID,  
  appPassword: process.env.MICROSOFT_APP_PASSWORD  
});  
var bot = new builder.UniversalBot(connector);  
server.post('/api/messages', connector.listen());
```



```
var connector = new builder.ConsoleConnector().listen();  
var bot = new builder.UniversalBot(connector);
```

UniversalBot (https://docs.botframework.com/en-us/node/builder/chat-reference/classes/_botbuilder_d_universalbot)

- has a lightweight connector model and includes ChatConnector and ConsoleConnector classes
- your bot can even utilize both the ChatConnector and ConsoleConnector and others at the same time if so desired
- replaces and unifies old classes like BotConnectorBot and TextBot

- updates and changes from <https://docs.botframework.com/en-us/node/builder/whats-new/>


In code editor: integration galore for Cognitive Services

Cognitive services (aka ProjectOxford)...

Language understanding for example...

```
// LUIS support classes in Node.js Bot Builder SDK
// - also have an entity recognizer
// and
// - a built-in dialog system for interpreting intents

// Main dialog with LUIS
var recognizer = new builder.LuisRecognizer(LuisModelUrl);
var intents = new builder.IntentDialog({ recognizers: [recognizer] })
.matches('SearchHotels', [
  function (session, args, next) {
    session.send('Welcome to the Hotels finder! we are analyzing your message: \'' + session.message.text);
    ...
  }
]);
```



Note on dialogs: Bot Builder breaks conversational applications up into components called dialogs. If you think about building a conversational application in the way you'd think about building a web application, each dialog can be thought of as route within the conversational application.

From: <https://docs.botframework.com/en-us/node/builder/guides/core-concepts>

Test conversing in BF Command Line Emulator

- Using the Emulator, you can:
- Send requests and receive responses to/from your bot endpoint on localhost or deployed in cloud
- Inspect the JSON response
- Emulate a specific user and/or conversation

Download tool for free

```
C:\Users\michha\Documents\bin\BotFrameworkEmulator-Console\BFEmulator.exe
Microsoft Bot Framework Channel Emulator v3.0.6043.30574

/exit or /quit to exit
/settings to change port, emulator serviceurl, bot endpoint, appId and appSecret settings
/dump [#] to show contents of last # activitys (default: 1)
/attachment [path] <- to add a file to your activity

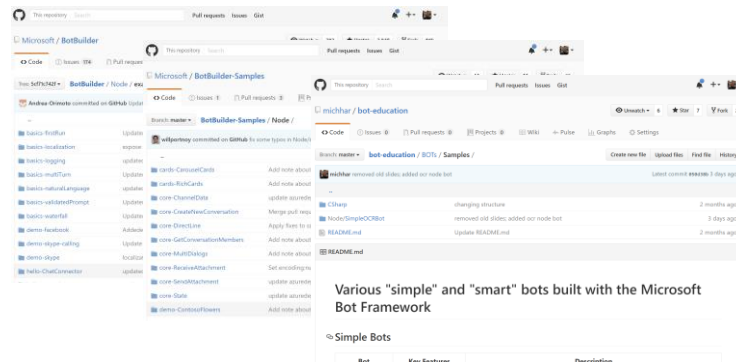
Current settings:
Port: 9000
Emulator ServiceUrl: http://localhost:9000/
Bot Endpoint: http://localhost:3978/api/messages
BotId: Bot1
AppId: disabled
AppPassword: disabled

Send message to bot:
hello ocrbot
User1 said:
hello ocrbot
Bot1 said:
Please give me an image link
https://img0.etsystatic.com/045/0/6267543/il_570xN.665155536_842h.jpg
User1 said:
https://img0.etsystatic.com/045/0/6267543/il_570xN.665155536_842h.jpg
Bot1 said:
Though this be madness, yet there is method in 't.
```

<https://docs.botframework.com/en-us/tools/bot-framework-emulator/#mac-and-linux-support-using-command-line-emulator>

Tons of sample code on GitHub

On github from Microsoft (and elsewhere):



<https://github.com/Microsoft/BotBuilder-Samples>

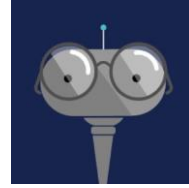
<https://github.com/Microsoft/BotBuilder/tree/master/CSharp/Samples>

<https://github.com/Microsoft/BotBuilder/tree/master/Node/examples>

<https://github.com/michhar/bot-education/tree/master/BOTs/Samples>

More out there...so many, can't list...

Parting thought (before our lab)



We can aim for our bots to: Be transparent and have algorithmic accountability so that humans can undo unintended harm. (Satya Nadella)

Ethical and societal considerations taken directly from an article by Satya Nadella:
<https://www.linkedin.com/pulse/partnership-future-how-humans-ai-can-work-together-solve-nadella>

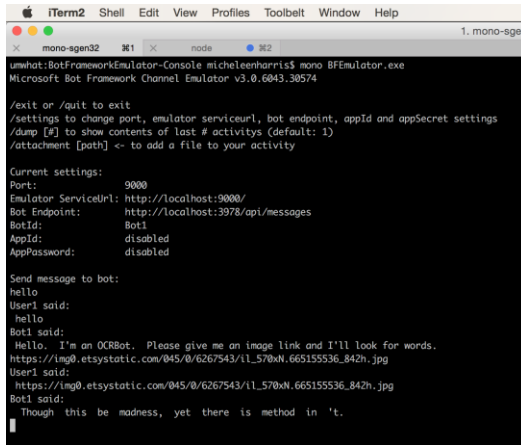
Resources

Support	Contact
Bot Builder SDK issues and suggestions	Use the issues tab on our github repo: https://github.com/Microsoft/BotBuilder/
Using a bot	Contact the bot's developer through their publisher e-mail
Community support	Use StackOverflow, with the hashtag #botframework
Reporting Abuse	Contact us at bf-reports@microsoft.com

Questions
(or use gitter chatroom at aka.ms/botedu-discuss)



Command line emulator

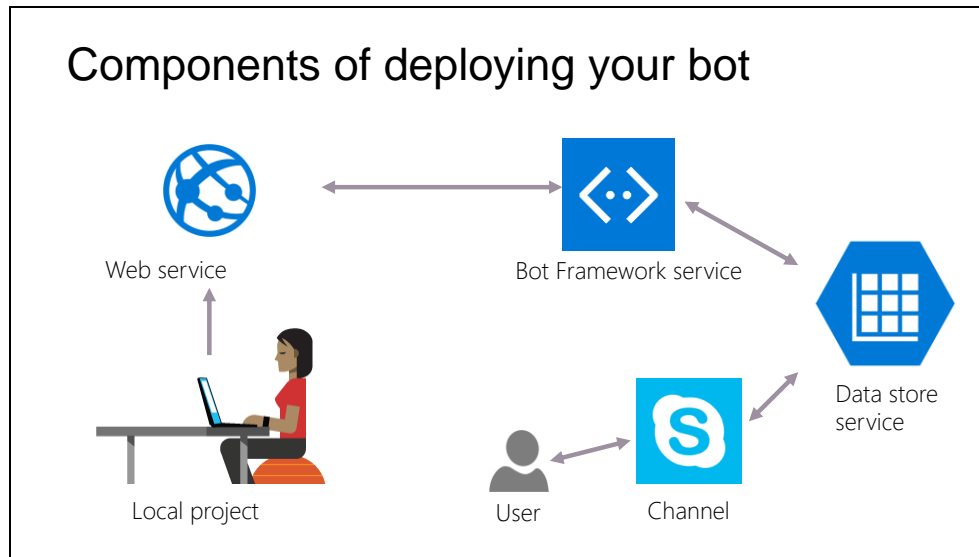


```
unwhat:BotFrameworkEmulator-Console michelenharris$ mono BFEmulator.exe
Microsoft Bot Framework Channel Emulator v3.0.6043.38574

/exit or /quit to exit
/settings to change port, emulator serviceurl, bot endpoint, appId and appSecret settings
/dump [#] to show contents of last # activitys (default: 1)
/attachment [path] <- to add a file to your activity

Current settings:
Port: 9000
Emulator ServiceUrl: http://localhost:9000/
Bot Endpoint: http://localhost:3978/api/messages
BotId: Bot1
AppId: disabled
AppPassword: disabled

Send message to bot:
hello
User1 said:
hello
Bot1 said:
Hello. I'm an OCRBot. Please give me an image link and I'll look for words.
https://img0.etsystatic.com/045/0/6267543/il_570xN.665155536_842h.jpg
User1 said:
https://img0.etsystatic.com/045/0/6267543/il_570xN.665155536_842h.jpg
Bot1 said:
Though this be madness, yet there is method in 't.
```

Steps:

The developer writes their bot code, leveraging the BF libraries in the SDKs and APIs available

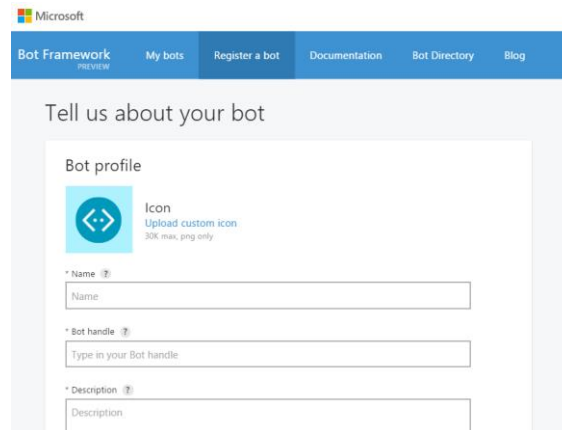
Create an endpoint for the bot to talk to in the cloud

Connect to the Bot Framework service (Connector and State services)

Pass user and conversation data between channel and Framework (data store in the state service, managed by the BF)

The user interacts with the bot on a channel of their choosing

Register a Bot in the Developer's Portal



The screenshot shows the Microsoft Bot Framework Developer's Portal registration page. The header includes the Microsoft logo and navigation links: Bot Framework (PREVIEW), My bots, Register a bot, Documentation, Bot Directory, and Blog. The main heading is "Tell us about your bot". Below this is the "Bot profile" section, which includes an "Icon" upload area (30K max, png only) and three required text input fields: "Name", "Bot handle" (with a hint "Type in your Bot handle"), and "Description".

Register on the developer portal by clicking the 'Register a bot' link: <https://dev.botframework.com/bots/new>

Name: TemplateBot

Bot Handle: templatebot (for referencing in Bot Directory and name for bot on web chat, NOT the app's URL used as endpoint)

Also, add a description here

Configuration

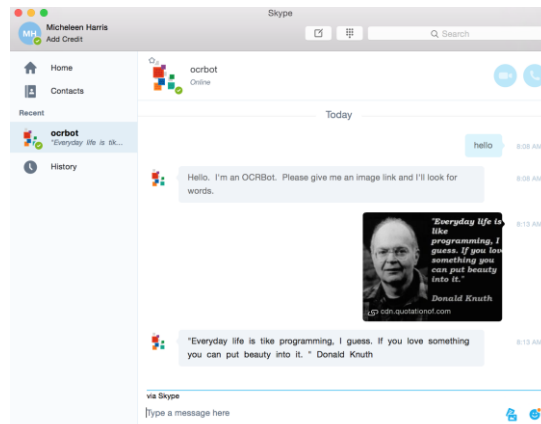
Remember the URL endpoint from deploying endpoint step. Should be something like:

"<https://botwebappname.azurewebsites.net/api/messages>"

You'll go through the "Generate App ID and password" wizard, then return to the registration page.

Go back and edit this profile anytime

Skype channel example



Slack channel example

