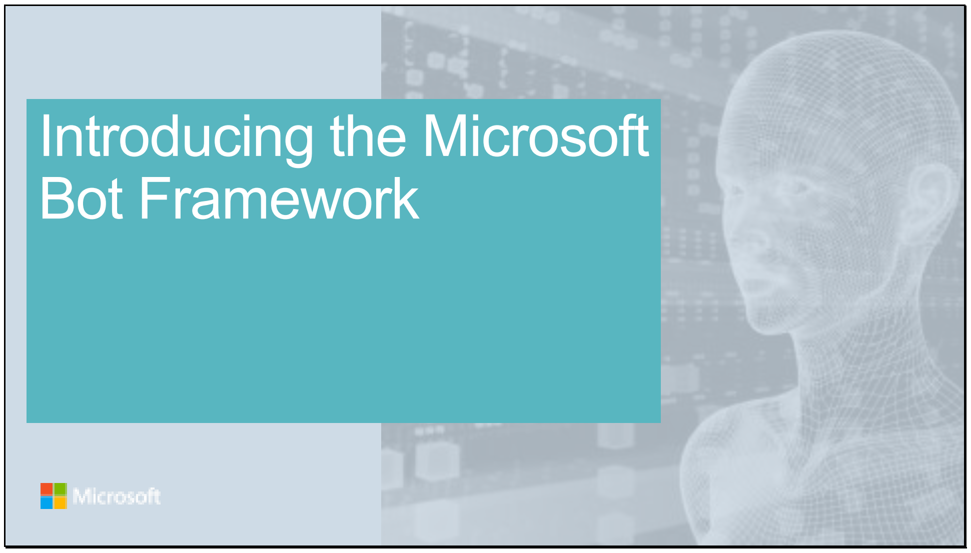
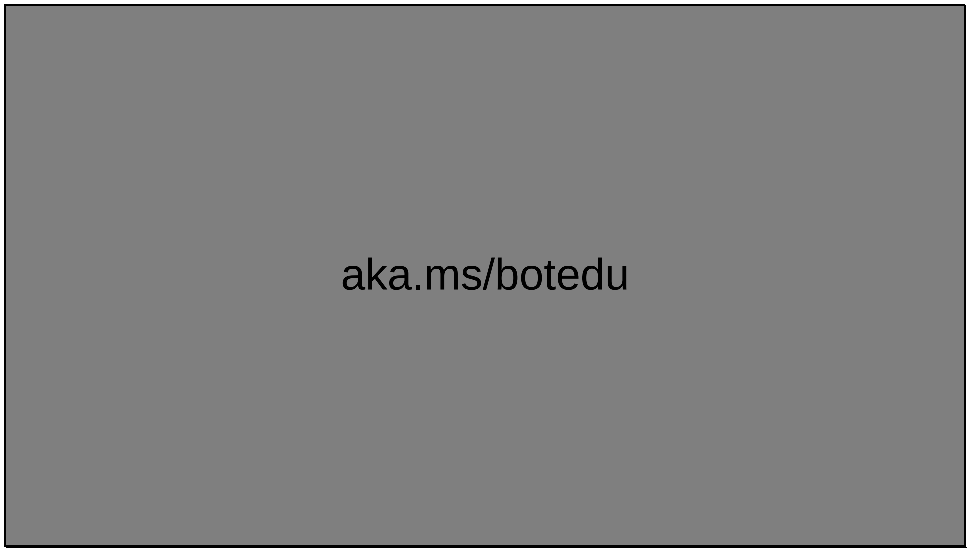


Introducing the Microsoft Bot Framework

Microsoft

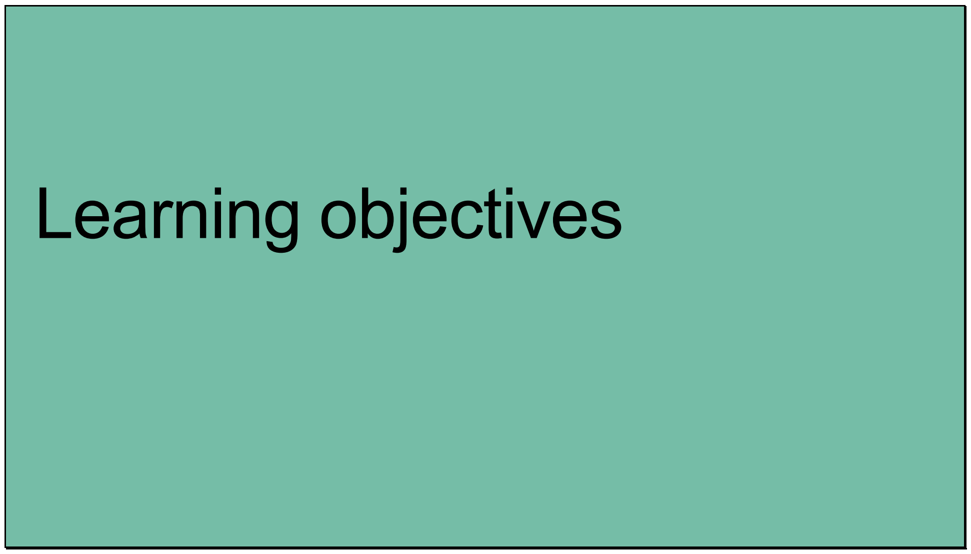
If the links in this deck are broken please let us know (mailto: michhar@Microsoft.com). Thanks in advance and enjoy learning about bots and the Microsoft Bot Framework.

Slide 2

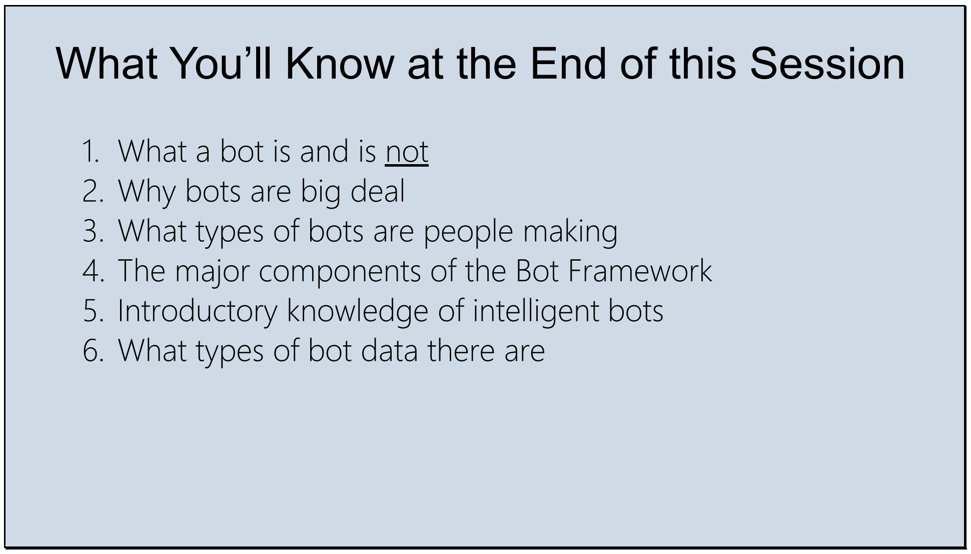


aka.ms/botedu

This link contains additional resources on the bot framework and related topics. mailto: michhar for questions/comments.
Also, check out the excellent Bot Framework FAQs: http://docs.botframework.com/en-us/faq/



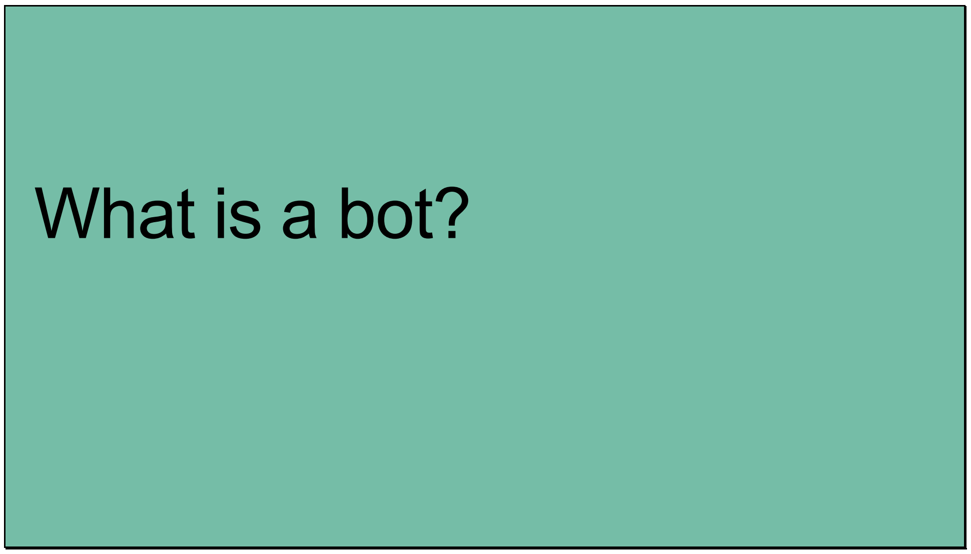
Learning objectives



What You'll Know at the End of this Session

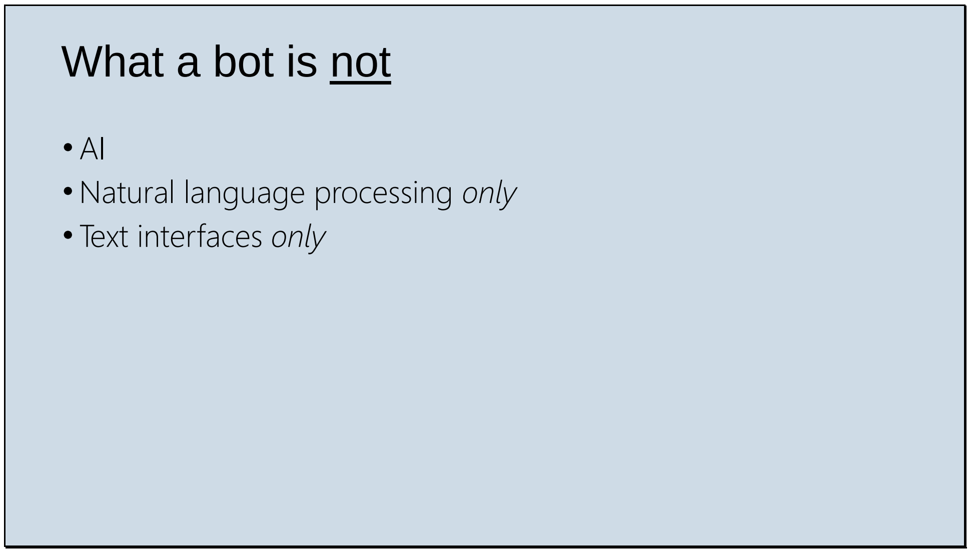
1. What a bot is and is not
2. Why bots are big deal
3. What types of bots are people making
4. The major components of the Bot Framework
5. Introductory knowledge of intelligent bots
6. What types of bot data there are

Learning objectives for this overview module on the Bot Framework



What is a bot?

Slide 6



What a bot is not

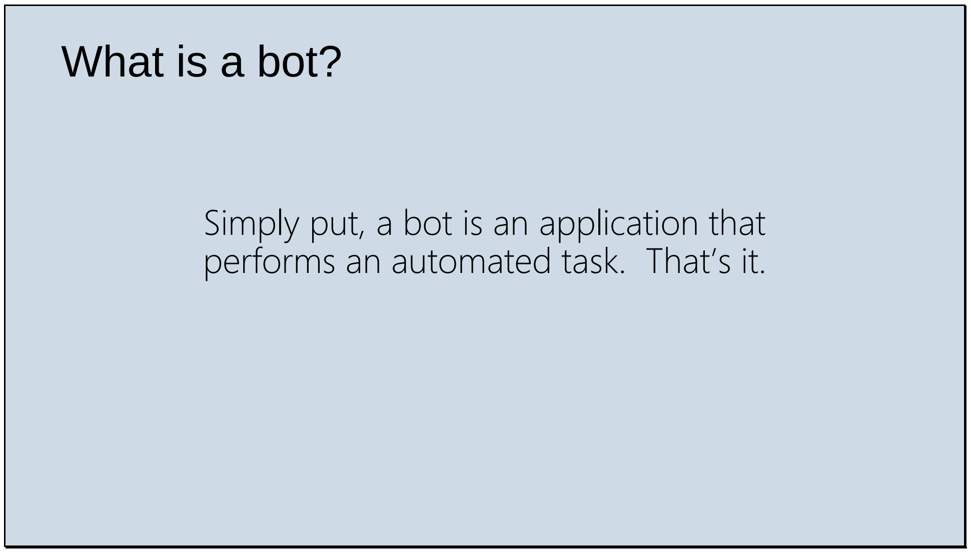
- AI
- Natural language processing *only*
- Text interfaces *only*

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 has limitations. The more your bot depends on natural language, the worse the experience gets. Hint: Typing isn't always the best option.
- Move away from natural language as quickly as possible
- "Drive" the user as much as you can (menus, choices, etc)
- Example: AzureBot "stop vm1" is a command, not natural language. Less typing = better

Not only text interfaces:
- Bot channels are evolving quickly to support richer experiences: Media, buttons, custom controls. These are coming. Text is not the best experience for everything.
- Examples:
  - Skype allows audio and 3D bots as well.
  - Slack, Facebook and Skype have buttons/custom UIs

Slide 7



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.

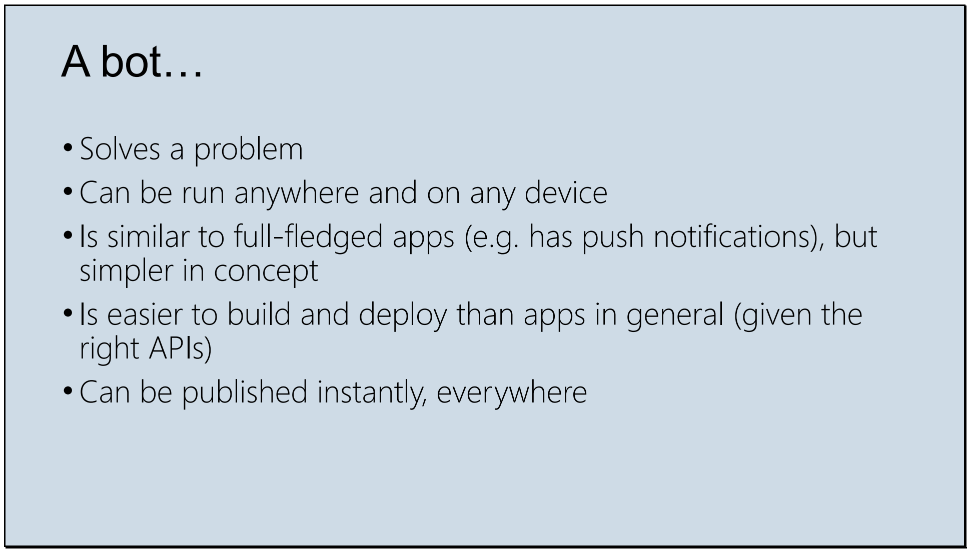
They can:

- Exist in different channels and across platforms.
- Do anything from simple task automation like taking food orders to sophisticated machine learning as is used by CaptionBot (https://www.captionbot.ai/ which describes the contents of an image in a human way) and even 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 concept to deployment much simpler and faster for develpers.

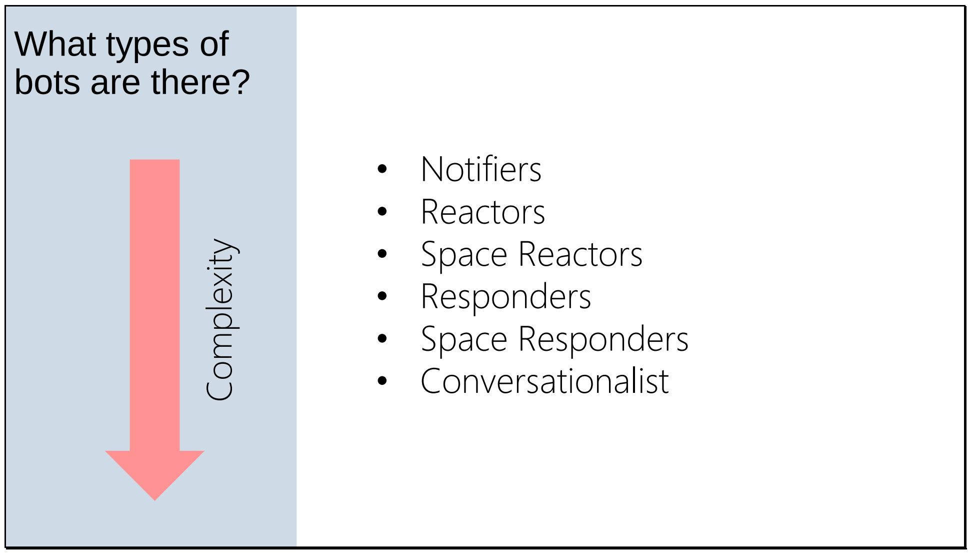
Slide 8



A bot…

- Solves a problem
- Can be run anywhere and on any device
- Is similar to full-fledged apps (e.g. has push notifications), but simpler in concept
- Is easier to build and deploy than apps in general (given the right APIs)
- Can be published instantly, everywhere

Slide 9



What types of bots are there?

Complexity

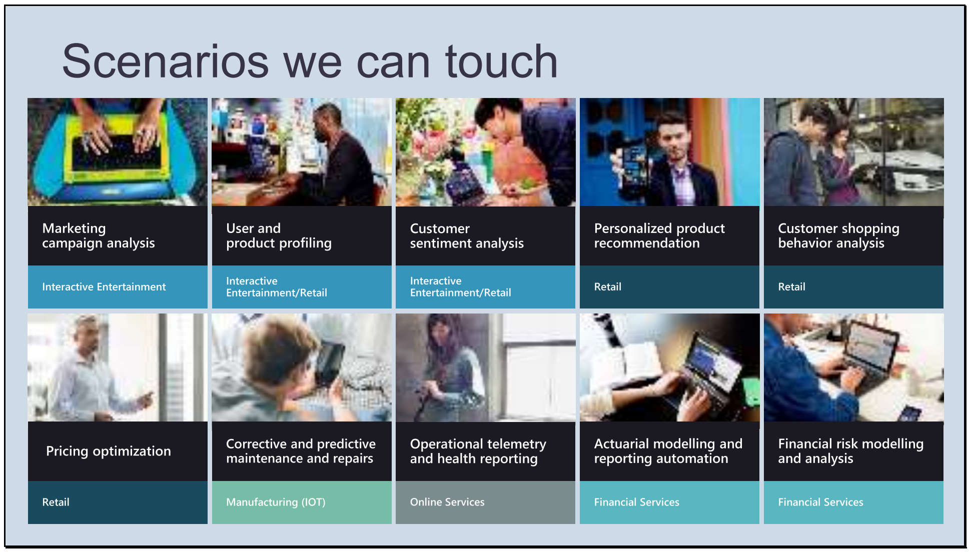
- Notifiers
- Reactors
- Space Reactors
- Responders
- Space Responders
- Conversationalist

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.

Definitions:
Notifier – simply broadcast messages from a source (doesn't mean the bot is not doing complicated things in the back ground)
Reactor – reacts to messages on service, but doesn't persist anything (message, user state, location) aka (what, who, where)
Space reactor – reacts to messages on service, persists location (so can respond based on this), doesn't persist message
Responder – reacts to messages on service, persists message (to get user state) and responds based on user state is association with the message
Space responder – same as Responder, but with persisting location (e.g. a particular channel type and maybe "room") to respond based on this
Conversationalist – same as Space Responder, but also with conversation state data so not only reacts, knows the user, persists the message, knows location of conversation, but can also respond based on conversation content and context

The Big Deal

Slide 11

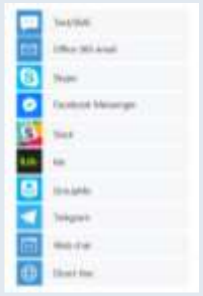


Scenarios we can touch

Marketing campaign analysis	User and product profiling	Customer sentiment analysis	Personalized product recommendation	Customer shopping behavior analysis
Interactive Entertainment	Interactive Entertainment/Retail	Interactive Entertainment/Retail	Retail	Retail
Pricing optimization	Corrective and predictive maintenance and repairs	Operational telemetry and health reporting	Actuarial modelling and reporting automation	Financial risk modelling and analysis
Retail	Manufacturing (IOT)	Online Services	Financial Services	Financial Services

1. /build bot video link: https://www.youtube.com/watch?v=7wNg18NYT6s

Slide 12

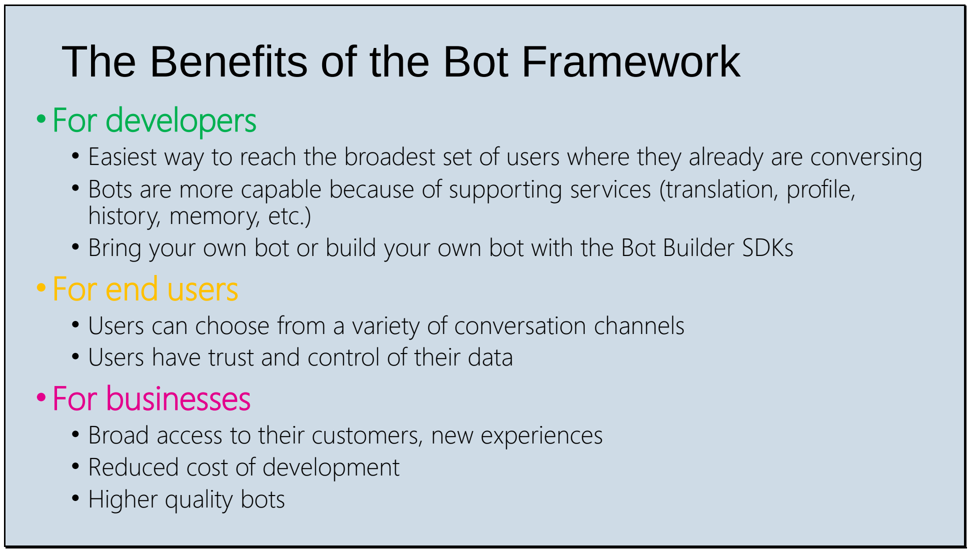


Make conversations rich, productive and fun

Enable a bot to see, hear, and interpret

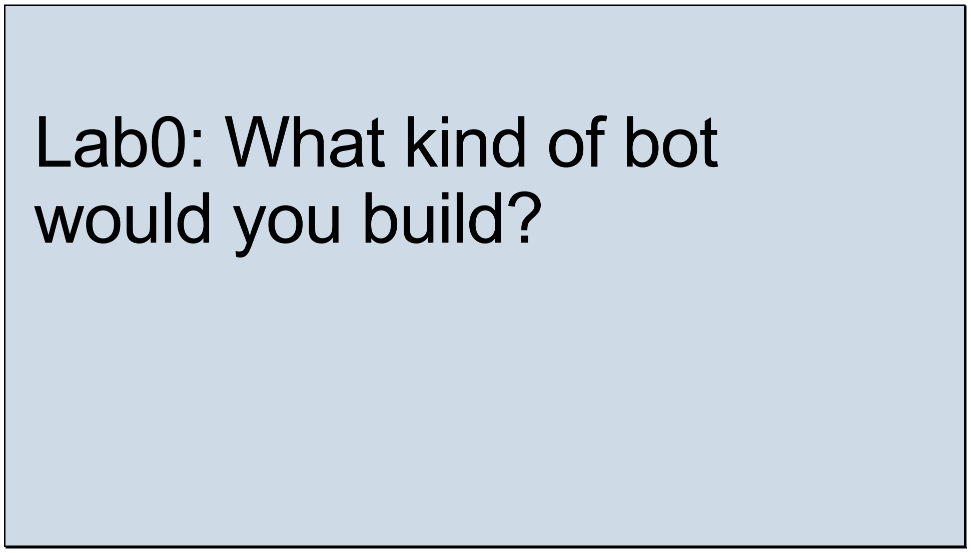
Create bots that interact in human ways

Slide 13



The Benefits of the Bot Framework

- For developers
  - Easiest way to reach the broadest set of users where they already are conversing
  - Bots are more capable because of supporting services (translation, profile, history, memory, etc.)
  - Bring your own bot or build your own bot with the Bot Builder SDKs
- For end users
  - Users can choose from a variety of conversation channels
  - Users have trust and control of their data
- For businesses
  - Broad access to their customers, new experiences
  - Reduced cost of development
  - Higher quality bots

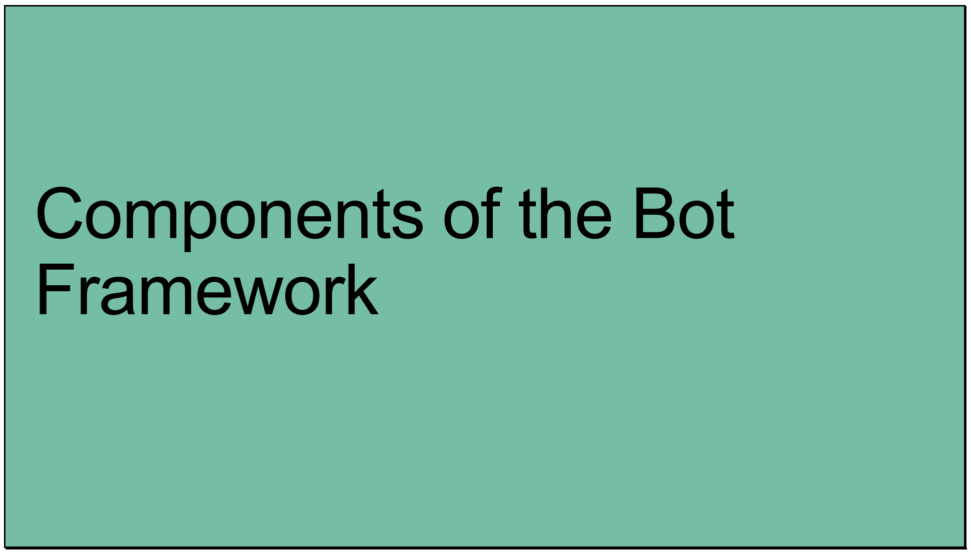


Lab0: What kind of bot would you build?

Choose from:
Notifiers
Reactors
Space Reactors
Responders
Space Responders
Conversationalist

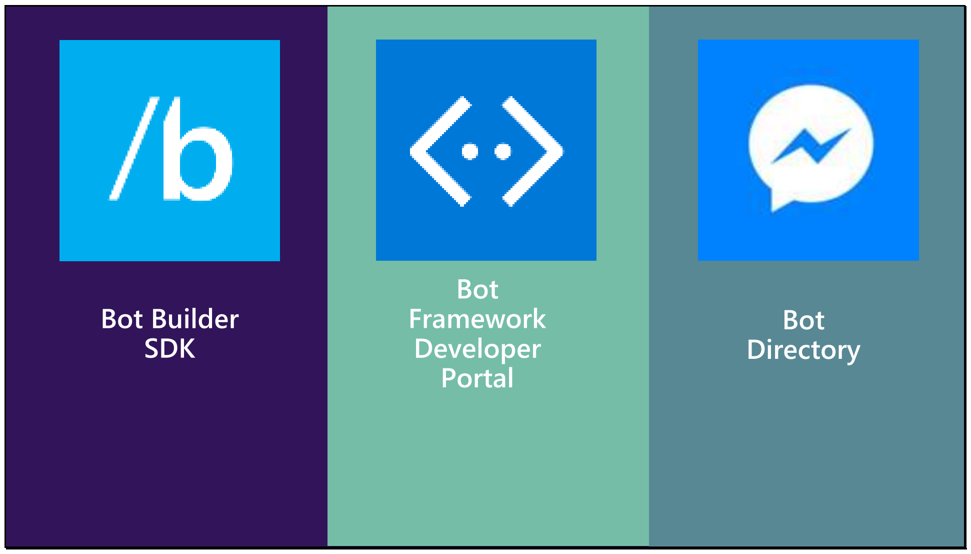
And what would your use case be?

Slide 15



Components of the Bot Framework

Slide 16



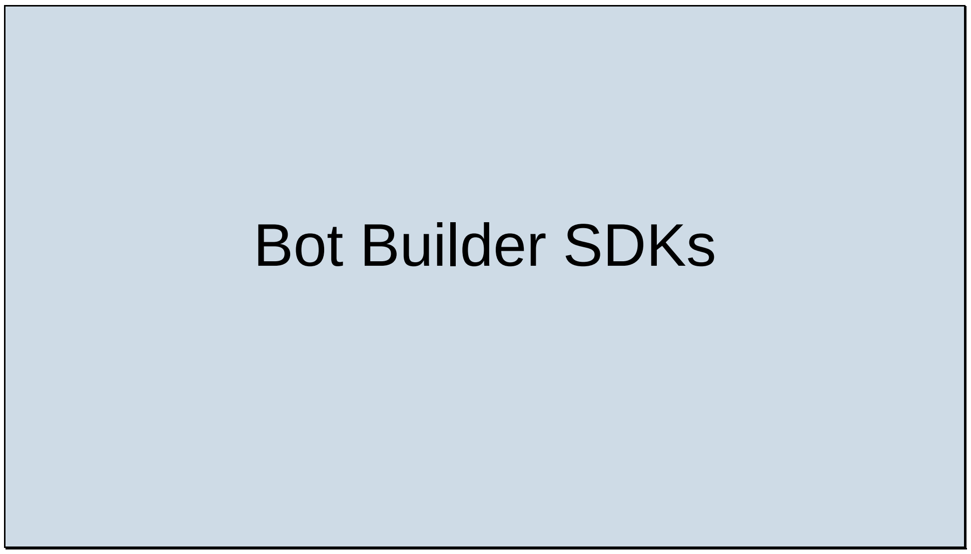
The Bot Builder SDK is an open source SDK hosted on GitHub that provides everything you need to build great dialogs within your Node.js-, .NET- or REST API-based bot. *

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.

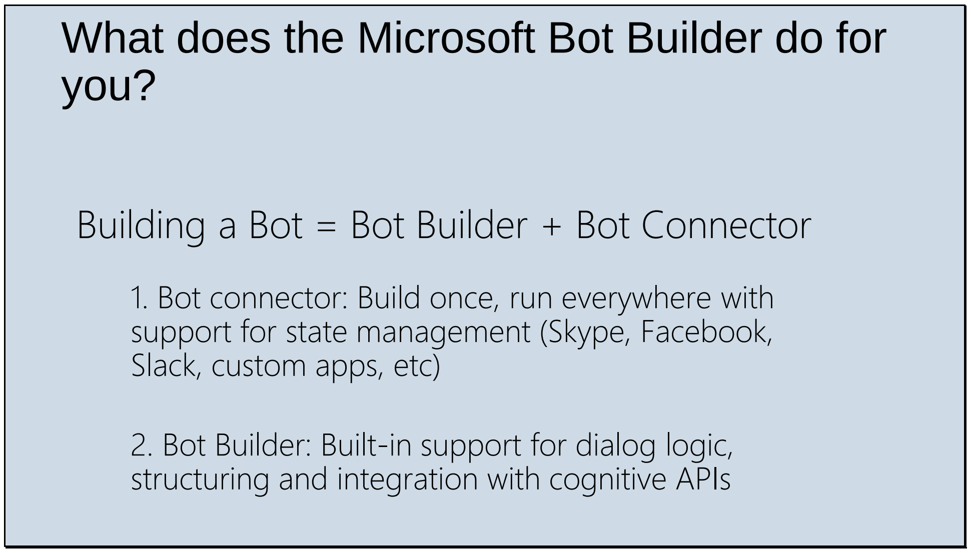
NB: Bot builder and bot connector SDK now one in V3 of framework: http://docs.botframework.com/en-us/support/upgrade-to-v3/#botbuilder-and-connector-are-now-one-sdk

Slide 17



Bot Builder SDKs

Slide 18



What does the Microsoft Bot Builder do for you?

Building a Bot = Bot Builder + Bot Connector

1. Bot connector: Build once, run everywhere with support for state management (Skype, Facebook, Slack, custom apps, etc)

2. Bot Builder: Built-in support for dialog logic, structuring and integration with cognitive APIs

Slide 19

Development Kits and REST

Bot Builder SDKs for:
- .NET framework for C#
- Node.js

and
- REST and REST State APIs

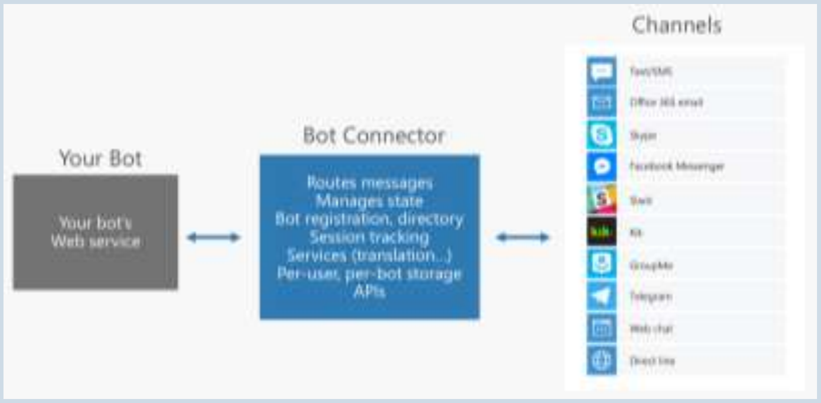
Microsoft Bot Connector API - v3.0

Client Libraries for the Bot Connector REST API

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

Slide 20

Bot Builder SDK: 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

Slide 21

Bot Developer Portal

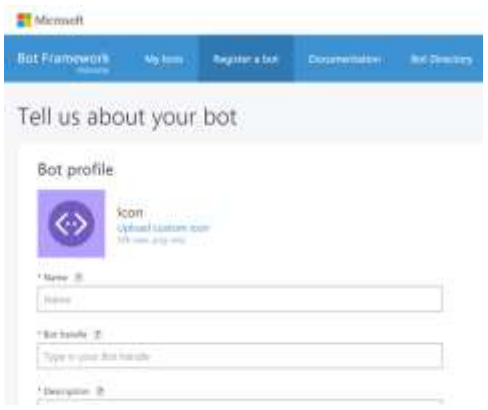
https://dev.botframework.com/

Slide 22

Bot Developer Portal: Register

Register, connect your service and create a profile

It’s an easy-to-use GUI



Tell us about your bot

Bot profile

Icon
Upload custom icon

* Name
Name

* Bot handle
Type in your Bot handle

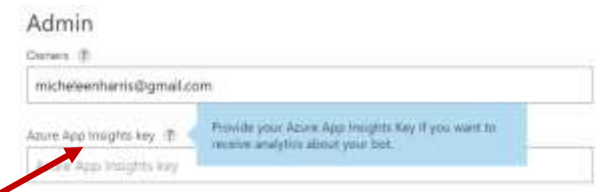
* Description

You only need login with your Microsoft account to do this (however will need a web app endpoint on Azure)

Slide 23

Bot Developer Portal: Diagnostics

Add analytics with Azure App Insights



Admin

Owners

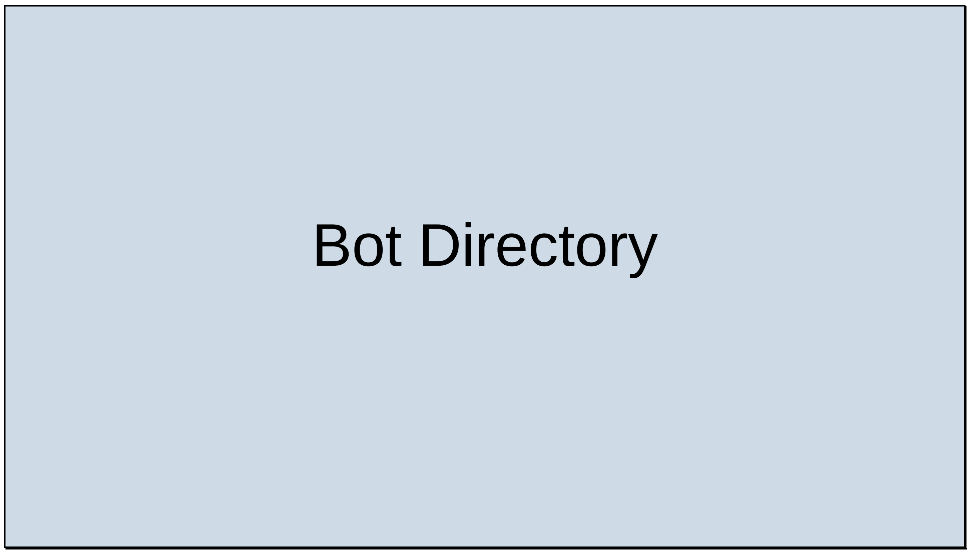
micheleenharris@gmail.com

Azure App Insights key

Provide your Azure App Insights Key if you want to receive analytics about your bot.

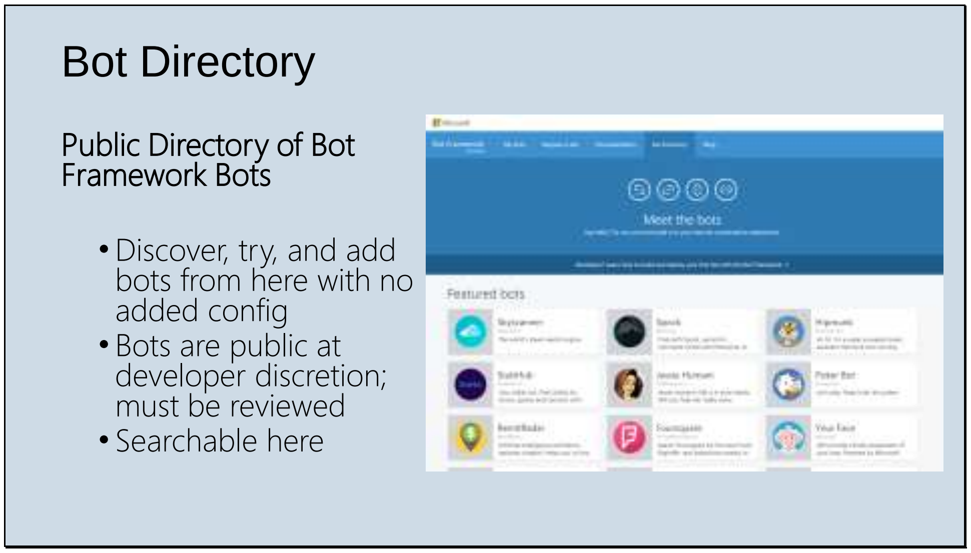
Azure App Insights key

Slide 24



Bot Directory

Slide 25



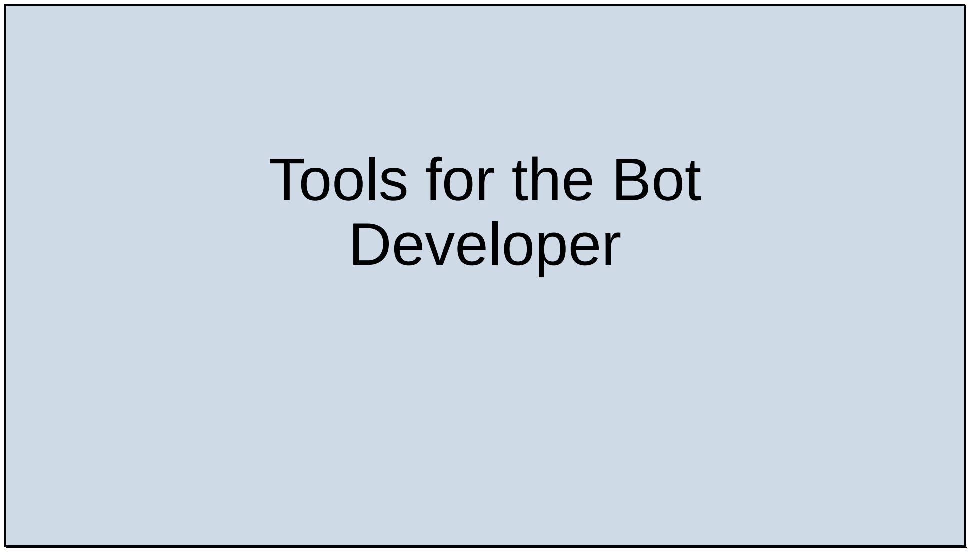
Bot Directory

Public Directory of Bot Framework Bots

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

Meet the bots
Featured bots

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



Tools for the Bot Developer

Slide 27

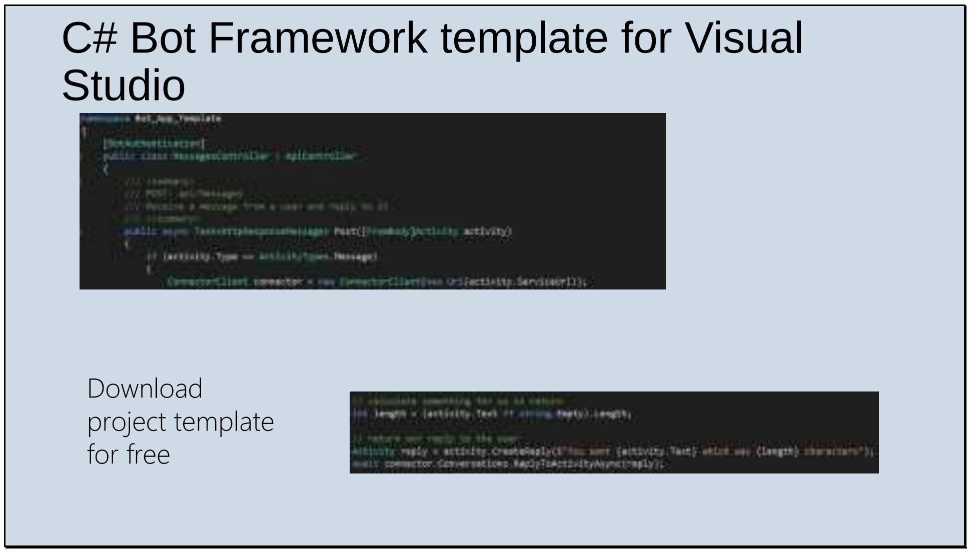
Bot Framework Emulator

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

Download tool for free

Hello Everyone! I'm the TestBot!

Slide 28



C# Bot Framework template for Visual Studio

```
namespace Bot_App_Template
{
    [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)
            {
                ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
```

Download project template for free

```
// calculate something for us to return
int length = (activity.Text ?? string.Empty).Length;

// return our reply to the user
Activity reply = activity.CreateReply($"You sent {activity.Text} which was {length} characters");
await connector.Conversations.ReplyToActivityAsync(reply);
```

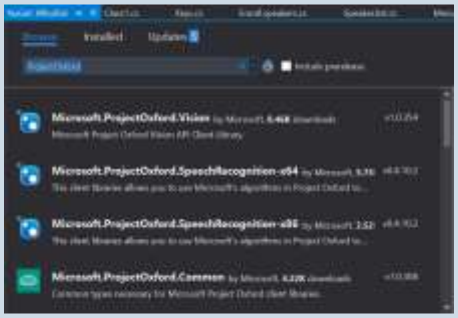
We’ll do this during the Setup phase if you haven’t already

Slide 29

NuGet packages galore for Cognitive Services

Cognitive services (aka ProjectOxford)

And many, many more – Connector and everything the .NET framework provides



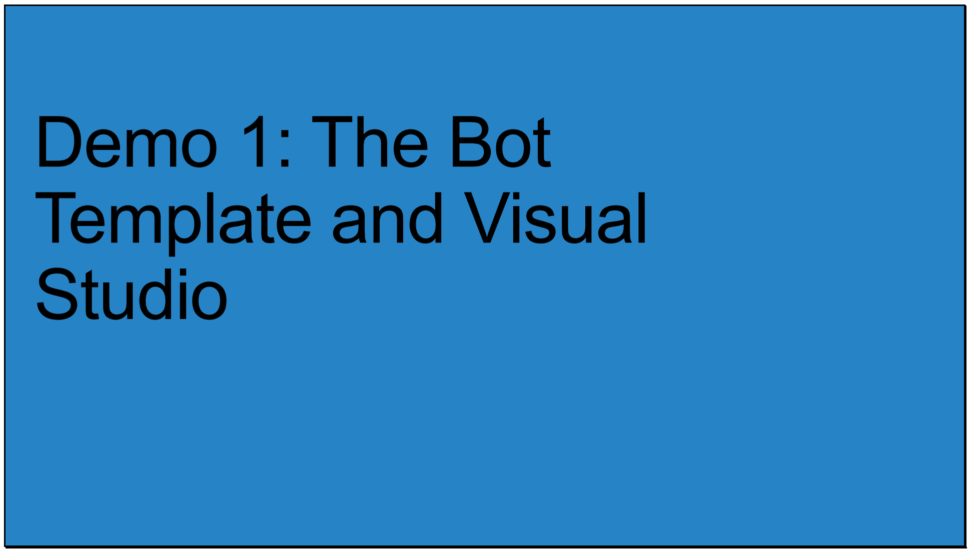
Microsoft.ProjectOxford.Vision by Microsoft, 6.46K downloads

Microsoft.ProjectOxford.SpeechRecognition-x64 by Microsoft

Microsoft.ProjectOxford.SpeechRecognition-x86 by Microsoft

Microsoft.ProjectOxford.Common by Microsoft

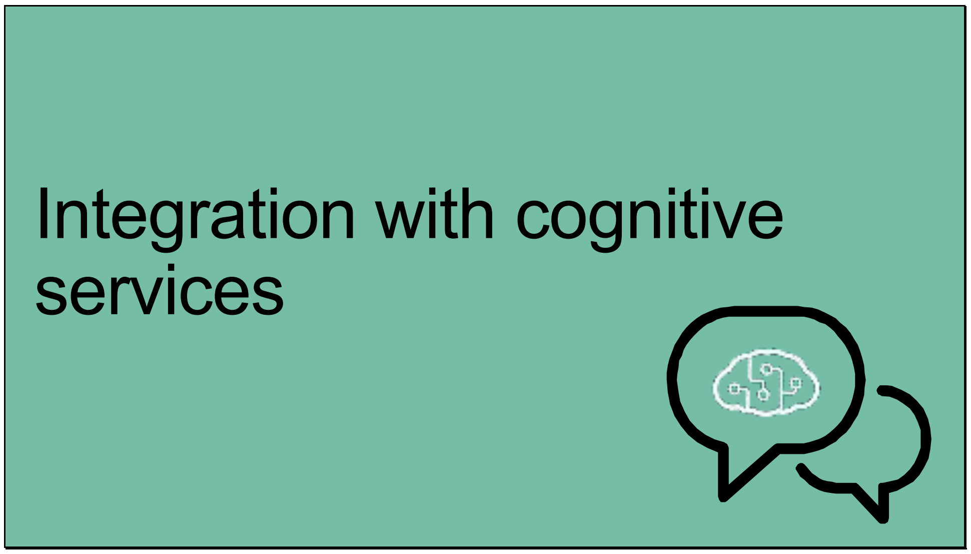
Slide 30



Demo 1: The Bot Template and Visual Studio

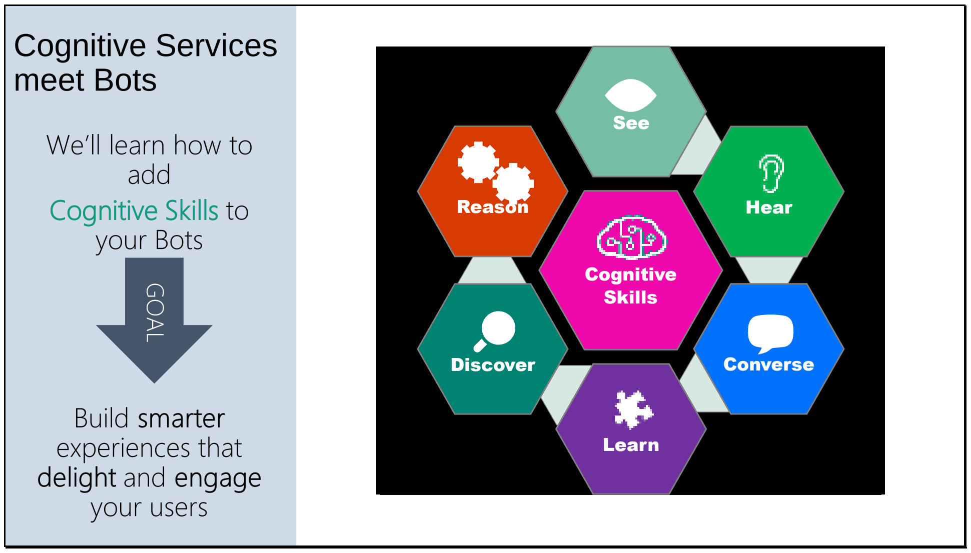
Demo of downloading, installing and starting a project with the bot template

Slide 31



Integration with cognitive services

Slide 32



Cognitive Services meet Bots

We'll learn how to add

Cognitive Skills to your Bots

GOAL

Build smarter experiences that delight and engage your users

See
Reason
Hear
Cognitive Skills
Discover
Converse
Learn

Slide 33

Cognitive services vs. bots

We can have a bot without cognitive services.

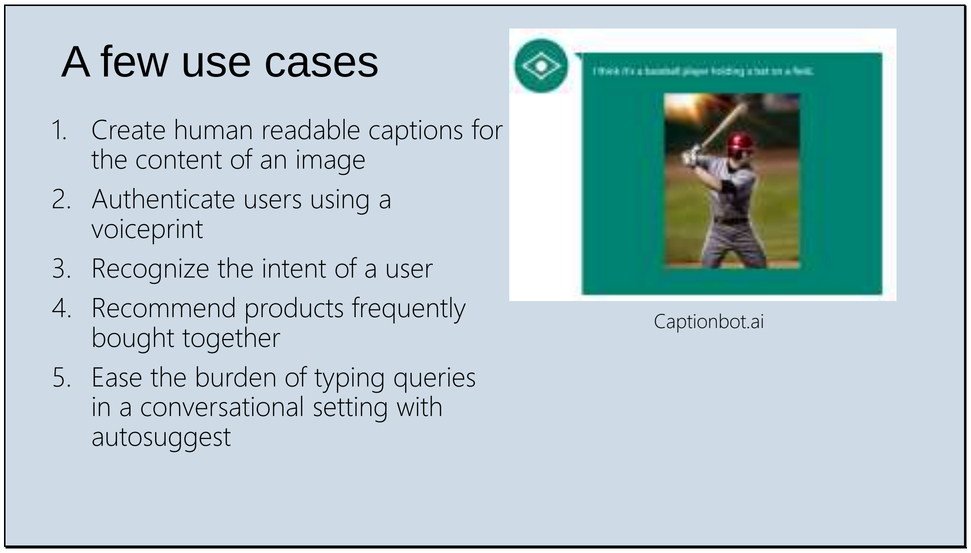
StubHub

We can have an intelligent app without a bot.

Stubhub: Find tickets for shows, games and concerts with StubHub.

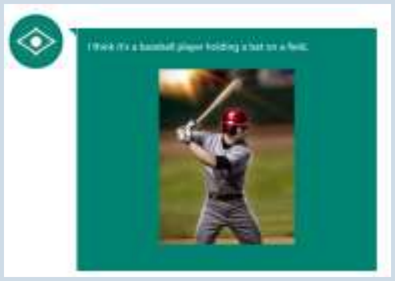
Computer vision API demo on this page: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

Slide 34



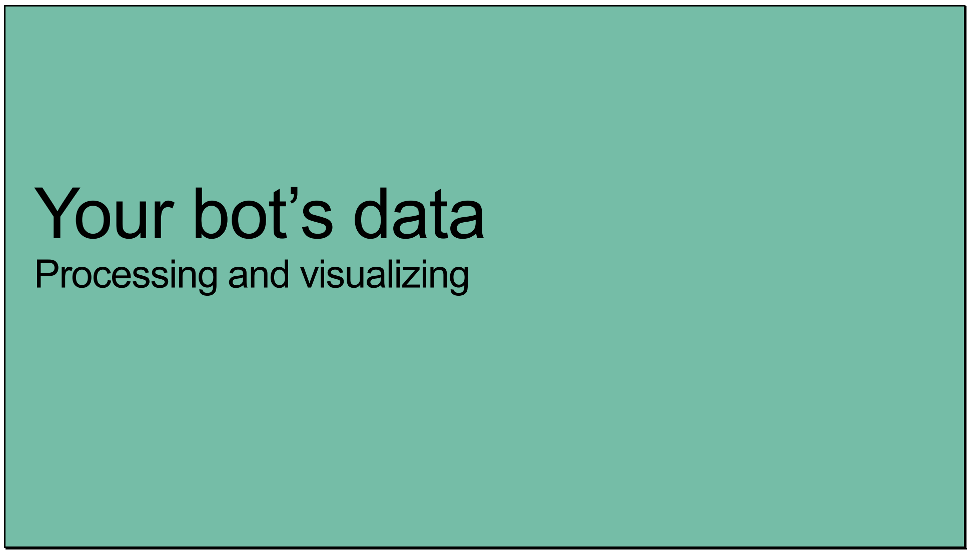
A few use cases

1. Create human readable captions for the content of an image
2. Authenticate users using a voiceprint
3. Recognize the intent of a user
4. Recommend products frequently bought together
5. Ease the burden of typing queries in a conversational setting with autosuggest


I think it's a baseball player holding a bat on a field.

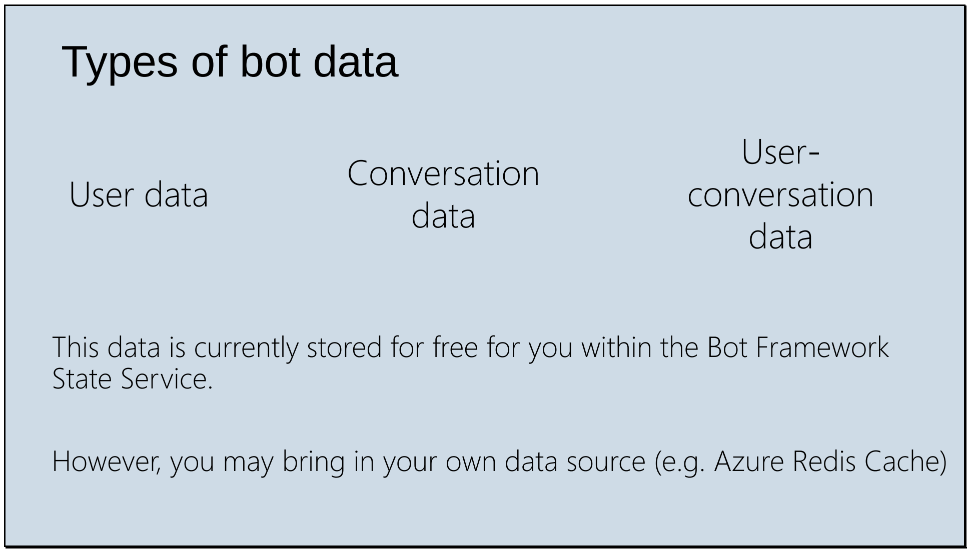
Captionbot.ai

- Computer Vision API - Like CaptionBot.ai – a bot that reports back in human way the contents of an image
- Speech API - Authentication as a user speaks to the bot with a speaker verification profile or "voiceprint"
- LUIS API for analysis of queries such as "What is the weather in Toyko today?" using entities to parse out intent (what the user is asking for)
- Knowledge API. Recommendations based on our knowledge and/or a user's history. Also, could search through a graph or user-defined database to return relevant academic papers from a natural language query
- Bing Search API for autosuggest - Search also has other capabilities such as returning the latest trending news on a topic for example



Your bot’s data

Processing and visualizing



Types of bot data

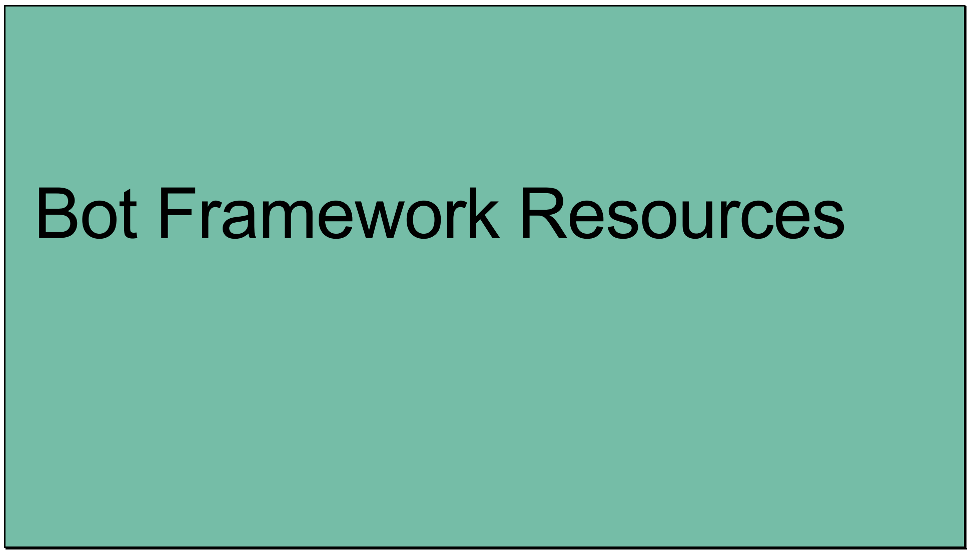
User data

Conversation data

User-conversation data

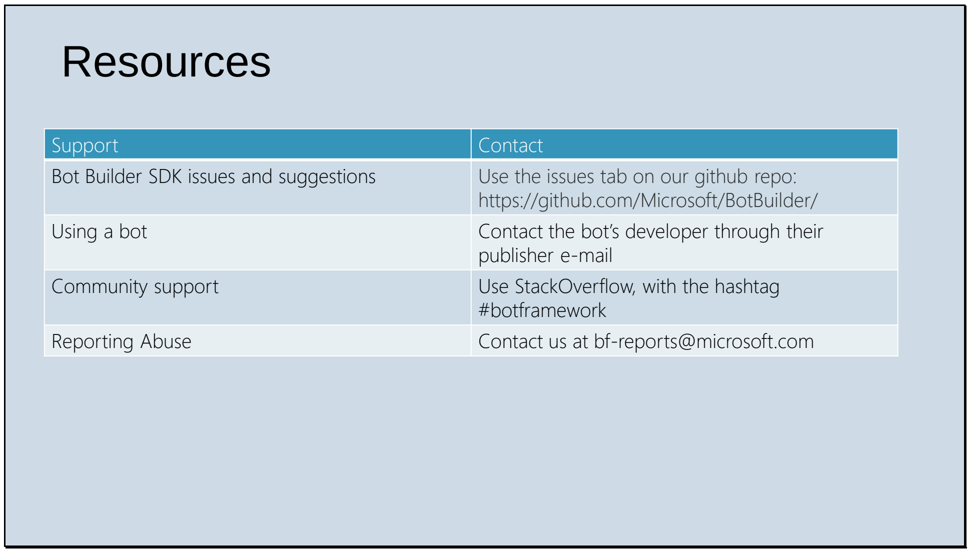
This data is currently stored for free for you within the Bot Framework State Service.

However, you may bring in your own data source (e.g. Azure Redis Cache)



Bot Framework Resources

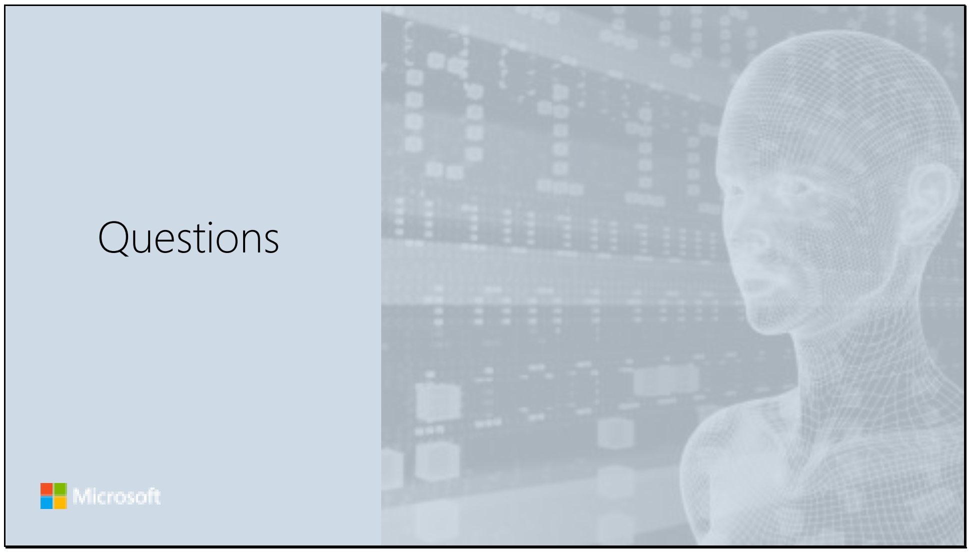
Slide 38



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

Slide 39



Questions

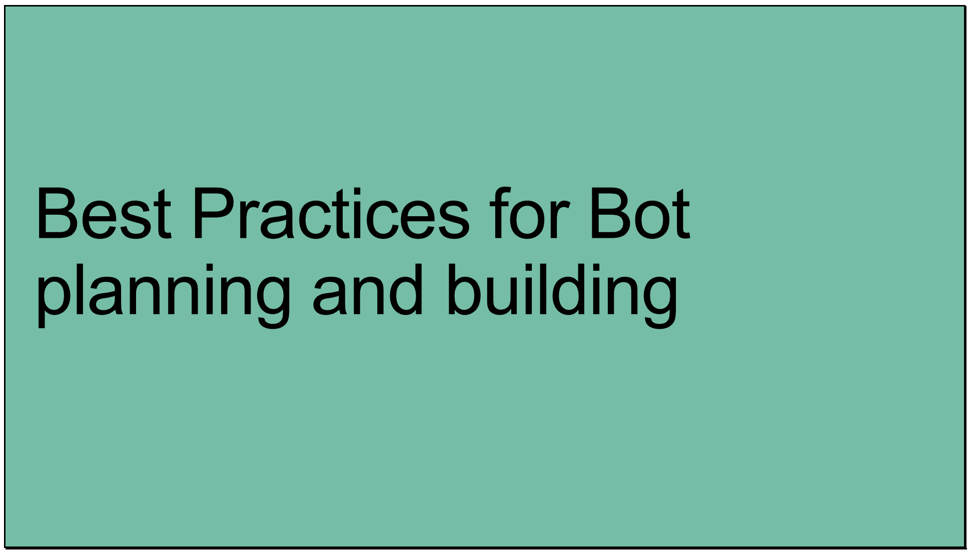
Microsoft

The User Experience

Experience Principles and Best Practices

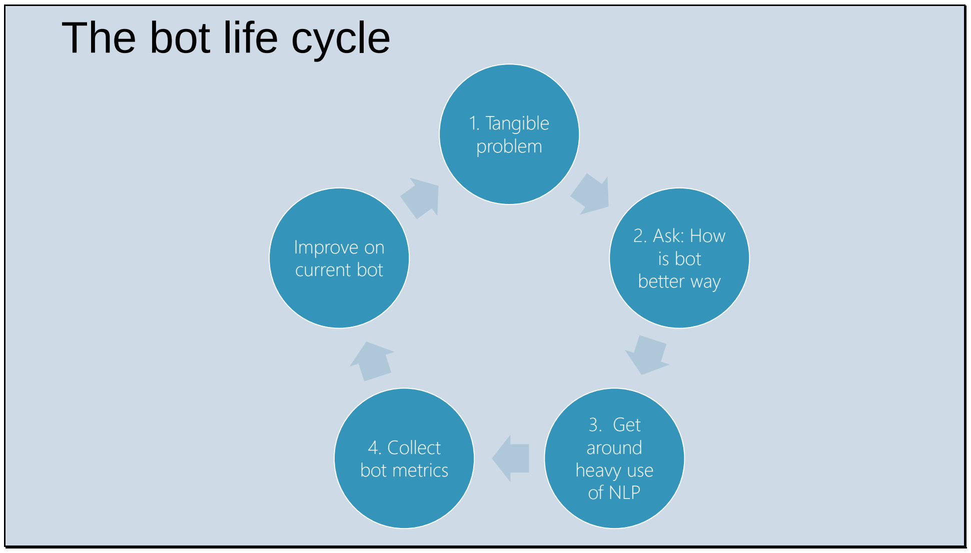
Presenter: Micheleen Harris

Microsoft



Best Practices for Bot planning and building

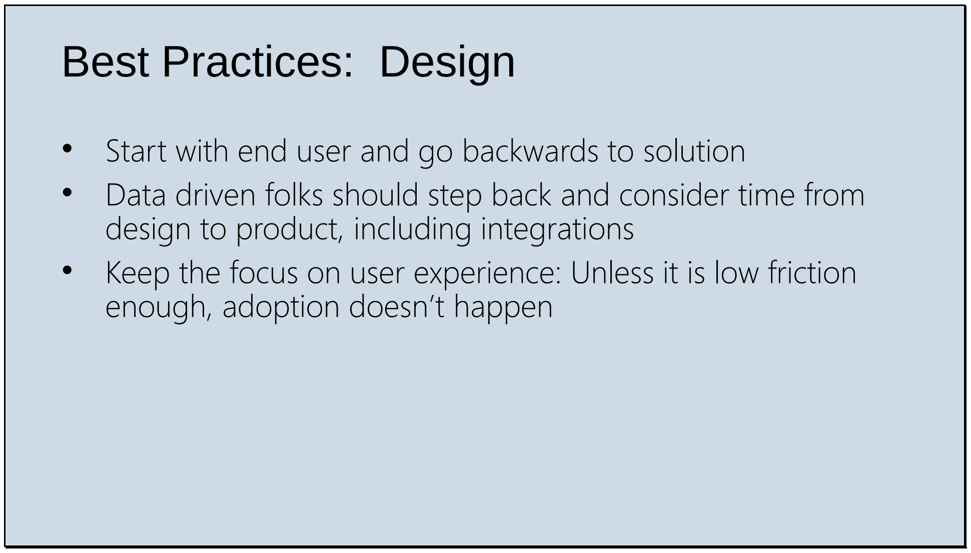
Slide 42



The bot life cycle

- 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.
- Collect data. Pump it to Azure. Use the same architecture you would use with IoT scenarios (IoT hub, stream analytics, power BI, etc). You can only analyze and improve your bot if you’re collecting metrics for it
- Iterate, improve

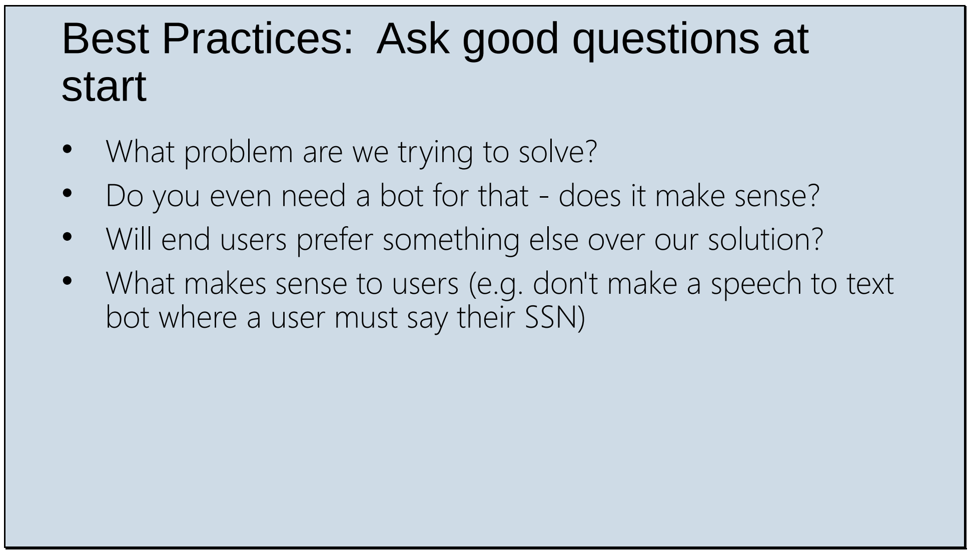
Slide 43



Best Practices: Design

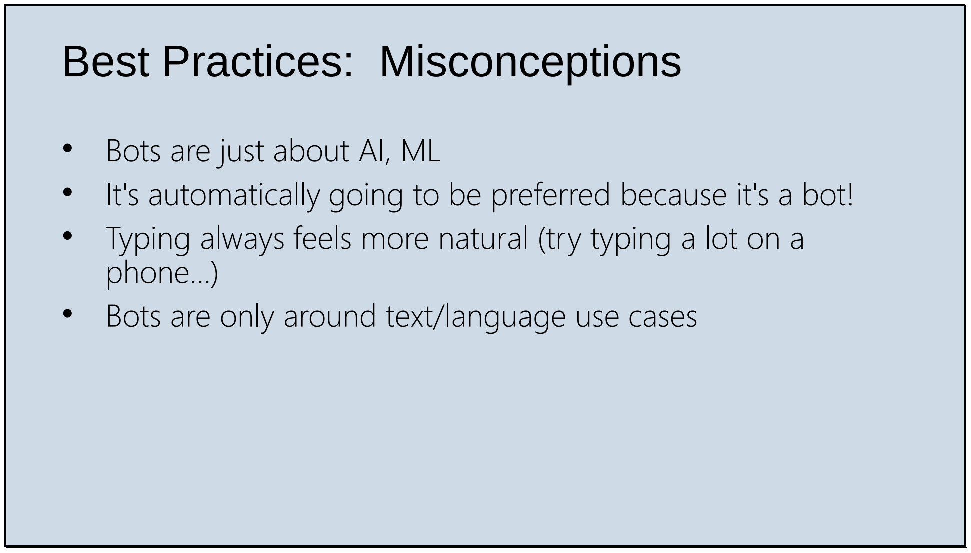
- Start with end user and go backwards to solution
- Data driven folks should step back and consider time from design to product, including integrations
- Keep the focus on user experience: Unless it is low friction enough, adoption doesn’t happen

Slide 44



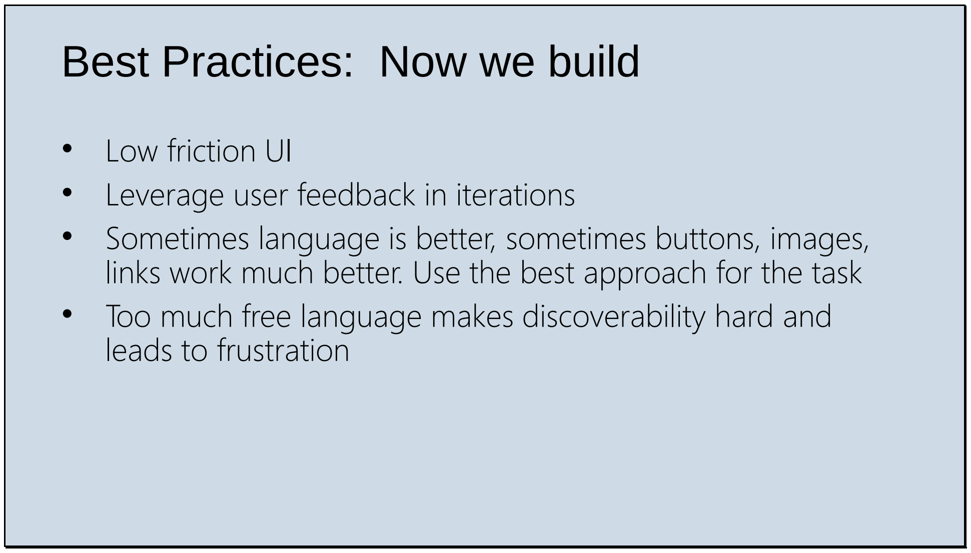
Best Practices: Ask good questions at start

- What problem are we trying to solve?
- Do you even need a bot for that - does it make sense?
- Will end users prefer something else over our solution?
- What makes sense to users (e.g. don't make a speech to text bot where a user must say their SSN)



Best Practices: Misconceptions

- Bots are just about AI, ML
- It's automatically going to be preferred because it's a bot!
- Typing always feels more natural (try typing a lot on a phone…)
- Bots are only around text/language use cases

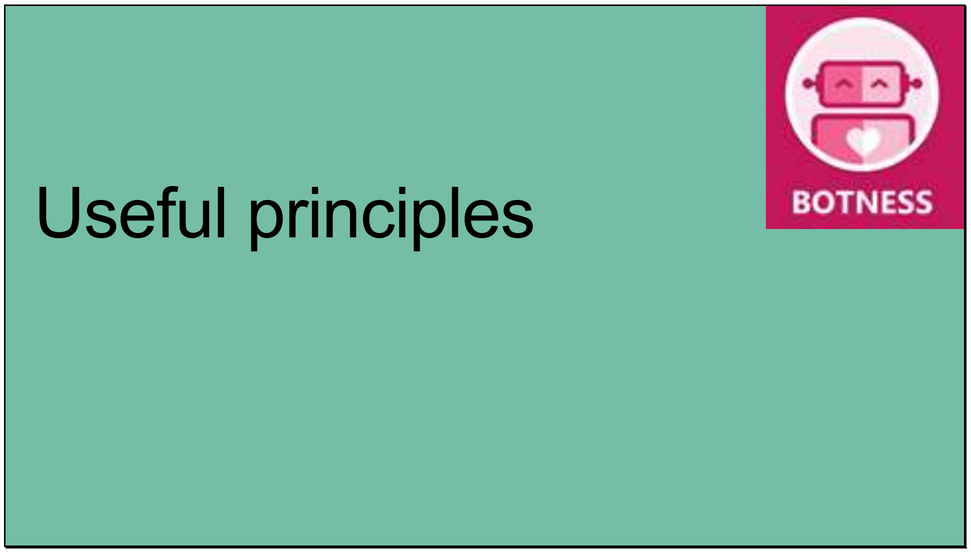


Best Practices: Now we build

- Low friction UI
- Leverage user feedback in iterations
- Sometimes language is better, sometimes buttons, images, links work much better. Use the best approach for the task
- Too much free language makes discoverability hard and leads to frustration

Would you play chess using a board, or just imagining it and speaking the moves out loud? Language is great, but it’s not the best experience in all cases. Typing adds friction, especially on mobile devices. Channels such as Skype allow you to leverage things like buttons and cards which are very nice ways to deal with some of this complexity. Do not over use language.

Don’t try to solve all scenarios with language. Menus work great. Simple tree based navigation paths work great.

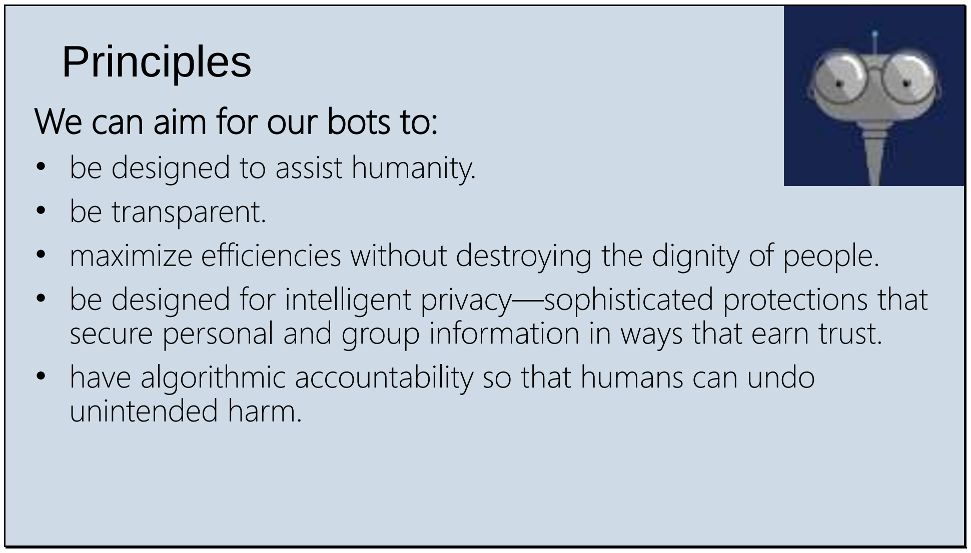


Useful principles

3 Scenarios: (Some customer names hidden – but all data real)

1. Azure customer support (tickets, complaints etc.,)
2. Health care industry (review about doctors etc.,)
3. Hospitality sector – hotel chain

Slide 48

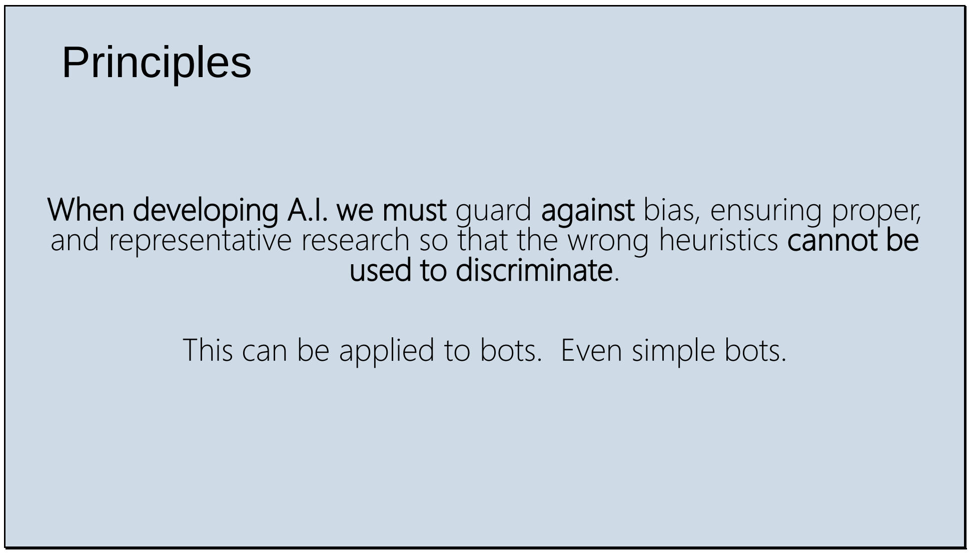


Principles

We can aim for our bots to:

- be designed to assist humanity.
- be transparent.
- maximize efficiencies without destroying the dignity of people.
- be designed for intelligent privacy—sophisticated protections that secure personal and group information in ways that earn trust.
- have algorithmic accountability so that humans can undo unintended harm.

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

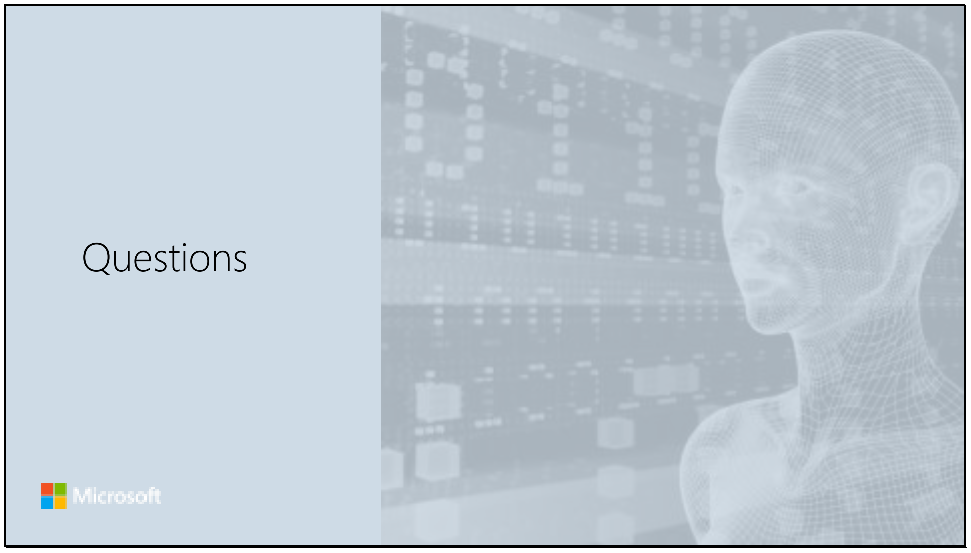


Principles

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

This can be applied to bots. Even simple bots.

Slide 50



Questions

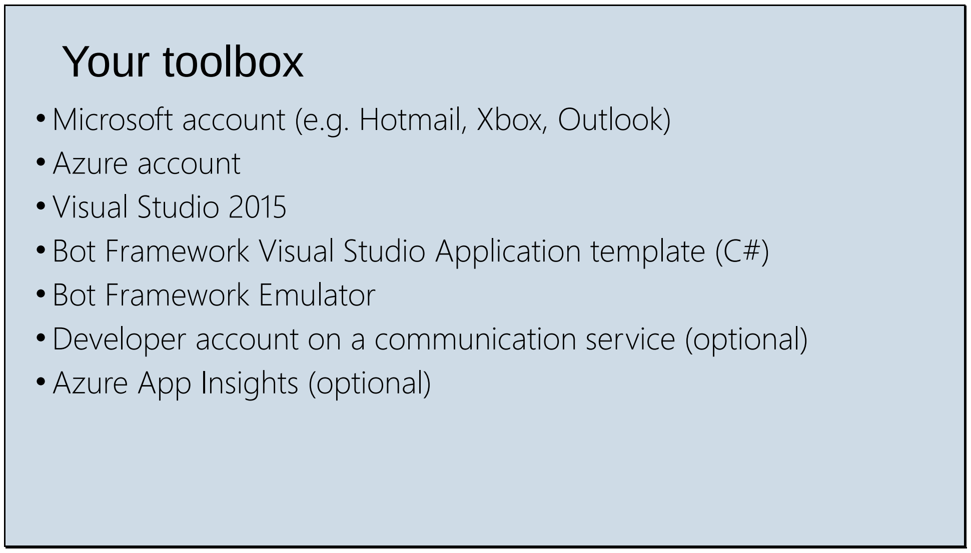
Microsoft

Slide 51

Bot Framework Developer Introduction with Hands-on-Lab

Getting started

Slide 53

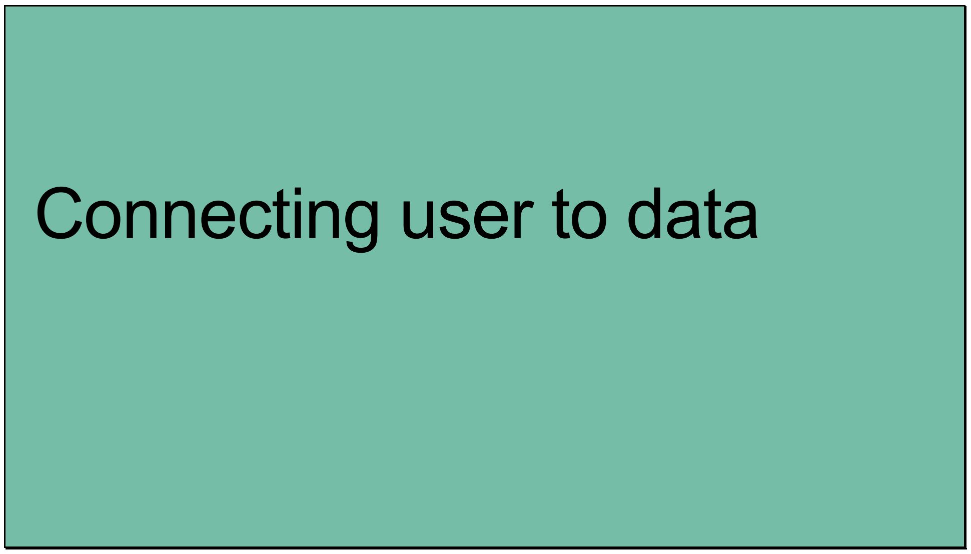


Your toolbox

- Microsoft account (e.g. Hotmail, Xbox, Outlook)
- Azure account
- Visual Studio 2015
- Bot Framework Visual Studio Application template (C#)
- Bot Framework Emulator
- Developer account on a communication service (optional)
- Azure App Insights (optional)

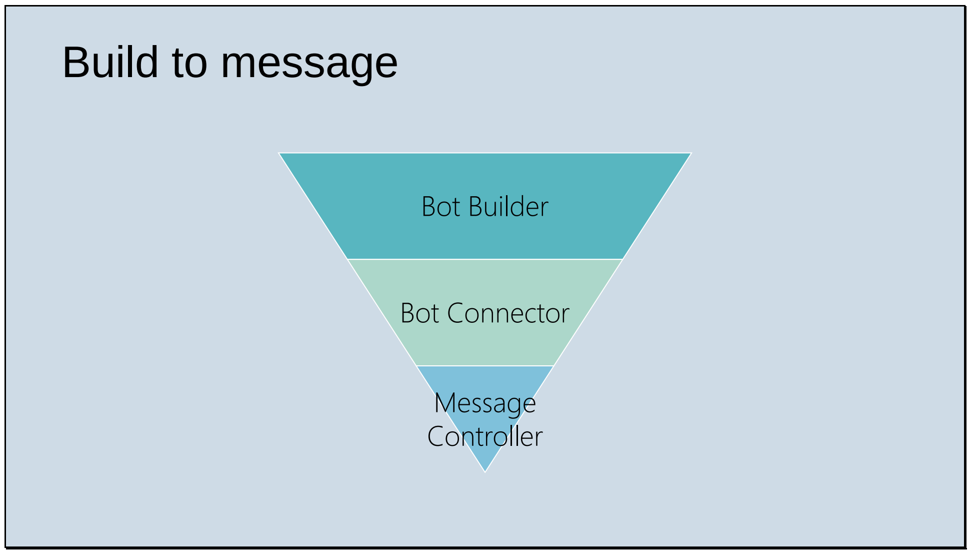
MS account - to log into the Bot Framework developer portal, which you will use to register your Bot
VS 2015 - www.visualstudio.com
Will need Azure account to publish bot as a web app service
Bot App template for VS 2015 – http://aka.ms/bf-bc-vstemplate
Emulator - https://aka.ms/bf-bc-emulator
Azure account – Bot Dev Portal and Azure accessible endpoint (REST endpoint for Connector Service callback)
Dev account on com service – if going beyond Skype
App Insights for collecting telemetry on bot

If interested in Node.js, there is a nice 3-part blog series on developing a bot in Node:



Connecting user to data

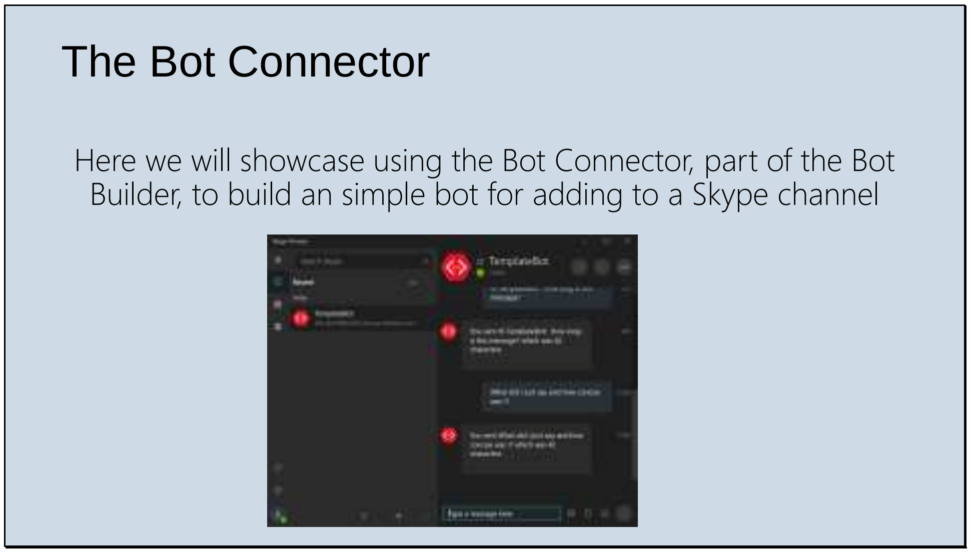
Slide 55



Build to message

User data <-> Create reply message

Slide 56

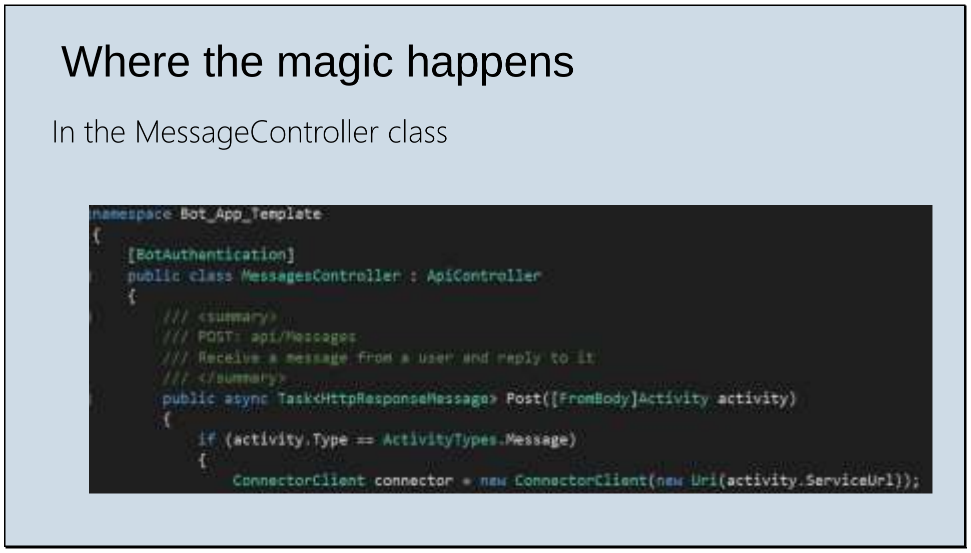


The Bot Connector

Here we will showcase using the Bot Connector, part of the Bot Builder, to build an simple bot for adding to a Skype channel

With the Bot Connector .NET template

Slide 57

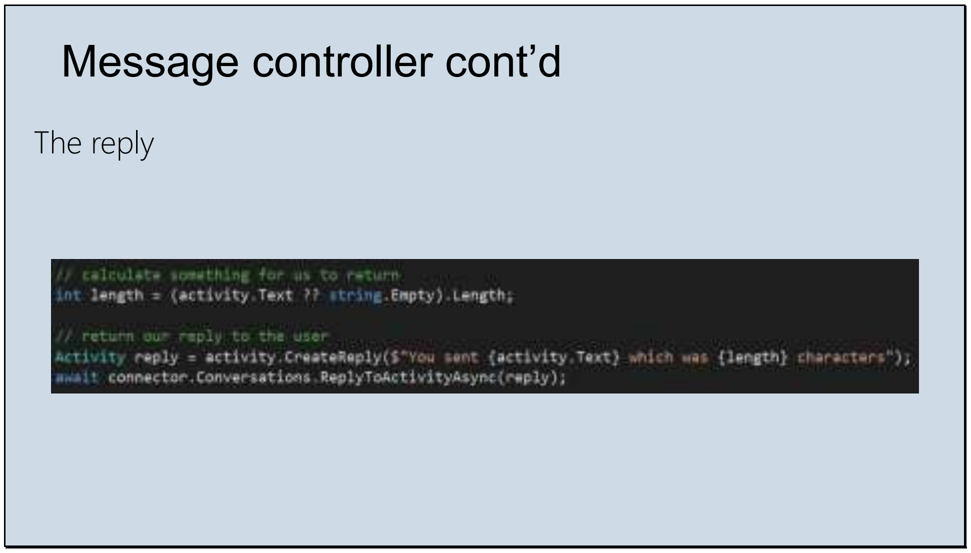


Where the magic happens

In the MessageController class

```
namespace Bot_App_Template
{
    [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)
            {
                ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
```

Slide 58

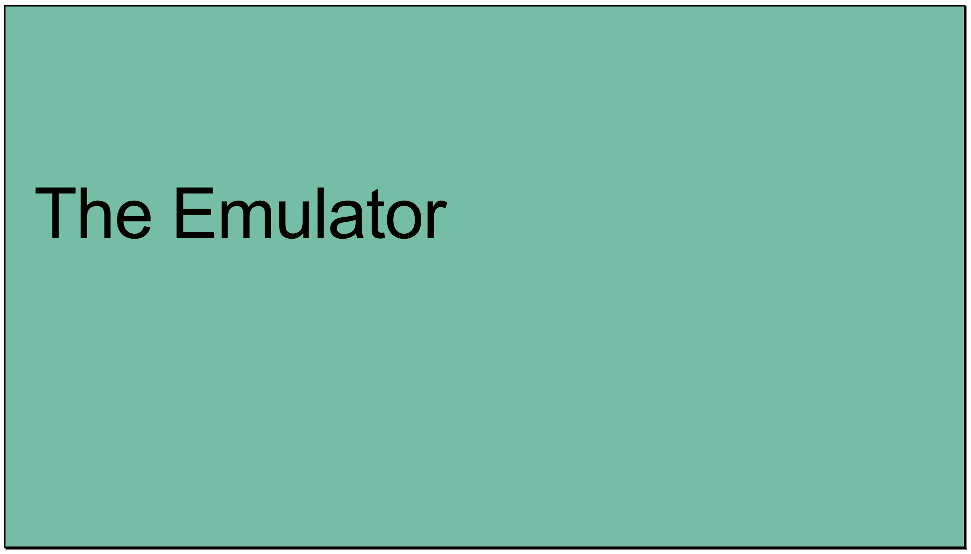


Message controller cont’d

The reply

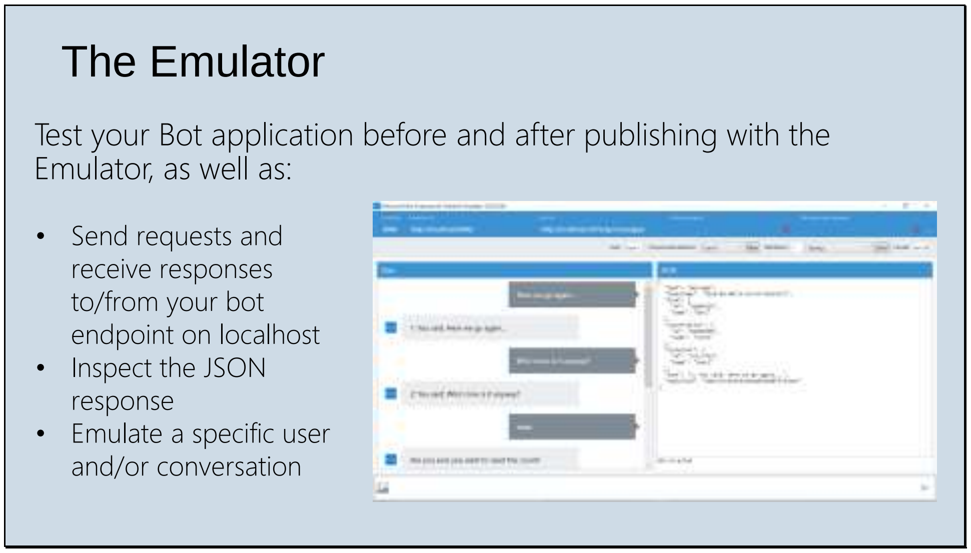
```
// calculate something for us to return
int length = (activity.Text ?? string.Empty).Length;

// return our reply to the user
Activity reply = activity.CreateReply($"You sent {activity.Text} which was {length} characters");
await connector.Conversations.ReplyToActivityAsync(reply);
```



The Emulator

Slide 60



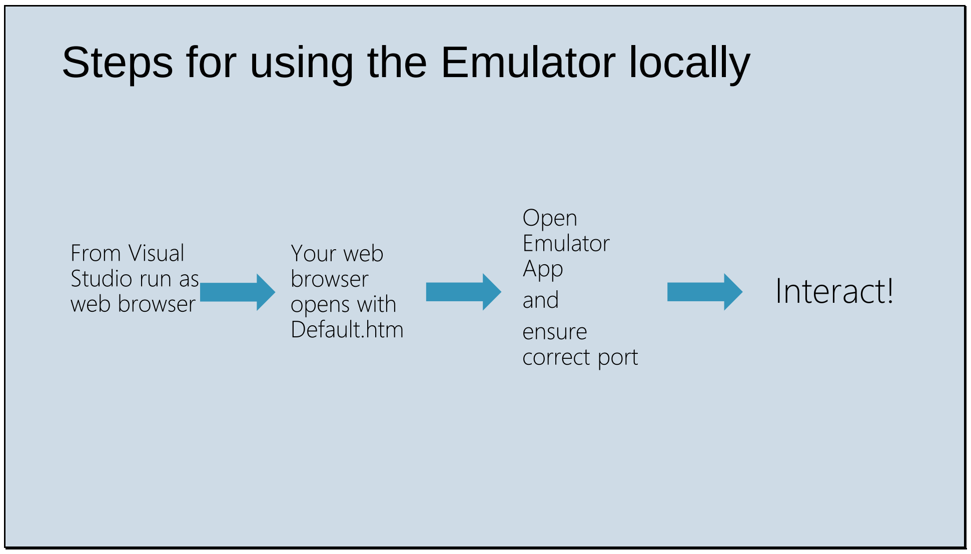
The Emulator

Test your Bot application before and after publishing with the Emulator, as well as:

- Send requests and receive responses to/from your bot endpoint on localhost
- Inspect the JSON response
- Emulate a specific user and/or conversation

Download here if you haven't already - https://aka.ms/bf-bc-emulator
Here we show the use of the Emulator with an endpoint on localhost of a running bot app

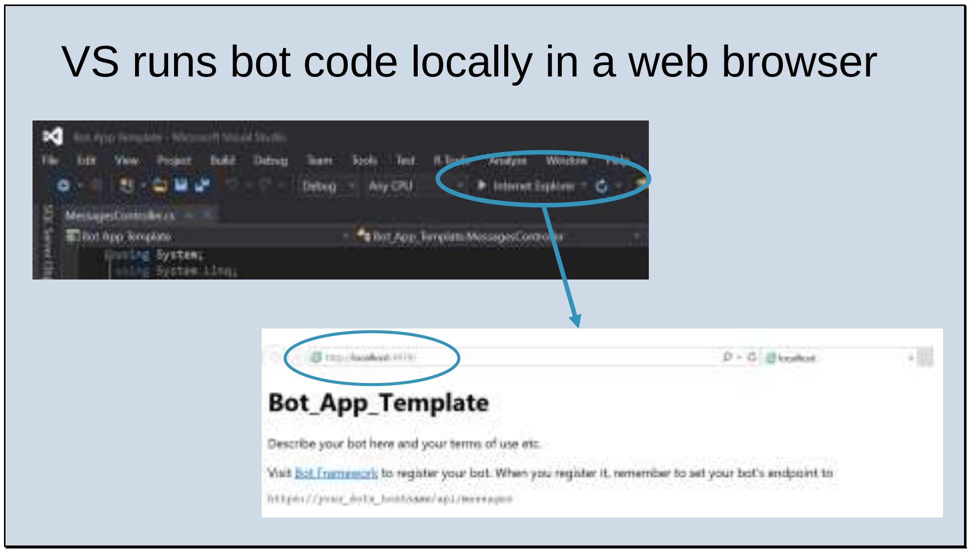
Slide 61



Steps for using the Emulator locally

From Visual Studio run as web browser → Your web browser opens with Default.htm → Open Emulator App and ensure correct port → Interact!

Slide 62



VS runs bot code locally in a web browser

```
using System;
using System.Linq;
```

Bot_App_Template

Describe your bot here and your terms of use etc.

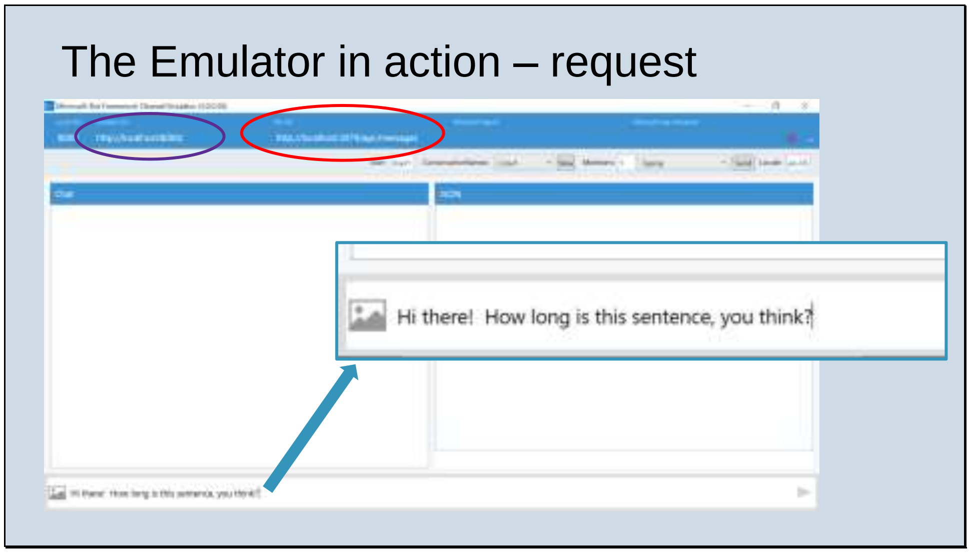
Visit Bot Framework to register your bot. When you register it, remember to set your bot's endpoint to

`https://your_bots_hostname/api/messages`

File Edit View Project Build Debug Team Tools Test R Tools Analyze Window Help
Debug
Any CPU
Internet Explorer
MessagesController.cs
Bot App Template
Bot_App_Template.MessagesController
http://localhost:3979/
localhost

Make a note of the port on the running app in browser (here, port 3979) and the endpoint URL. Next…test interactions locally with the Bot Framework Emulator

Slide 63



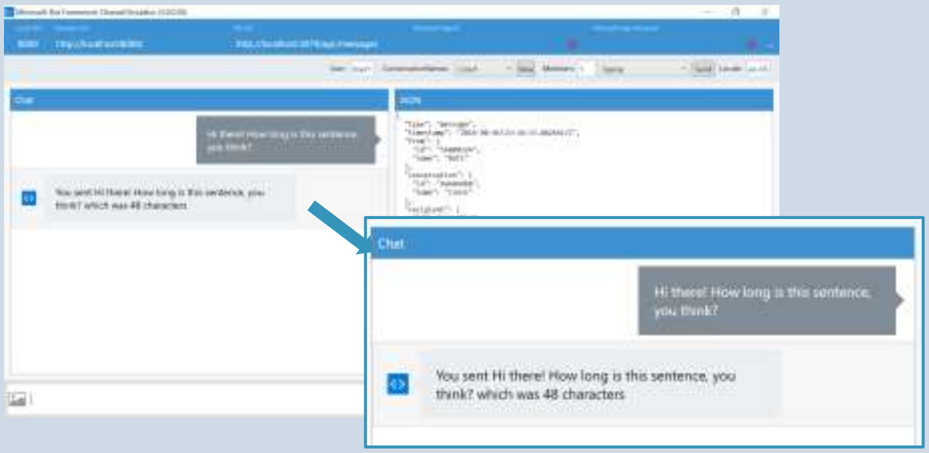
The Emulator in action – request

Hi there! How long is this sentence, you think?

Check the Bot Url port - that is corresponds to the app deployed by VS in browser (here, port 3979)
Emulator Url (purple circle) is the forwarding URL w/ https support (local port should agree with this)
The Bot Url (red circle) is the endpoint URL w/ https support

Slide 64

The Emulator in action – the response

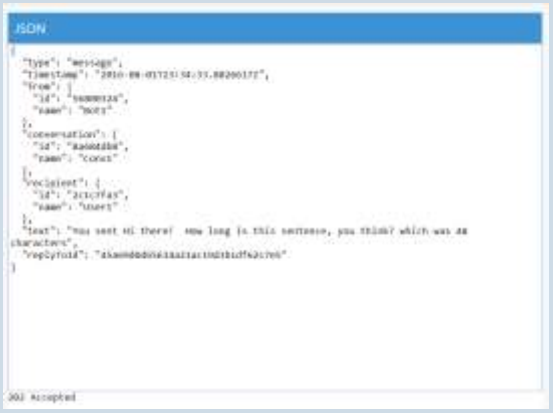


Hi there! How long is this sentence, you think?

You sent Hi there! How long is this sentence, you think? which was 48 characters

Slide 65

The Emulator in action – the response JSON



JSON

```
{
  "type": "message",
  "timestamp": "2016-08-01T23:34:33.88200172",
  "from": {
    "id": "56800324",
    "name": "Bot1"
  },
  "conversation": {
    "id": "8a684db8",
    "name": "Conv1"
  },
  "recipient": {
    "id": "2c1c7fa3",
    "name": "User1"
  },
  "text": "You sent Hi there!  How long is this sentence, you think? which was 48 characters",
  "replyToId": "[illegible]"
}
```

```
202 Accepted
```

Slide 66

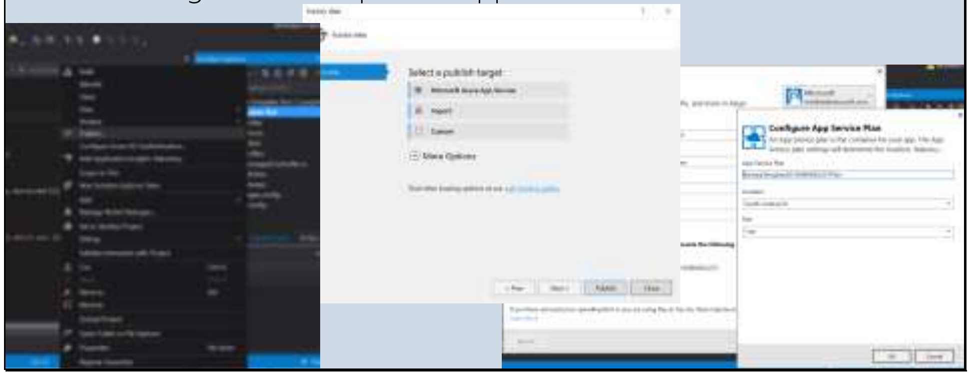
Publish

Pre-req: Azure Subscription from Azure account

Slide 67

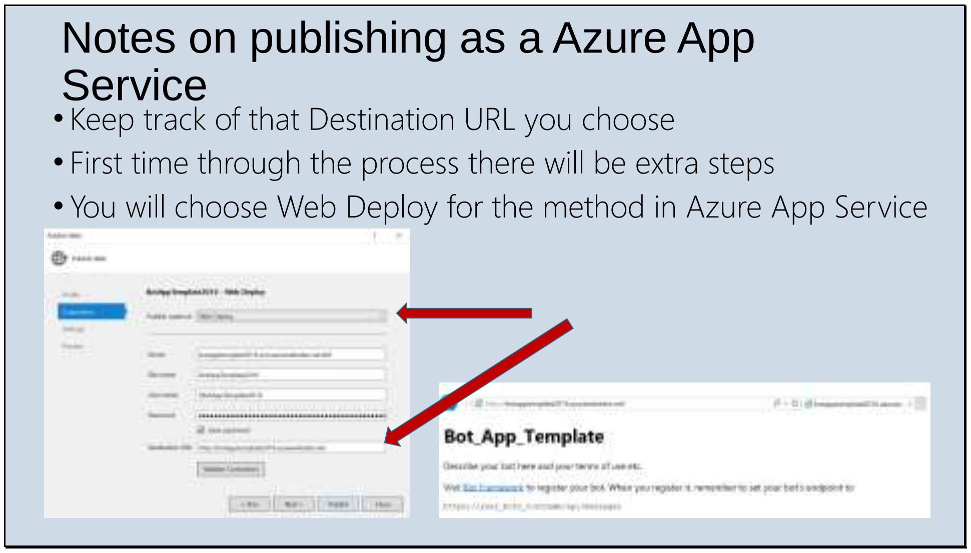
Publishing from VS 2015

- Publish as a Microsoft Azure App Service
- Go through wizard to publish app to Azure



Easy to publish directly from VS (but can do it other ways). Click on the name of the bot in VS and then ‘Publish…’

Slide 68



Notes on publishing as a Azure App Service

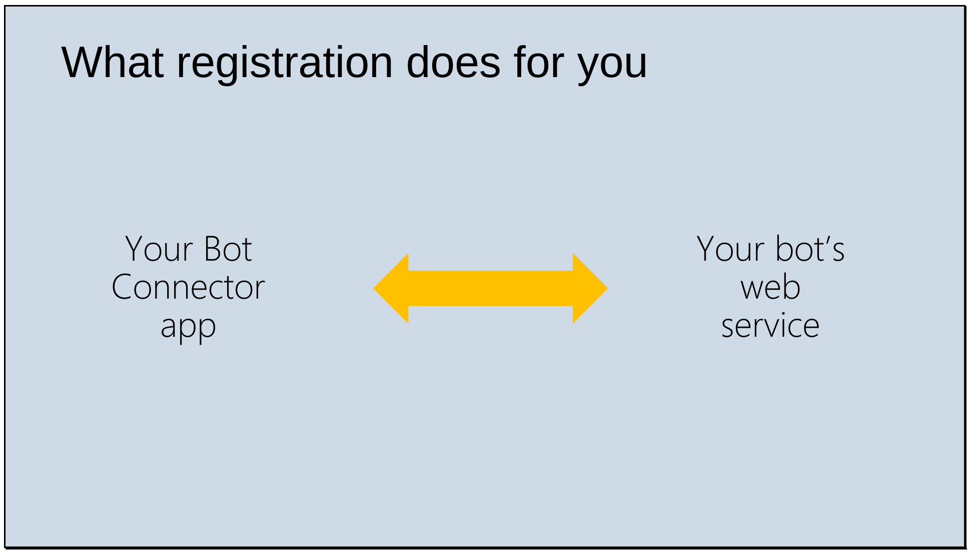
- Keep track of that Destination URL you choose
- First time through the process there will be extra steps
- You will choose Web Deploy for the method in Azure App Service

Bot_App_Template

URL will be needed to update BF registration endpoint
The extra steps will only have to be performed once

Register

Slide 70



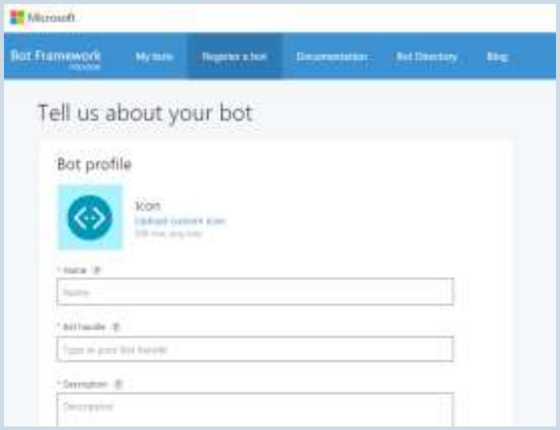
What registration does for you

Your Bot Connector app
Your bot's web service

Happens in the MS Bot Framework portal

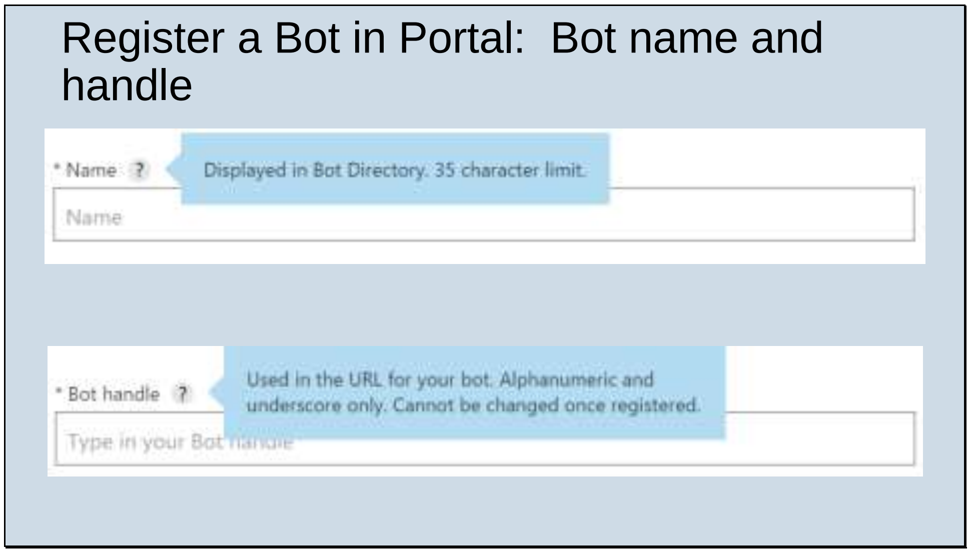
Slide 71

Register a Bot



Register on the developer portal by clicking the ‘Register a bot’ link:
https://dev.botframework.com/bots/new

Slide 72



Register a Bot in Portal: Bot name and handle

* Name ? Displayed in Bot Directory. 35 character limit.

Name

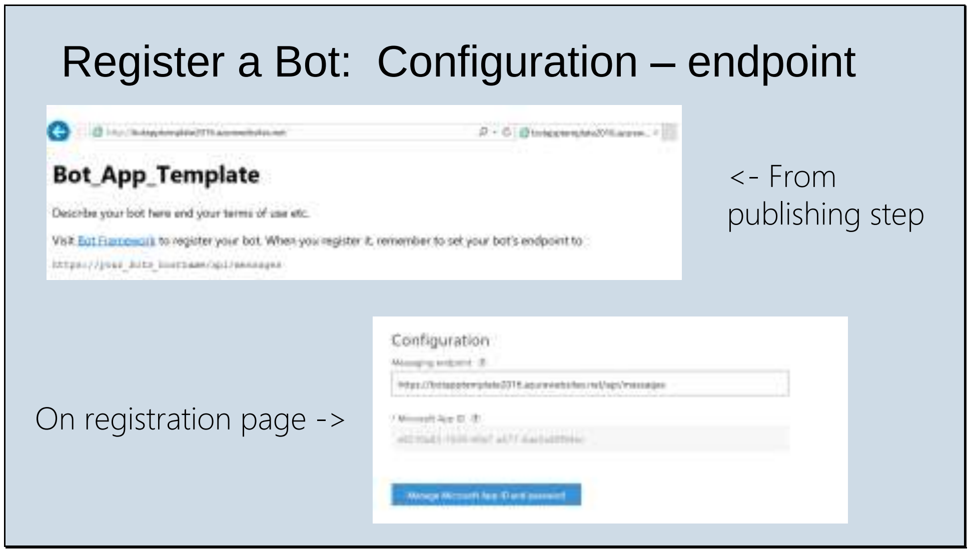
* Bot handle ? Used in the URL for your bot. Alphanumeric and underscore only. Cannot be changed once registered.

Type in your Bot handle

E.g.
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

Slide 73



Register a Bot: Configuration – endpoint

Bot_App_Template

Describe your bot here and your terms of use etc.

Visit Bot Framework to register your bot. When you register it, remember to set your bot's endpoint to

```
https://your_bots_hostname/api/messages
```

<- From publishing step

On registration page ->

Configuration

Messaging endpoint

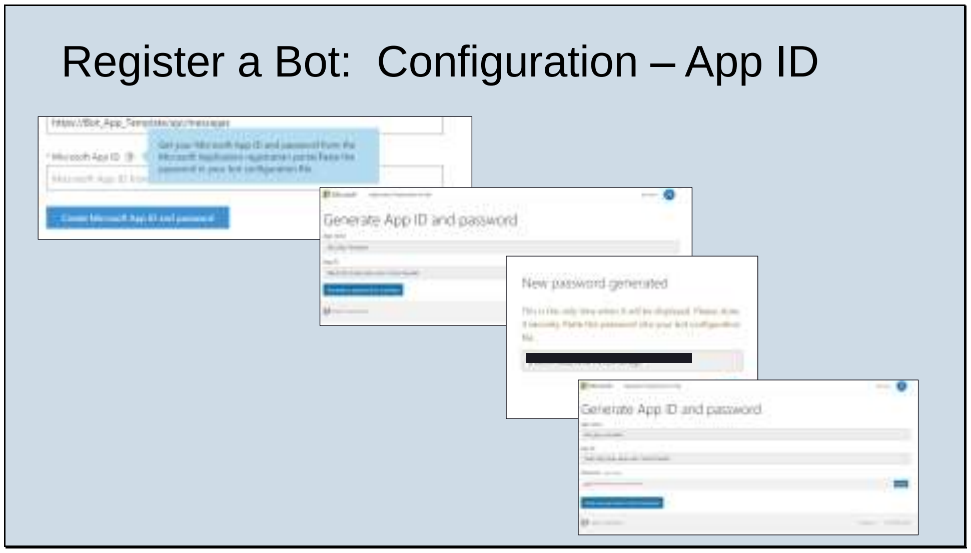
[illegible]

Microsoft App ID

[illegible]

Remember the URL endpoint from publishing step (in VS) and the browser window opened.
Should be something like: “https://*botwebappname.azurewebsites.net*/api/messages”

Slide 74



Register a Bot: Configuration – App ID

https://Bot_App_Template/api/messages

* Microsoft App ID

Get your Microsoft App ID and password from the Microsoft Application registration portal. Paste the password in your bot configuration file.

Create Microsoft App ID and password

Generate App ID and password

New password generated

This is the only time when it will be displayed. Please store it securely. Paste this password into your bot configuration file.

[redacted]

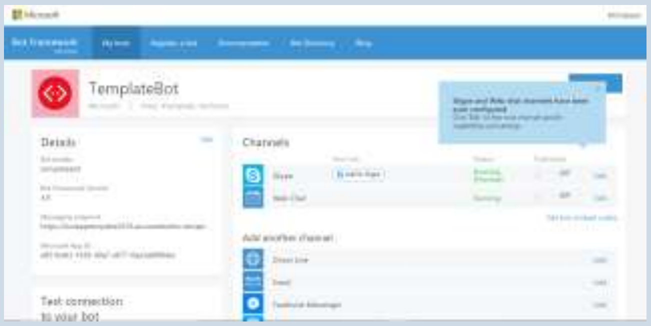
Generate App ID and password

You’ll go through the “Generate App ID and password” wizard, then return to the registration page.

Slide 75

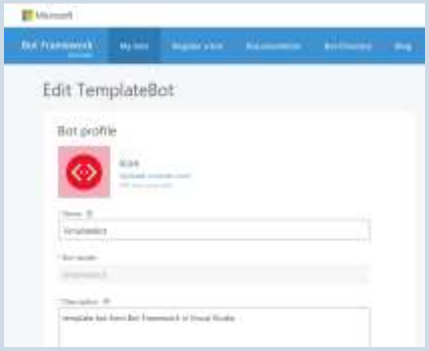
Update web configuration in VS

- In VS update the Web.config file with the Microsoft App ID and password
- Republish in VS



Slide 76

Edit profile anytime


Microsoft

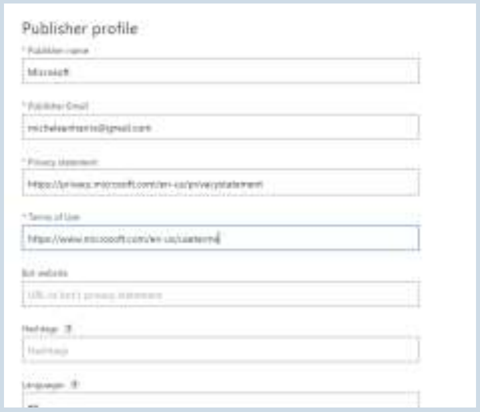
Bot Framework | My bots | Register a bot | Documentation | Bot Directory | Blog

Edit TemplateBot

Bot profile

Name: TemplateBot

Description: template bot from Bot Framework in Visual Studio


Publisher profile

* Publisher name: Microsoft

* Publisher Email: [illegible]@gmail.com

* Privacy statement: https://privacy.microsoft.com/en-us/privacystatement

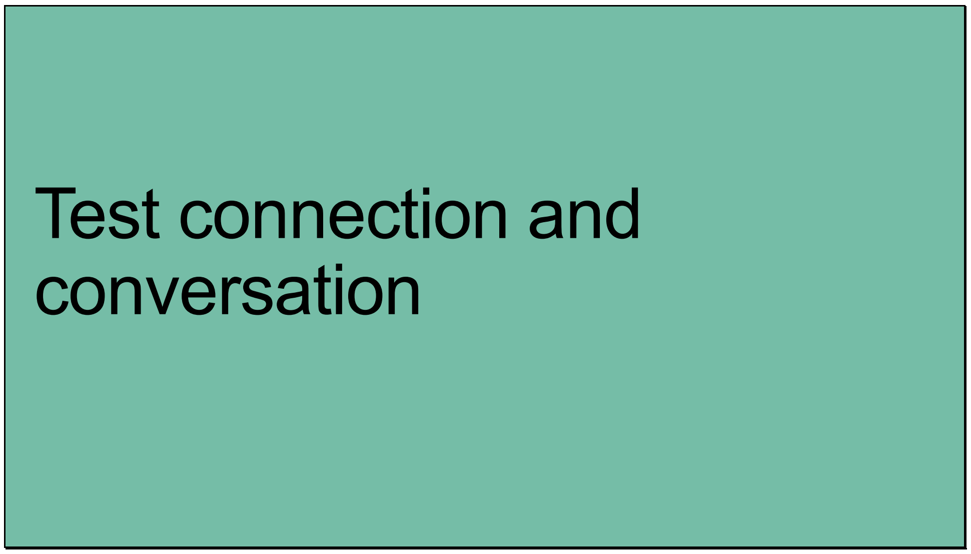
* Terms of Use: https://www.microsoft.com/en-us/useterms

Bot website: URL to bot's privacy statement

Hashtags: Hashtags

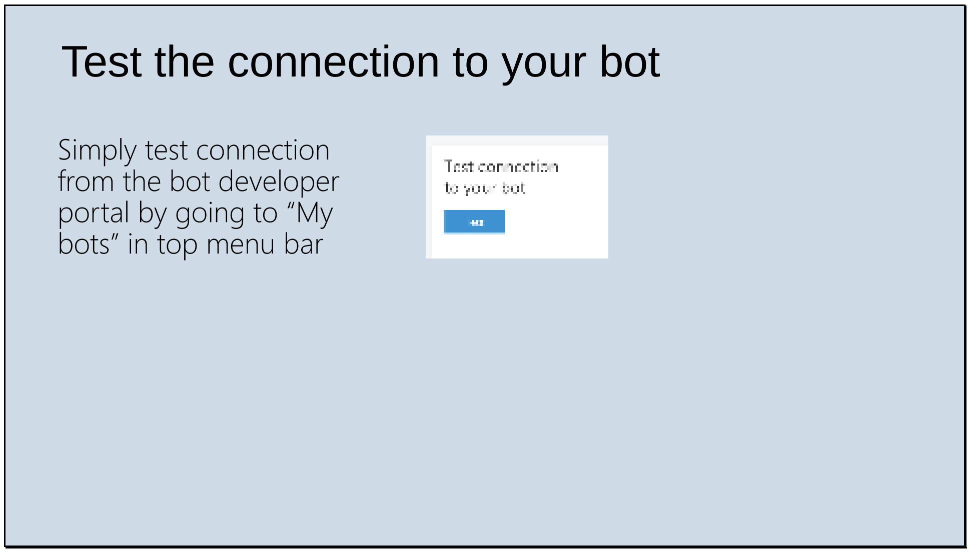
Languages

Could also just have http://microsoft.com for the Privacy statement and Terms of Use



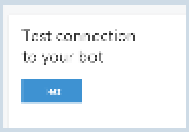
Test connection and conversation

Slide 78



Test the connection to your bot

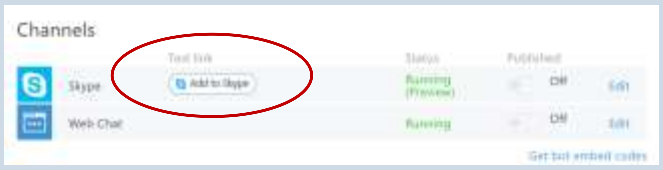
Simply test connection from the bot developer portal by going to “My bots” in top menu bar



Slide 79

Test connection with channels

Test bot with a pre-configured channels



Channels

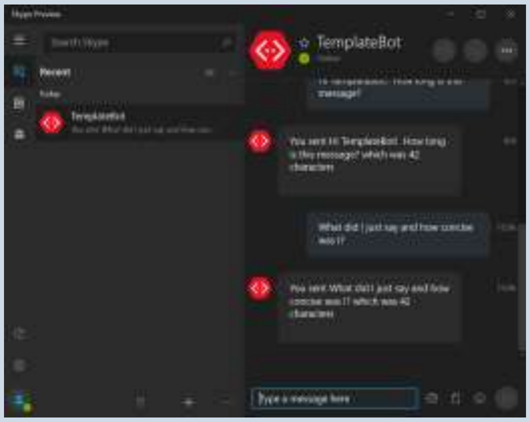
	Test link	Status	Published	
Skype	Add to Skype	Running (Preview)	Off	Edit
Web Chat		Running	Off	Edit

Get bot embed codes

Slide 80

Test connection with skype channel

Add bot to contacts and began a chat…



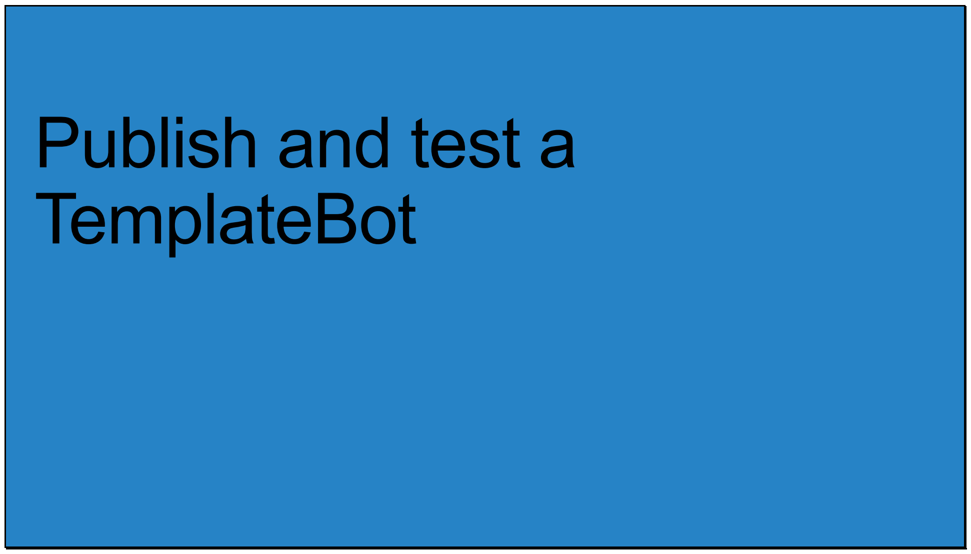
message?

You sent Hi TemplateBot. How long is this message? which was 42 characters

What did I just say and how concise was I?

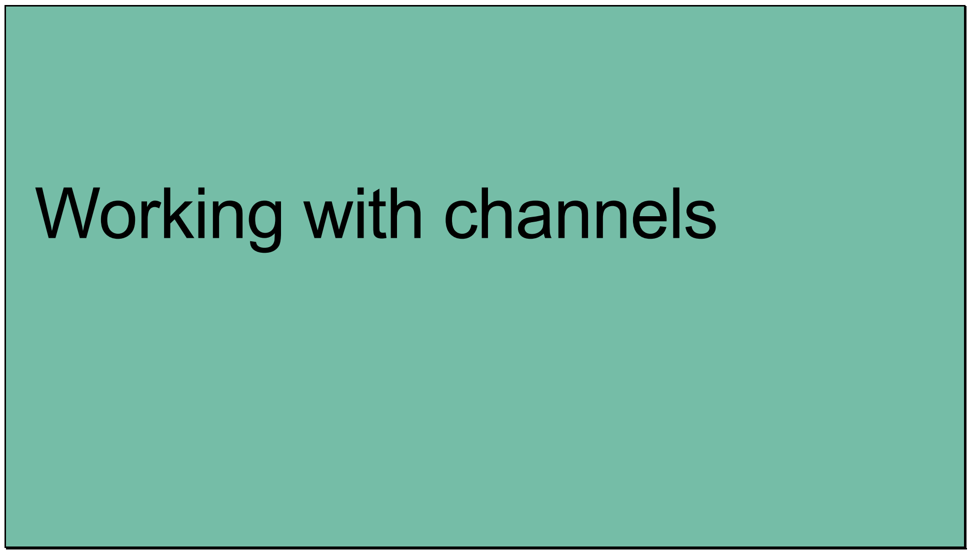
You sent What did I just say and how concise was I? which was 42 characters

From developer portal page, clicked on test link “Add to Skype” and added bot to my contacts for testing.



Publish and test a TemplateBot

We will do this together in Visual Studio and the portal by going through these slides step by step

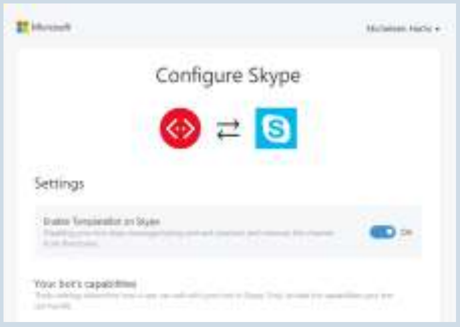


Working with channels

Slide 83

Editing a channel

Skype for instance:



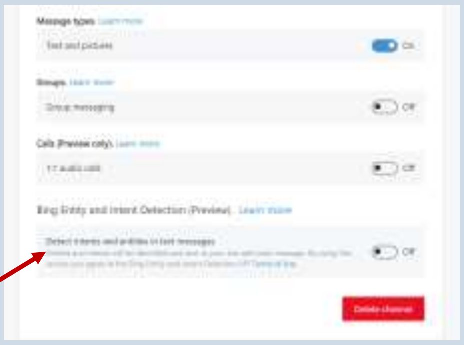
Microsoft

Configure Skype

Settings

Enable Templatebot on Skype — On

Your bot's capabilities



Message types Learn more

Text and pictures — On

Groups Learn more

Group messaging — Off

Calls (Preview only). Learn more

1:1 audio calls — Off

Bing Entity and Intent Detection (Preview). Learn more

Detect intents and entities in text messages — Off

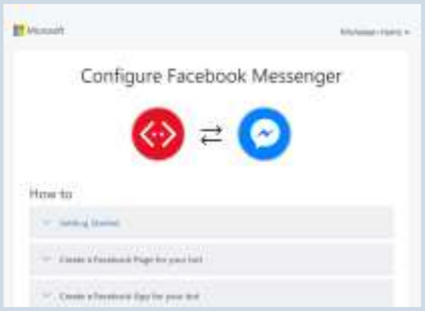
Delete channel

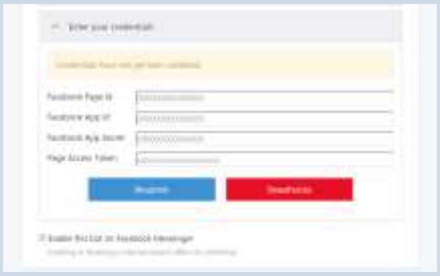
Being replaced shortly with LUIS

Slide 84

Adding a channel

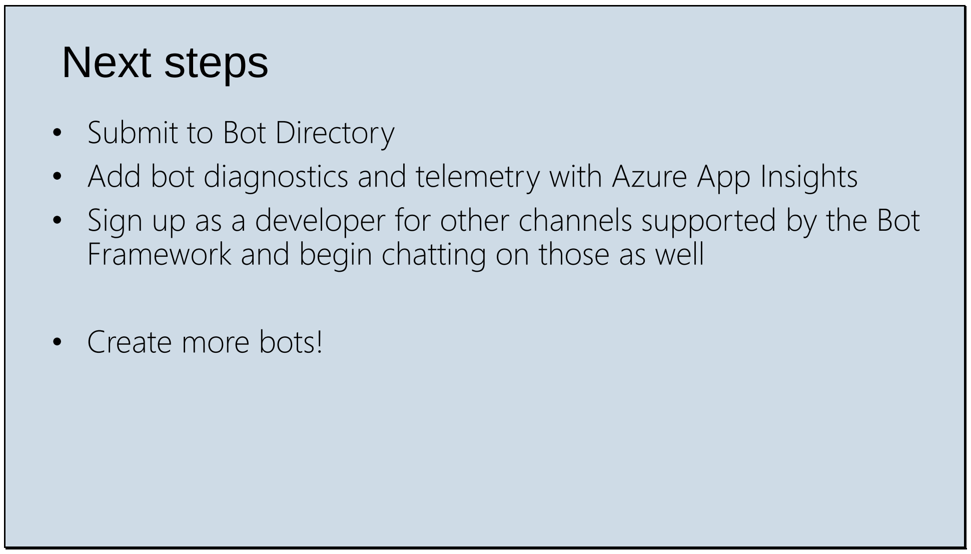
Many channels will require your credentials as a developer on the service, e.g., Facebook channel





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

Next steps

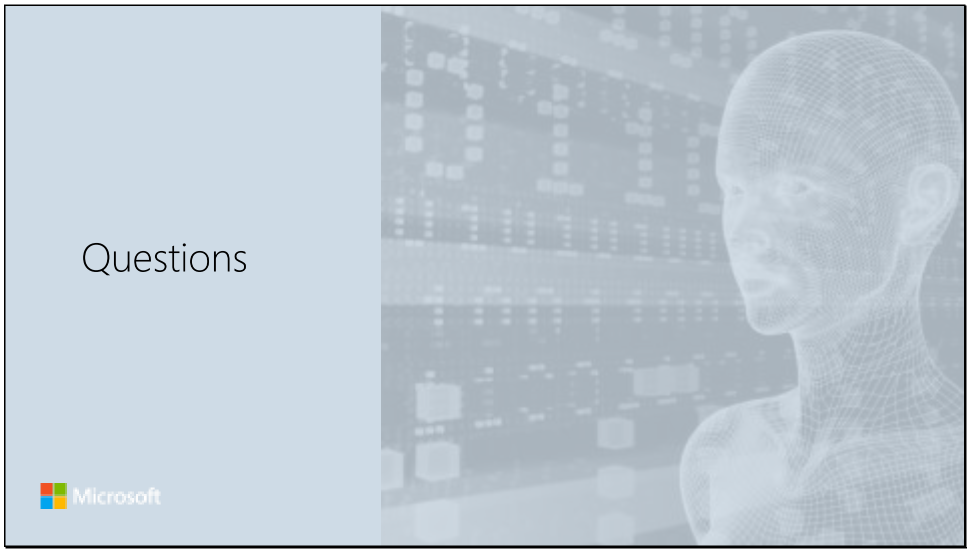


Next steps

- Submit to Bot Directory
- Add bot diagnostics and telemetry with Azure App Insights
- Sign up as a developer for other channels supported by the Bot Framework and begin chatting on those as well

- Create more bots!

Slide 87



Questions

Microsoft