

Making a Clever Chatbot for World Domination

Rob Miles
Author and Raconteur
www.robmiles.com

Agenda

An introduction to chatbots

Making a simple chatbot

Making an FAQ chatbot

Chatbots and robots

Adding language understanding



What is a chatbot or “bot”?

A chatbot chats with users

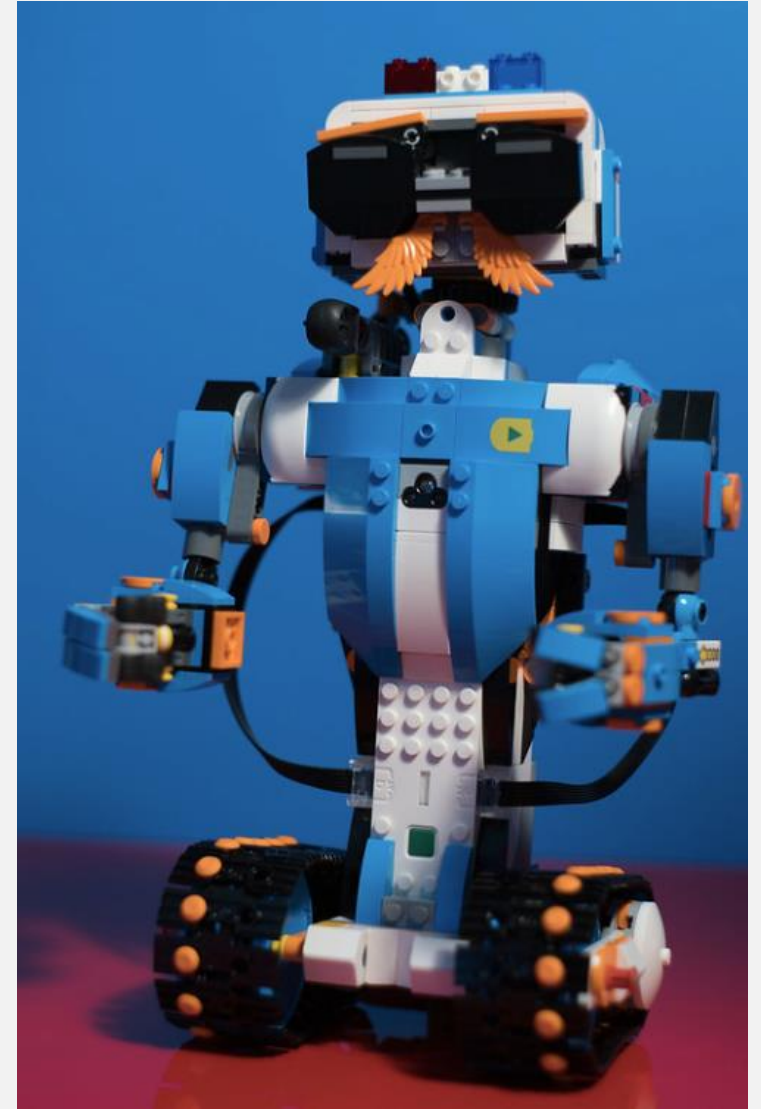
It can use a variety of different channels including voice or a web page

Preferable to installing an application – which users just don’t bother with

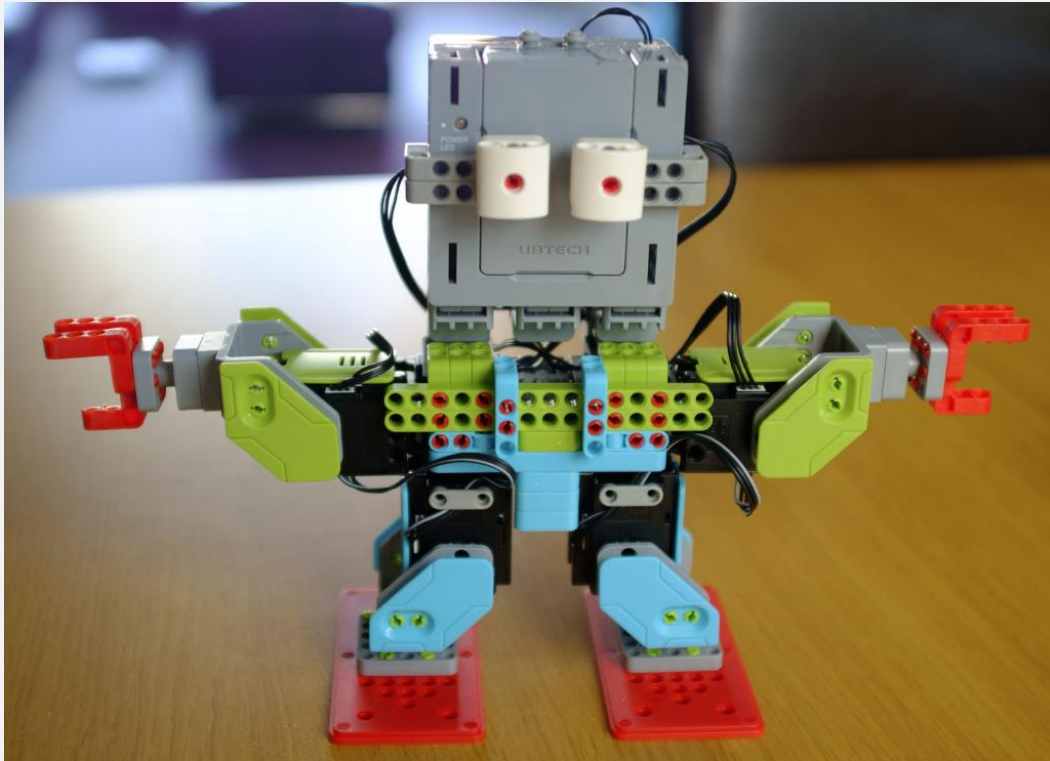
I’m going to talk about the Microsoft Bot Framework

Open Source, can be hosted anywhere

Grown to version 4.0



Bots are best as dumb devices



We don't have humanoid robots helping us do the housework, but we do have washing machines and vacuum cleaners.

These are robots that have been specifically designed to perform a simple task

The same is true when we think about bots.

Don't consider them as "your plastic pal who's fun to be with".

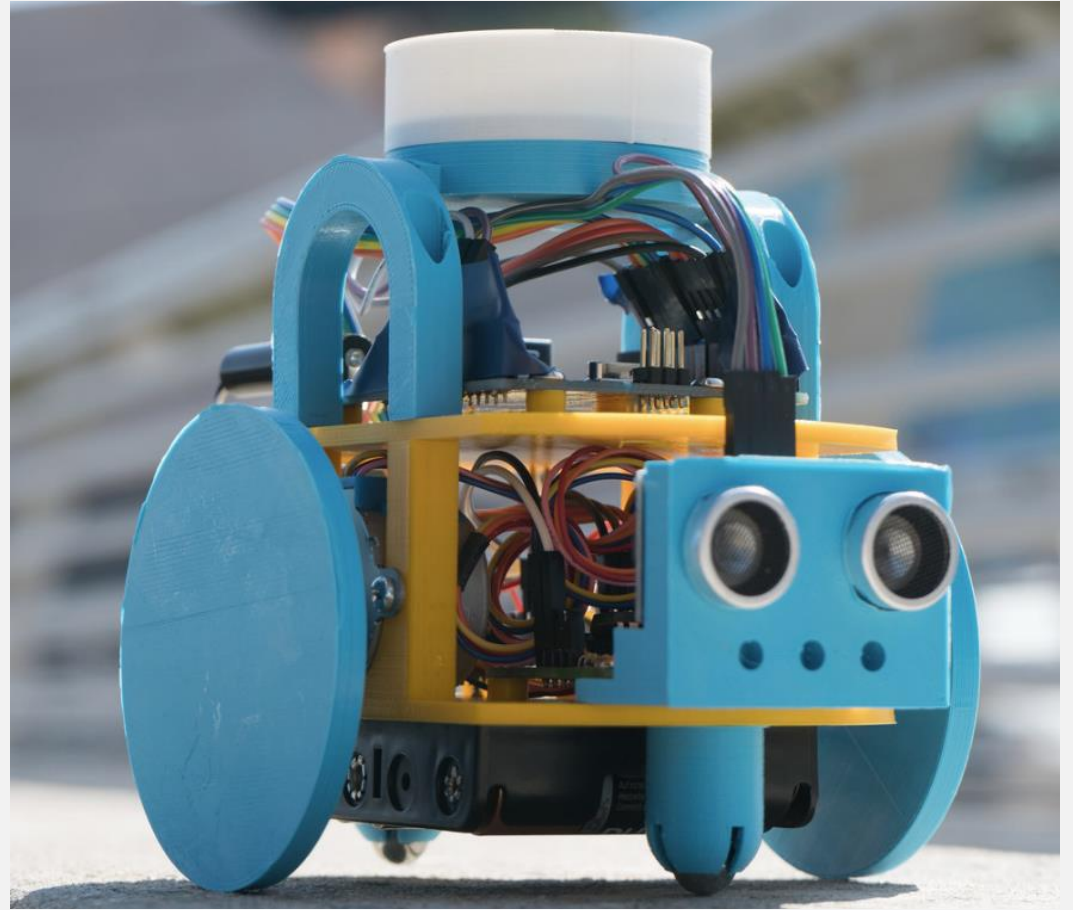
Think of them as something that you can use to book a flight, reserve a game or ask about the weather.

Bots must be helpful rather than human

A bit of humanity is a good idea, but it is much more useful if the bot can respond in a helpful way when it fails to understand what is going on

We will see how to give bots “personality”

You can also build in the ability to “escalate” if a bot user needs human attention



Making bots

A bot is a data processor

In its simplest form a bot is takes something in, does something with it, and then produces an output
This is a classic model of a computer. When we look at the bot framework in a minute we'll see something that lets us take something and push back a reply

Bots vs Wizards

A good way to regard a bot is as a wizard that guides the user through a process. When you are booking a haircut or changing your password you don't want conversation, just a particular outcome. Considering it this way and recognising the importance of context will make for great bots

You don't need to understand natural language

You can get a long way with regular expressions or just simple text analysis

Context is key

A crucial way to improve your bot is to have it maintain context during the conversation. So that when the user says "it" meaning "the book I'm buying" you can respond sensibly.

Clever Bots

You can make clever bots with LUIS

The LUIS (Language Understanding Intelligent Service) can be used to improve how well your bot understands natural language

It adds voice control

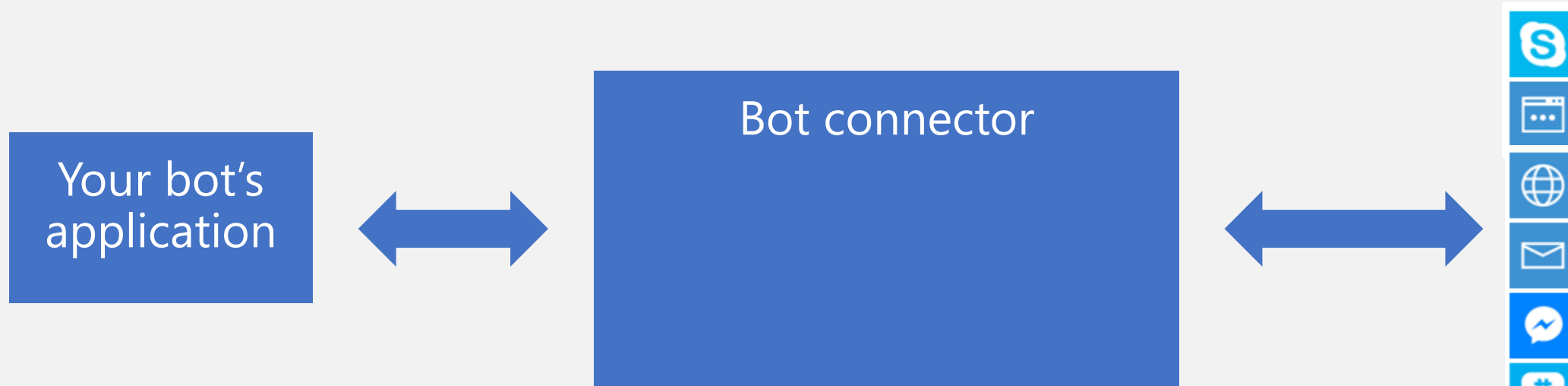
You can train LUIS to work in your particular context

You can make bots that have empathy with their users

There are even tools that let your bot determine how happy your user is – so that you can automatically escalate a conversation

We're not going to do anything like that though, we are just going to take in text, do something with it and then send back a response

Microsoft bot framework architecture



The Bot connector provides the glue

Your bot sits behind a url and talks to the bot services

The services then connect to a range of different connection technologies to provide the required interaction

The connector manages the conversion of the conversation type to the channels

Bot Options

C# and .NET

Write C# applications that contain bot instances

JavaScript and Node

Embed in web page using JavaScript and Node

REST API

Create restful calls from any device that you fancy

Getting Started

<https://dev.botframework.com/>

Bot Options

C# and .NET

Write C# applications that contain bot instances (this is what we are going to do)

JavaScript and Node

Embed in web page using JavaScript and Node

REST API

Create restful calls from any device that you fancy

Getting Started

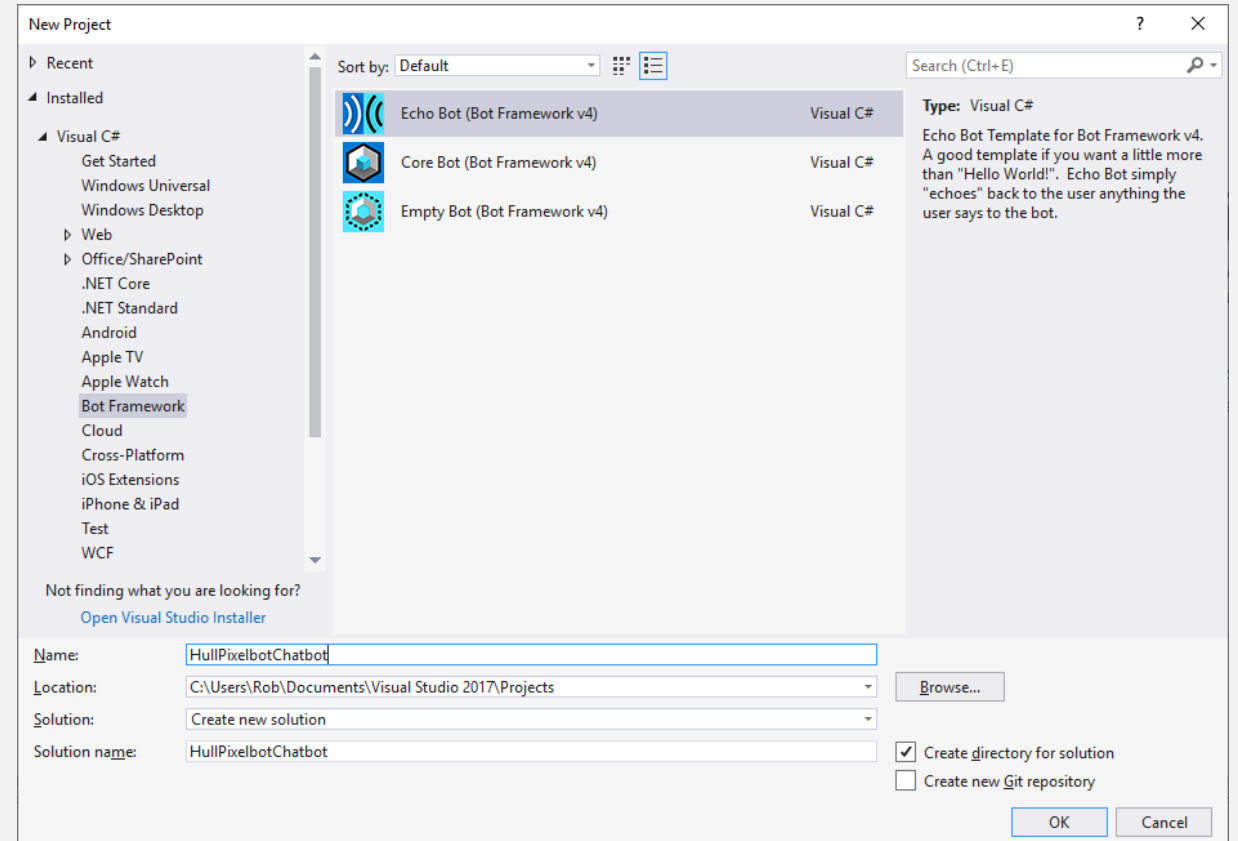
<https://dev.botframework.com/>

Chatbot project templates

Download the template

Make a new C# project from the template

There are numerous versions, I'm starting with the echo one



Template available from visx:

<https://marketplace.visualstudio.com/items?itemName=BotBuilder.botbuilderv4>

Talking to the chatbot



HullPixelbotChatbot Bot

Your bot is ready!

You can test your bot in the Bot Framework Emulator by connecting to <http://localhost:3978/api/messages>.

[Download the Emulator](#)

HOW TO BUILD A BOT



Plan:

Review the bot [design guidelines](#)



Build:

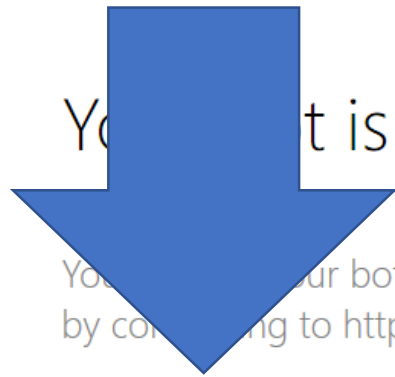
Create a bot from [Azure](#) or [locally](#)
Download [Command-line tools](#)
Add services such as [Language Understanding](#),
[QnA Maker](#) and [Dispatch](#)

We can fire up our chatbot and talk to it

Talking to the chatbot



HullPixelbotChatbot Bot



Your bot is ready!

You can run your bot in the Bot Framework Emulator by connecting to <http://localhost:3978/api/messages>.

[Download the Emulator](#)

HOW TO BUILD A BOT



Plan:

Review the bot [design guidelines](#)

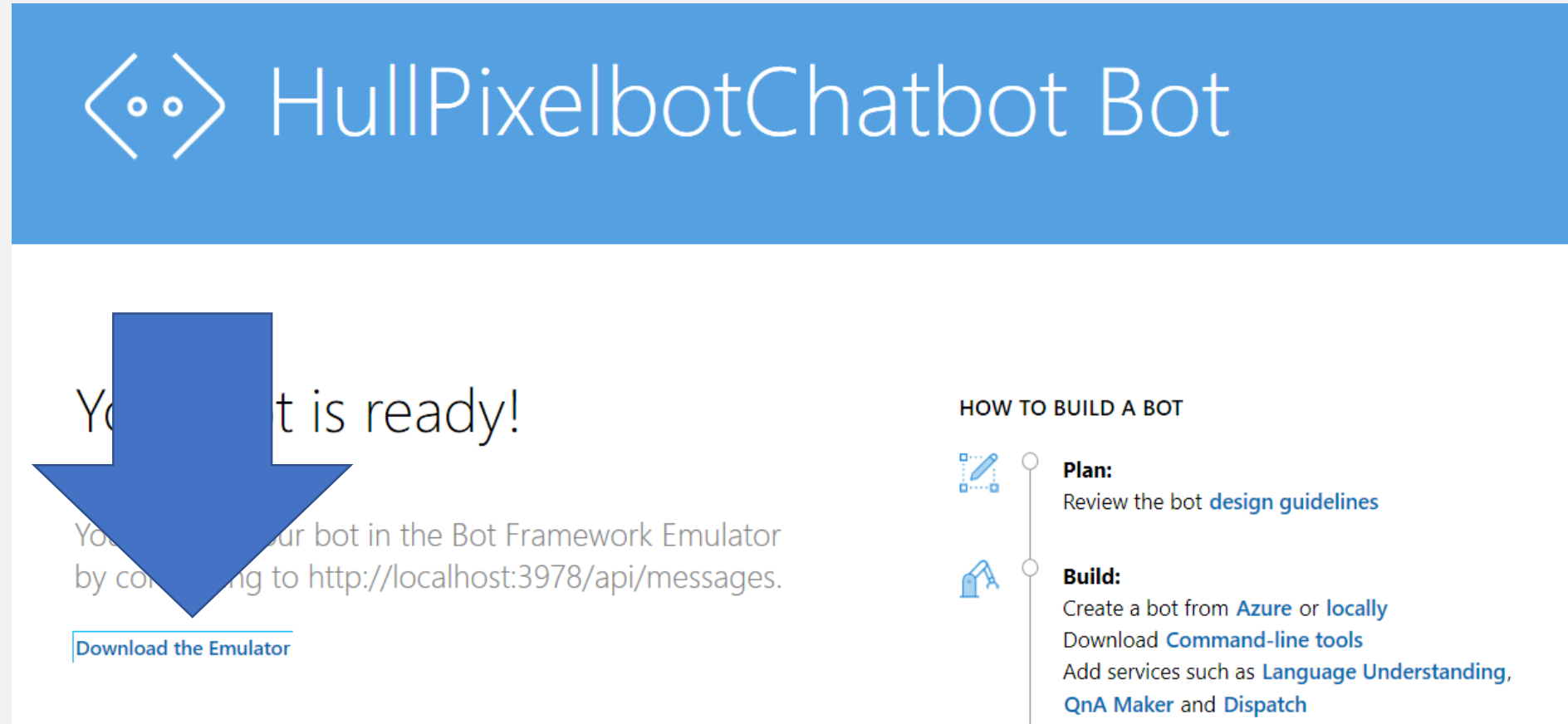



Build:

Create a bot from [Azure](#) or [locally](#)
Download [Command-line tools](#)
Add services such as [Language Understanding](#),
[QnA Maker](#) and [Dispatch](#)

Use the emulator for this

Talking to the chatbot





 HullPixelbotChatbot Bot

Your bot is ready!

You can run your bot in the Bot Framework Emulator by connecting to <http://localhost:3978/api/messages>.

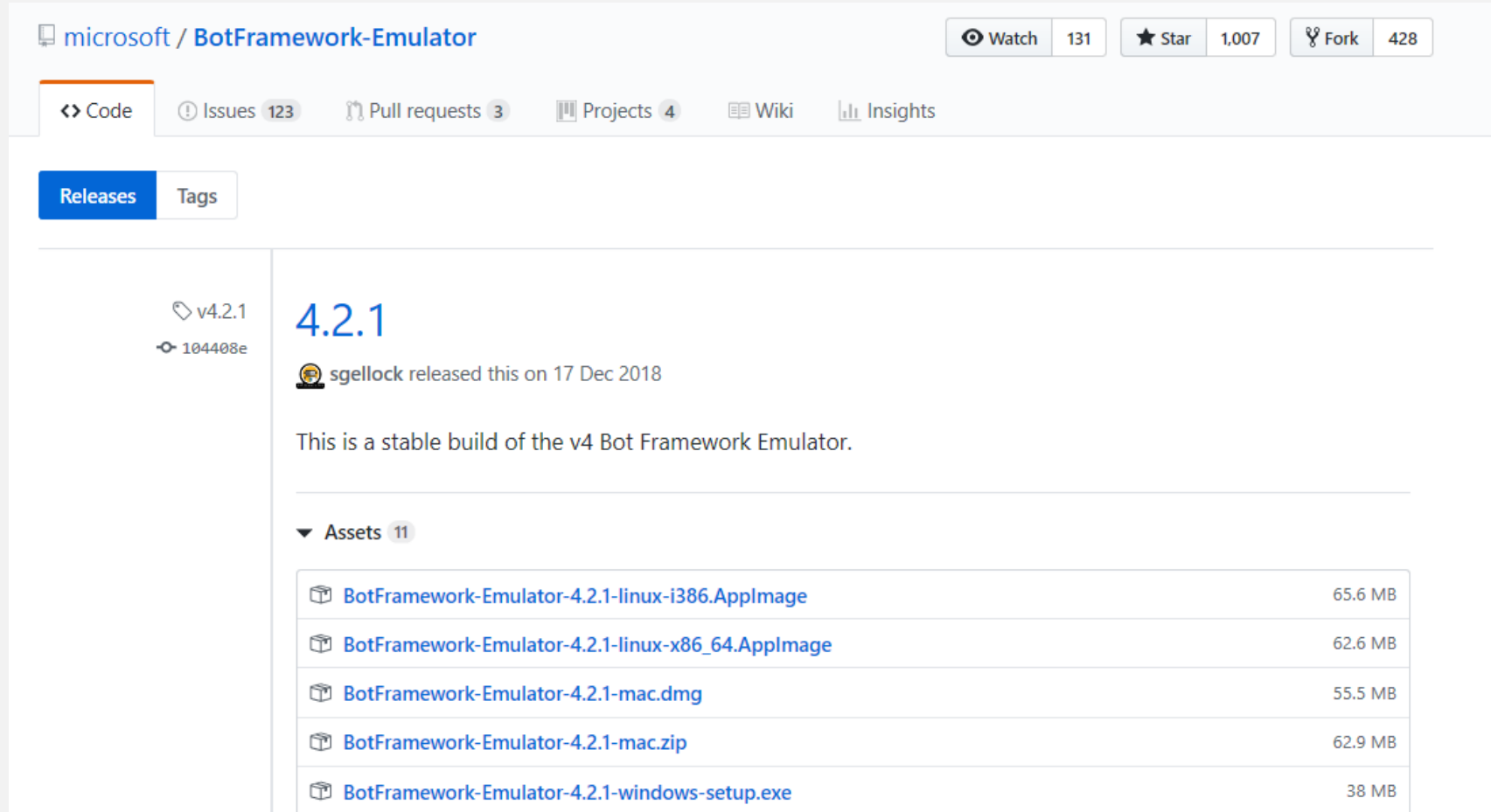
[Download the Emulator](#)

HOW TO BUILD A BOT

-  **Plan:**
Review the bot [design guidelines](#)
-  **Build:**
Create a bot from [Azure](#) or [locally](#)
Download [Command-line tools](#)
Add services such as [Language Understanding](#), [QnA Maker](#) and [Dispatch](#)

The emulator can send and receive messages to the bot directly

Getting the emulator



The screenshot shows the GitHub repository page for `microsoft / BotFramework-Emulator`. The repository has 131 watchers, 1,007 stars, and 428 forks. The navigation bar includes links for Code, Issues (123), Pull requests (3), Projects (4), Wiki, and Insights. The 'Releases' tab is selected, showing the v4.2.1 release. The release was made by `sgellock` on 17 Dec 2018. The description states: 'This is a stable build of the v4 Bot Framework Emulator.' Below the description, there are 11 assets listed in a table.

Asset Name	Size
BotFramework-Emulator-4.2.1-linux-i386.AppImage	65.6 MB
BotFramework-Emulator-4.2.1-linux-x86_64.AppImage	62.6 MB
BotFramework-Emulator-4.2.1-mac.dmg	55.5 MB
BotFramework-Emulator-4.2.1-mac.zip	62.9 MB
BotFramework-Emulator-4.2.1-windows-setup.exe	38 MB

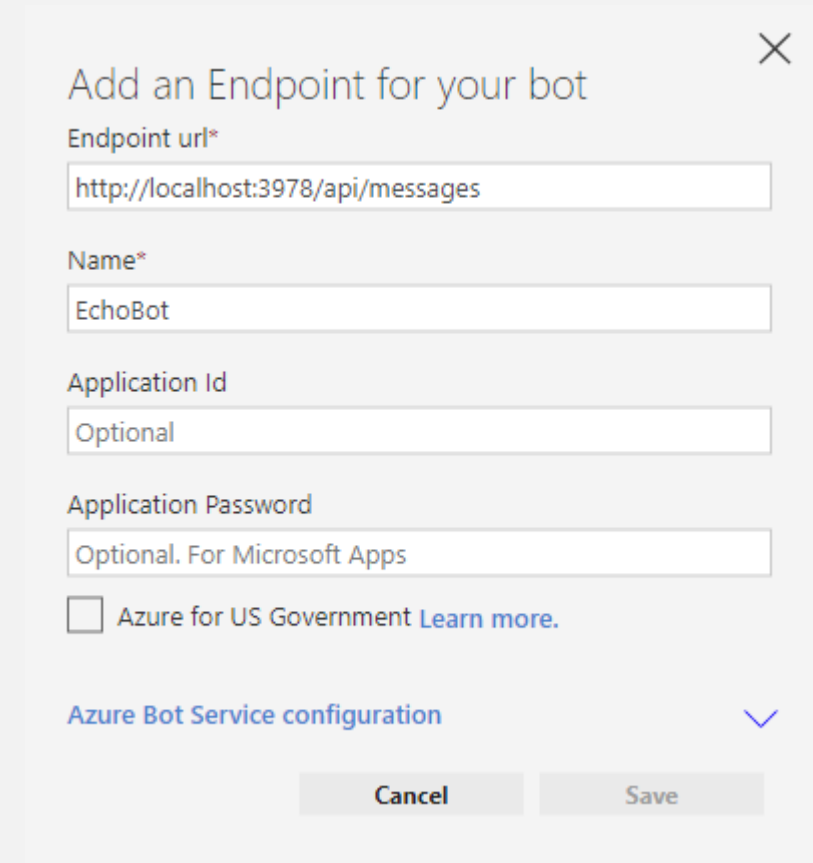
The emulator lives on GitHub

Configuring the emulator

For testing we connect the emulator to the locally hosted bot

This means that we can debug the code behind the bot

We can also connect to a bot that is hosted remotely so that we can see the messages exchanged by the devices

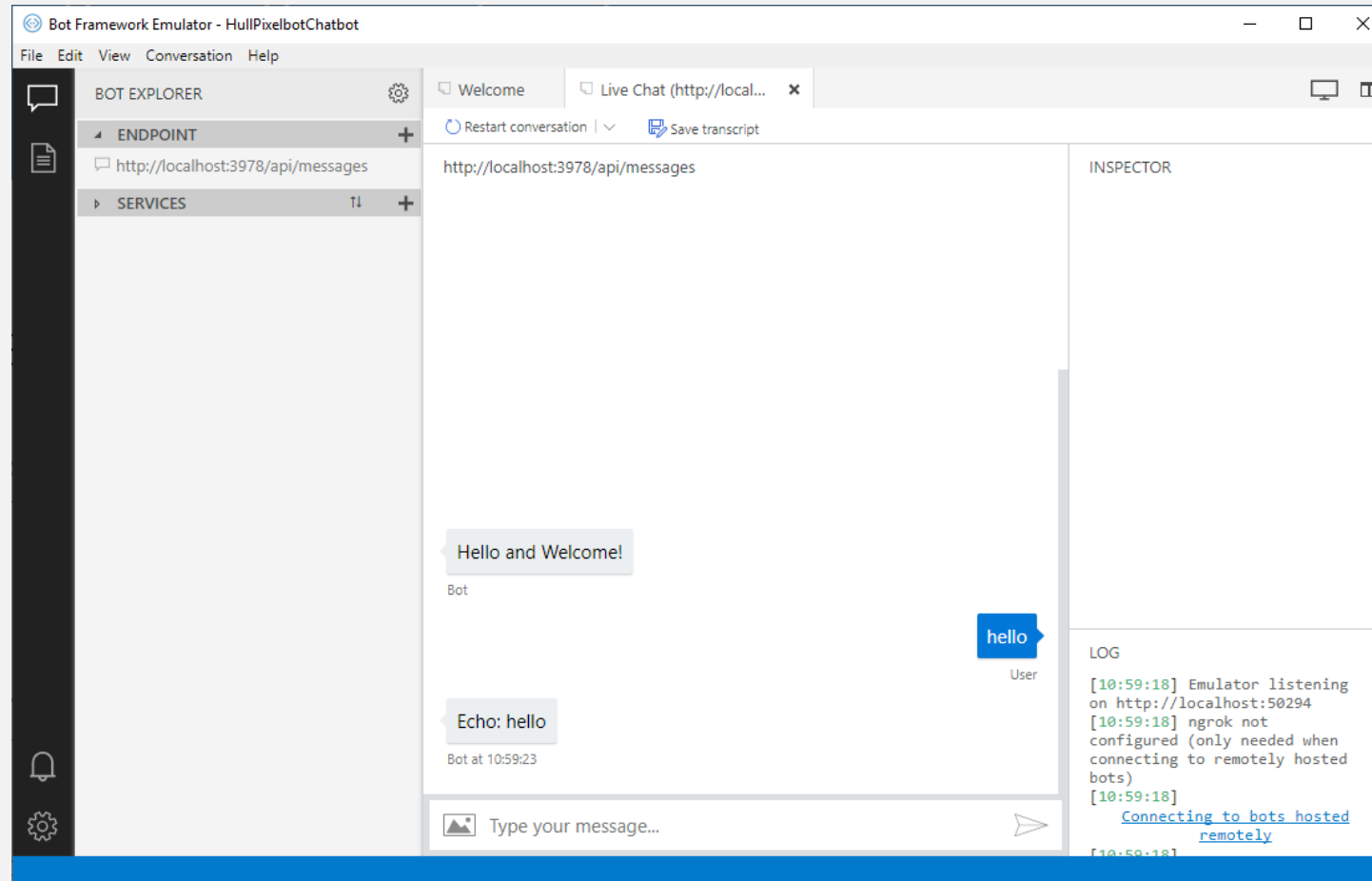


The screenshot shows a dialog box titled "Add an Endpoint for your bot" with a close button (X) in the top right corner. The dialog contains several input fields and a checkbox:

- Endpoint url***: A text input field containing the URL "http://localhost:3978/api/messages".
- Name***: A text input field containing the name "EchoBot".
- Application Id**: A text input field containing the value "Optional".
- Application Password**: A text input field containing the value "Optional. For Microsoft Apps".
- Azure for US Government** [Learn more.](#)

At the bottom of the dialog, there is a section labeled "Azure Bot Service configuration" with a downward-pointing chevron icon. Below this section are two buttons: "Cancel" and "Save".

Talking to the Echo bot



The echo bot just echoes back whatever was sent

How the echo bot works

```
public class EchoBot : ActivityHandler
{
    protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
                                                         Cancellation token cancellationToken)
    {
        await turnContext.SendActivityAsync(
            MessageFactory.Text($"Echo: {turnContext.Activity.Text}"), cancellationToken);
    }
}
```

The EchoBot class contains methods that are called when messages arrive for a bot

Your code can manage the incoming messages at this point

How the echo bot works

```
public class EchoBot : ActivityHandler
{
    protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
                                                         Cancellation token cancellationToken)
    {
        await turnContext.SendActivityAsync(
            MessageFactory.Text($"Echo: {turnContext.Activity.Text}"), cancellationToken);
    }
}
```

This is the method that sends the message back to the client

The message is just a string of text

Demo

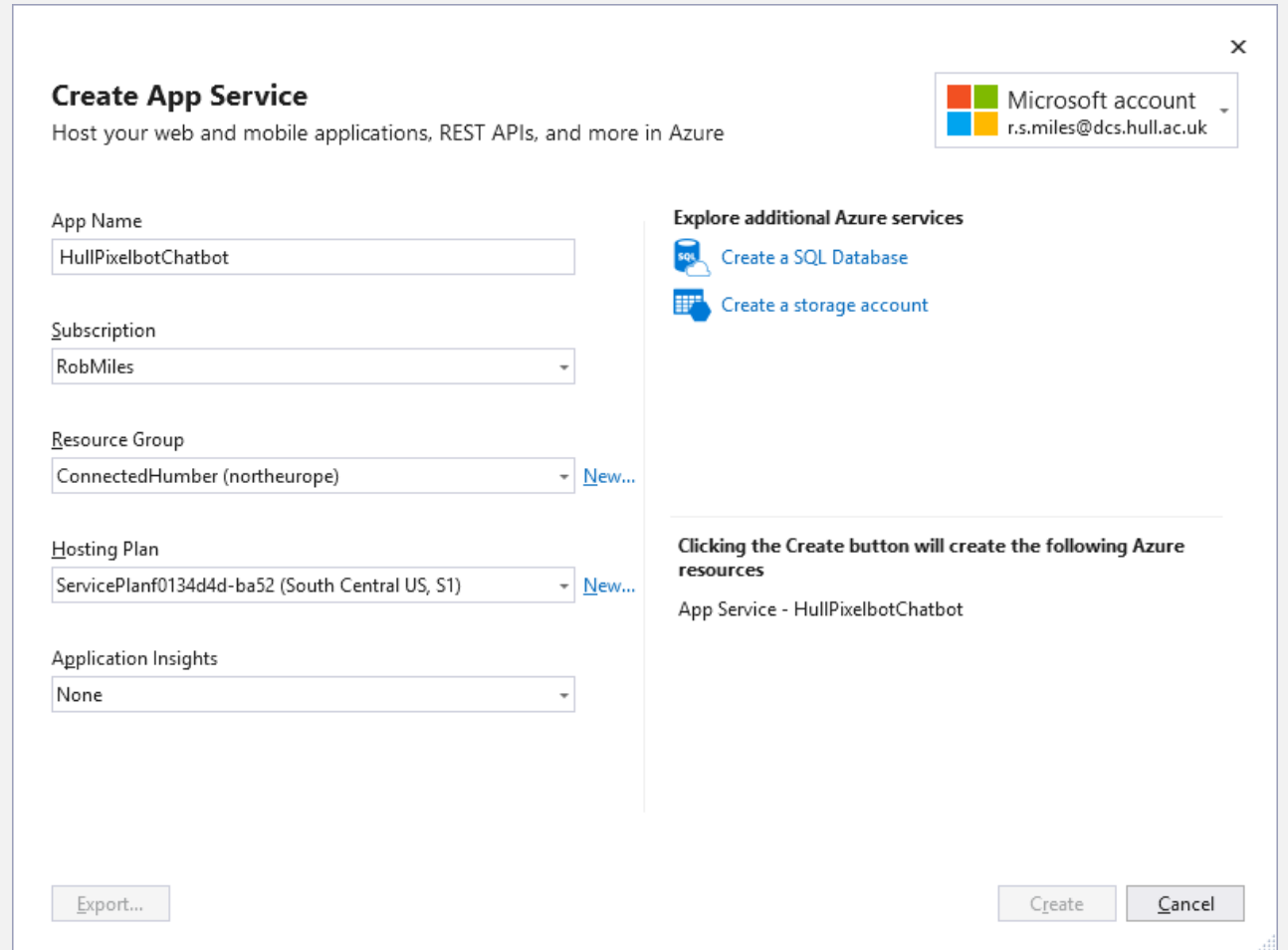
Build an echo bot and talk to it

Deploying the bot service to the cloud

Once you have made your bot service and tested it you can deploy it to the cloud

This just provides the service behind the endpoint

We can set up a Channel registration service to provide the connection to users




The screenshot shows the 'Create App Service' dialog box in the Azure portal. The dialog is titled 'Create App Service' and has a subtitle 'Host your web and mobile applications, REST APIs, and more in Azure'. In the top right corner, there is a Microsoft account dropdown menu showing 'Microsoft account' and the email 'r.s.miles@dcs.hull.ac.uk'. The main form contains several fields: 'App Name' with the value 'HullPixelbotChatbot', 'Subscription' with 'RobMiles', 'Resource Group' with 'ConnectedHumber (northeurope)', 'Hosting Plan' with 'ServicePlanf0134d4d-ba52 (South Central US, S1)', and 'Application Insights' with 'None'. To the right of the form, there is a section titled 'Explore additional Azure services' with two links: 'Create a SQL Database' and 'Create a storage account'. Below this, a section titled 'Clicking the Create button will create the following Azure resources' lists 'App Service - HullPixelbotChatbot'. At the bottom of the dialog, there are three buttons: 'Export...', 'Create', and 'Cancel'.

Bot registration

[Home](#) > [Bot Services](#) > [Bot Service](#) > Bot Channels Registration

Bot Channels Registration

Microsoft



Bot Channels Registration

Microsoft

[Create](#) [Save for later](#)

Your own Bot hosted where you want, registered with the Azure Bot Service. Build, connect, and manage Bots to interact with your users wherever they are - from your app or website to Cortana, Skype, Messenger and many other services.

A separate service manages the channels for a bot

The channels and the service can be on completely different servers

Bot registration

The Channel Registration service is managed as any other Azure service via the Azure portal

The screenshot shows the 'HullPixelbot - Settings' page in the Azure portal. The page is divided into a left-hand navigation pane and a main content area. The navigation pane includes sections for 'Bot management' (Test in Web Chat, Analytics, Channels, Settings, Speech priming, Bot Service pricing) and 'Support + troubleshooting' (New support request). The 'Settings' option is currently selected. The main content area has a breadcrumb 'Home > HullPixelbot - Settings' and a title 'HullPixelbot - Settings' with the subtitle 'Bot Channels Registration'. Below the title are 'Save' and 'Discard' buttons. The settings are organized into sections: 'Icon' (with a custom icon upload button and a 30K max, png only limit), 'Display name' (set to 'HullPixelbot'), 'Bot handle' (set to 'HullPixelbot'), 'Description' (set to 'Find out about HullPixelbot and how to control one.'), 'Configuration' (with a 'Messaging endpoint' set to 'https://hullpixelbotchatbot.azurewebsites.net/api/messages' and an unchecked 'Enable Streaming Endpoint' checkbox), and 'Microsoft App ID' (set to '0ab4ba7a-b2f7-4364-99eb-bb05c9b48594' with a 'Manage' link).

Home > HullPixelbot - Settings


HullPixelbot - Settings

Bot Channels Registration

Search (Ctrl+ /)

Save Discard

Icon

Upload custom icon  30K max, png only

* Display name ⓘ

HullPixelbot

Bot handle ⓘ

HullPixelbot

Description ⓘ

Find out about HullPixelbot and how to control one.

Configuration

Messaging endpoint

https://hullpixelbotchatbot.azurewebsites.net/api/messages

Enable Streaming Endpoint

* Microsoft App ID [Manage](#) ⓘ

0ab4ba7a-b2f7-4364-99eb-bb05c9b48594

- Overview
- Activity log
- Access control (IAM)
- Tags

Bot management

- Test in Web Chat
- Analytics
- Channels
- Settings
- Speech priming
- Bot Service pricing

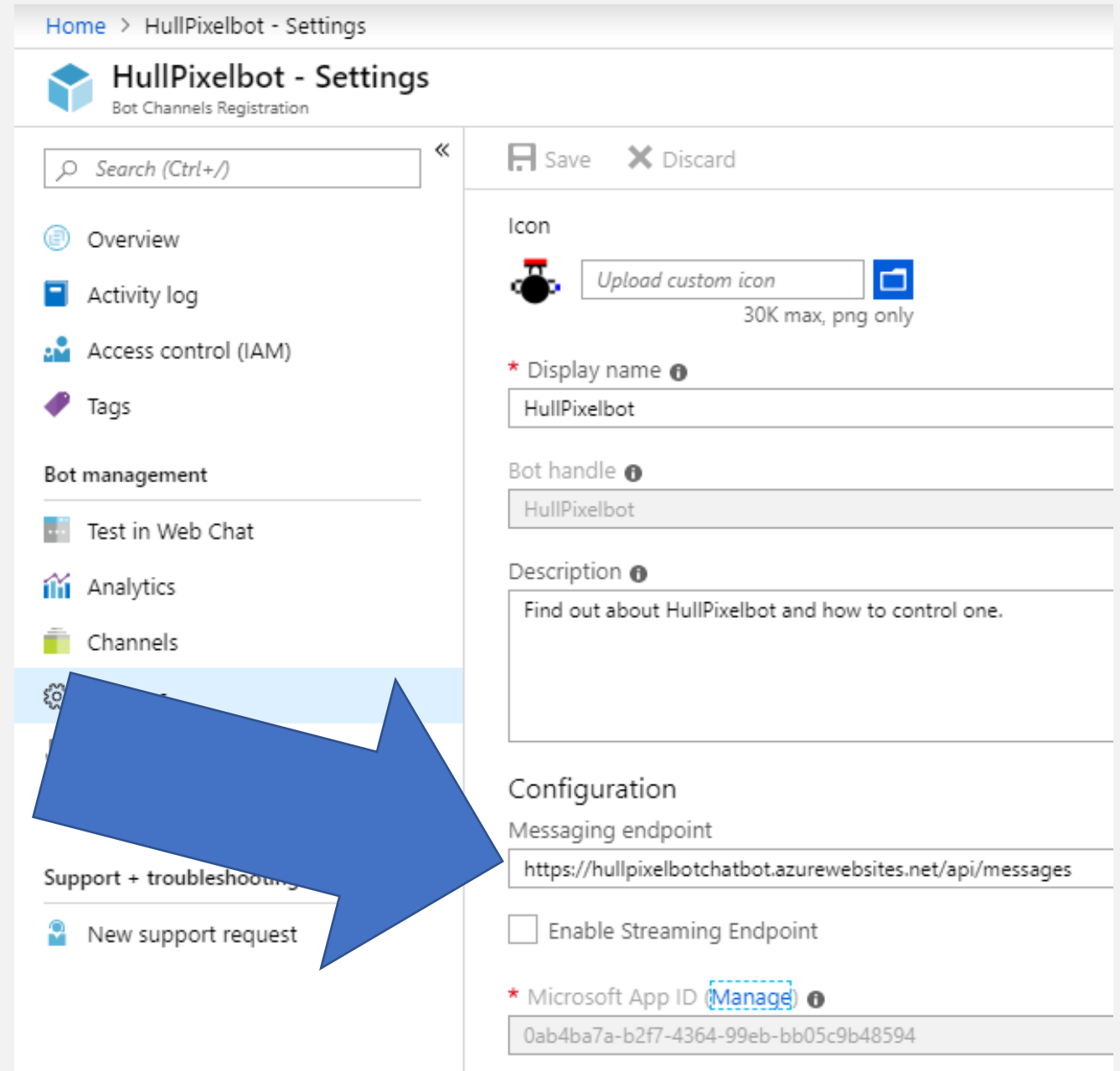
Support + troubleshooting

- New support request

Bot registration

We configure the channel registration with the messaging endpoint

This is the connection between the channels and the bot service

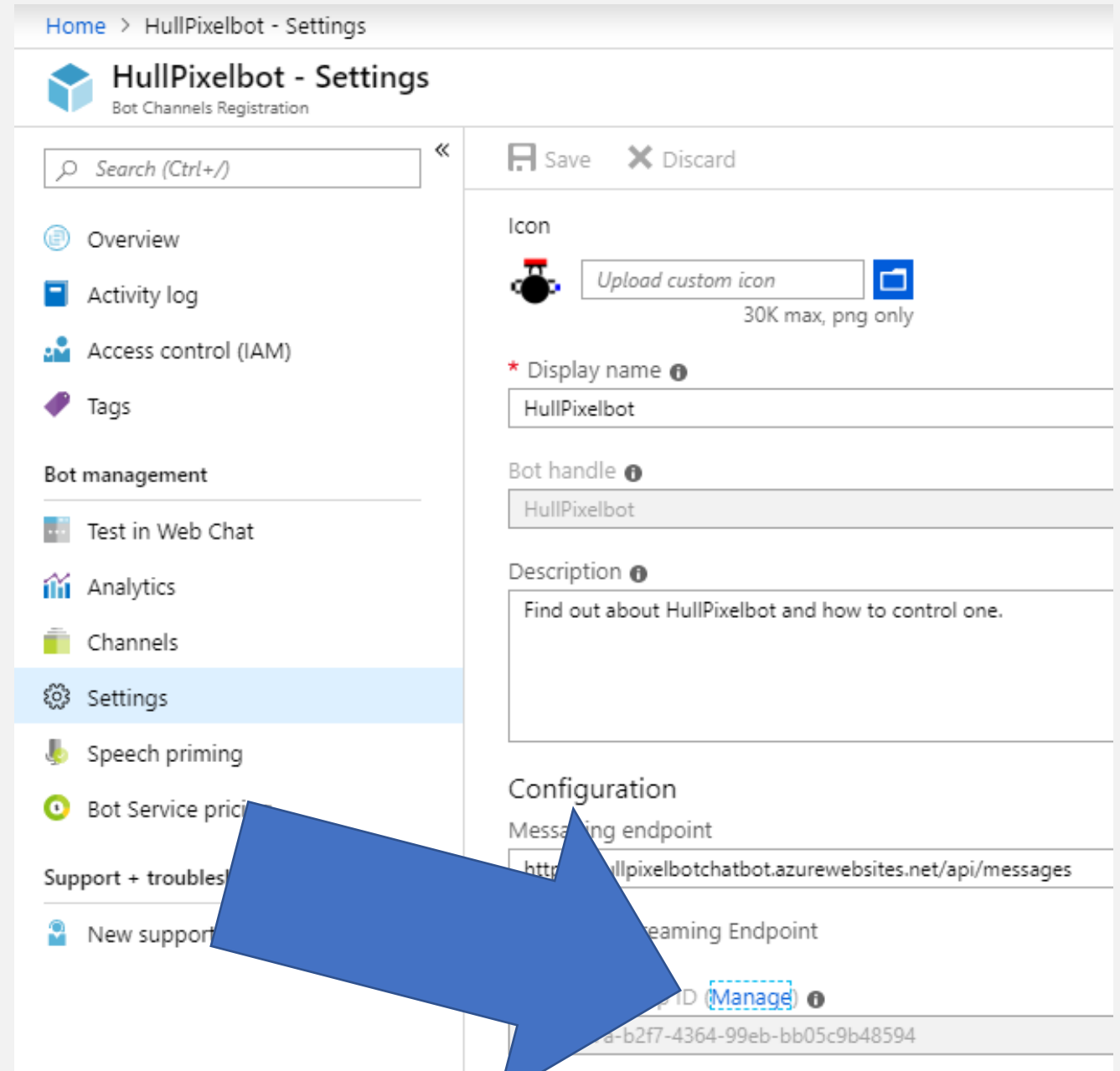


The screenshot shows the 'HullPixelbot - Settings' page in the Microsoft Bot Framework console. The page is titled 'HullPixelbot - Settings' and 'Bot Channels Registration'. On the left, there is a navigation menu with options: Overview, Activity log, Access control (IAM), Tags, Bot management, Test in Web Chat, Analytics, Channels, and Support + troubleshooting. The 'Channels' option is highlighted with a blue bar. A large blue arrow points from the 'Channels' option to the 'Configuration' section on the right. The 'Configuration' section includes a 'Messaging endpoint' field with the value 'https://hullpixelbotchatbot.azurewebsites.net/api/messages' and an 'Enable Streaming Endpoint' checkbox. The 'Microsoft App ID' field contains the value '0ab4ba7a-b2f7-4364-99eb-bb05c9b48594' and has a 'Manage' link next to it. Other fields include 'Icon', 'Display name' (HullPixelbot), and 'Bot handle' (HullPixelbot).

Authentication

A client must authenticate to the bot when it connects

We need to set the app id and key in the client service



The screenshot shows the 'HullPixelbot - Settings' page in the Microsoft Bot Framework console. The left sidebar contains navigation options: Overview, Activity log, Access control (IAM), Tags, Bot management, Test in Web Chat, Analytics, Channels, Settings (highlighted), Speech priming, Bot Service pricing, Support + troubleshooting, and New support request. The main content area is divided into sections: 'Icon' with an upload button and a 30K max, png only limit; 'Display name' set to 'HullPixelbot'; 'Bot handle' set to 'HullPixelbot'; 'Description' with the text 'Find out about HullPixelbot and how to control one.'; 'Configuration' with a 'Messaging endpoint' field containing 'http://hullpixelbotchatbot.azurewebsites.net/api/messages'; and a 'Streaming Endpoint' field with a 'Manage' button. A large blue arrow points from the 'Manage' button towards the text on the left.

Getting the keys

I can create a password for inclusion in the application

Need to be careful with this, we only see the password once, when it is generated

HullPixelbot Registration

[Click here for help integrating your application with Microsoft.](#)

Properties

Name

HullPixelbot

Application Id

0ab4ba7a-b2f7-4364-99eb-bb05c9b48594

Application Secrets

Generate New Password

Generate New Key Pair

Upload Public Key

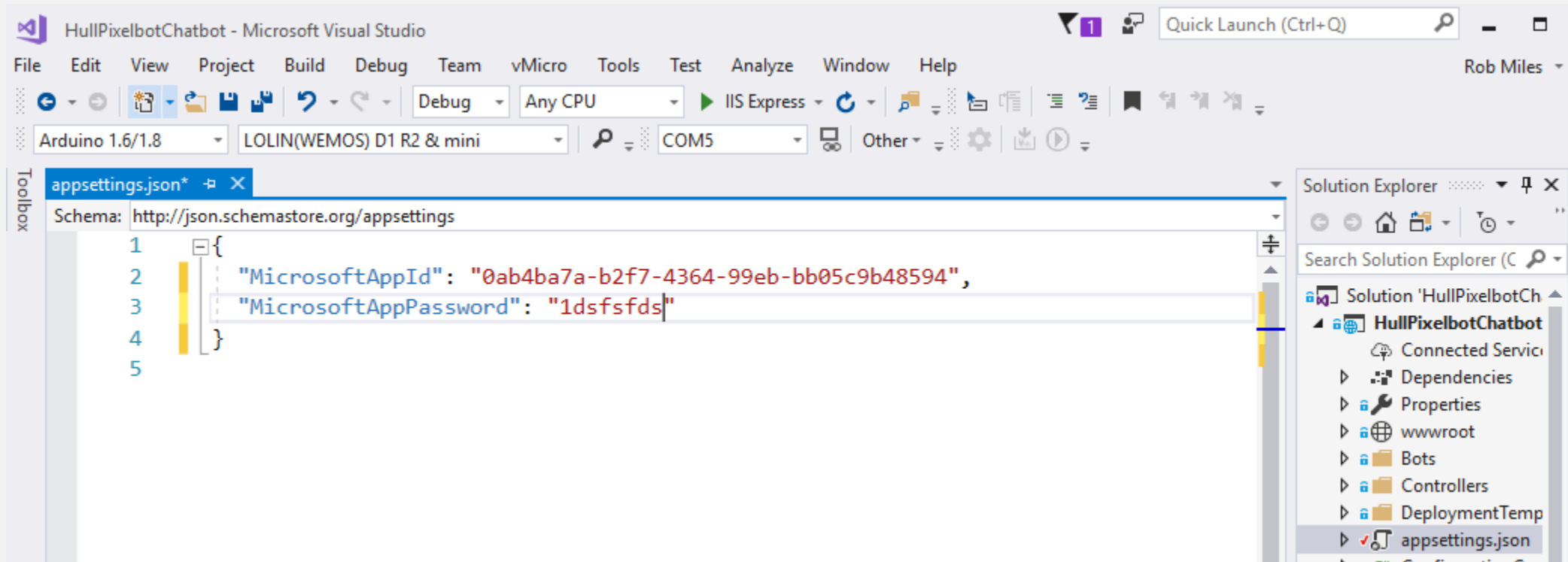
Type

Password/Public Key

Password

149*****

Setting the keys



The bot service contains the app id and password in a config file

These must be set before we host the service in the cloud

To recap:

We create our service that implements a bot

The service responds to text inputs and generates text outputs

We place the application in the cloud as a service

We create a channel connector that is mapped to the endpoint of our service

We use an application ID and key to allow the connector to authenticate the service

To recap:

We create our service that implements a bot

The service responds to text inputs and generates text outputs

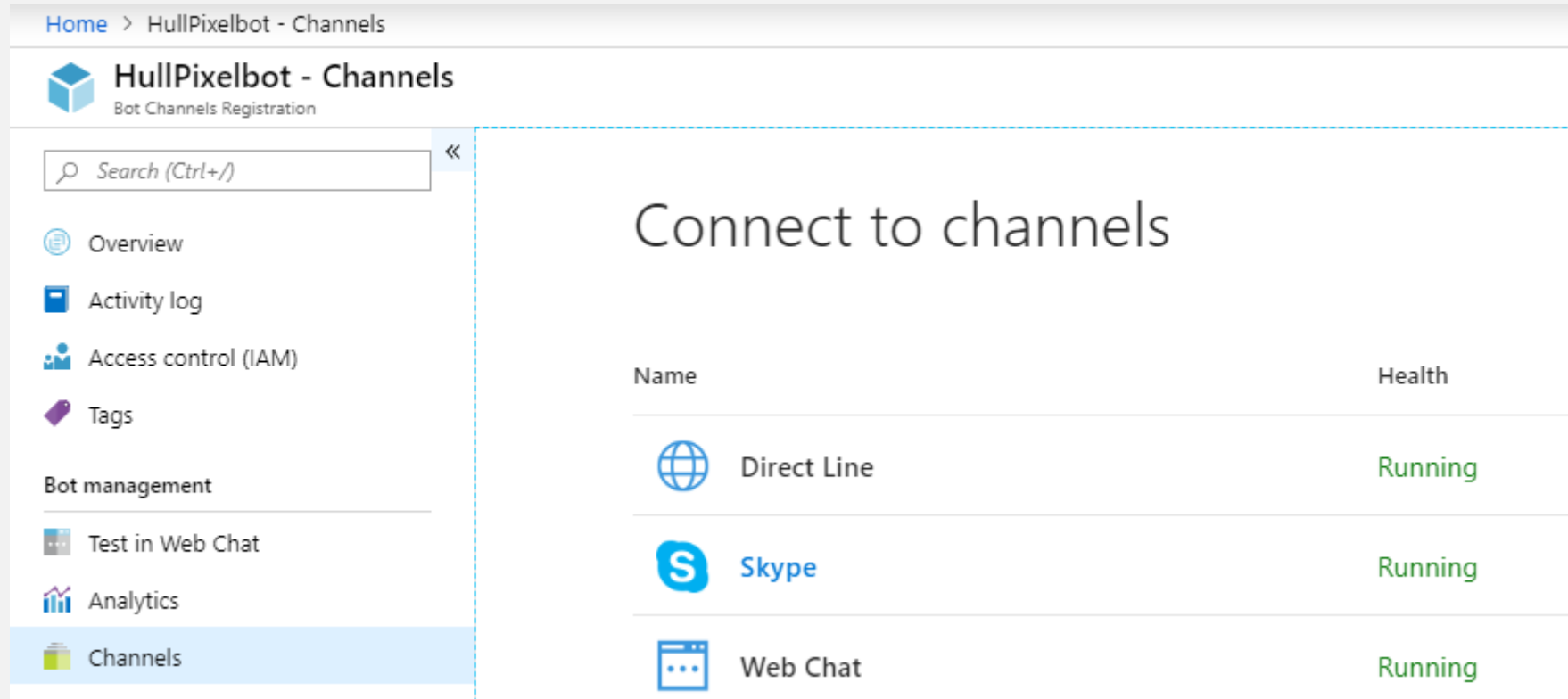
We place the application in the cloud as a service

We create a channel connector that is mapped to the endpoint of our service




We use an application ID and key to allow the connector to authenticate the service

Now we tell the connector which channels we want to use

Connecting channels

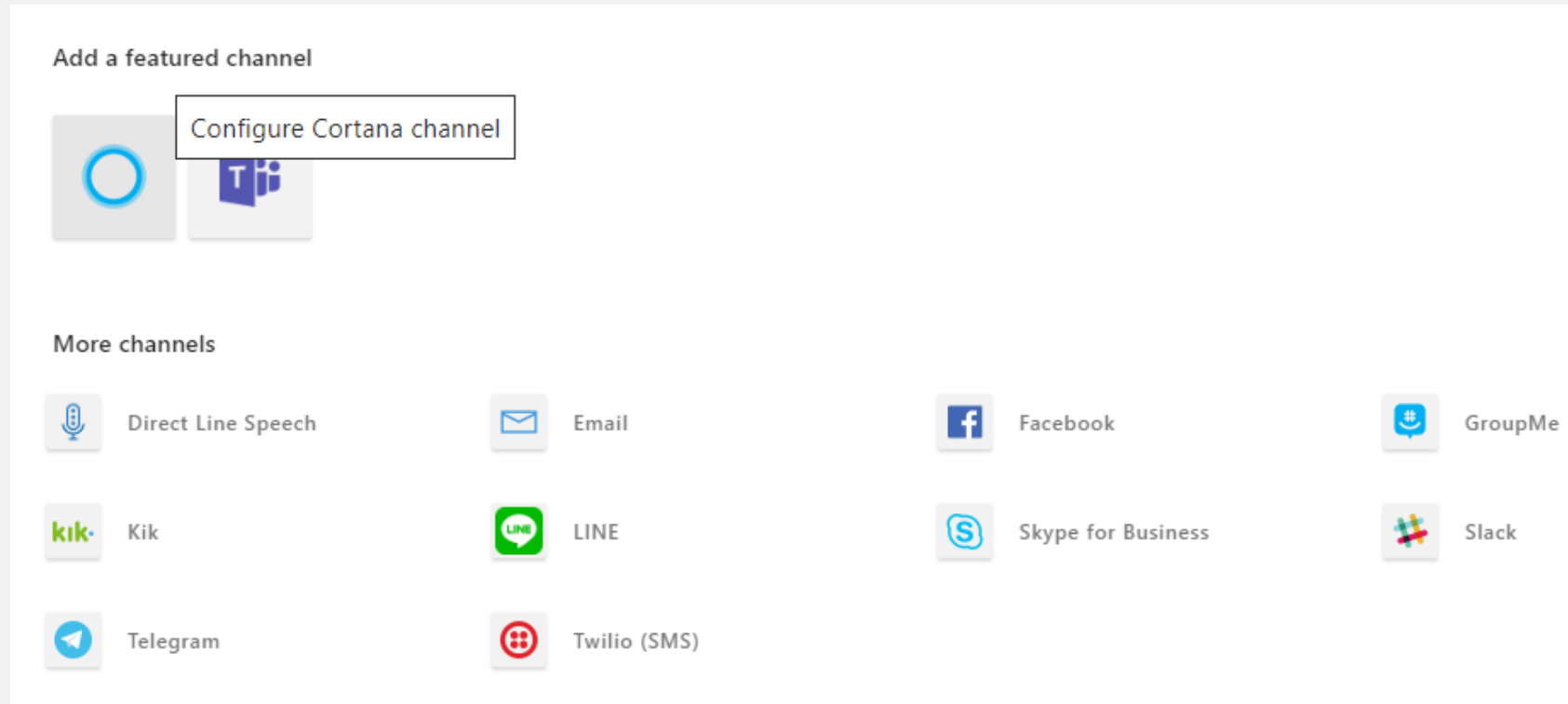


The screenshot shows the 'HullPixelbot - Channels' management interface. The breadcrumb path is 'Home > HullPixelbot - Channels'. The page title is 'HullPixelbot - Channels' with the subtitle 'Bot Channels Registration'. A search bar is present with the placeholder text 'Search (Ctrl+ /)'. The left sidebar contains navigation options: Overview, Activity log, Access control (IAM), Tags, Bot management, Test in Web Chat, Analytics, and Channels (which is highlighted). The main content area is titled 'Connect to channels' and displays a table of channel connections.

Name	Health
 Direct Line	Running
 Skype	Running
 Web Chat	Running

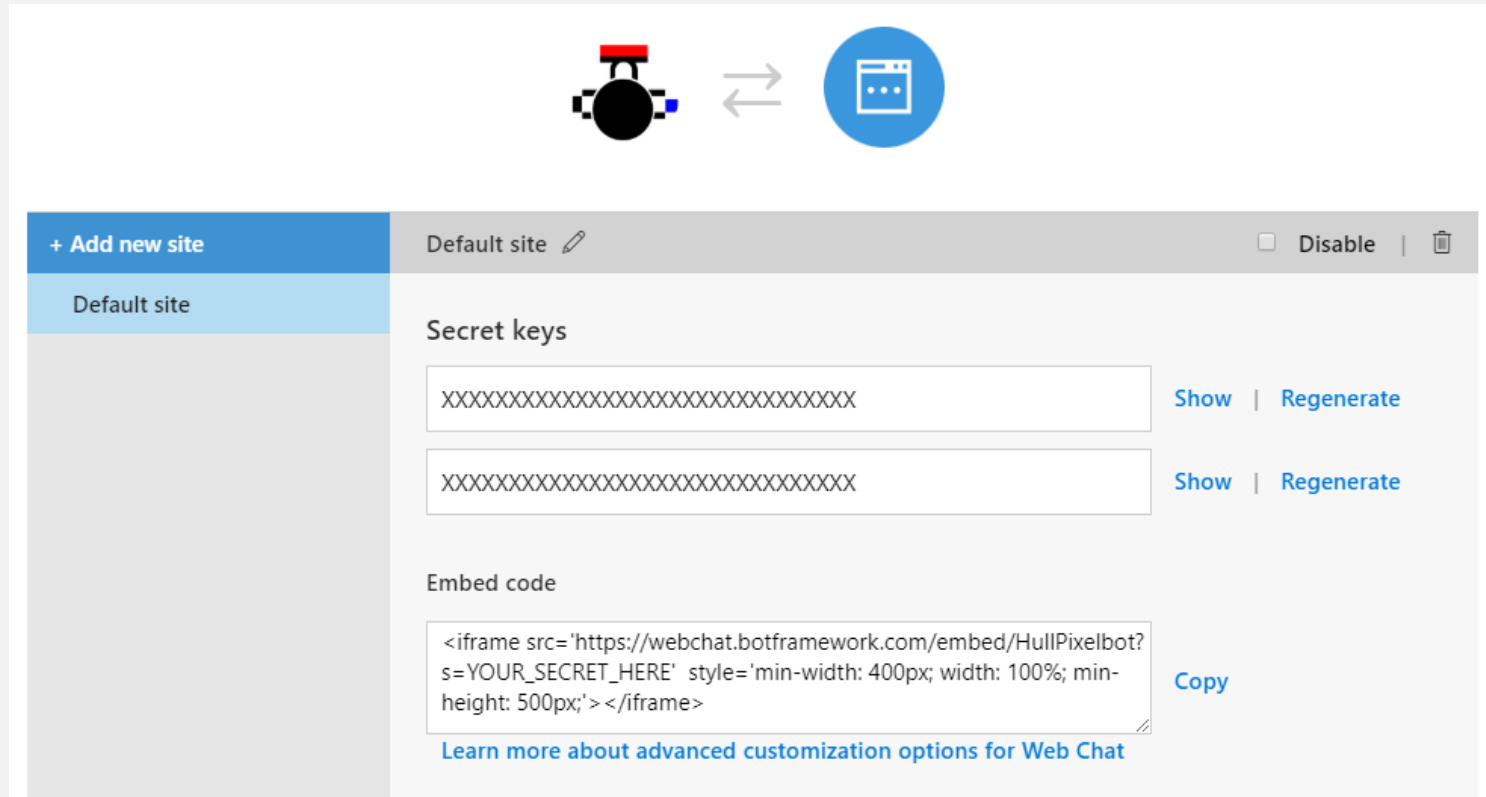
We can manage each channel connection individually

Adding channels



There are lots of channel connectors available including Cortana, Teams, Slack, email and Skype

Web embed codes



The screenshot displays a user interface for configuring a web chat. At the top, there are two icons: a black robot head and a blue circle with a white chat bubble icon, connected by a double-headed arrow. Below this is a sidebar with a blue button labeled '+ Add new site' and a selected item 'Default site'. The main content area is titled 'Default site' and includes a 'Disable' checkbox and a trash icon. Under the heading 'Secret keys', there are two text input fields, each containing a long string of 'X' characters. To the right of each field are 'Show' and 'Regenerate' links. Below the secret keys is the 'Embed code' section, which contains a text area with the following HTML code:

```
<iframe src='https://webchat.botframework.com/embed/HullPixelbot?s=YOUR_SECRET_HERE' style='min-width: 400px; width: 100%; min-height: 500px;'></iframe>
```

 A 'Copy' link is positioned to the right of the code. At the bottom of the embed code section is a blue link: [Learn more about advanced customization options for Web Chat](#).

We can generate lumps of html to drop into web pages to embed the chatbot into web pages

Skype

We can use skype to have a chat with the chatbot

You can ask this robot questions about how to program it

Skype

crazyrobmiles

People, groups & messages

Chats Calls Contacts Notifications

RECENT CHATS + Chat

- hullpixelbotqna-bot 14:39
Move over an arc
- HullPixelbot 08:44
Card
- P D Foster 15/12/2017
Call ended - 59m 53s
- Alfred Thompson
- ronald ramon
- John Scott Tynes
- home
- Joniel Rodriguez
- David Grey
- +4401482840186
- +4401482492219

hullpixelbotqna-bot

Move the robot

The move command can be used to start the robot moving, and to set a distance for the move. Move commands can also be configured to move in a particular time.

14:39

move in an arc

hullpixelbotqna-bot, 14:39

Move over an arc

This statement allows the robot to move in an arc. This can be a bit hard to understand. Experiment with arcs to understand how they work.

arc 0 angle 90

The first parameter is the radius of the arc, the second is the angle giving the angular distance around the arc.

The figure above shows how this works. If the radius is positive the curve will be about a point which is to the right of the robot, and the robot will turn clockwise as it moves. If the radius is negative the curve will be about a point that is to the left of the robot, and the robot will turn counterclockwise. The centre point of the arc is on a line drawn between the two wheels. If the angle is positive the robot will turn clockwise.

In the statement above the radius is 0, i.e. the centre of the arc is the point midway between the wheels. This means that the robot would rotate about its axis.

Type a message here

Cortana

Configure Cortana



Use this bot to power a Cortana skill

Adding the Cortana channel means that Cortana becomes aware of your bot's capabilities as a skill. This registration allows users to invoke and converse with your bot through Cortana's user interface. To learn more about Cortana skills, please take a look at the [Cortana Skills Kit developer docs](#).

By default, your skill will be available to Cortana in all devices associated with your Microsoft account. You can also publish to a group for testing and publish to world when you are ready to make your skill available world-wide.

You can also create a channel that links a chatbot to a Cortana skill

These can be published for the whole world to use

Adding richness to the interface

A bot sends and receives text

We can add richness to the interface by using adaptive cards

These are an open standard that we can use to display graphics and implement things like buttons for the user to press

How they work will depend on the channel used to deliver the interface

At the moment we can't use adaptive cards with the Skype channel which is rather sad 😞

Adaptive Cards

Adaptive cards are an exchange format for user interface elements

They are lumps of JSON that describes the user interface intent

Then the target framework can render them however they see fit

We can use them with our chatbot

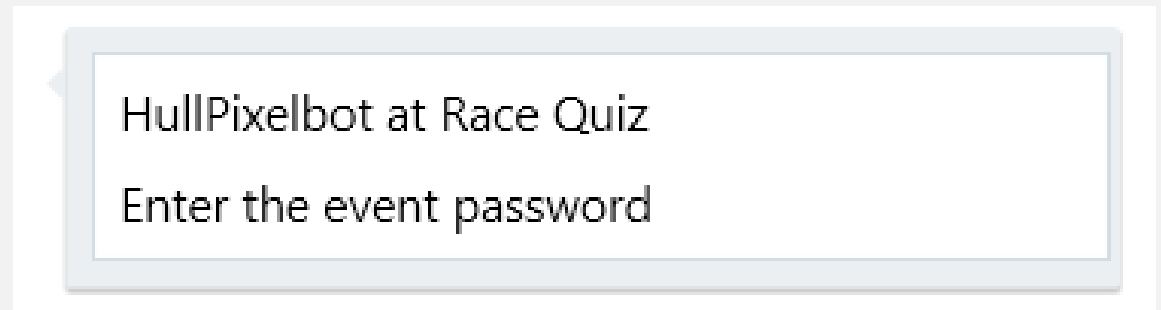
You can find out more about them here:

<https://adaptivecards.io/>

Filling in a card

```
AdaptiveCard result = new AdaptiveCard();

result.Body.Add(new AdaptiveTextBlock()
{
    Text = "HullPixelbot at " + eventName,
    Size = AdaptiveTextSize.Medium
});
result.Body.Add(new AdaptiveTextBlock()
{
    Text = "Enter the event password",
    Size = AdaptiveTextSize.Medium
});
```

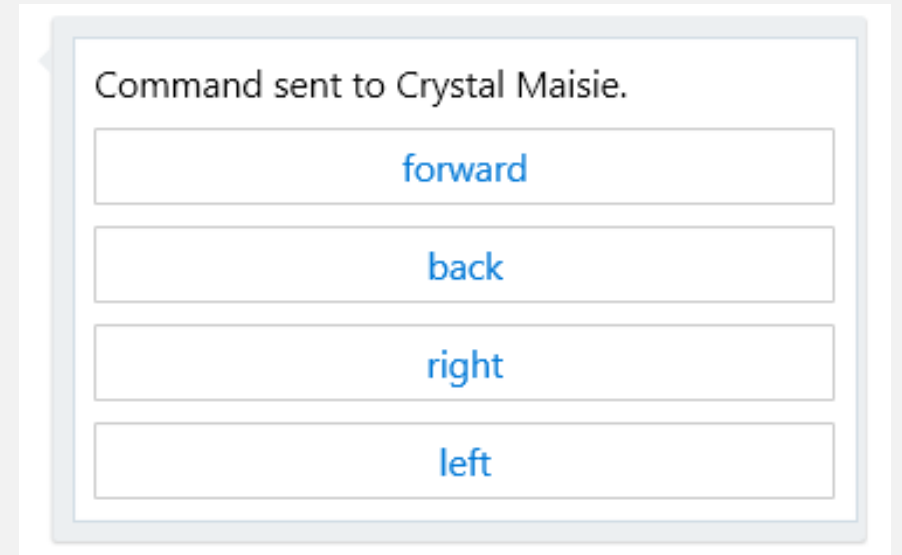


- An **AdaptiveCard** is a container object
- We can add elements and set the properties of the elements to inform the rendering process

Getting card options

```
AdaptiveCard result = new AdaptiveCard();

private void addOptionsToCard(AdaptiveCard card,
                             IEnumerable<string> options)
{
    foreach (string str in options)
    {
        card.Actions.Add(new AdaptiveSubmitAction()
            { Title = str, Data = str });
    }
}
```



- We can use **AdaptiveSubmitAction** items on the cards to allow the user to input by clicking the option
- The data item of the item is submitted as if it had been entered at the keyboard

Adding a Quiz

I added a quiz to the chatbot

This asks questions in between letting you control the robot

The players can control the robot but must answer questions every now and then.

Command sent to Crystal Maisie.

And now, a chance to show how clever you...

The name of Ed Sheeran's new album is:

Not a bit like Eggheads

Divide

Multiply

Plus equals

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    SetupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
        turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

This is the code that links the chatbots to my robot event

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    setupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
        turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

Set the event up (only does something the first time it is called)

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    SetupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

This is the text entered by the user

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    SetupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

This is the Id for this conversation (this is how I maintain state)

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    SetupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

The ActOnCommand method returns an AdaptiveCard

Linking a chatbot to a robot

```
protected override async Task OnMessageActivityAsync(ITurnContext<IMessageActivity> turnContext,
CancellationTokens cancellationTokens)
{
    SetupEvent();

    AdaptiveCard actionResult = await ActiveEvent.ActOnCommand(turnContext.Activity.Text,
turnContext.Activity.Conversation.Id);

    Attachment actionAttachment = new Attachment(
        contentType: "application/vnd.microsoft.card.adaptive", content: actionResult);

    Activity response = CreateResponse(turnContext.Activity, actionAttachment);

    await turnContext.SendActivityAsync(response, cancellationTokens);
}
```

This builds a response and sends it back to the user

Demo

Controlling Robots with Chatbots

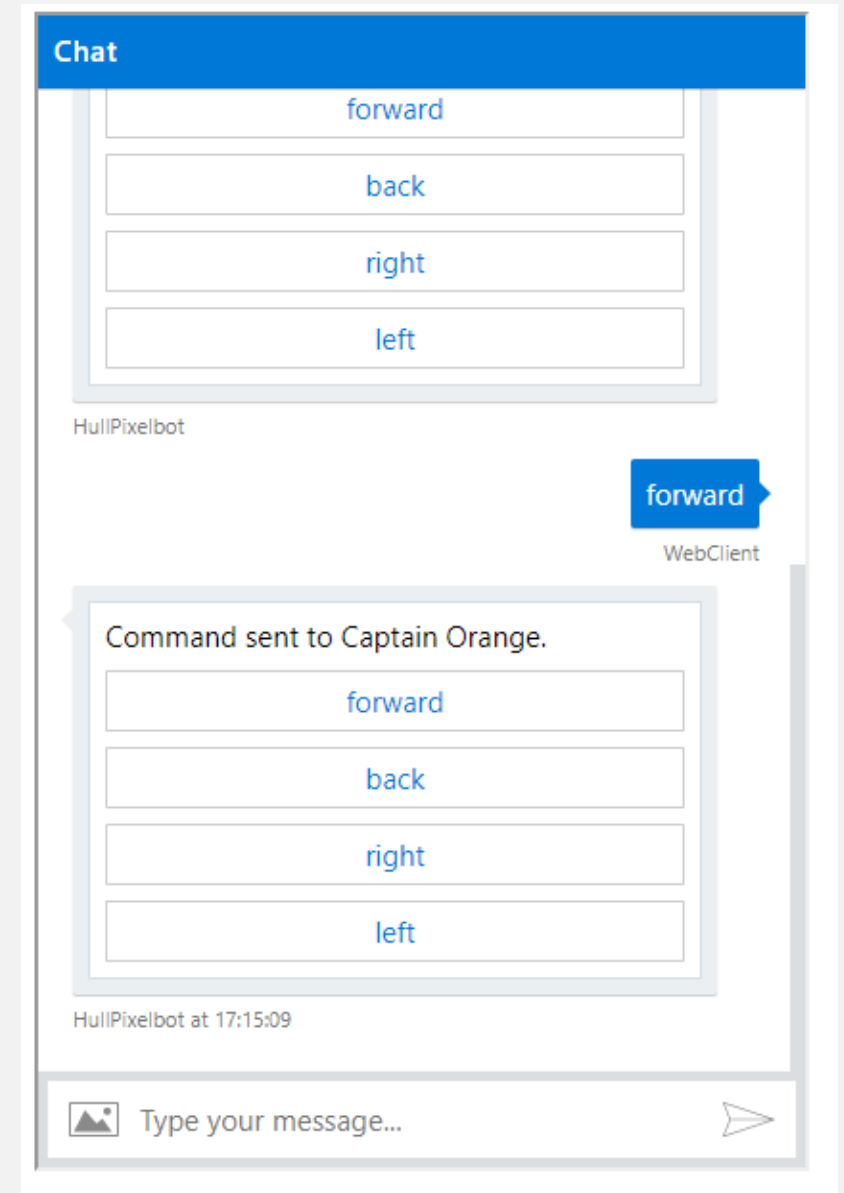
Connect to the robot quiz

Go to:

`hpb.host`

Select the quiz

The first three people will be allocated a robot and can control it via the chatbot



Adding Artificial Intelligence

Chatbots are all very well, but so far we've not seen any AI

There are two ways I'm going to use AI with our chatbots

The first one is to use a Microsoft QnA maker to create a bot that can handle simple queries

The second is to take a look at natural language understanding in bots

Microsoft QnA Generator

From FAQ to Bot in
minutes.

Build, train and publish a simple question and answer bot based on FAQ URLs, structured documents, product manuals or editorial content in minutes.

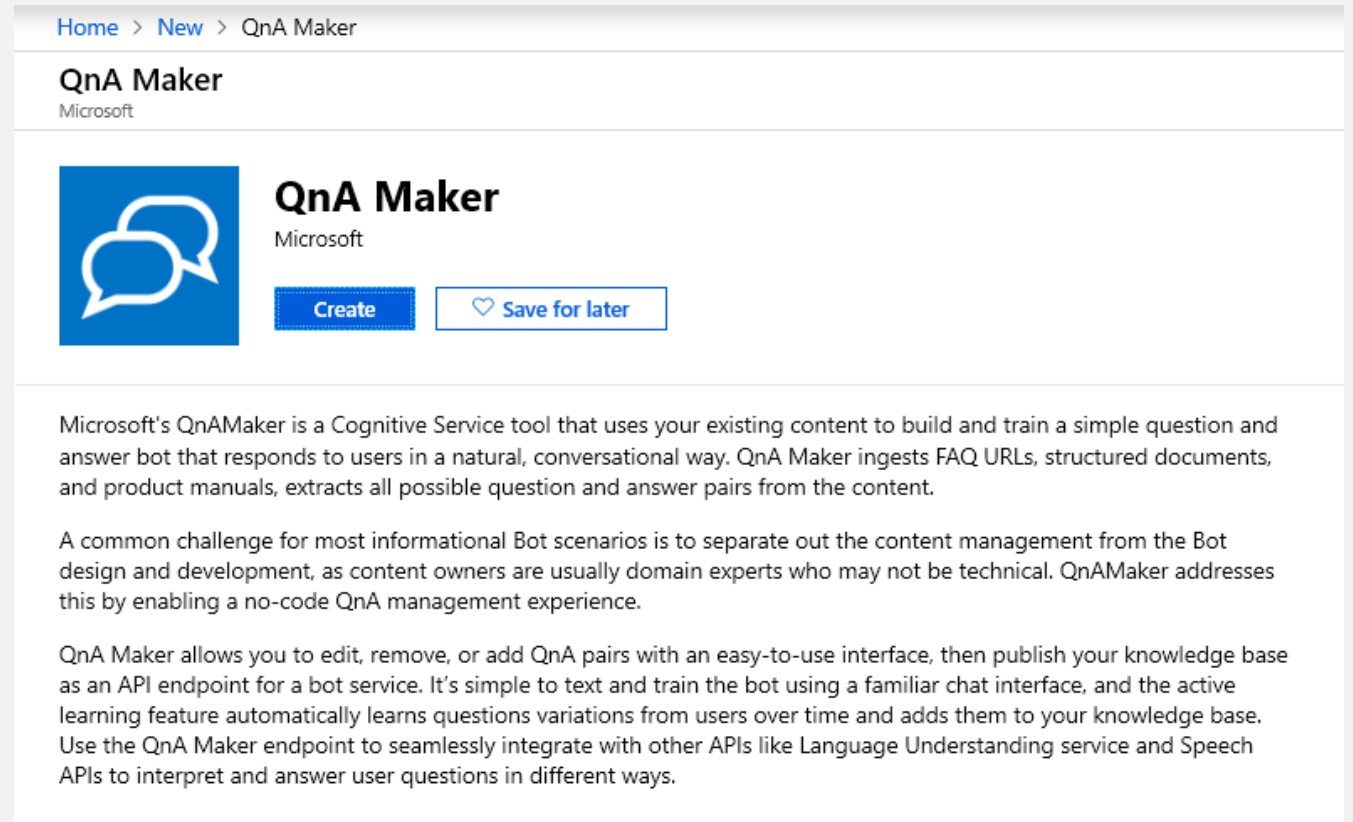
If you want to present your documents as an intelligent 'bot' you can do this using the Microsoft QnA generator

It will work on web sites, manuals, documents or brochures

Creating a QnA Maker

The QnA maker is a tool that runs in the cloud and processes the input to produce the data for the bot to use


You have to provision one of these on your Azure host



The screenshot shows the Microsoft QnA Maker product page. At the top, there is a breadcrumb navigation: Home > New > QnA Maker. Below this, the product name "QnA Maker" is displayed with the Microsoft logo underneath. To the left of the product name is a blue square icon containing two white speech bubbles. Below the icon and product name are two buttons: a blue "Create" button and a white "Save for later" button with a heart icon. The main content area contains three paragraphs of text describing the tool's capabilities and usage.

Home > New > QnA Maker

QnA Maker
Microsoft

 **QnA Maker**
Microsoft

[Create](#) [Save for later](#)

Microsoft's QnAMaker is a Cognitive Service tool that uses your existing content to build and train a simple question and answer bot that responds to users in a natural, conversational way. QnA Maker ingests FAQ URLs, structured documents, and product manuals, extracts all possible question and answer pairs from the content.

A common challenge for most informational Bot scenarios is to separate out the content management from the Bot design and development, as content owners are usually domain experts who may not be technical. QnAMaker addresses this by enabling a no-code QnA management experience.

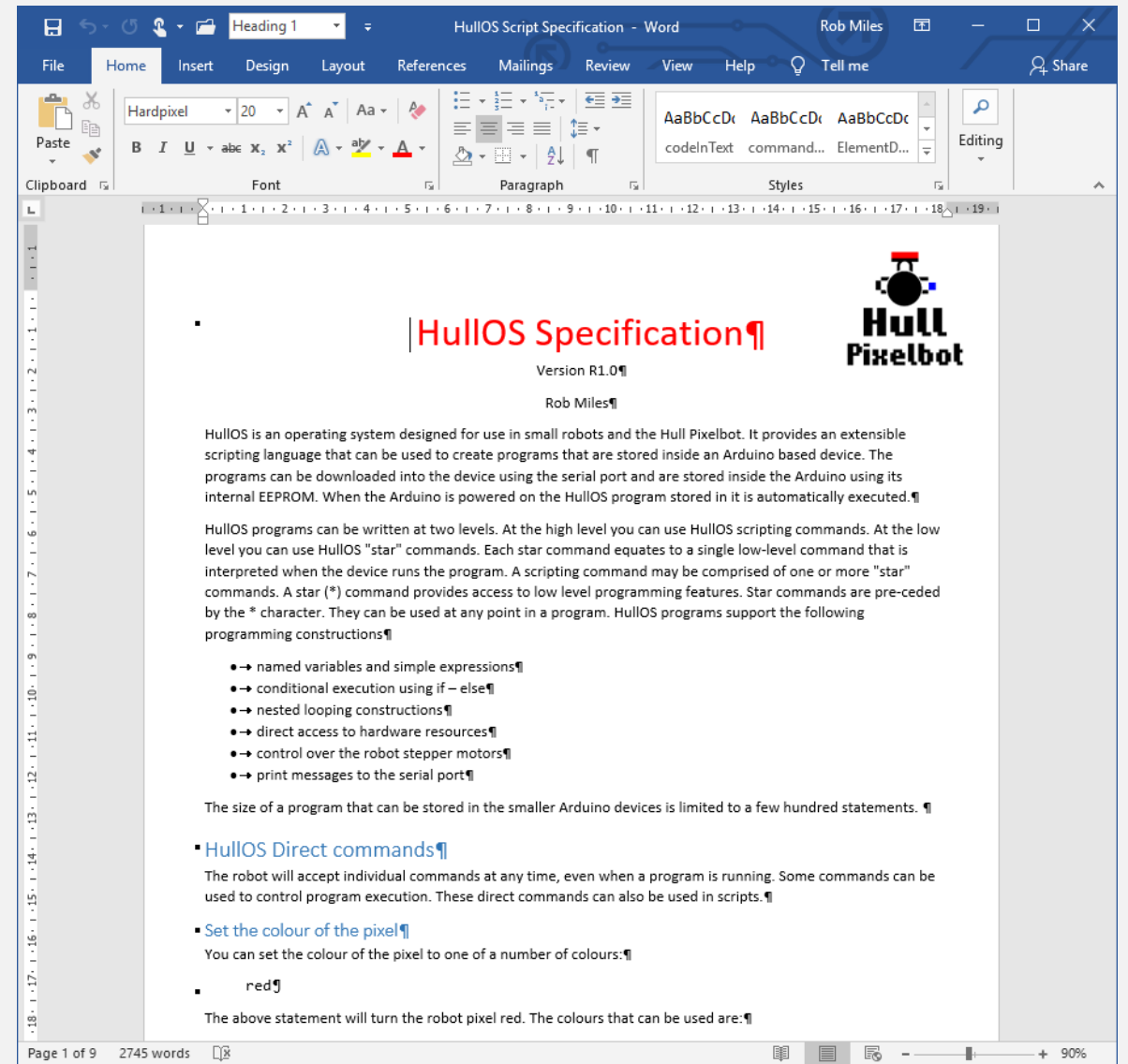
QnA Maker allows you to edit, remove, or add QnA pairs with an easy-to-use interface, then publish your knowledge base as an API endpoint for a bot service. It's simple to text and train the bot using a familiar chat interface, and the active learning feature automatically learns questions variations from users over time and adds them to your knowledge base. Use the QnA Maker endpoint to seamlessly integrate with other APIs like Language Understanding service and Speech APIs to interpret and answer user questions in different ways.

What I did

I took a word document that describes the scripting commands for my Hull Pixelbot robots

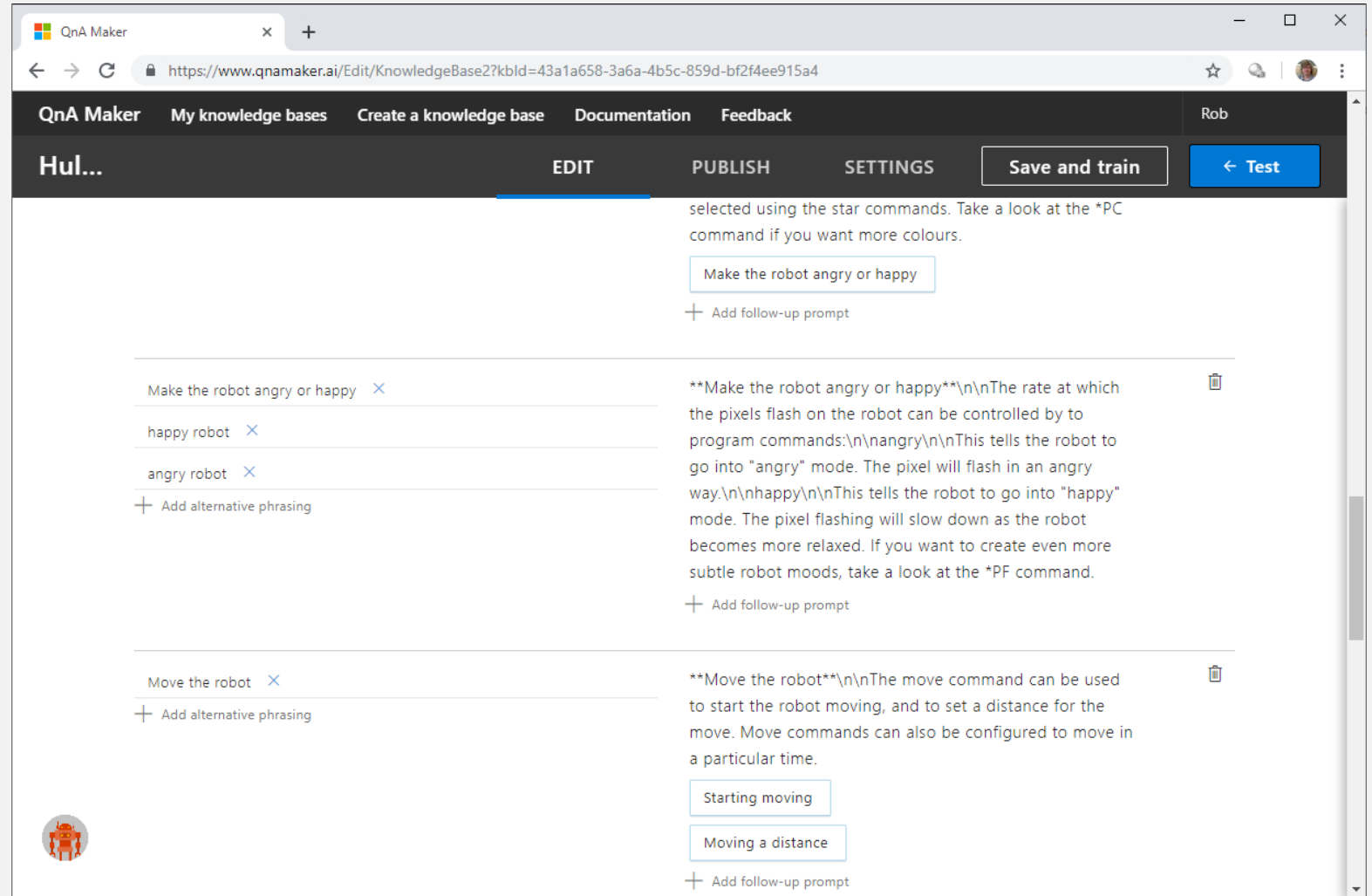
I fed this raw document into the QnA generator to see what it did

I used the QnA generator to automatically make a web chatbot that exposes the information in the document



QnA maker

We can use the web editor to modify the questions and answers that have been extracted from the data



The screenshot shows the QnA Maker web editor interface. The browser address bar displays the URL: <https://www.qnamaker.ai/Edit/KnowledgeBase2?kbId=43a1a658-3a6a-4b5c-859d-bf2f4ee915a4>. The navigation bar includes links for "QnA Maker", "My knowledge bases", "Create a knowledge base", "Documentation", "Feedback", and a user profile "Rob". The main interface is divided into sections: "Hul...", "EDIT", "PUBLISH", "SETTINGS", "Save and train", and "← Test".

The main content area displays a list of prompts and their corresponding answers. Each prompt is followed by a "Make the robot angry or happy" button and an "Add follow-up prompt" link. The answers are formatted with bold text and include instructions on how to control the robot's mood and movement.

Prompt	Answer
Make the robot angry or happy	selected using the star commands. Take a look at the *PC command if you want more colours.
happy robot	**Make the robot angry or happy**\n\nThe rate at which the pixels flash on the robot can be controlled by to program commands:\n\nangry\n\nThis tells the robot to go into "angry" mode. The pixel will flash in an angry way.\n\nhappy\n\nThis tells the robot to go into "happy" mode. The pixel flashing will slow down as the robot becomes more relaxed. If you want to create even more subtle robot moods, take a look at the *PF command.
angry robot	**Move the robot**\n\nThe move command can be used to start the robot moving, and to set a distance for the move. Move commands can also be configured to move in a particular time.

QnA Maker

We can also test the interactions with the data set

The screenshot displays the QnA Maker web application interface. The browser address bar shows the URL: <https://www.qnamaker.ai/Edit/KnowledgeBase2?kblId=43a1a658-3a6a-4b5c-859d-bf2f4ee915a4>. The navigation bar includes "QnA Maker", "My knowledge bases", "Create a knowledge base", "Documentation", and "Feedback". The user profile "Rob" is visible in the top right.

The main interface is divided into sections for editing knowledge base items. The "EDIT" tab is active, showing a list of items:

- Make the robot angry or happy** (with a close icon):
 - happy robot (with a close icon)
 - angry robot (with a close icon)
 - + Add alternative phrasing
- Move the robot** (with a close icon):
 - + Add alternative phrasing

Each item has a corresponding description and a "Add follow-up prompt" button. The description for "Make the robot angry or happy" includes instructions on using star commands and program commands like `\n\nangry` and `\n\nhappy`.

A chat window is overlaid on the right side of the interface. It shows a "Published KB" section with a "Start over" button. The chat history includes a blue message box with the text "hello" and a response from the bot: "Welcome. This is the HulloS scripting expert. I can tell you how to control your robot using the HulloS scripting commands." The chat is timestamped "HulloS Scripting (Test) at 4:08 PM". At the bottom, there is a text input field with the placeholder "Type your message here ..." and a checkbox for "Enable multi-turn" with a "Learn more" link.

Editing

I've added some extra questions and answers to make some responses make more sense



hullpixelbotqna-bot, 15:51

Print a single value

The print command will print a single value to the serial port.

```
print "hello"
```

The text to be printed is enclosed in double quotes, as shown above. If you print something else it will be appended to the line that is being printed.

A print statement can also print the value of an expression. You'll find out what these are later.

hello



hullpixelbotqna-bot, 15:53

Welcome. This is the HullIOS scripting expert. I can tell you how to control your robot using the HullIOS scripting commands.

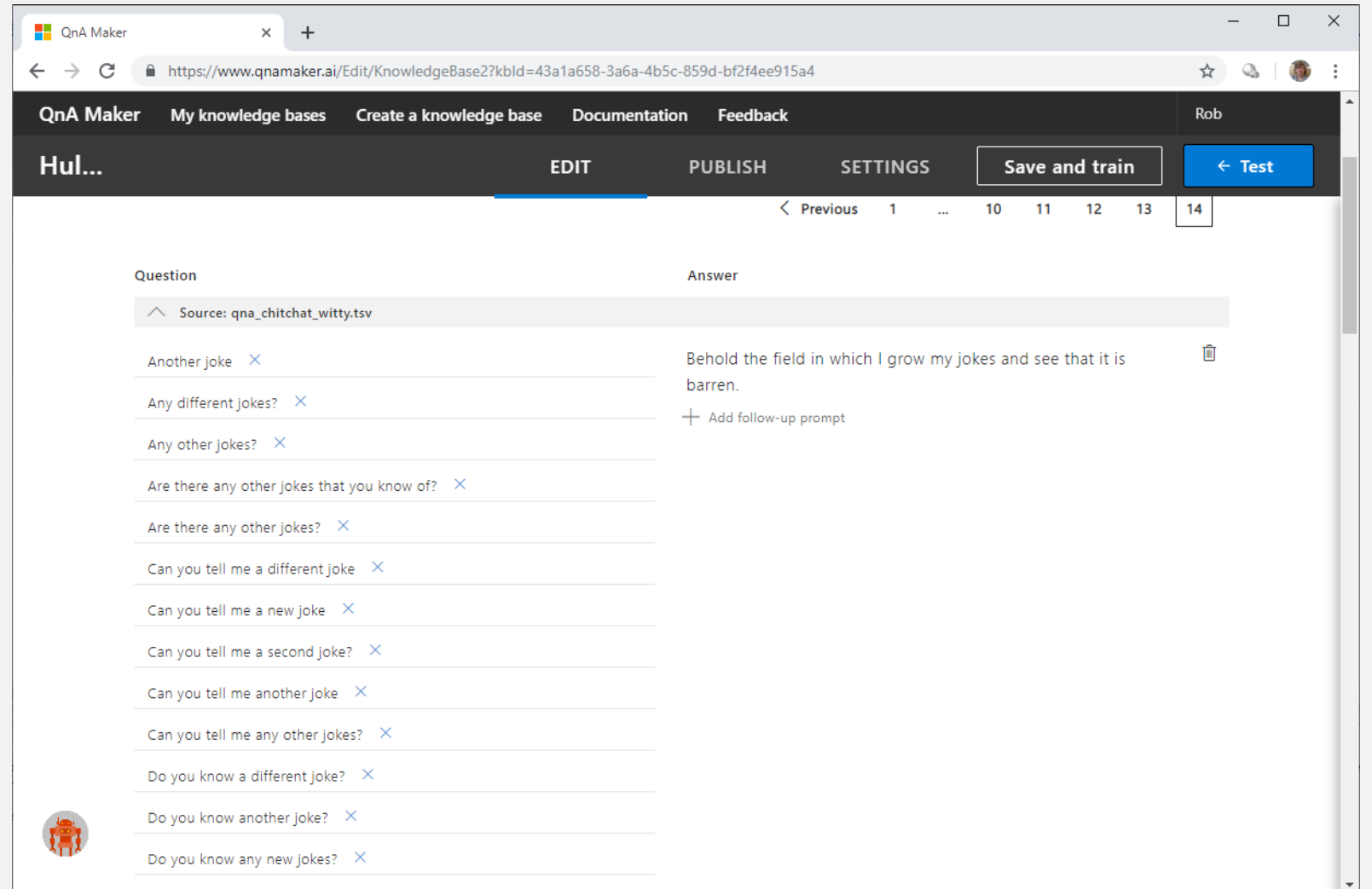
15:52

hello

Adding Personality

There are some “chit chat” personalities that you can add to your bot

These are supplied as question sets that you can add



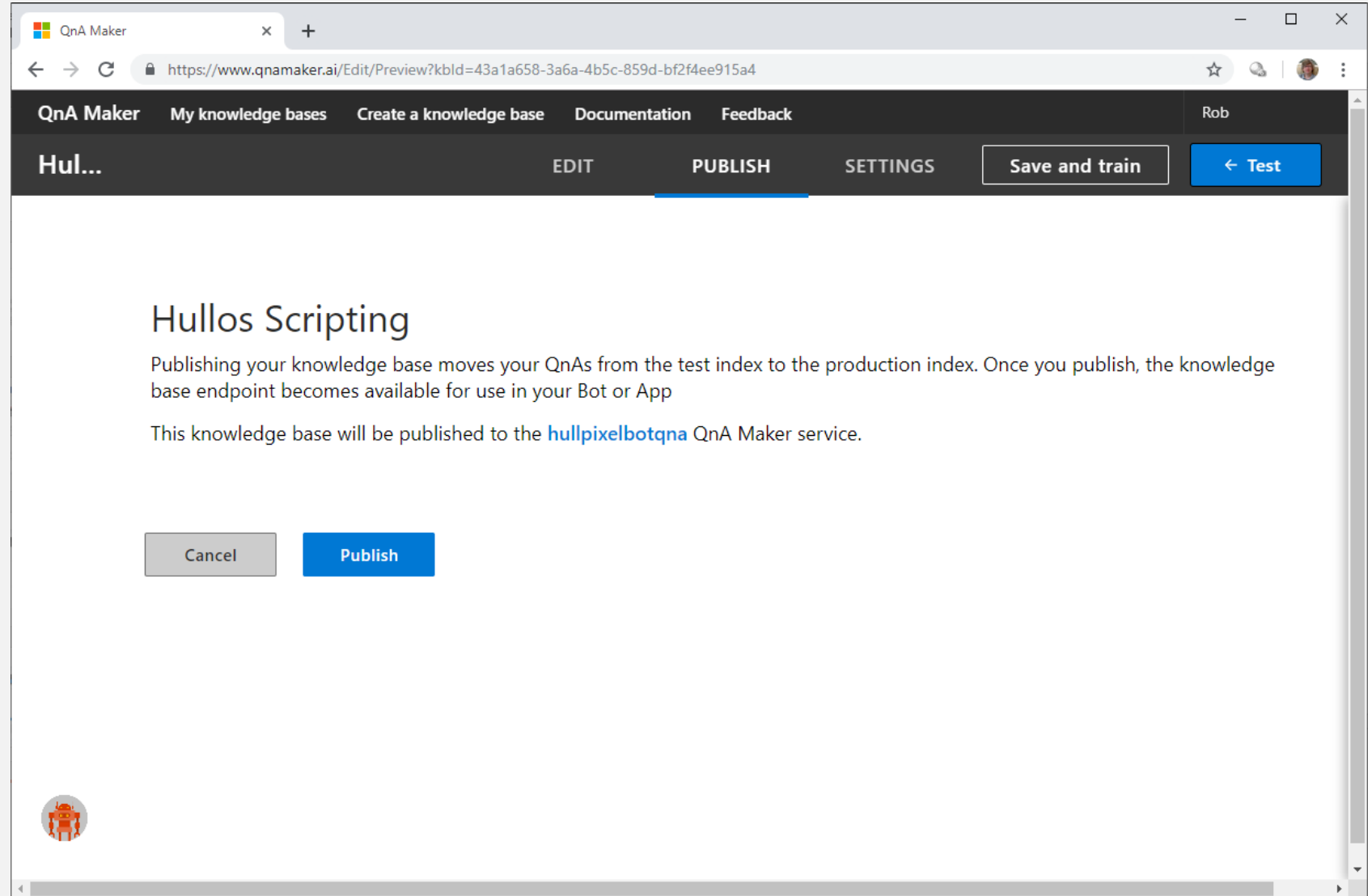
The screenshot shows the QnA Maker web interface. The browser address bar displays the URL: <https://www.qnamaker.ai/Edit/KnowledgeBase2?kblid=43a1a658-3a6a-4b5c-859d-bf2f4ee915a4>. The navigation bar includes links for 'QnA Maker', 'My knowledge bases', 'Create a knowledge base', 'Documentation', and 'Feedback'. The user's name 'Rob' is visible in the top right corner. The main interface is in 'EDIT' mode, with tabs for 'EDIT', 'PUBLISH', and 'SETTINGS'. A 'Save and train' button and a blue 'Test' button are present. A pagination control shows 'Previous', '1', '...', '10', '11', '12', '13', and '14'. The main content area is a table with two columns: 'Question' and 'Answer'. The source is identified as 'qna_chitchat_witty.tsv'. The table contains several rows of questions and their corresponding answers, such as 'Another joke' and 'Behold the field in which I grow my jokes and see that it is barren.' A '+ Add follow-up prompt' button is also visible.

Question	Answer
Source: qna_chitchat_witty.tsv	
Another joke	Behold the field in which I grow my jokes and see that it is barren.
Any different jokes?	+ Add follow-up prompt
Any other jokes?	
Are there any other jokes that you know of?	
Are there any other jokes?	
Can you tell me a different joke	
Can you tell me a new joke	
Can you tell me a second joke?	
Can you tell me another joke	
Can you tell me any other jokes?	
Do you know a different joke?	
Do you know another joke?	
Do you know any new jokes?	

Publishing the data set

When we are happy with the content we can publish the data to create a service

We can then create a bot that serves out this knowledge



The screenshot shows a web browser window with the URL <https://www.qnamaker.ai/Edit/Preview?kblid=43a1a658-3a6a-4b5c-859d-bf2f4ee915a4>. The page title is "Hullos Scripting". The navigation bar includes "QnA Maker", "My knowledge bases", "Create a knowledge base", "Documentation", "Feedback", and "Rob". The main content area has tabs for "EDIT", "PUBLISH", and "SETTINGS". The "PUBLISH" tab is active, and the "Save and train" button is highlighted. Below the tabs, the text reads: "Publishing your knowledge base moves your QnAs from the test index to the production index. Once you publish, the knowledge base endpoint becomes available for use in your Bot or App. This knowledge base will be published to the [hullpixelbotqna](#) QnA Maker service." At the bottom, there are two buttons: "Cancel" and "Publish".

Demo

Talking to my document with Skype

Using language recognition in your apps

```
var response = await qnaMaker.GetAnswersAsync(turnContext);  
if (response != null && response.Length > 0)  
{  
    await turnContext.SendActivityAsync(MessageFactory.Text(response[0].Answer + " " +  
        response[0].Score), cancellationToken);  
}
```

The pre-built bot displays the most likely response

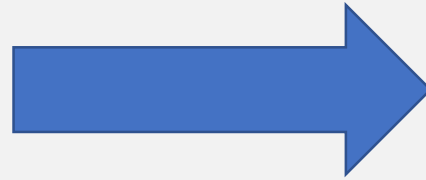
If we use the QnA service in our own application we can pull out a score value that will allow use to decide how much confidence the system has that it has a valid response

Demo

Talking to my document with C#

Adding more understanding

Book me a flight to Cairo



You can also use LUIS as a way of automating response to user queries

This requires a bit of training, mapping intent to input

```
{
  "query": "Book me a flight to Cairo",
  "topScoringIntent": {
    "intent": "BookFlight",
    "score": 0.9887482
  },
  "intents": [
    {
      "intent": "BookFlight",
      "score": 0.9887482
    }
  ],
  "entities": [
    {
      "entity": "cairo",
      "type": "Location",
      "score": 0.956781447
    }
  ]
}
```

Summary

A chatbot takes user input and generates responses

A response can take the form of an Adaptive Card that can contain buttons and images

The chatbot is deployed as a service that is resident in the cloud

A channel registration service connects the chatbot service with the channels it is to use

Channels can include web chat, Skype, email, Slack

The QnA service can convert input into questions and answers that can be deployed via a chatbot

The QnA service can be made "chatty" using "chit chat" resources

The QnA service can also be used in a regular chatbot

The LUIS framework allows user intent to be determined from input

www.robmiles.com