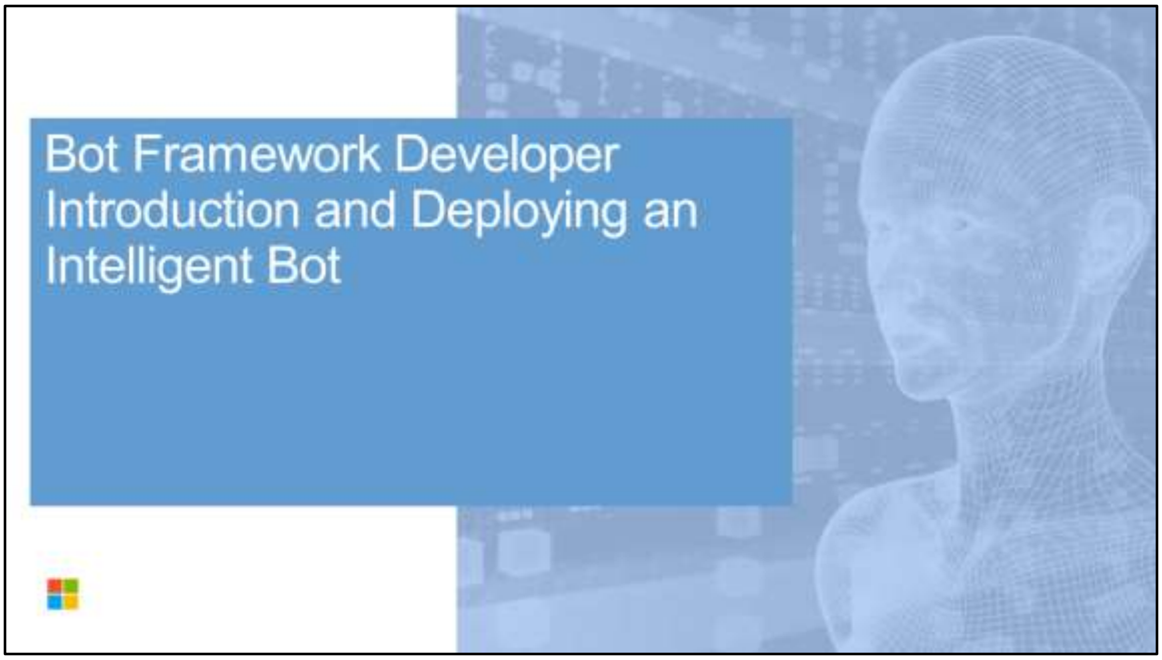
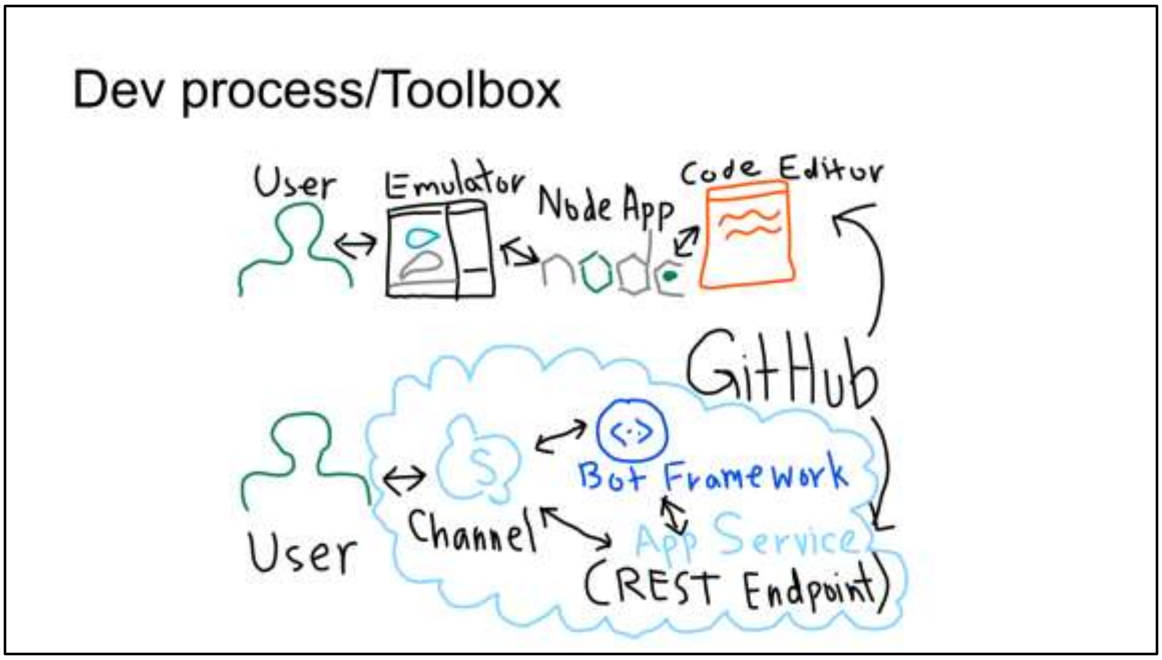


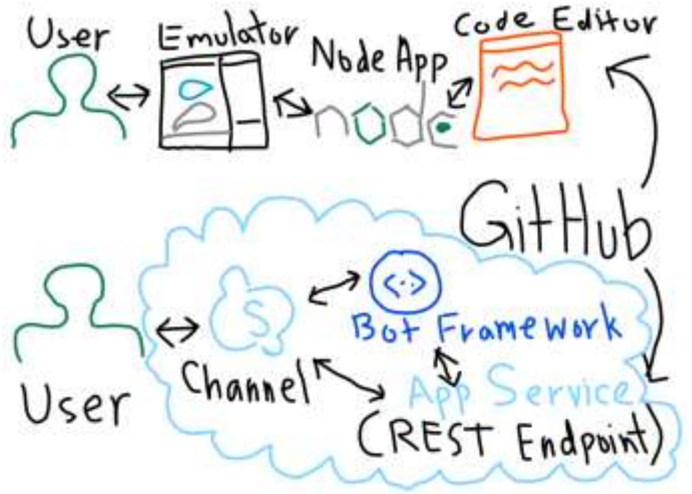
Bot Framework Developer Introduction and Deploying an Intelligent Bot

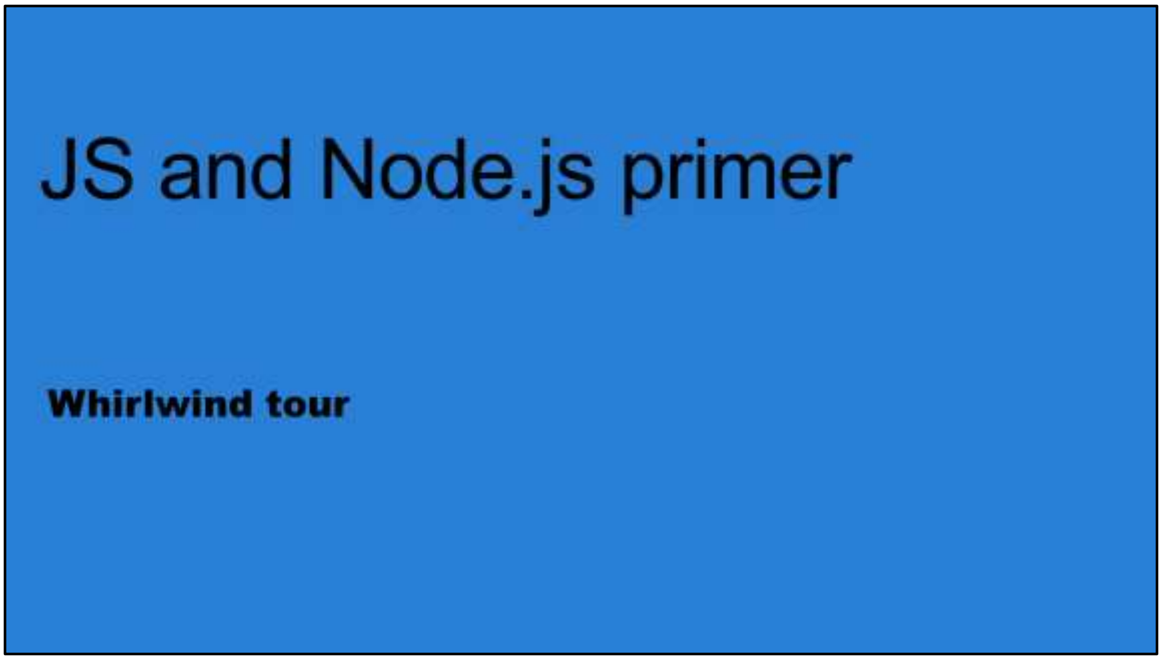
Getting started

Core concepts: https://docs.botframework.com/en-us/node/builder/guides/core-concepts/#navtitle



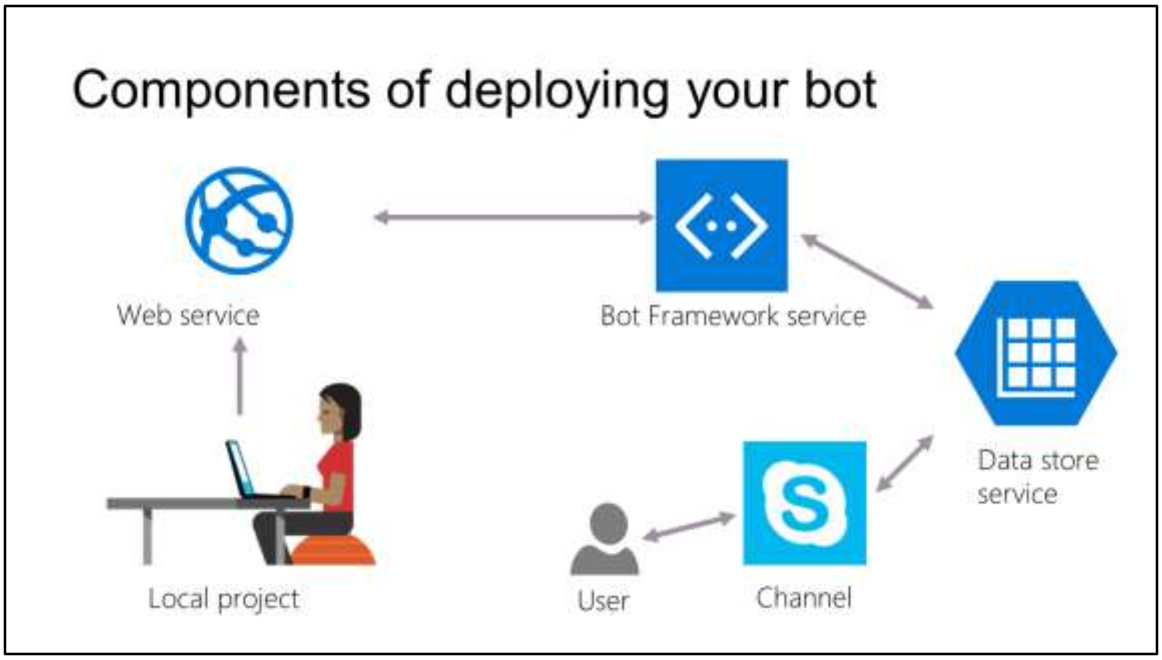
Dev process/Toolbox





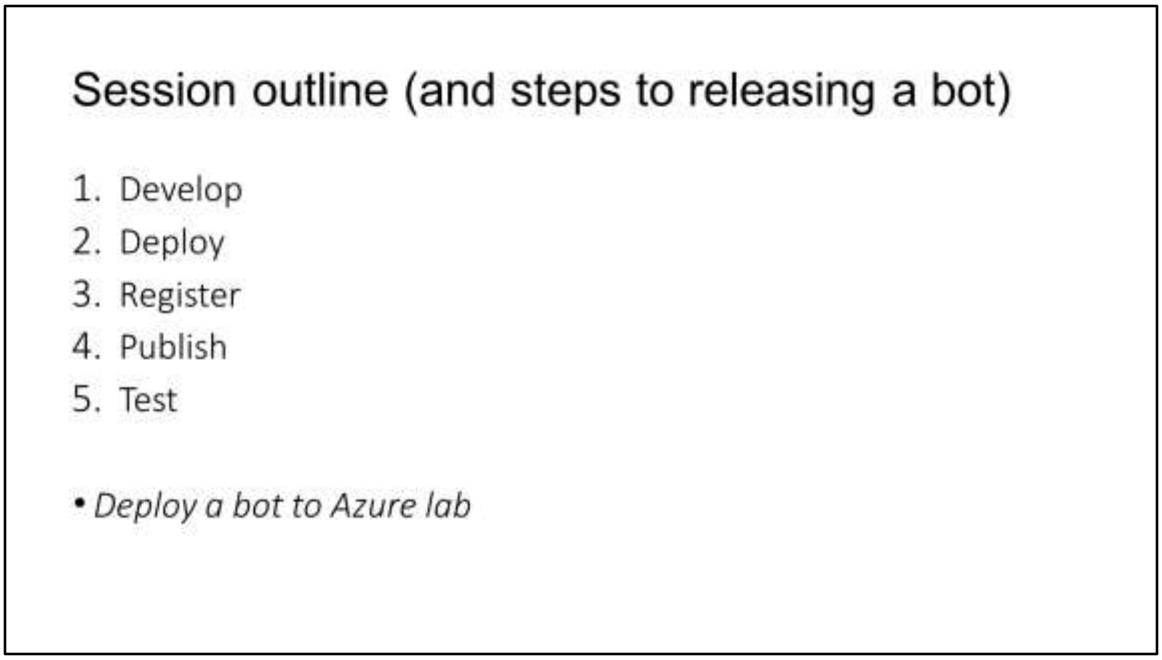
JS and Node.js primer

Whirlwind tour



Components of deploying your bot

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



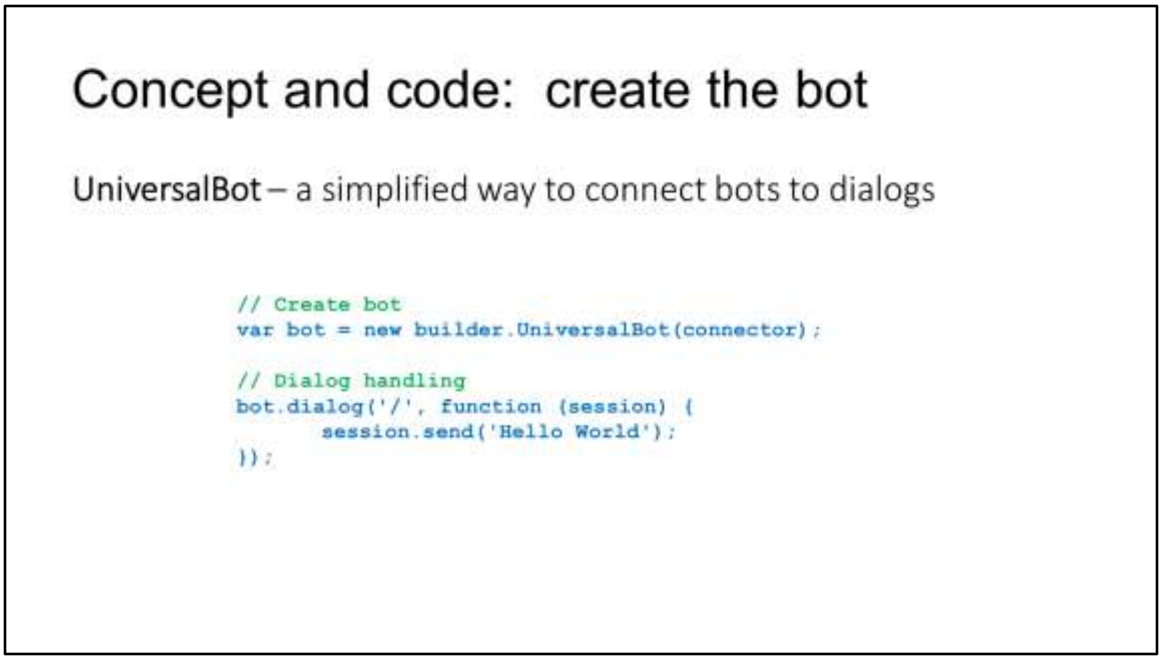
Session outline (and steps to releasing a bot)

1. Develop
2. Deploy
3. Register
4. Publish
5. Test

- *Deploy a bot to Azure lab*

Develop

in code editor



Concept and code: create the bot

UniversalBot – a simplified way to connect bots to dialogs

```
// Create bot
var bot = new builder.UniversalBot(connector);

// Dialog handling
bot.dialog('/', function (session) {
       session.send('Hello World');
});
```

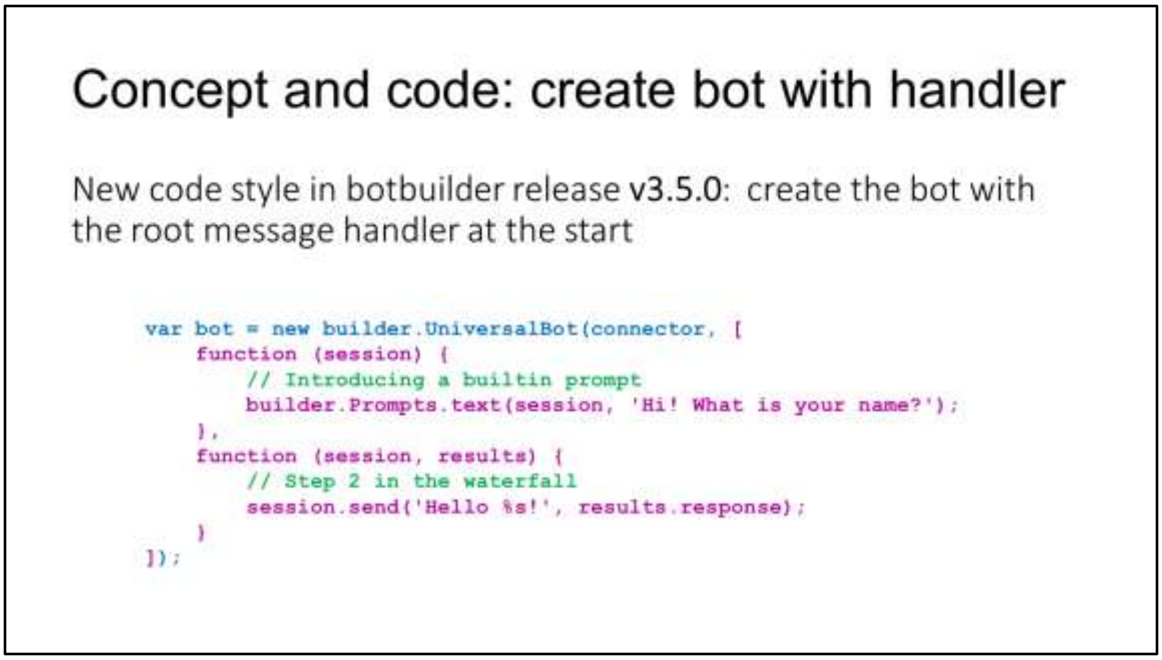
You will use your setup environment in VSCode (or elsewhere) to work through these examples. The code is also at: https://github.com/Azure/bot-education/blob/master/Student-Resources/BOTs/Node/bot-playground/server.js

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

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/



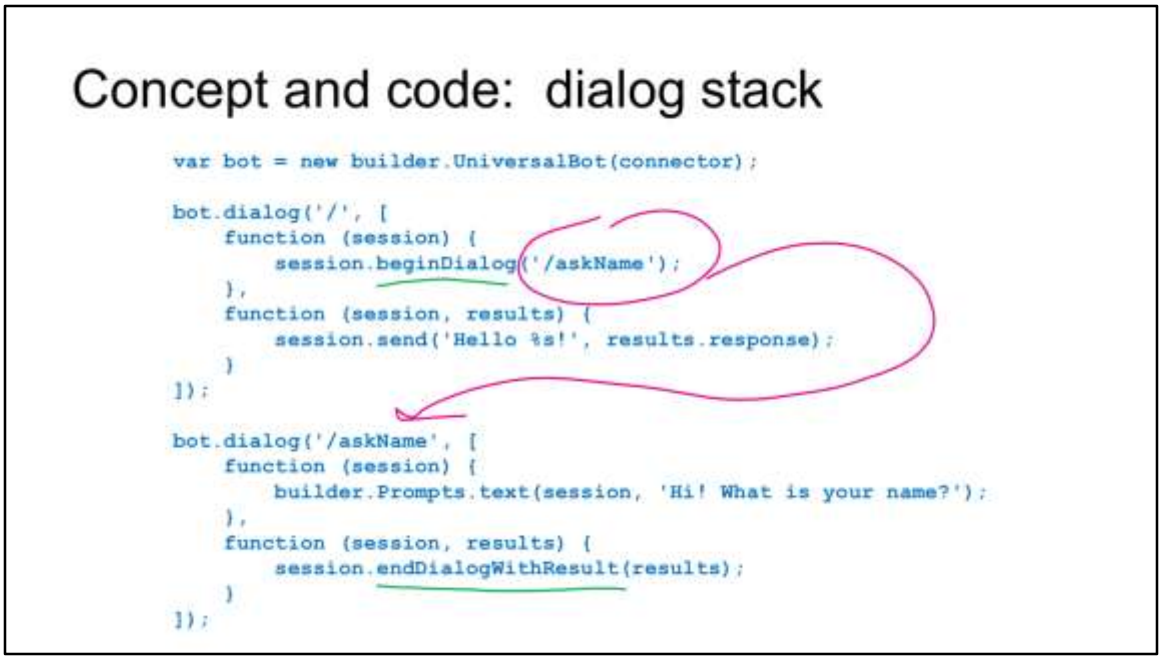
Concept and code: create bot with handler

New code style in botbuilder release **v3.5.0**: create the bot with the root message handler at the start

```
var bot = new builder.UniversalBot(connector, [
    function (session) {
        // Introducing a builtin prompt
        builder.Prompts.text(session, 'Hi! What is your name?');
    },
    function (session, results) {
        // Step 2 in the waterfall
        session.send('Hello %s!', results.response);
    }
]);
```

You will use your setup environment in VSCode (or elsewhere) to work through these examples.

Really a style choice
Here we are working with user data and message data



Concept and code: dialog stack

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

bot.dialog('/', [
    function (session) {
        session.beginDialog('/askName');
    },
    function (session, results) {
        session.send('Hello %s!', results.response);
    }
]);

bot.dialog('/askName', [
    function (session) {
        builder.Prompts.text(session, 'Hi! What is your name?');
    },
    function (session, results) {
        session.endDialogWithResult(results);
    }
]);
```

You will use your setup environment in VSCode (or elsewhere) to work through these examples.

Here let’s assume we’ve created out bot as in the previous slide (var bot = …)

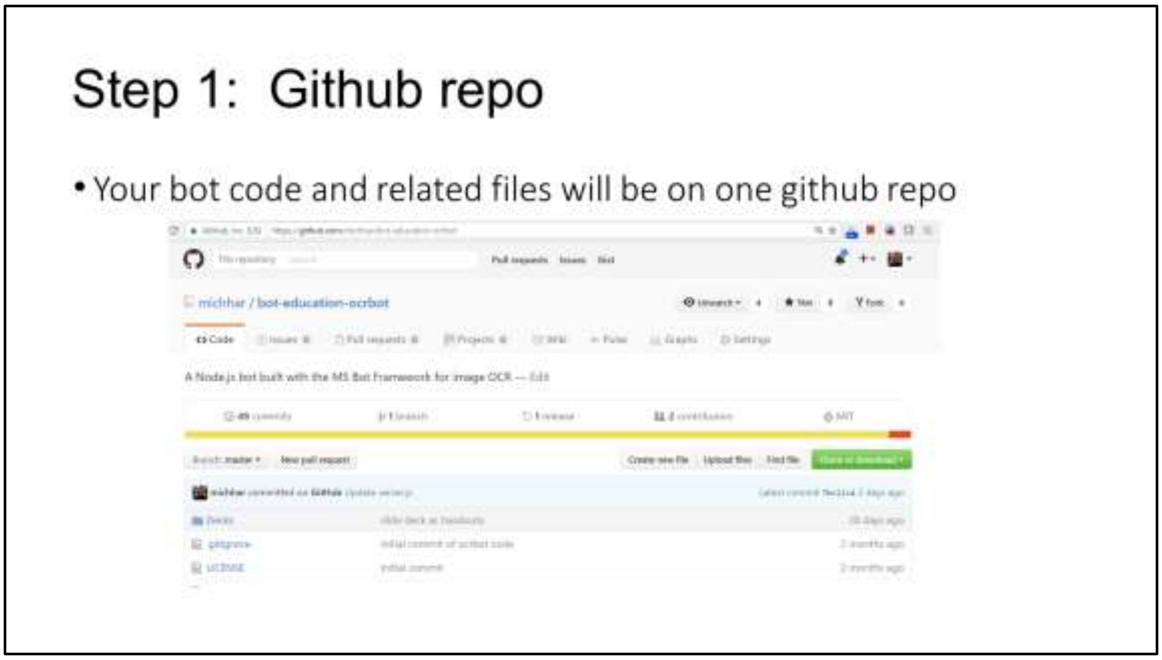
beginDialog – pushes data onto the “stack”
endDialogWithResults – pops data off of the “stack” and in this case returns the results (we could have done some more interesting things with the data of course…)

Note: both dialog “routes” have small waterfalls

Deploy

to Azure

https://docs.botframework.com/en-us/node/builder/guides/deploying-to-azure/#step-1-get-a-github-repo



Step 1: Github repo

- Your bot code and related files will be on one github repo

michhar / bot-education-ocrbot

Code | Issues | Pull requests | Projects | Wiki | Pulse | Graphs | Settings

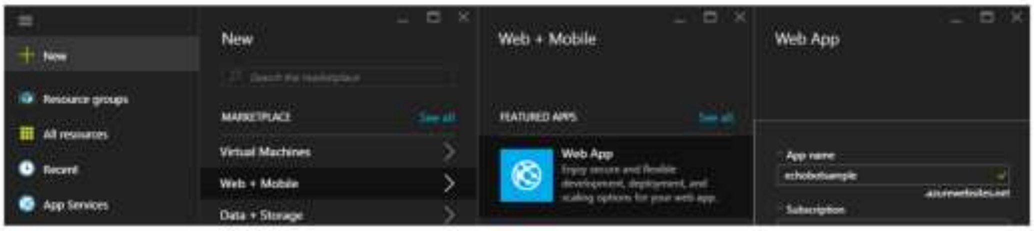
A Node.js bot built with the MS Bot Framework for image OCR — Edit

For example: https://github.com/michhar/bot-education-ocrbot

Either fork this repo or create your own and place your bot code in it (the server.js and any additional necessary files; I usually like to include the package.json)

Note: there are other deployment methods

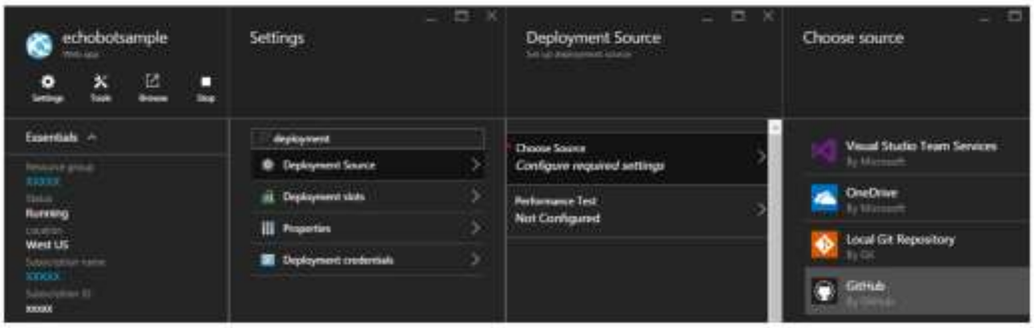
Step 2: Create an Azure Web App



App name: echobotsample .azurewebsites.net

In azure portal: https://portal.azure.com

Step 3: Set up for continuous deployment

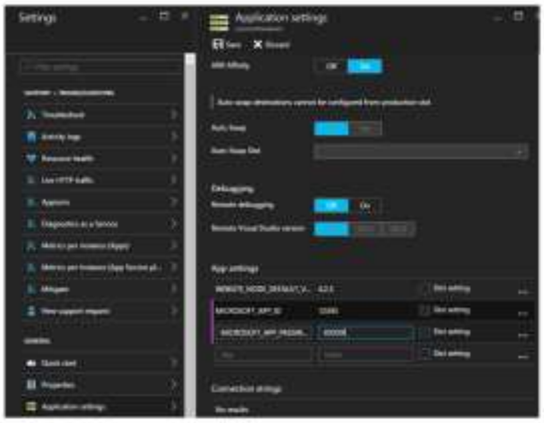


In azure portal: https://portal.azure.com

Select github and enter your credentials for GitHub when asked.

Click on Browse to test the deployment. Should see the message from the index.html file.

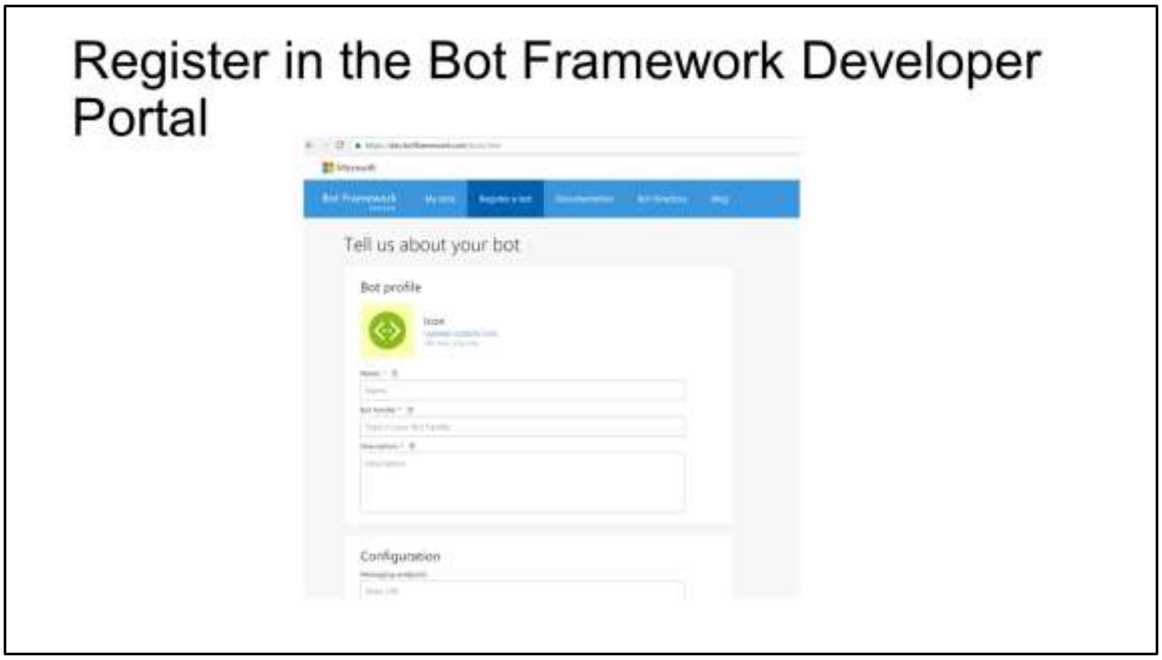
Step 4: Enter in App Id and App Password



This allows the password to be safe and guarded.

Register

in the Developer Portal

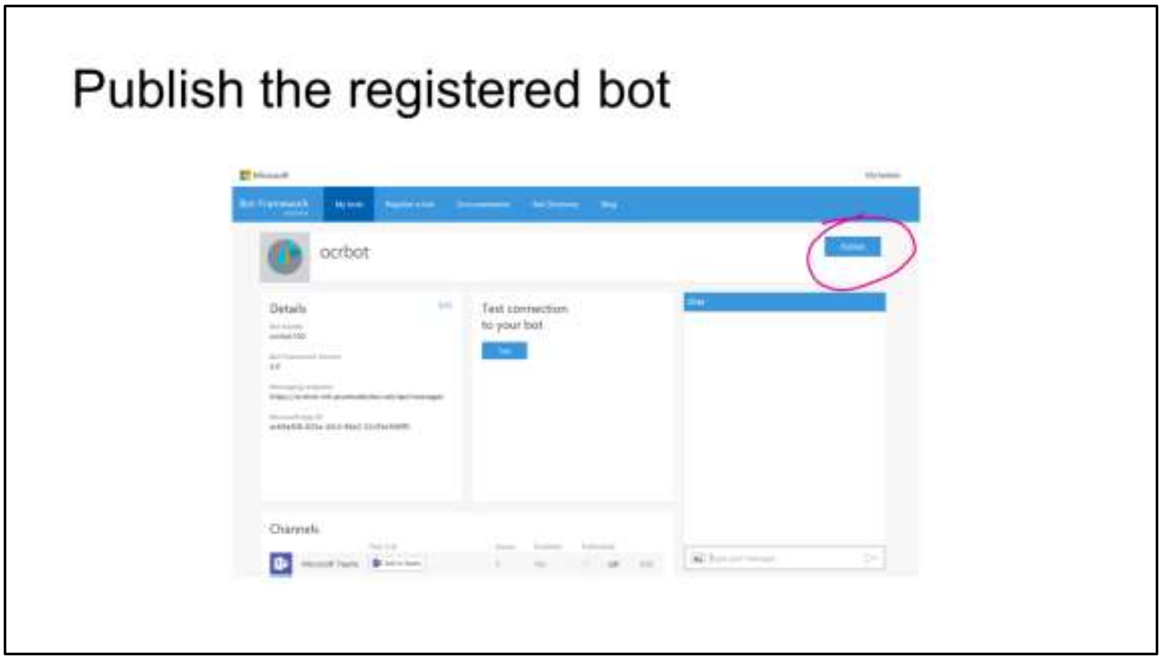


Register in the Bot Framework Developer Portal

On https://dev.botframework.com click on “Register a bot”

Publish to the Bot Directory

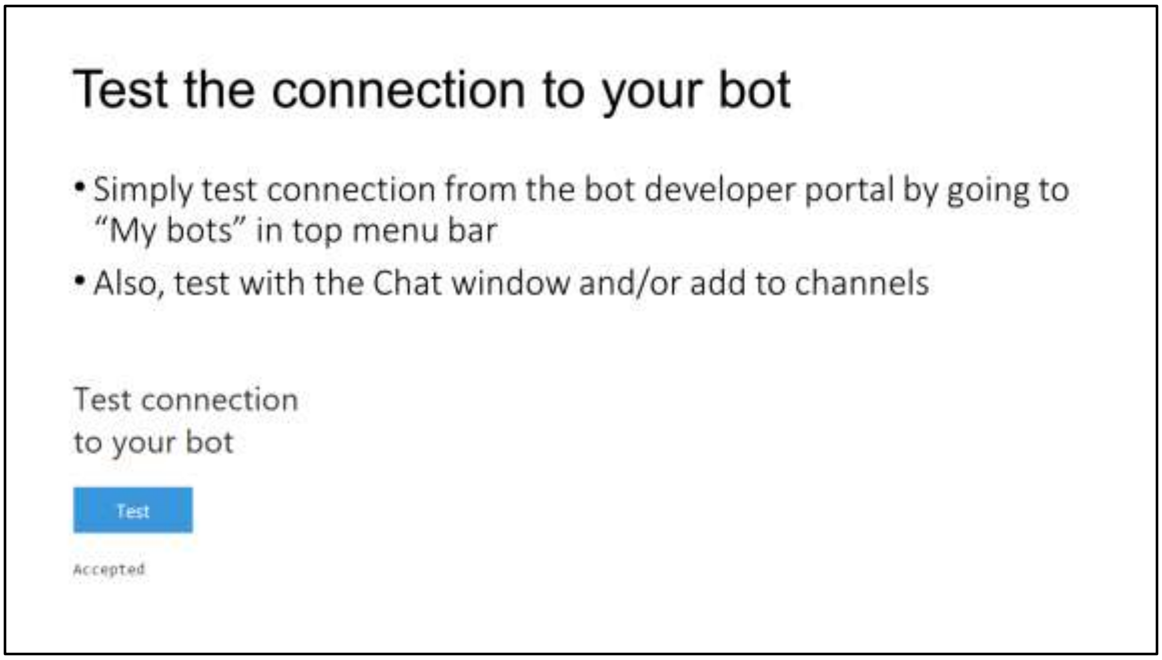
In the Developer Portal



Publish the registered bot

If you wish – it will then be reviewed and surface in the Bot Directory
Review guidelines: https://docs.botframework.com/en-us/directory/review-guidelines/

Test connection and conversation

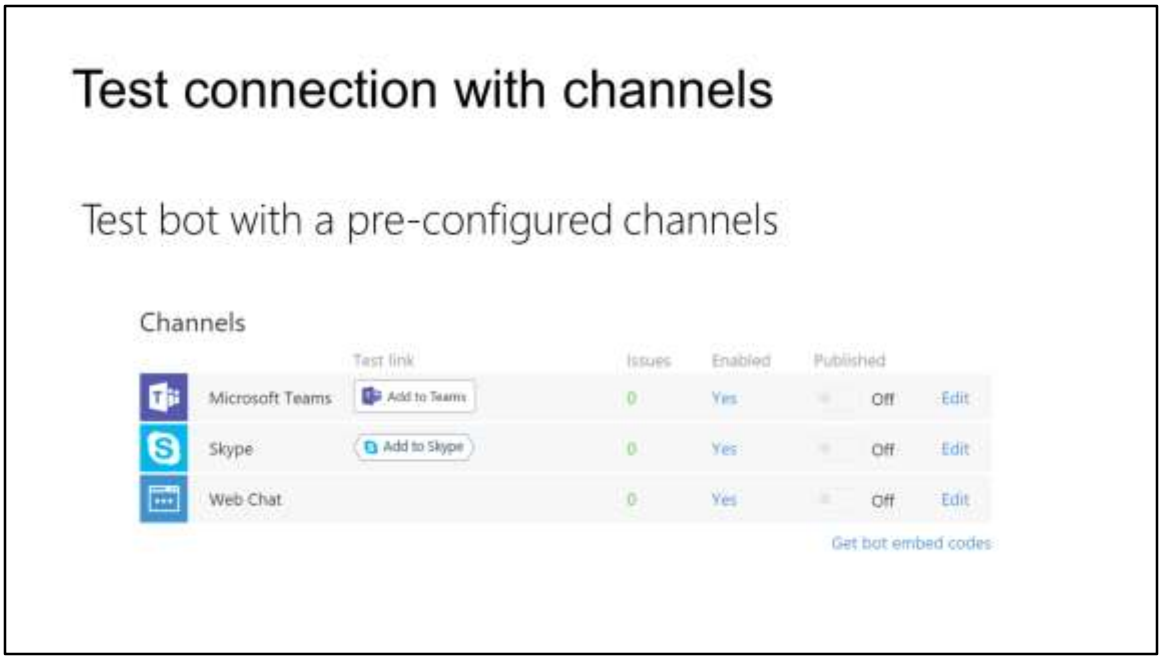


Test the connection to your bot

- Simply test connection from the bot developer portal by going to "My bots" in top menu bar
- Also, test with the Chat window and/or add to channels

Test connection
to your bot

```
Accepted
```



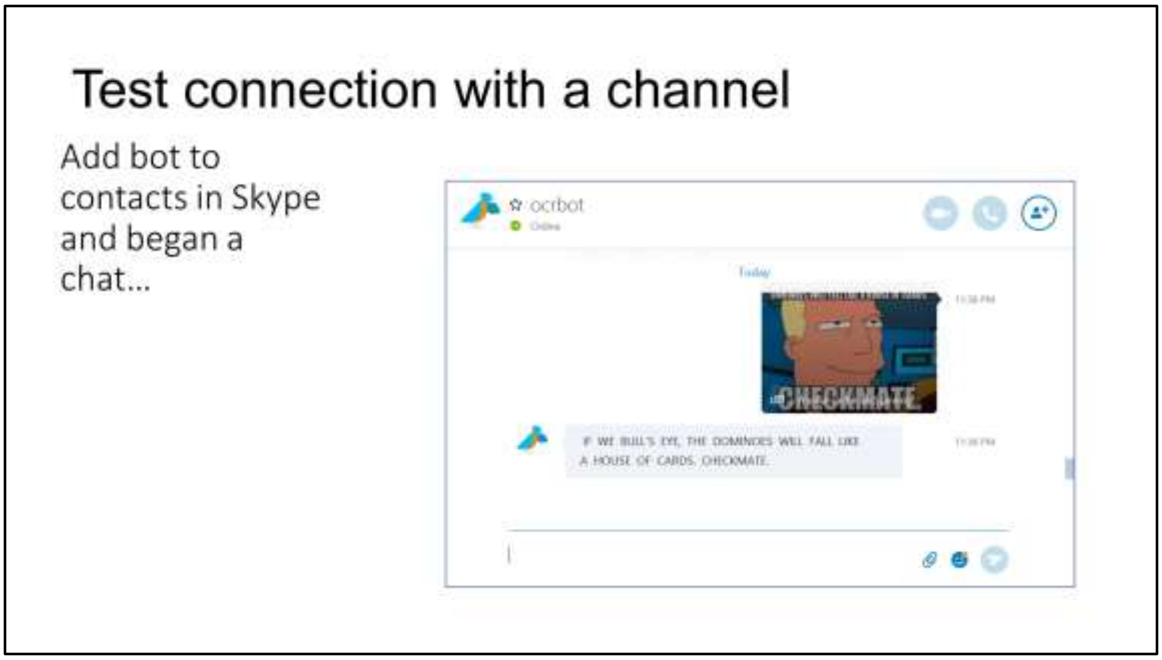
Test connection with channels

Test bot with a pre-configured channels

Channels

		Test link	Issues	Enabled	Published	
	Microsoft Teams	Add to Teams	0	Yes	Off	Edit
	Skype	Add to Skype	0	Yes	Off	Edit
	Web Chat		0	Yes	Off	Edit

Get bot embed codes



Test connection with a channel

Add bot to contacts in Skype and began a chat...

ocrbot

Online

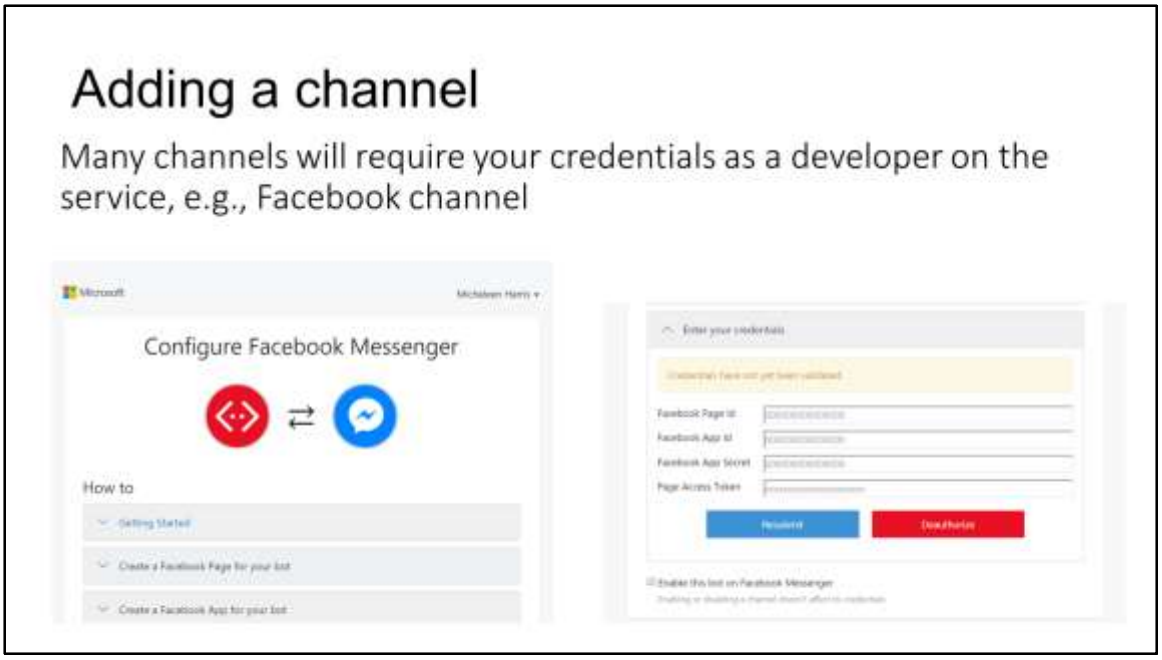
Today

11:38 PM

IF WE BULL'S EYE, THE DOMINOES WILL FALL LIKE A HOUSE OF CARDS. CHECKMATE.

11:38 PM

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



Adding a channel

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

Microsoft   Michaleen Harris ▾

Configure Facebook Messenger

How to

- Getting Started
- Create a Facebook Page for your bot
- Create a Facebook App for your bot

Enter your credentials

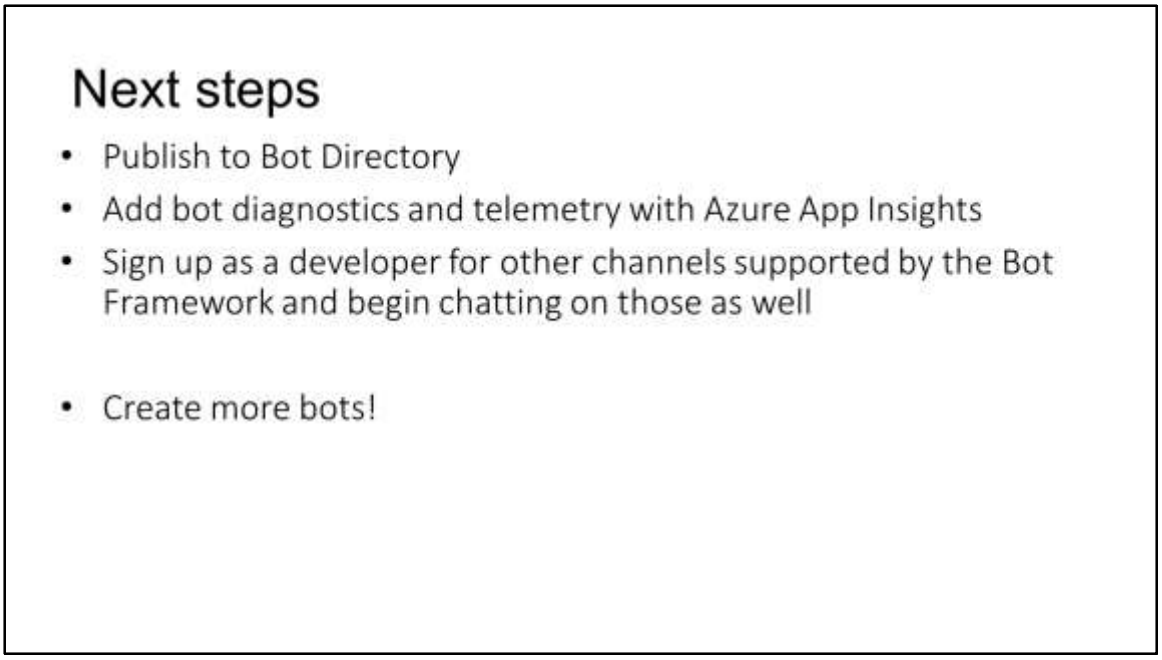
Credentials have not yet been validated

Facebook Page Id	
Facebook App Id	
Facebook App Secret	
Page Access Token	

☐ Enable this bot on Facebook Messenger

Enabling or disabling a channel doesn't affect its credentials

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



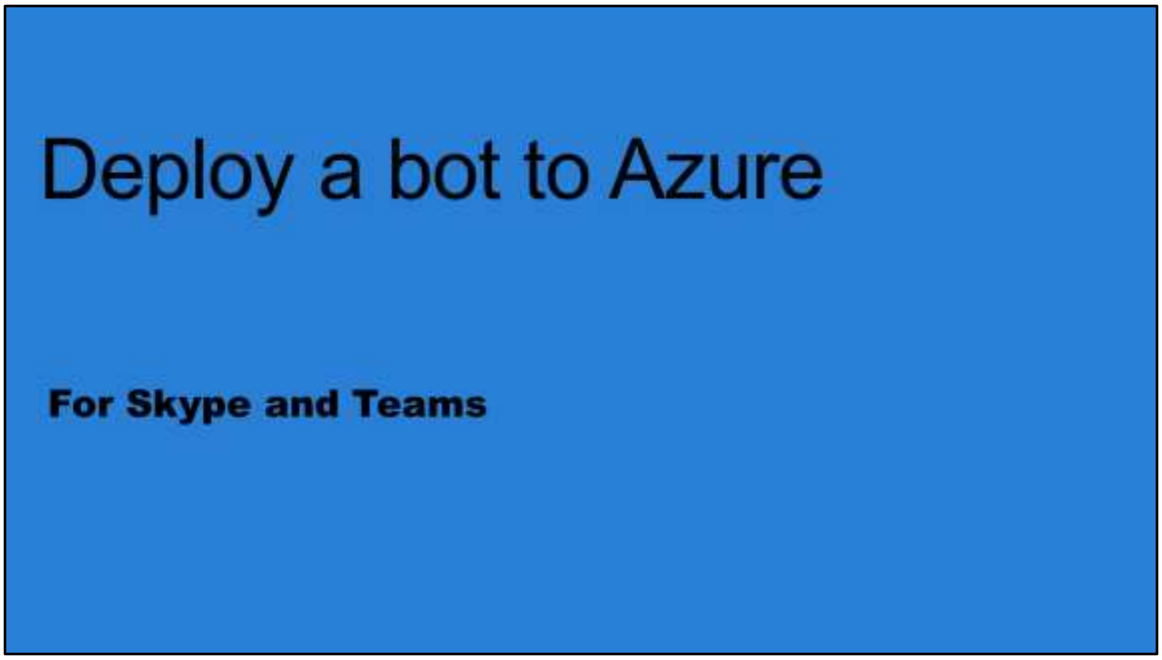
Next steps

- Publish 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!

Questions?

Also, try the course gitter chatroom at aka.ms/botedu-discuss



Deploy a bot to Azure

For Skype and Teams

Fork this project into your own github account: https://github.com/michhar/bot-education-ocrbot

Another good set of instructions here: https://michhar.github.io/notes_posts/ocrbot-makes-a-connection