

+Creating a bot using the Microsoft Bot  
Framework  
In Node.js  
Hands-on Lab Manual

---

# Table of Contents

---

Creating a bot using the Microsoft Bot Framework In Node.js .....	1
Hands-on Lab Manual .....	1
Lab Introduction .....	3
Objectives .....	3
Prerequisites .....	3
Lab Scenarios .....	3
Configuration and Setup .....	4
Copy/Paste of Code .....	9
Exercise 1: Basic Bot using BotBuilder .....	10
Exercise 2: Using prompts in a bot .....	16
Exercise 3: Using Intent Dialogs (LUIS) .....	19
Exercise 4: Connecting to Skype and Webchat .....	25
Additional Resources .....	40
Copyright .....	41

---

# Lab Introduction

---

## Objectives

After completing these self-paced labs, you will be able to:

- Have an understanding of the basics of the Bot Framework

## Prerequisites

- Node.js
- A command line utility
- Basic understanding of Node.js & JavaScript

## Lab Scenarios

This series of exercises is designed to show you how to get started using the Microsoft Bot Framework.

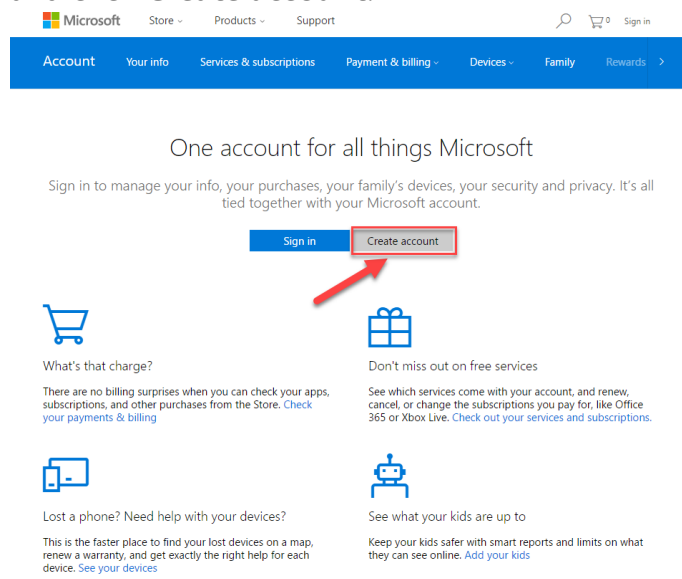
# Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

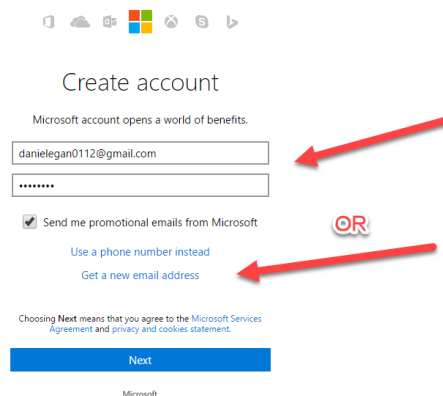
Page 4 of 41

## Configuration and Setup

1. Install prerequisite software
  - **Node.js** : <https://nodejs.org/en/> (this will include NPM)
  - **NGrok** : <https://ngrok.com/>
  - **Skype** : <http://skype.com> (if you want to test a Skype Bot)
  - **Visual Studio Code**. VS Code is cross platform, you can use others if you like (like sublime, Atom, Notepad++, Vim, etc..) but you will be on your own with debugging and usage.
    1. <https://code.visualstudio.com/>
2. **Create a Microsoft ID** (if you don't already have one)
  - Go to the Microsoft account sign-up page <https://account.microsoft.com/> and **click Create account**.



- In the User name box enter your existing email address, or click Get a new email address to create an Outlook or Hotmail address.



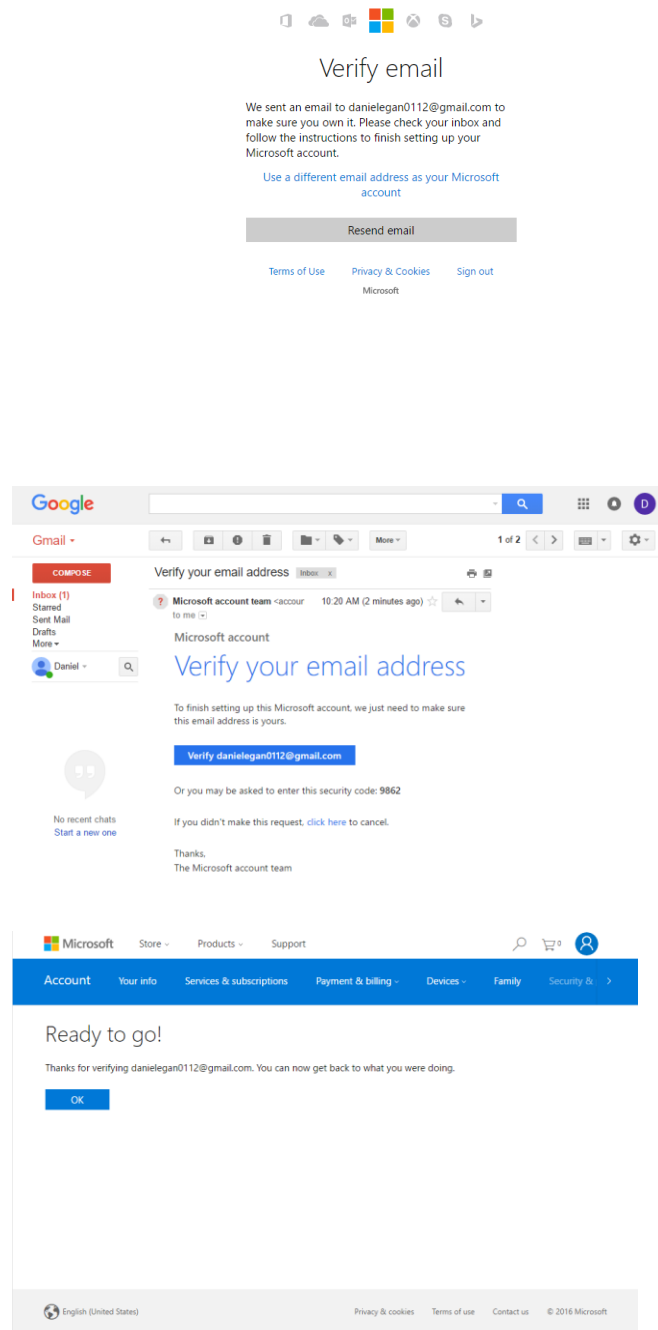
**NOTE: If you use an existing email address you will need to verify it**

# Creating a bot using the Microsoft Bot Framework

## Node.js Hands-on Labs

Page 5 of 41

**before moving on.**

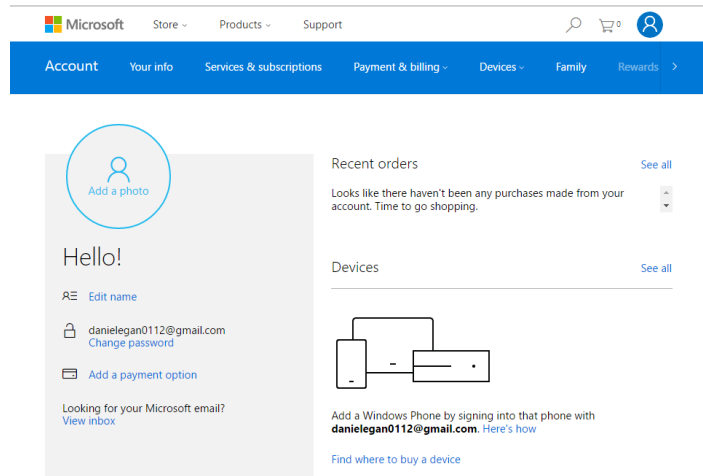


- Either path will take you to this screen

# Creating a bot using the Microsoft Bot Framework

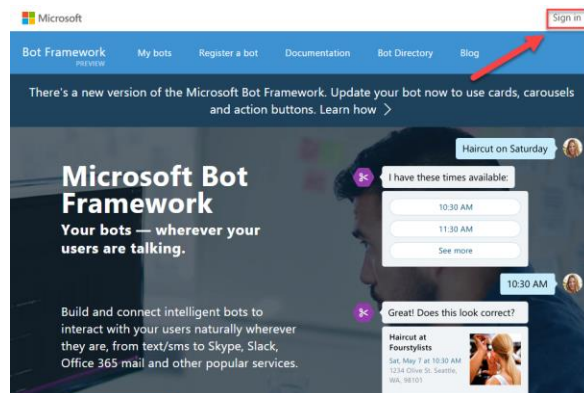
Node.js Hands-on Labs

Page 6 of 41



## 3. Create a BotFramework account

- Navigate to <http://BotFramework.com>
- Click on sign in

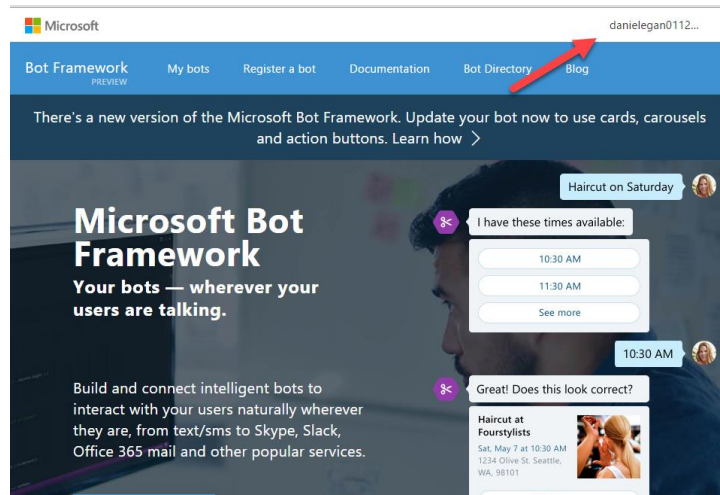


- If you are using the same browser that you used to create your Microsoft ID then you will be signed in automatically, otherwise you will need to use the ID you just created to sign in.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

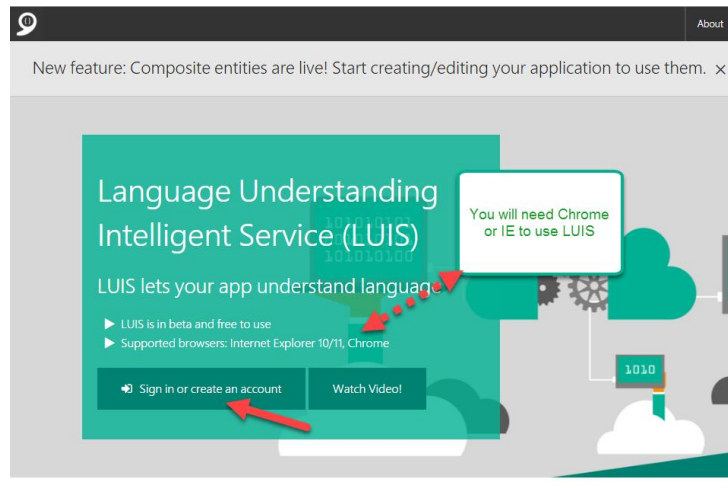
Page 7 of 41



- You can leave this window open, we will be using it later.

### 4. Sign-up for LUIS. Language Understanding Intelligent Services

- <https://www.luis.ai/>
- Click on :Sign In or Create Account button

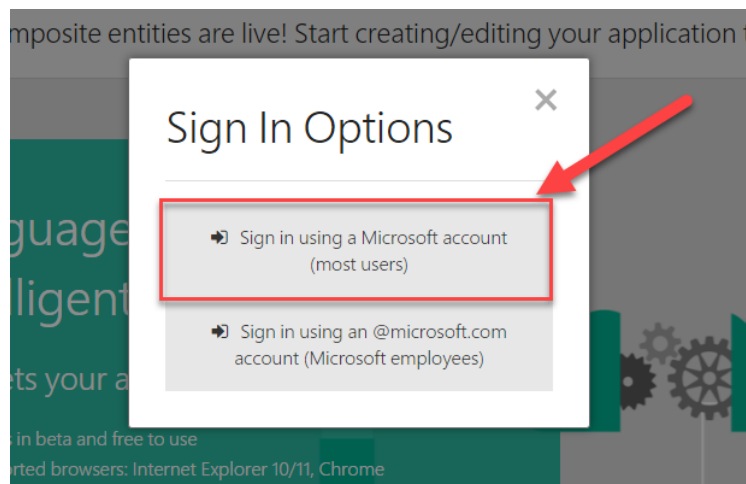


- Select **Sign in using a Microsoft Account** (most users)

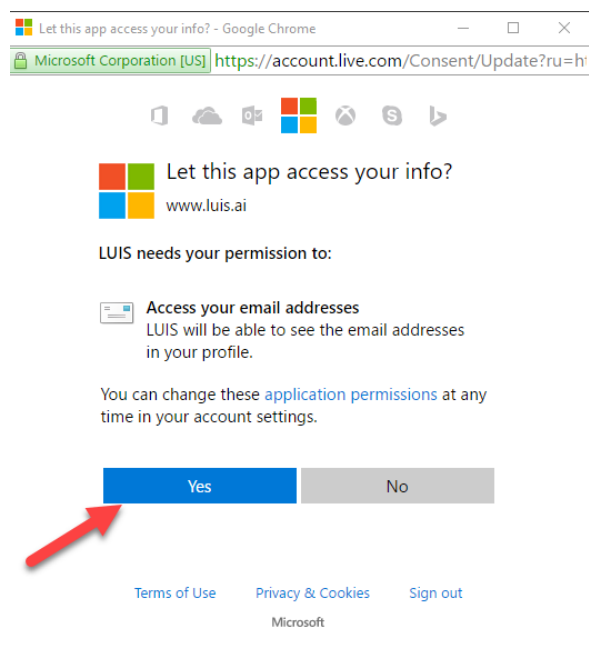
## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 8 of 41



- If you are still signed in it will ask you to say Yes to accept permissions. Otherwise you will need to sign in with the Microsoft ID you created earlier.



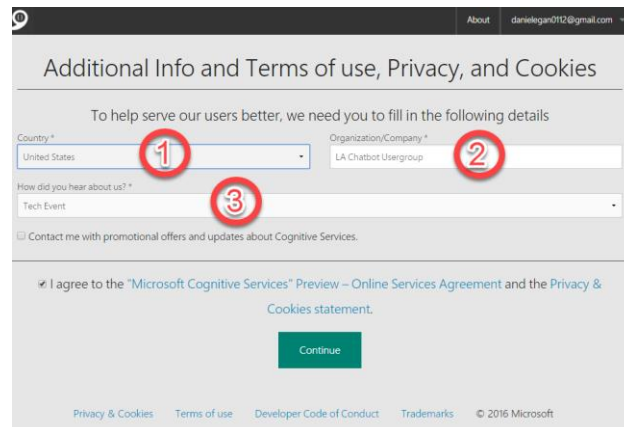
- In the next form enter
  1. United States
  2. LA Chatbot Usergroup
  3. Tech Event
  4. Uncheck the Contact Me checkbox if you want
  5. CHECK the I agree checkbox
  6. Click Continue



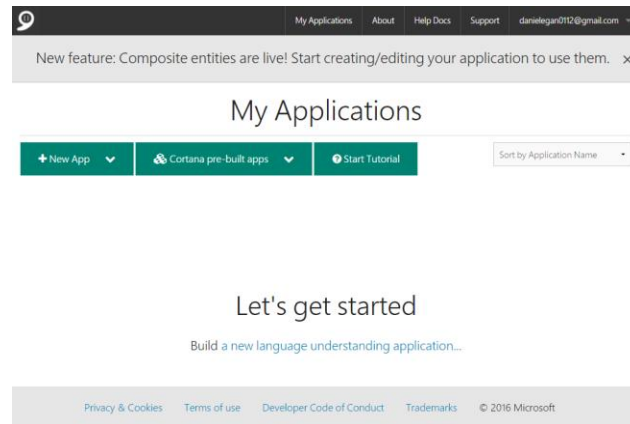
## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 9 of 41



- You can walk through the quick walkthrough if you would like or click the x to close it. When finished, your screen should look like below.



- We will explain and use this later for our bot.

### Copy/Paste of Code

You will have the option to copy/paste code snippets from this document to complete this lab. You will learn much more by typing it in yourself but sometimes in a lab format speed is needed to get through all the exercises in time.

**NOTE:** If you are on a mac, you will be using the PDF file. Do not copy and paste from the PDF file. There is a separate file called SNIPS.txt that contain the snips you need.

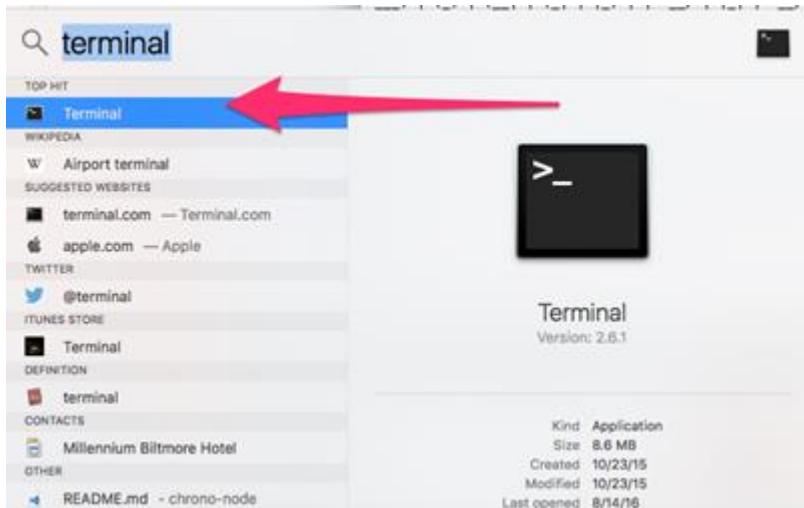
# Exercise 1: Basic Bot using BotBuilder

In this exercise, you will create a simple bot using the BotBuilder and

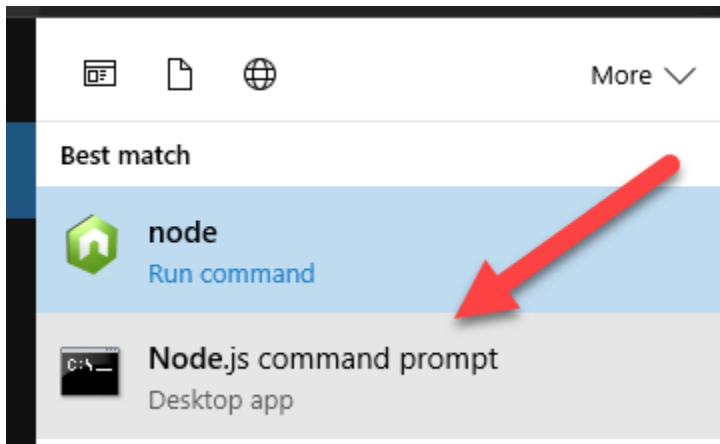
### Detailed Steps

1. Open up your Terminal. (This could be Terminal on Mac, PowerShell on windows, or a terminal of your choice (cmndr, iTerm, etc...))

On Mac hit **Command** → **Spacebar** and type terminal



From windows hit the windows key and type node or PowerShell

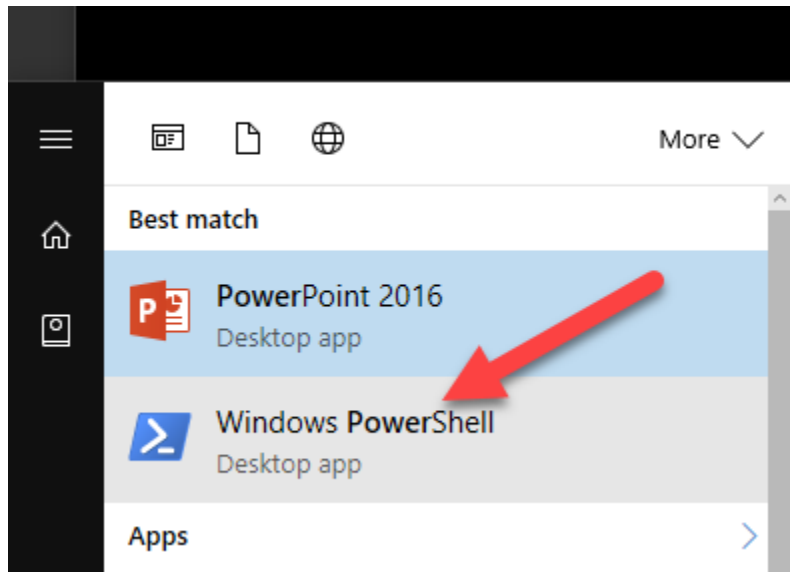


## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 11 of 41

### Detailed Steps



2. Next we want to create a folder to hold our project and add some initial files. (personally I hold all project folders in a folder called projects (**C:\Projects on PC or my 'userfolder'/projects on mac**)).

To create the folder, type the following from the command line to make a directory (mkdir)

```
~$ mkdir botworkshop
```

next we want to change to that directory (cd)

```
~$ cd botworkshop
```

next we will initialize it with a **package.json** file. We will use the **-y** so that it gives us the default values.

```
~$ npm init -y
```

the next step is to install the botbuilder npm module

```
~$ npm install botbuilder --save
```

finally, we want to open all of this up in **VSCode** type the following (the word code with a space then a period)

```
~$ code .
```

**NOTE:** You can also open up VSCode and select **File → Open (MAC) File → Open folder (PC)** and select the botworkshop folder we created)

## Creating a bot using the Microsoft Bot Framework

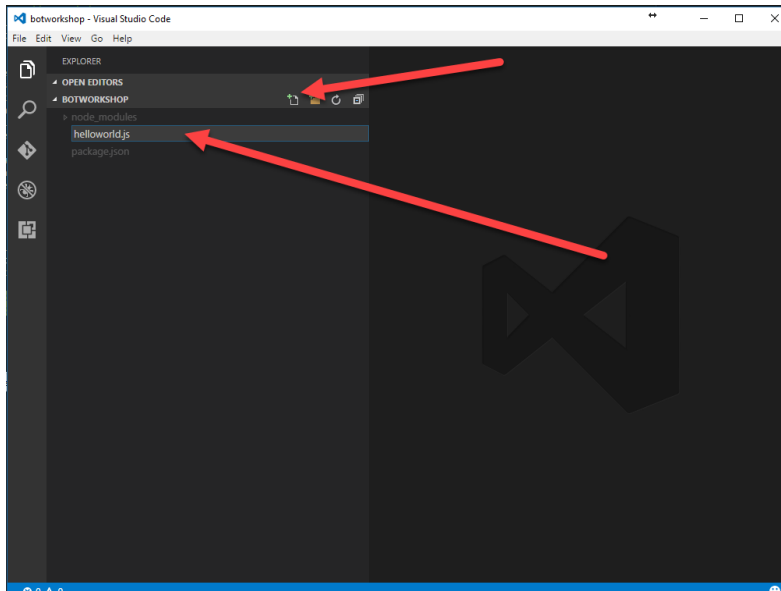
Node.js Hands-on Labs

Page 12 of 41

### Detailed Steps

This will open up our project in VSCode (again, you are free to choose your own editor if you like)

- Now we want to create our first simple hello world bot. Click on the **Add File** icon, and name the file **helloworld.js**



- In the file you just created, either type in or paste the following code.

**NOTE:** if you are using the PDF file, you can find the snippets in a separate file called SNIPS.txt

-----SNIP1-----

```
//Talking with the user.. as simple as possible
var builder = require('botbuilder');

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

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

- Now go back to your command prompt (terminal, powershell, etc) and type the following (make sure you are still in the projects/botworkshop folder)

```
~$ node helloworld.js
```

Nothing should happen, Well, you should not get any errors anyway. If all went well, it should be "listening" for you.

Type the word Hello (or anything really) and you should get back a "Hello World" from your bot.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 13 of 41

### Detailed Steps



```
Windows PowerShell (Admin)

C:\projects\botworkshop
λ node helloworld.js
Hello
Hello World
C:\projects\botworkshop
λ
```

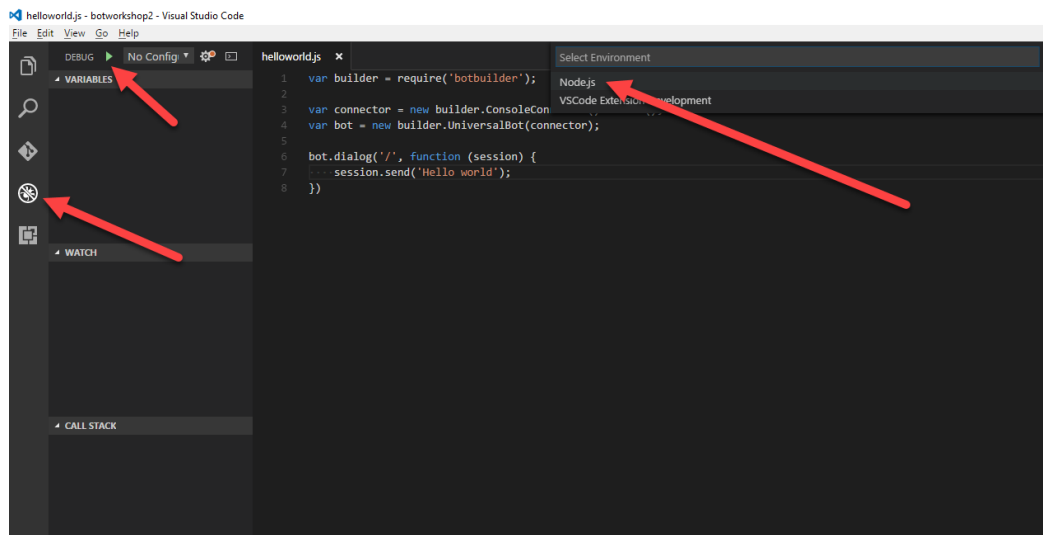
6. Hit **Ctrl +C** to exit out of the bot and back to command line.

7. Now we want to setup a debug session. It is best to do this on the simple bot so you can get used to using the debugger.

> In VSCode, click on the bug icon (Left side, second from bottom)

> On the top you can see that there is no config file. Click on the green arrow top left

> This will produce a drop down, select Node.js



8. This will create a **launch.json** file. Open this file and change the **“program”** attribute from **index.js** to **helloworld.js**

## Creating a bot using the Microsoft Bot Framework

### Node.js Hands-on Labs

Page 14 of 41

#### Detailed Steps

```
1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "name": "Launch",
6       "type": "node",
7       "request": "launch",
8       "program": "${workspaceRoot}/index.js",
9       "stopOnEntry": false,
10      "args": [],
11      "cwd": "${workspaceRoot}",
12      "preLaunchTask": null,
13      "runtimeExecutable": null,
14      "runtimeArgs": [
15        "--nolazy"
16      ],
17      "env": {
18        "NODE_ENV": "development"
19      },
20      "externalConsole": false,
21      "sourceMaps": false,
22      "outDir": null
23    },
24    {
25      "name": "Attach",
26      "type": "node",
27      "request": "attach",
28      "port": 5858,
29      "address": "localhost",
30      "restart": false,
31      "sourceMaps": false,
32      "outDir": null,
33      "localRoot": "${workspaceRoot}",
```

**NOTE:** The last step is not used in this HOL because we will be using Attach as opposed to LAUNCH but it is good to be aware of that field for the future.

9. Next open the **helloworld.js** file and put a breakpoint next to the **session.send('Hello World');** line by clicking next to the line number.

```
launch  ▾  helloworld.js  x  launch.json
1  var builder = require('botbuilder');
2
3  var connector = new builder.ConsoleConnector().listen();
4  var bot = new builder.UniversalBot(connector);
5
6  bot.dialog('/', function (session) {
7    session.send('Hello world');
8  })
```

10. Now we want to run our program in debug mode. Open up your console (Terminal, Powershell, etc) and type the following:

```
~$ node --debug-brk helloworld.js
```

11. That should run the node process in debug mode listening on port 5858

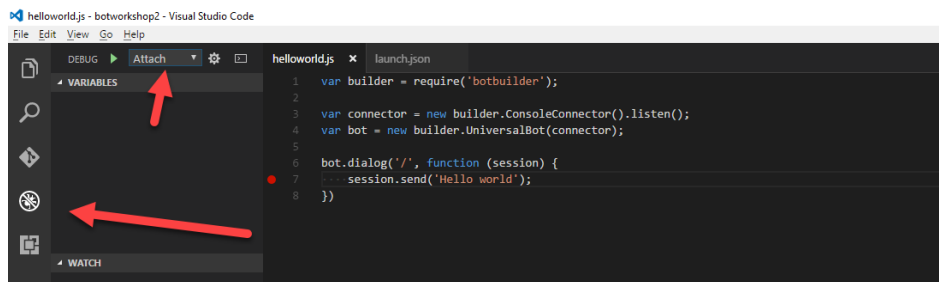
12. In VS Code, make sure you are still in the debug panel, in the debug dropdown make sure you select **attach** (to attach to the process we just started that is running on port 5858) and click on the green debug arrow to run it.

## Creating a bot using the Microsoft Bot Framework

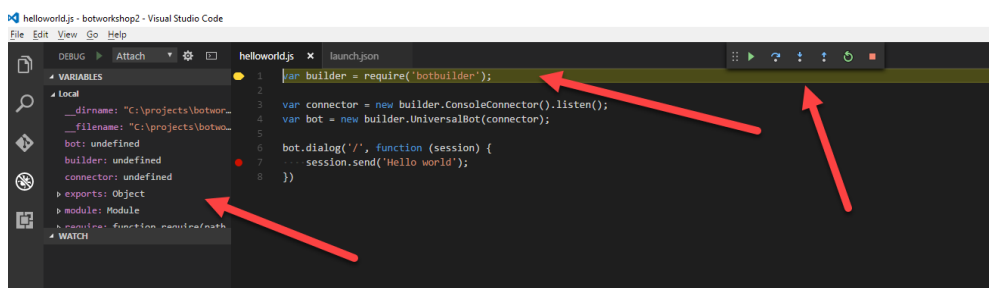
### Node.js Hands-on Labs

Page 15 of 41

#### Detailed Steps



13. This will attach to the process and the debug will be stopped on the first line of the program (that's what using **--debug-brk** does as opposed to just **--debug**) you will be able to view all the local variables on the left, step through the code (Function keys or buttons on top) and inspect variables by hovering over them. You can hit F5 to run the program (or the green arrow on the top bar).



Spend time debugging and looking around in this simple example so you can debug more complex ones later.

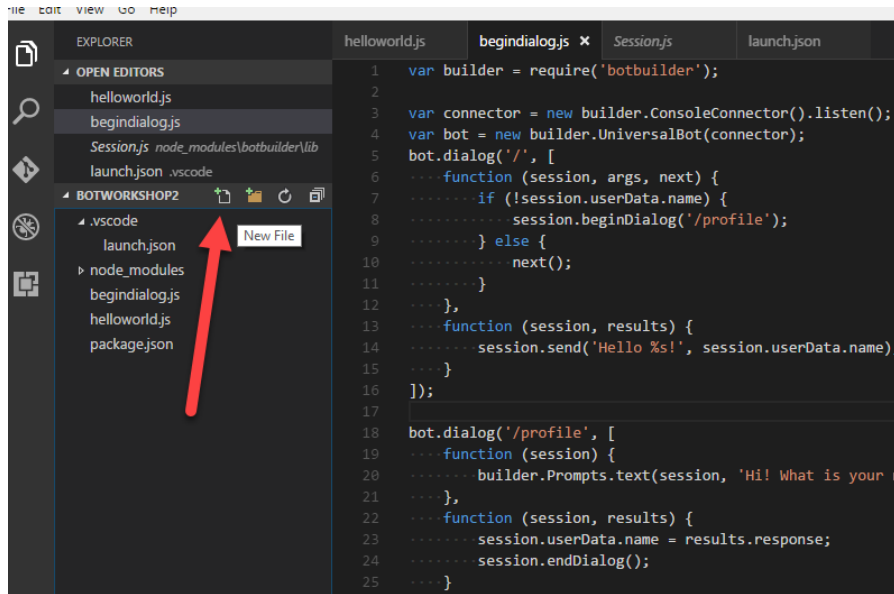
That is the end of exercise one. Now that we have everything set up and a simple bot running we will talk about how to handle greater complexity when working with your bot.

## Exercise 2: Using prompts in a bot

In this exercise we will create a very simple bot so that you can get used to using different prompts in a waterfall, and collecting data.

### Detailed Steps

1. Open up Visual Studio Code in the same folder (project) we started earlier. Add a new file by clicking on the new file icon and name it **promptbot.js**



2. First we need to add the require for botbuilder, and to new up our connector and bot just like we have done before.  
Add the following code to the top of the **promptbot.js** file.

----- SNIP2-----

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

Next we need to add our dialog. It is one dialog, so let's look at the whole thing at once. (It makes it easier to copy/paste as well) Paste the following code directly below our bot variable (**var bot = new ...**) in **promptbot.js**

-----SNIP3-----

```
bot.dialog('/', [  
  function (session) {  
    builder.Prompts.text(session, "Hello... What's your name?");  
  },  
  function (session, results) {  
    session.userData.name = results.response;  
  }  
]);
```



## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 17 of 41

### Detailed Steps

```
        builder.Prompts.number(session, "Hi " + results.response + ",  
How many years have you been coding?");  
    },  
    function (session, results) {  
        session.userData.coding = results.response;  
        builder.Prompts.choice(session, "What language do you code Node  
using?", ["JavaScript", "CoffeeScript", "TypeScript"]);  
    },  
    function (session, results) {  
        session.userData.language = results.response.entity;  
        session.send("Got it... " + session.userData.name +  
            " you've been programming for " +  
session.userData.coding +  
            " years and use " + session.userData.language +  
".");  
    }  
]);
```

In this code we are creating a function array (between the []) so it will cascade from one question to the next. Within the waterfall we are using 3 different prompts (text, number, choice) and storing data for the user in **userData**. We then pull the data out for the final **session.send** message.

4. Now we can run the code. Go to your command prompt and type in the following.

```
~$ node --debug promptbot.js
```

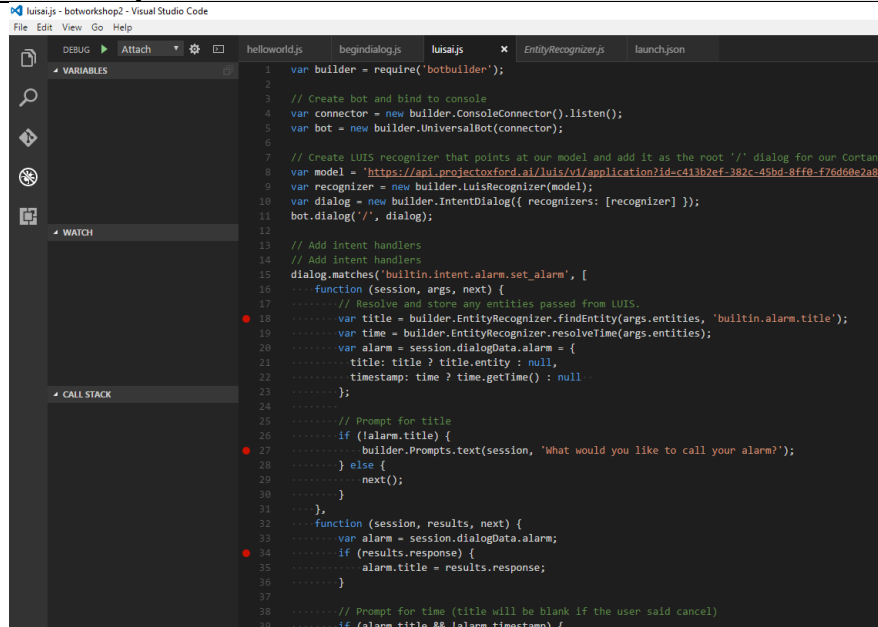
5. Next go to Visual Studio Code and place some breakpoints in your code so you can step through and inspect the objects and flow.

# Creating a bot using the Microsoft Bot Framework

## Node.js Hands-on Labs

Page 18 of 41

### Detailed Steps



```
1 var builder = require('botbuilder');
2
3 // Create bot and bind to console
4 var connector = new builder.ConsoleConnector().listen();
5 var bot = new builder.UniversalBot(connector);
6
7 // Create LUIS recognizer that points at our model and add it as the root '/' dialog for our Cortana
8 var model = 'https://api.projectoxford.ai/luis/v1/application?id=c413b2ef-382c-45bd-8ff0-f76d6e2a821';
9 var recognizer = new builder.LuisRecognizer(model);
10 var dialog = new builder.IntentDialog({ recognizers: [recognizer] });
11 bot.dialog('/', dialog);
12
13 // Add intent handlers
14 // Add intent handlers
15 dialog.matches('builtin.intent.alarm.set_alarm', [
16   function (session, args, next) {
17     // Resolve and store any entities passed from LUIS.
18     var title = builder.EntityRecognizer.findEntity(args.entities, 'builtin.alarm.title');
19     var time = builder.EntityRecognizer.resolveTime(args.entities);
20     var alarm = session.dialogData.alarm = {
21       title: title ? title.entity : null,
22       timestamp: time ? time.getTime() : null
23     };
24
25     // Prompt for title
26     if (!alarm.title) {
27       builder.Prompts.text(session, 'What would you like to call your alarm?');
28     } else {
29       next();
30     }
31   },
32   function (session, results, next) {
33     var alarm = session.dialogData.alarm;
34     if (results.response) {
35       alarm.title = results.response;
36     }
37
38     // Prompt for time (title will be blank if the user said cancel)
39     if (!alarm.title || !alarm.timestamp) {
```

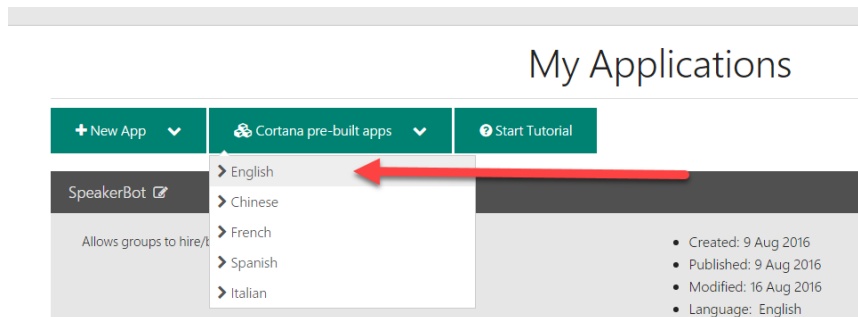
6. Step through the code as we had before to watch as things are running.

### Exercise 3: Using Intent Dialogs (LUIS)

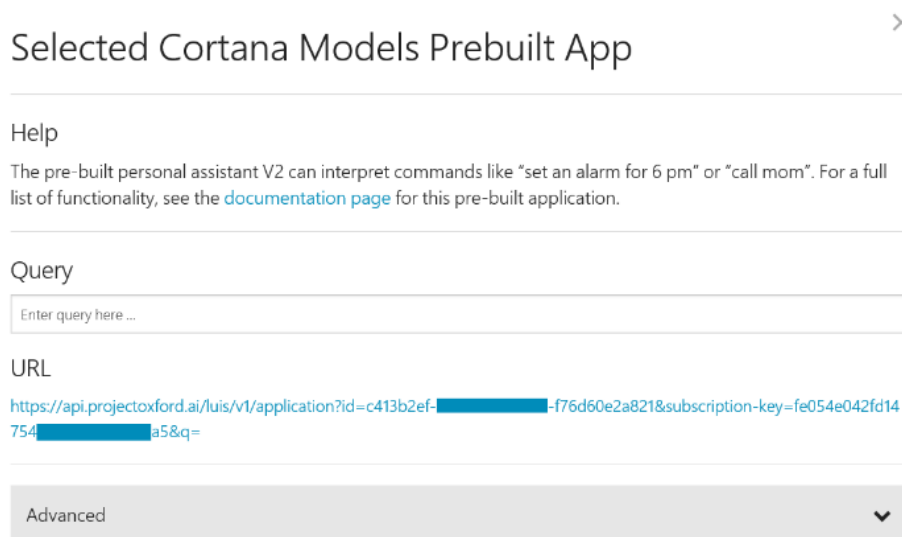
In this exercise we will create a LUIS Model. As discussed in the talk, we can create our own domain specific models, but for this HOL we will use Cortana Prebuilt App. The advantage is that we can quickly see the power of Natural Language in our bot without the training. This will create a personal assistant. To see all that this prebuilt model can do see the documentation here: <https://www.luis.ai/Help/Index#PreBuiltApp>

#### Detailed Steps

1. Sign on to <http://www.LUIS.ai>. You should have set this up in the first exercise, if not go back to the first section.
2. From your dashboard Select the **Cortana pre-built apps → English**



3. Once you click on your language, a Model dialog will pop up. **Copy the URL** from this screen, we will need it for our application.

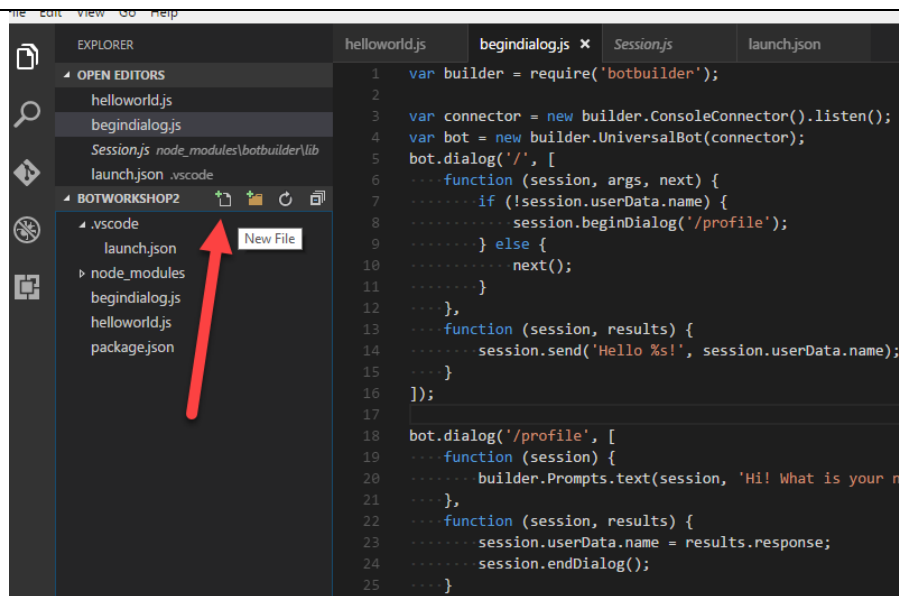


4. Open up Visual Studio Code and in your project, create a file called luisai.js

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 20 of 41



5. Add the following code to the **luisai.js** file.

-----SNIP4-----

```
var builder = require('botbuilder');

// Create bot and bind to console
var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);

// Create LUIS recognizer that points at our model and add it as the
root '/' dialog for our Cortana Bot.
var model = '<your models url>';
var recognizer = new builder.LuisRecognizer(model);
var dialog = new builder.IntentDialog({ recognizers: [recognizer] });
bot.dialog('/', dialog);

// Add intent handlers
dialog.matches('builtin.intent.alarm.set_alarm',
builder.DialogAction.send('Creating Alarm'));
dialog.matches('builtin.intent.alarm.delete_alarm',
builder.DialogAction.send('Deleting Alarm'));
dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't
understand. I can only create & delete alarms."));
```

6. Replace '**<your models url>**' with the url that we copied in step 3
7. Move over to your console and run the following

```
~$ node luisai.js
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 21 of 41

8. You can ask it to create or delete alarms for you. Try different ways of saying it and notice that you can ask it the same thing in many different ways and it understands (NLP).

```
C:\projects\botworkshop>
λ node luisai.js
help
I'm sorry I didn't understand. I can only create & delete alarms.
create an alarm for 5 minutes from now
Creating Alarm
what alarms do I have set
I'm sorry I didn't understand. I can only create & delete alarms.
do an alarm for 10 oclock
Creating Alarm
11pm alarm please
```

Right now we are just gathering intent, we are not actually performing any actions except for printing to the screen things like **‘Creating and Alarm’** or **‘Deleting an Alarm’**. Let’s add some functionality. This will allow us to inspect the intents coming back from LUIS and respond to them. We will start with the **.set\_alarm intent**

So replace the following code:

Replace this whole section

```
13 // Add intent handlers
14 dialog.matches('builtin.intent.alarm.set_alarm', builder.DialogAction.send('Creating Alarm'));
15 dialog.matches('builtin.intent.alarm.delete_alarm', builder.DialogAction.send('Deleting Alarm'));
16 dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't understand. I can only create & delete alarms."));
17
```

With this code:

-----SNIP5-----

```
// Add intent handlers
dialog.matches('builtin.intent.alarm.set_alarm', [
  function (session, args, next) {
    // Resolve and store any entities passed from LUIS.
    var title = builder.EntityRecognizer.findEntity(args.entities, 'builtin.alarm.title');
    var time = builder.EntityRecognizer.resolveTime(args.entities);
    var alarm = session.dialogData.alarm = {
      title: title ? title.entity : null,
      timestamp: time ? time.getTime() : null
    };

    // Prompt for title
    if (!alarm.title) {
      builder.Prompts.text(session, 'What would you like to call your alarm?');
    } else {
      next();
    }
  }
]);
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 22 of 41

```
    }
  },
  function (session, results, next) {
    var alarm = session.dialogData.alarm;
    if (results.response) {
      alarm.title = results.response;
    }

    // Prompt for time (title will be blank if the user said cancel)
    if (alarm.title && !alarm.timestamp) {
      builder.Prompts.time(session, 'What time would you like to set the alarm
for?');
    } else {
      next();
    }
  },
  function (session, results) {
    var alarm = session.dialogData.alarm;
    if (results.response) {
      var time = builder.EntityRecognizer.resolveTime([results.response]);
      alarm.timestamp = time ? time.getTime() : null;
    }

    // Set the alarm (if title or timestamp is blank the user said cancel)
    if (alarm.title && alarm.timestamp) {
      // Save address of who to notify and write to scheduler.
      alarm.address = session.message.address;
      alarms[alarm.title] = alarm;

      // Send confirmation to user
      var date = new Date(alarm.timestamp);
      var isAM = date.getHours() < 12;
      session.send('Creating alarm named "%s" for %d/%d/%d %d:%02d%s',
        alarm.title,
        date.getMonth() + 1, date.getDate(), date.getFullYear(),
        isAM ? date.getHours() : date.getHours() - 12, date.getMinutes(), isAM ?
'am' : 'pm');
    } else {
      session.send('Ok... no problem.');
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 23 of 41

In the previous section of code, we are using a number of the techniques we have discussed. We are using **intent matching, text prompts, time prompts, and a waterfall**. Next we need to add an intent dialog for deleting an alarm.

Paste the following code **below** the last section in **luisai.js**.

-----SNIP6-----

```
dialog.matches('builtin.intent.alarm.delete_alarm', [
  function (session, args, next) {
    // Resolve entities passed from LUIS.
    var title;
    var entity = builder.EntityRecognizer.findEntity(args.entities,
    'builtin.alarm.title');
    if (entity) {
      // Verify its in our set of alarms.
      title = builder.EntityRecognizer.findBestMatch(alarms,
      entity.entity);
    }

    // Prompt for alarm name
    if (!title) {
      builder.Prompts.choice(session, 'Which alarm would you like
to delete?', alarms);
    } else {
      next({ response: title });
    }
  },
  function (session, results) {
    // If response is null the user canceled the task
    if (results.response) {
      delete alarms[results.response.entity];
      session.send("Deleted the '%s' alarm.",
results.response.entity);
    } else {
      session.send('Ok... no problem.');
```

As you can see, it is very similar to the add alarm section with the addition of using the **choice prompt**. Now we need to add two more pieces to make it complete. We need to add back our default match section (one line of code) and a very simple implementation of an alarm. Paste the following code at the **bottom** of the **luisai.js** file.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 24 of 41

-----SNIP-----

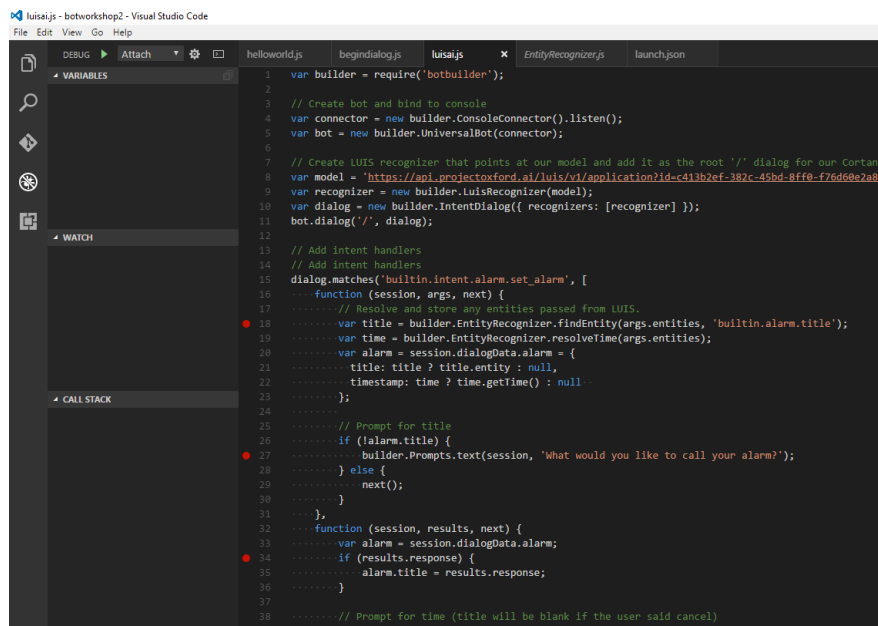
```
dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't
understand. I can only create & delete alarms.));

// Very simple alarm scheduler
var alarms = {};
setInterval(function () {
    var now = new Date().getTime();
    for (var key in alarms) {
        var alarm = alarms[key];
        if (now >= alarm.timestamp) {
            var msg = new builder.Message()
                .address(alarm.address)
                .text("Here's your '%s' alarm.", alarm.title);
            bot.send(msg);
            delete alarms[key];
        }
    }
}, 15000);
```

9. Now we can run the code. Go to your command prompt and type in the following.

```
~$ node --debug luisai.js
```

10. Next go to Visual Studio Code and place some breakpoints in your code so you can step through and inspect the entities, as they are being set.



11. Step through the code as we had before to watch as things are running.



### Exercise 4: Connecting to Skype and Webchat

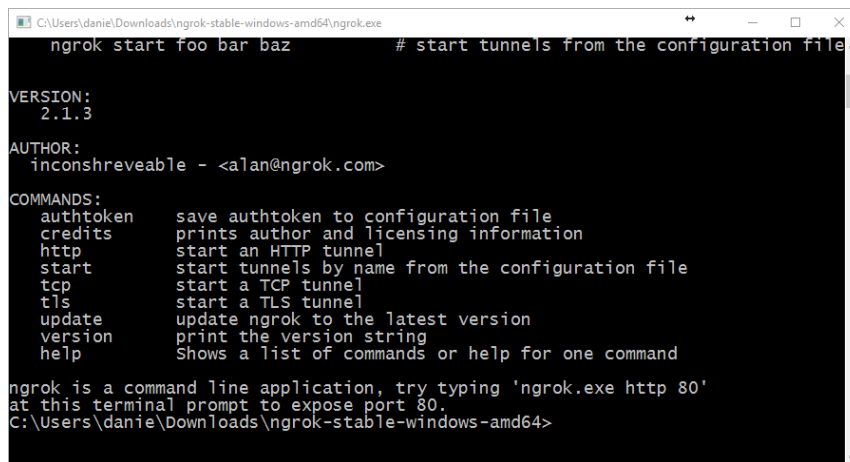
In this exercise we are going to connect your bot to a Skype and a Webchat channel. In doing so, we ARE NOT going to be hosting our bot in the cloud but running them from our laptops. We are doing this for 3 reasons:

- 1) It takes out some of the complexity of the workshop by not having to troubleshoot uploading and running a node app in your cloud of choice.
- 2) We don't have to walk 30 + people through setting up a cloud account.
- 3) It is valuable to use this technique to troubleshoot your bot prior to production so it is a valuable skill.

The first thing we need to do is to run NGrok. You should have downloaded NGrok from <https://ngrok.com/> as it states on the website, NGrok allows you to **create "Secure tunnels to localhost"**. This means we can make the bot framework think that our bot is hosted in the cloud by having an internet accessible URL for the bot hosted on our laptop.

#### Detailed Steps

1. Download and unzip Ngrok on your computer
2. Once unzipped double-click on the **ngrok.exe** file. This will open its own command window.



```
C:\Users\danie\Downloads\ngrok-stable-windows-amd64\ngrok.exe
ngrok start foo bar baz # start tunnels from the configuration file

VERSION:
  2.1.3

AUTHOR:
  inconshreveable - <alan@ngrok.com>

COMMANDS:
  authtoken  save authtoken to configuration file
  credits    prints author and licensing information
  http       start an HTTP tunnel
  start      start tunnels by name from the configuration file
  tcp        start a TCP tunnel
  tls        start a TLS tunnel
  update     update ngrok to the latest version
  version    print the version string
  help       Shows a list of commands or help for one command

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\danie\Downloads\ngrok-stable-windows-amd64>
```

3. At the prompt in this window type the following.

```
~$ ngrok http 3978
```

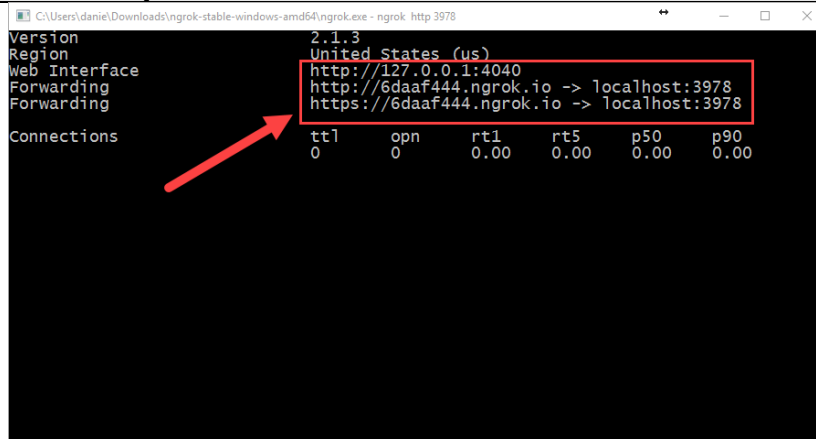
You should see the following in your command window.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 26 of 41

### Detailed Steps



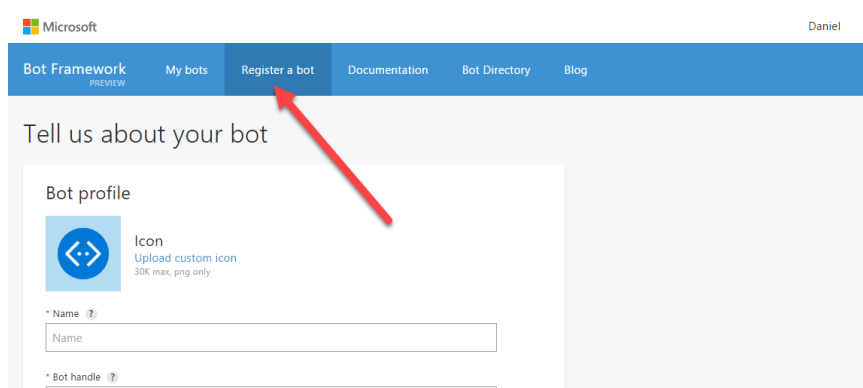
```
C:\Users\daniel\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 3978
Version                2.1.3
Region                 United States (us)
Web Interface          http://127.0.0.1:4040
Forwarding              http://6daaf444.ngrok.io -> localhost:3978
                       https://6daaf444.ngrok.io -> localhost:3978
Connections
  ttl    opn    rt1    rt5    p50    p90
    0     0     0.00  0.00  0.00  0.00
```

We care about two things in this window.

1. The forwarding URLs (both http and https) that will be our external address for our bot to reach our local machine.
2. The Web interface address. We will use this to track traffic coming into this port for our bot.

**Leave this running.** We will need this for the rest of this walkthrough.

4. Next we need to set up our bot on the BotFramework page. Open a browser and go to <http://BotFramework.com>. You should have already set up an account at the beginning of this HOL.
5. If you are not already signed in, sign in with the MS account you created earlier. When you are signed in click on the Register a bot in the menu.



6. Fill out the bot registration form. We will go through it section by section and touch on the important fields.

**Name:** <Name of your bot> Self-explanatory

**Bot Handle:** <your handle here> this will be used in the C# SDK when referencing your bot (not in Node.js SDK)

**Description:** Self-explanatory

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 27 of 41

### Detailed Steps

#### Bot profile



Icon

Upload custom icon  
30K max, png only

\* Name ?

DanielLuisBot

\* Bot handle ?

danielluisbot

\* Description ?

My workshop LUIS bot

7. This next section is only two fields but many steps. The first box is the Messaging endpoint. If you were hosting your bot in the cloud, then this would be the address of the site that is hosting it. Something like `http://DanielSpeakerBot.com/api.messages` but since we are hosting it locally we need to use the address that Ngrok gave us when we used it.

```
C:\Users\danie\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 3978
Version                2.1.3
Region                 United States (us)
Web Interface          http://127.0.0.1:4040
Forwarding              http://6daaf444.ngrok.io -> localhost:3978
Forwarding              https://6daaf444.ngrok.io -> localhost:3978
Connections
  ttl    opn    rt1    rt5    p50    p90
   0     0     0.00   0.00   0.00   0.00
```

append the address from Ngrok with `/api/messages`

#### Configuration

Messaging endpoint ?

`https://6daaf444.ngrok.io/api/messages`

\* Microsoft App ID ?

Microsoft App ID from the Microsoft App registration portal

Create Microsoft App ID and password

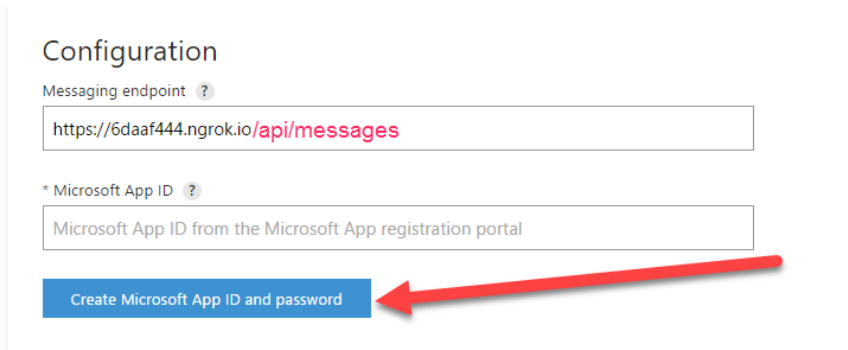
## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 28 of 41

### Detailed Steps

Next you need to create an AppID and password for your bot. click on the “**Create Microsoft App ID and password**” button.



When you do this a new page will pop up and give you an **App ID**.

### Generate App ID and password



**SAVE THIS APP ID SOMEWHERE. WE WILL NEED IT LATER.**

Next click on the **Generate a password to continue** button.

### Generate App ID and password



This will pop up a modal dialog with your 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.



## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 29 of 41

### Detailed Steps

This is the only time it will be shown.

**SAVE THIS PASSWORD ID SOMEWHERE. WE WILL NEED IT LATER**

Click **ok** to continue.

Next click on the **Finish and go back to Bot Framework** button to continue.

### Generate App ID and password

App name  
DanielLuisBot

App ID  
2f29ce2c...

Password [Learn More](#)  
Mzq... [Delete](#)

[Finish and go back to Bot Framework](#)

8. In the next section you need to fill out the publisher information (YOU). For the publisher name, make it whatever you would like since we are just testing. When you create a real bot, you can change this. For privacy and terms of use just put the url of your blog or any url. Once published these will need to point to real documents but for now they do not need to.

### Publisher profile

\* Publisher name  
Chicago Island Group

\* Publisher Email  
daniel.egan@live.com

\* Privacy statement  
<http://www.TheSociableGeek.com>

\* Terms of Use  
<http://TheSociableGeek.com>

Bot website  
URL to bot's website

Hashtags ?  
Hashtags

Languages ?  
en

Default Conversation Language ?  
Default Conversation Language

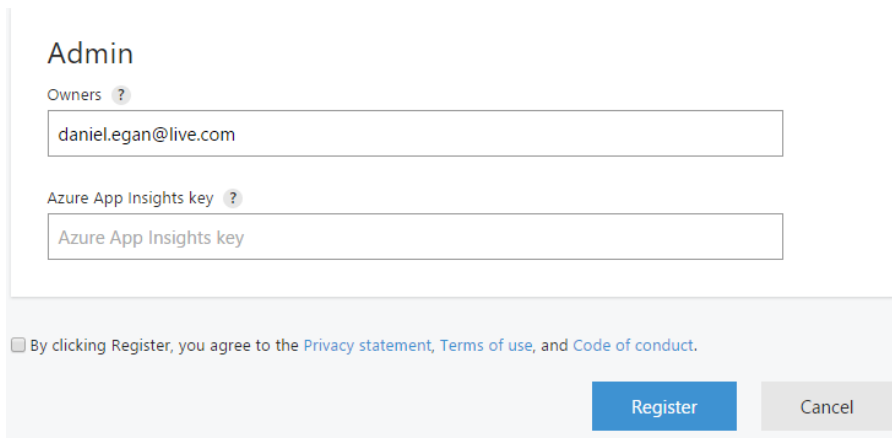
## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 30 of 41

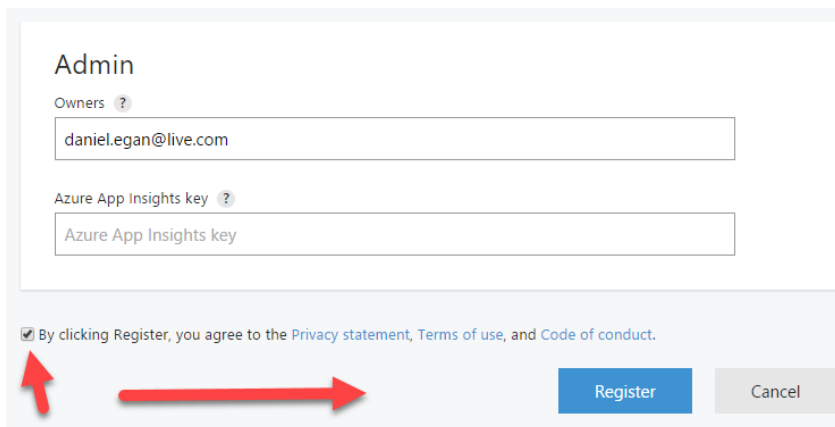
### Detailed Steps

9. In the final section, we do not need to add anything. (although in the future, setting up and using an App insights key will give you a bunch of great reporting)



The screenshot shows the 'Admin' registration page. It has two input fields: 'Owners' with the value 'daniel.egan@live.com' and 'Azure App Insights key' which is empty. Below the fields is a checkbox labeled 'By clicking Register, you agree to the Privacy statement, Terms of use, and Code of conduct.' which is currently unchecked. At the bottom right are 'Register' and 'Cancel' buttons.

10. **Agree** to the terms and click **Register** to create your bot.



This screenshot is identical to the previous one, but the checkbox 'By clicking Register, you agree to the Privacy statement, Terms of use, and Code of conduct.' is now checked. A red arrow points to the checked checkbox, and another red arrow points from the checkbox towards the 'Register' button.

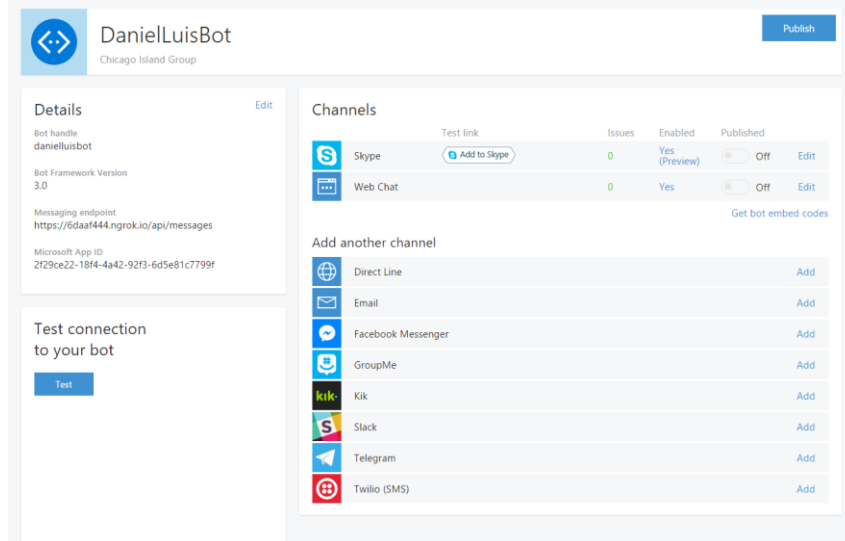
11. Leave this page up and running. We will need to come back here after we modify our bot to link it to the bot framework.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 31 of 41

### Detailed Steps



12. Bring up Visual Studio Code and open up the **luisai.js** file. We will be modifying the following section of that file.

```
3
4  var builder = require('botbuilder');
5
6  // Create bot and bind to console
7  var connector = new builder.ConsoleConnector().listen();
8  var bot = new builder.UniversalBot(connector);
9
```

Up until now, we have been using the ConsoleConnector. Now we are going to be using the ChatConnector. But first we need to add a node module that will help us with serving this app. It is called Restify.

13. Open up your console (Terminal, Powershell, etc..). Make sure you are in the botworkshop folder (or whatever you called it) and type the following.

```
~$ npm install restify -save
```

**Restify** is a node module that helps make rest calls easier.

14. Once that is done, open up the luisai.js file and add the following code underneath the **var builder = require('botbuilder');** code.

```
var restify = require('restify');
```

The code should look similar to this

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 32 of 41

### Detailed Steps

```
var builder = require('botbuilder');
var restify = require('restify');

// Create bot and bind to console
var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);
```

15. Since we will be working with the new connector. Delete the connector and the bot lines so we can recreate them.

```
var builder = require('botbuilder');
var restify = require('restify');

// Create bot and bind to console
var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);
```

With that gone, we need to set up the restify server. To do so, add the following lines under the restify require statement.

-----SNIP8-----

```
// Setup Restify Server
var server = restify.createServer();
server.listen(process.env.port || process.env.PORT || 3978, function ()
{
    console.log('%s listening to %s', server.name, server.url);
});
```

We are setting up a server that will look for an environment variable called PORT, if it does not find one, it will start on port **3978** (which is why we used that port for NGrok).

Next we want to create the new ChatConnector (instead of the ConsoleConnector). Add the following code directly under the last code you pasted.

-----SNIP9-----

```
// Create chat bot
var connector = new builder.ChatConnector({
    appId: process.env.MICROSOFT_APP_ID,
```



## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 33 of 41

### Detailed Steps

```
appPassword: process.env.MICROSOFT_APP_PASSWORD
});
```

Notice that the connector requires and **appId** and **appPassword**. These are what we saved when we create our bots on BotFramework.com.

Now we need to new up our bot and pass in our connector like we did before and set up where the post route is (/api/messages).

Add this code below the last code you pasted.

-----SNIP10-----

```
var bot = new builder.UniversalBot(connector);
server.post('/api/messages', connector.listen());
```

There is one last step. Adding our **appId** and **appPassword**. To protect them, you should **ALWAYS** put them in **Environment Variables** (or other safe place) . If you plan to put this in production **OR** plan to save this code in github or another repository, **DO NOT** do what we are going to do right now.

Modify your connector to add your **appId** and **appPassword**.

```
// Create chat bot
var connector = new builder.ChatConnector({
  ...appId: '2f29ce22-...',
  ...appPassword: 'MzqY...',
});
```

16. Now lets run out bot. Go to your console and run the following command.

```
~$ node luisai.js
```

You should see the restify server running on the port we specified.

```
C:\projects\botworkshop2> node luisai.js
restify listening to http://[::]:3978
```

17. We can test the connection by going to your bot on **BotFramework.com**

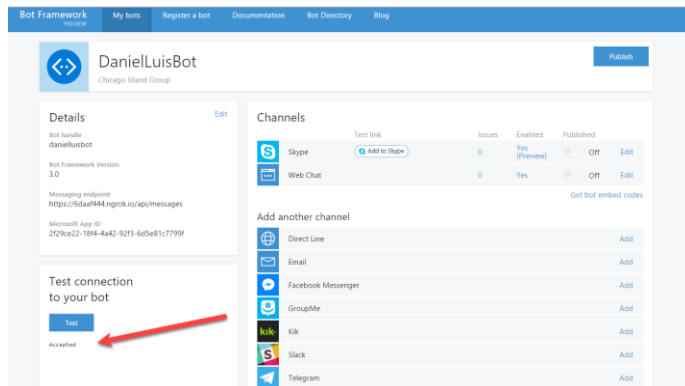
If everything works out fine when clicking the test button, you will get back an accepted message.

# Creating a bot using the Microsoft Bot Framework

## Node.js Hands-on Labs

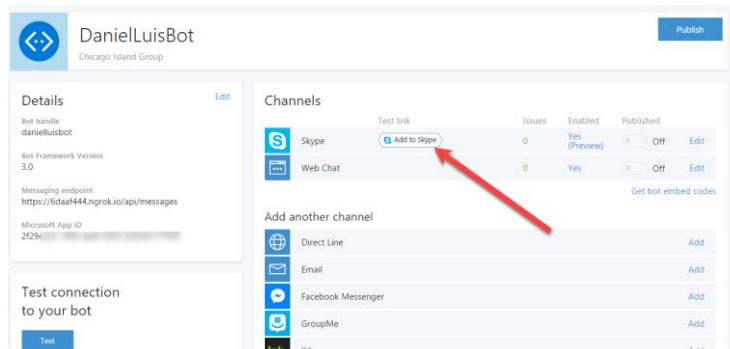
Page 34 of 41

### Detailed Steps

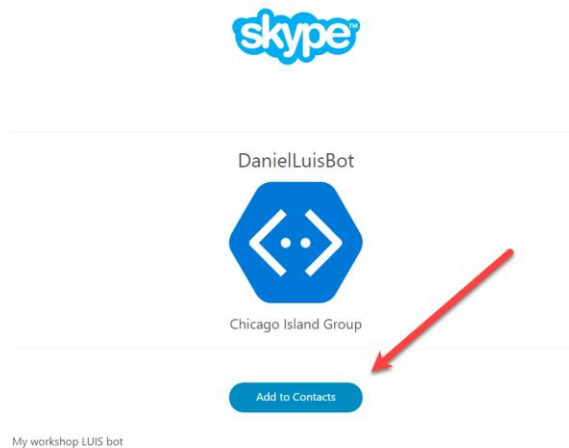


18. Now we can test Skype (you must have Skype installed).

In your portal, click on the Add to Skype Button



When the windows pops up, click on add you Contacts.



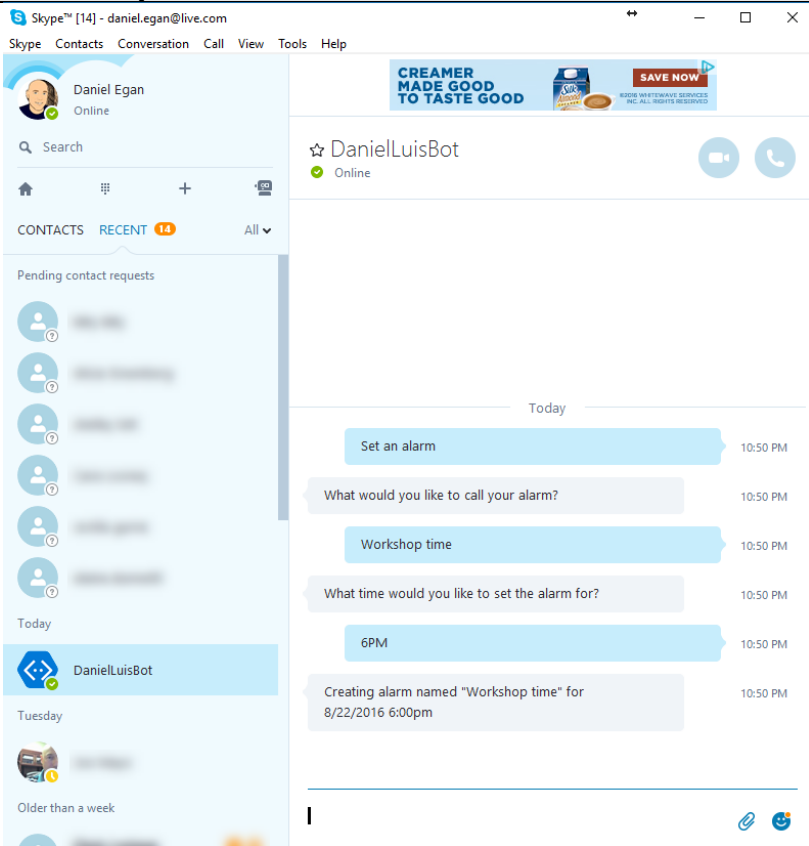
Once it is added to your contacts, you can chat away.

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 35 of 41

### Detailed Steps



The screenshot shows a Skype chat window titled "Skype™ [14] - daniel.egan@live.com". The contact list on the left includes "Daniel Egan" and "DanielLuisBot". The chat history with "DanielLuisBot" shows the following messages:

- Set an alarm (10:50 PM)
- What would you like to call your alarm? (10:50 PM)
- Workshop time (10:50 PM)
- What time would you like to set the alarm for? (10:50 PM)
- 6PM (10:50 PM)
- Creating alarm named "Workshop time" for 8/22/2016 6:00pm (10:50 PM)

If you don't have Skype installed you can test it out by using a webchat window.

The first thing we need to do is to create a page to host the webchat control.

**19.** Open up Visual Studio Code and add a file called index.html

-----SNIP11-----

```
<!doctype html>
<html>
  <head>
    <title>MyAppID</title>
  </head>
  <body>

  </body>
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 36 of 41

### Detailed Steps

```
</html>
```

20. Next, go to your bot on BotFramework.com and click on the Get bot embedded codes

**DanielLuisBot**  
Chicago Island Group

**Details**

Bot handle: danielluisbot

Bot Framework Version: 3.0

Messaging endpoint: https://6daaf444.ngrok.io/api/messages

Microsoft App ID: 2f21...

**Test connection to your bot**

Test

Accepted

**Channels**

	Test link	Issues	Enabled	Published	
Skype	Add to Skype	0	Yes (Preview)	Off	Edit
Web Chat		0	Yes	Off	Edit

[Get bot embed codes](#)



**Add another channel**

- Direct Line
- Email
- Facebook Messenger
- GroupMe
- Kik
- Slack

Click on the Web Chat Icon and copy the Embed code

**Bot embed code**

Choose a channel to get the embed code for it. Copy and paste the code to your website.

**Note:** you must manually place your secret into the iframe code (replacing "YOUR\_SECRET\_HERE") for the iframe to work. If you do this, you allow others to embed your bot into their pages. For information on securing your chat control, see the Bot Connector documentation.

**Embed code for**

```
<iframe src="https://webchat.botframework.com/embed/danielluisbot?s=YOUR_SECRET_HERE" style="height: 502px; max-height: 502px;" ></iframe>
```

Copy that embed code into the body section on index.html

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>MyAppID://title</title>
5   </head>
6   <body>
7     <div>
8       <iframe src="https://webchat.botframework.com/embed/danielluisbot?s=YOUR_SECRET_HERE" style="height: 502px; max-height: 502px;" ></iframe>
9     </div>
10  </body>
11 </html>
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 37 of 41

### Detailed Steps

As you can see in red (highlighted in yellow) You will need to replace the section that says YOUR\_SECRET\_HERE with the secret for your bot this is NOT your ID or Password.

#### Bot embed code

Choose a channel to get the embed code for it. Copy and paste the code to your website.



Note: you must manually place your secret into the iframe code (replacing "YOUR\_SECRET\_HERE" for the iframe to work). If you do this, you allow others to embed your bot into their pages. For information on securing your chat control, see the Bot Connector documentation.

Embed code for



```
<iframe src="https://webchat.botframework.com/embed/danielluisbot?s=YOUR_SECRET_HERE" style="height: 502px; max-height: 502px;"></iframe>
```

To get the secret for your bot. Go to your bots page and click on edit next to your Web Chat Channel.

Channels

	Test link	Issues	Enabled	Published	
Skype	<a href="#">Add to Skype</a>	0	Yes (Preview)	<input type="checkbox"/> Off	<a href="#">Edit</a>
Web Chat		0	Yes	<input type="checkbox"/> Off	<a href="#">Edit</a>

[Get bot embed codes](#)

Add another channel

Direct Line	<a href="#">Add</a>
Email	<a href="#">Add</a>
Facebook Messenger	<a href="#">Add</a>

Click on Regenerate Web Chat Secret

Copy the Secret

Click on I'm done...


## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 38 of 41

### Detailed Steps

Configure Web Chat



How to

Web Chat secret

Credentials have been validated.

Secret: No secret generated

Embed template: No secret generated

Regenerate Web Chat secret Deauthorize

Note: you must manually copy your secret into the iframe code (replacing 'YOUR\_SECRET\_HERE') for the iframe to work. If you do this, you allow others to embed your bot into their pages. For information on securing your chat control, see the [Bot Connector documentation](#).

☒ Enable this bot on Web Chat  
Enabling or disabling a channel doesn't affect its credentials.

I'm done configuring Web Chat >

Paste this secret into your `<iframe>` into the body section of `Index.html`

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>MyAppID</title>
5   </head>
6   <body>
7     <iframe src="https://webchat.botframework.com/embed/danielhuisbot?s=ldX48r9nVWJ." style="width: 100%; height: 100%; border: none;" >
8   </body>
9 </html>
```

Finally, to be able to access this page, you need to add a `server.get` to your `luisai.js` file.

Underneath the `server.post('/api/messages', connector.listen());` line, add the following code.

-----SNIP12-----

```
server.get('/', restify.serveStatic({
  directory: __dirname,
  default: '/index.html'
}));
```

## Creating a bot using the Microsoft Bot Framework

Node.js Hands-on Labs

Page 39 of 41

### Detailed Steps

This will route incoming get requests (in the browser) to our **index.html** page.

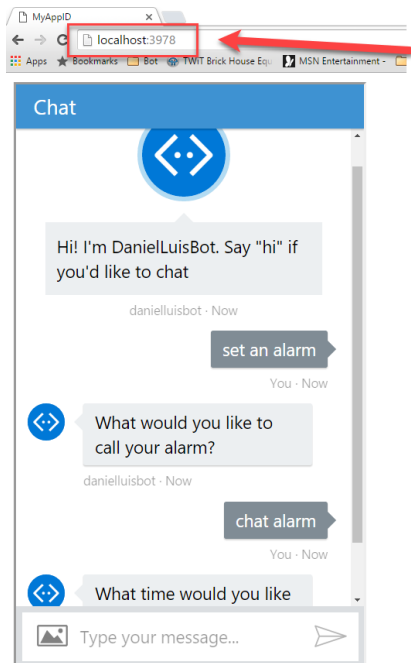
To test it out, go to your console and run the following command.

```
~$ node luisai.js
```

You should see the restify server running on the port we specified.

```
C:\projects\botworkshop2> node luisai.js
restify listening to http://[::]:3978
```

Next browse to <http://localhost:3978> in a browser



Now you can chat away.

---

## Additional Resources

---



# Copyright

---

Information in this document, including URL and other Internet Web site references, is subject to change without notice and is provided for informational purposes only. The entire risk of the use or results from the use of this document remains with the user, and Microsoft Corporation makes no warranties, either express or implied. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2009 Microsoft Corporation. All rights reserved.

Microsoft and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.