

Micro Design Centre Inc.

Developing Alexa Skills

Setting up a Local Development Environment
Version 1.0.2

Table of Contents

| | |
|---|----|
| Setting up a Local Development Environment..... | 2 |
| Installing Visual Studio Code | 2 |
| Installing nodejs..... | 4 |
| Installing the Alexa Skills Kit Command Line Interface (ASK CLI)..... | 5 |
| Bespoken Tools | 27 |
| Creating an Alexa Skill locally..... | 28 |
| Debugging the local Alexa Skill with Bespoken Tools | 35 |
| Summary | 61 |

Setting up a Local Development Environment

An Alexa Skill is made up of two parts, the Skill Interface and the Skill Service. The Skill Service usually lives up in the cloud on an Amazon Server in what Amazon calls a Lambda function.

We can code this Lambda Function using the browser-based editor that Amazon provides for creating Lambdas, but that is not really a development environment. There is no Source Control, Debugging Tools, Testing Tools etc. that we really need for our development efforts.

Alternately, a Local Development Environment can be setup where we can write/test/debug our code. Later then when the code is finished and fully debugged, we can simply deploy to Amazon so the code can be used by our Alexa Skill.

To create a Local Development Environment, we will be setting up the following items

- Visual Studio Code
- Node.js
- Alexa Skills Kit Command Line Interface (ASK CLI)
- Bespoken Tools

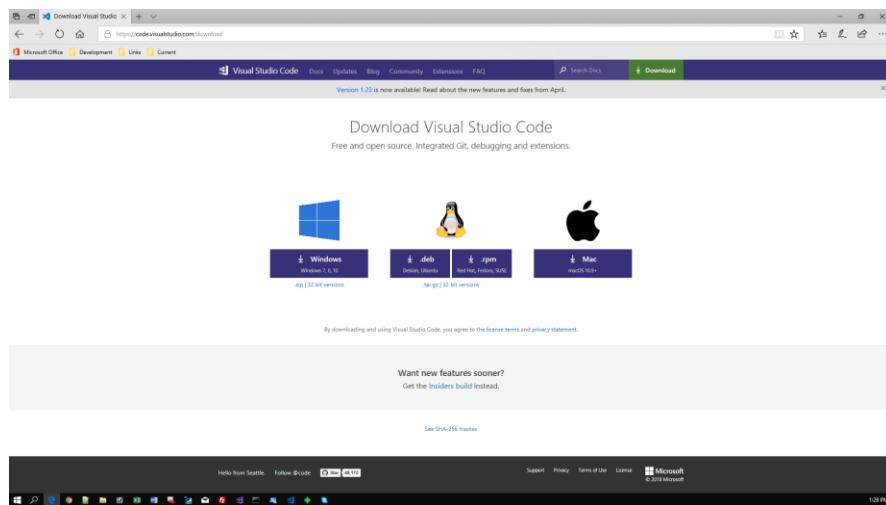
Installing Visual Studio Code

We will be using Visual Studio Code for locally coding our Alexa Skill Service. Over the past few years, Visual Studio code has become one of the better text editors and it is available from Microsoft for free.

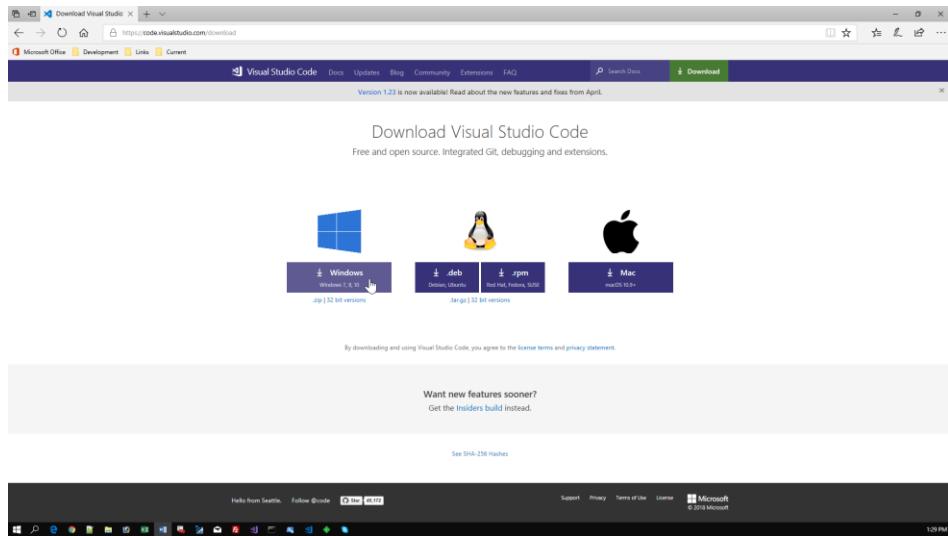
Open a browser and navigate to

<https://code.visualstudio.com/download>

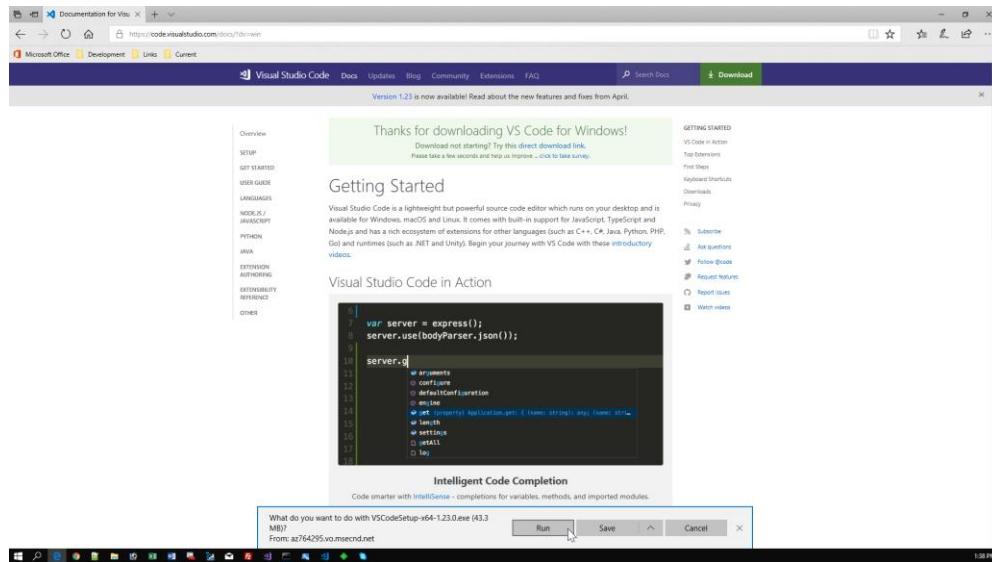
You will be taken to the following page



Click on the appropriate link for the Operating System you are running. Since I am running on a Windows PC, I will download the Windows version of Visual Studio Code.



You will be taken to the download page for Visual Studio Code. Click ‘Run’ in the resulting pop up as shown below

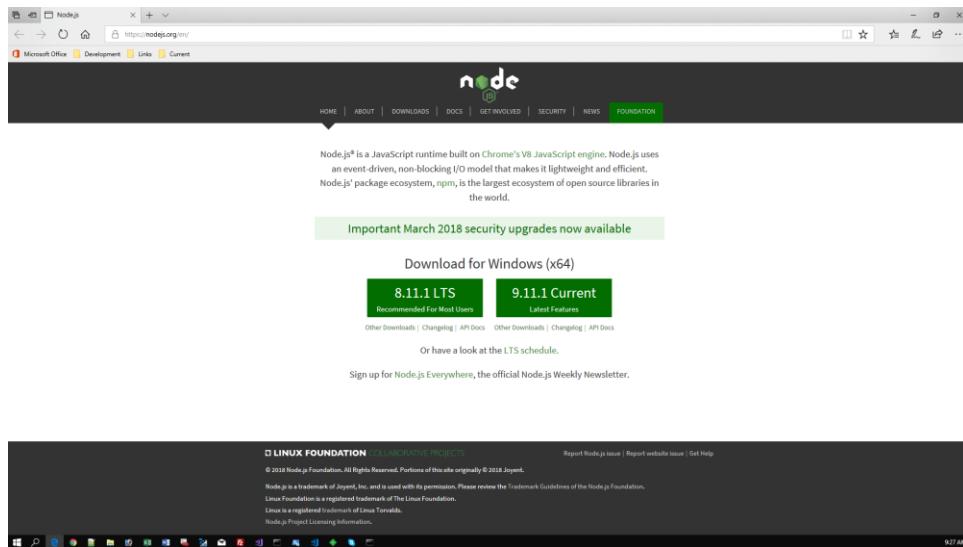


You will be presented with the Visual Studio Code Setup Wizard. Follow along with the steps in the Wizard and setup for Visual Studio Code will take place.

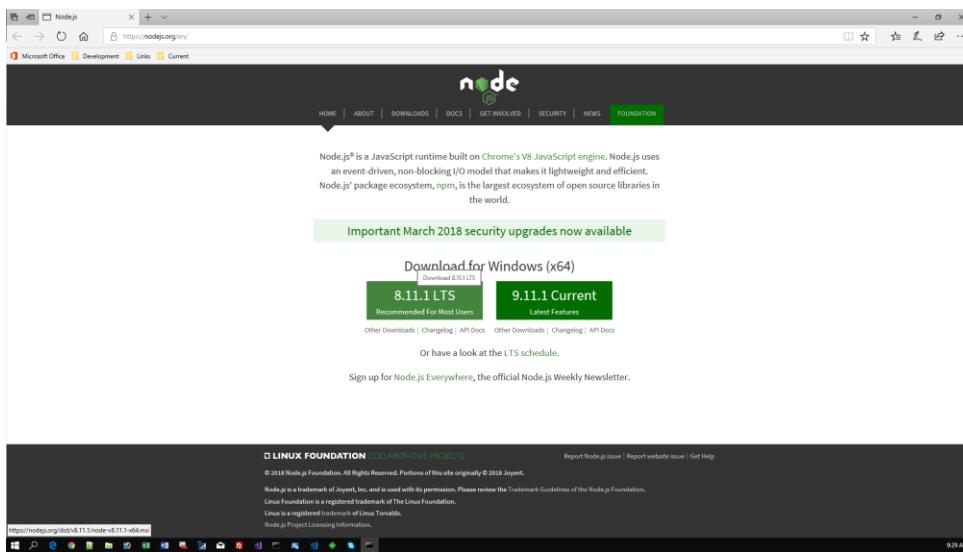
Installing nodejs

Since we will be creating the Skill Service using the Alexa Skills Kit SDK V2 for Node.js, we will need to have Node.js running on our local machine.

Open a browser and navigate to <https://nodejs.org> as shown below



Click on the Download button as shown below



Run the msi file and you will be presented with the Node JS Setup Wizard. Follow along with the steps in the Wizard and setup for Node JS will take place.

Installing the Alexa Skills Kit Command Line Interface (ASK CLI)

Once we have developed the Skill Service locally we will need to deploy the code up to the Lambda Service running in the Amazon Cloud. To make this as easy as possible, we will be installing and using the ‘Alexa Skills Kit Command Line Interface’ (ASK CLI for short).

However, before we install the ASK CLI we will need to setup credentials for a special AWS Account that the ASK CLI can use. When the ASK CLI does anything with AWS, it will do so using the special User Credentials we are setting up.

There are 5 Steps involved

- Create a new Security Policy
- Create a new User and assign Security Policy
- Setting up a Local AWS Profile
- Install the ASK CLI
- Associate the User to the ASK CLI

Step 1 - Create a new Security Policy

Open a browser and navigate to

<https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html>

You will be presented with the following page

The screenshot shows a Microsoft Edge browser window with the URL <https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html>. The page title is "Set Up Credentials for an Amazon Web Services (AWS) Account". On the left, there is a sidebar menu for the Alexa Skills Kit, including sections like "Build Skills with the Alexa Skills Kit", "Custom Skills", "Smart Home Skills", "Skills with Device Certification", "Skills for Business", "Capability Verification Reference", "Video Skills", "Game Skills", "ASK Skills", "APIs and Skill Management API", and "Set Up Credentials for an Amazon Web Services (AWS) Account". The main content area contains instructions for managing an Alexa skill through the ASK CLI if it uses AWS Lambda functions. It includes a code snippet for a JSON policy:

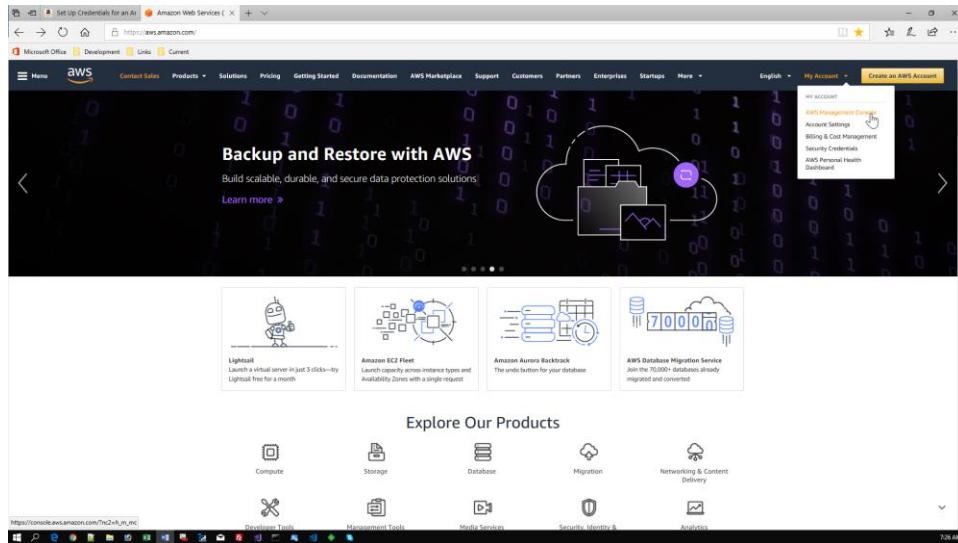
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:CreateFunction",
                "lambda:UpdateFunction",
                "lambda:GetFunction"
            ],
            "Resource": "arn:aws:lambda:*:*/skill/*"
        }
    ]
}
```

On this page you will find some JSON code that is the specifications for the new Policy that we will be creating. Make a copy of this JSON as we will need it in the next step.

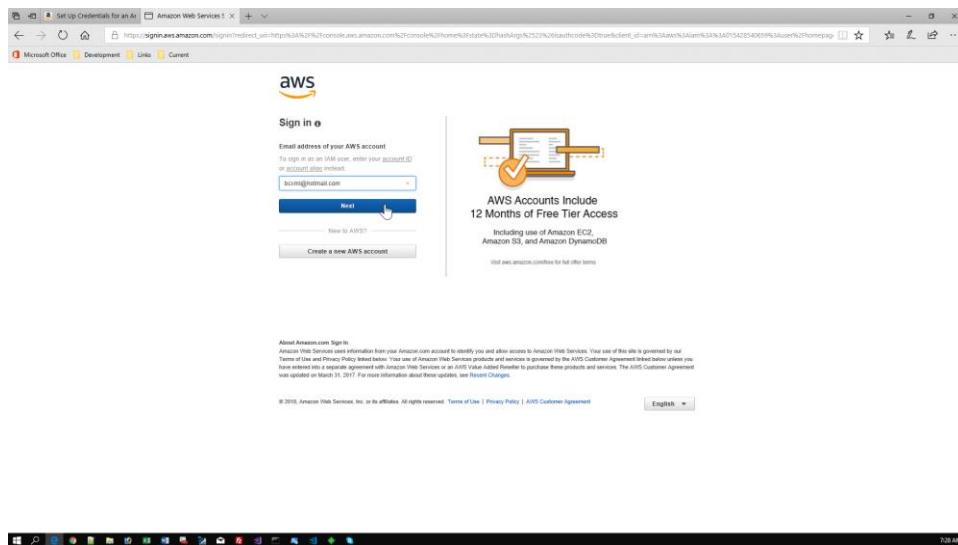
Next, navigate to the AWS home page located at

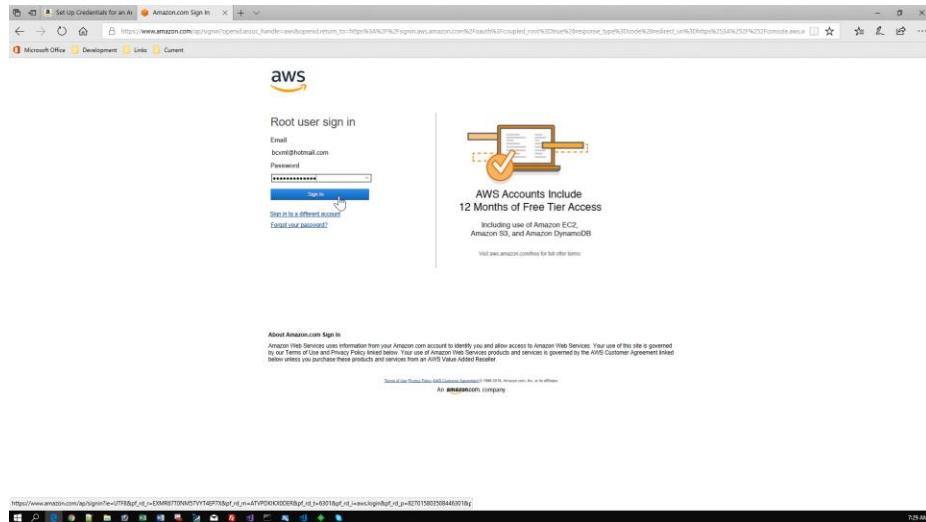
<https://aws.amazon.com>

Click on the ‘My Account’ link and select ‘AWS Console’ as shown below

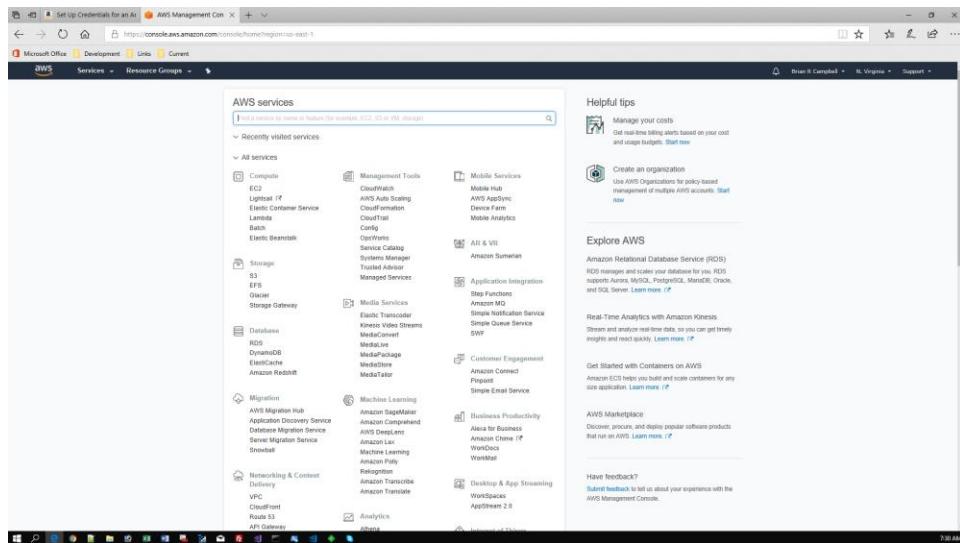


Enter the email address of your AWS Account and password as shown below

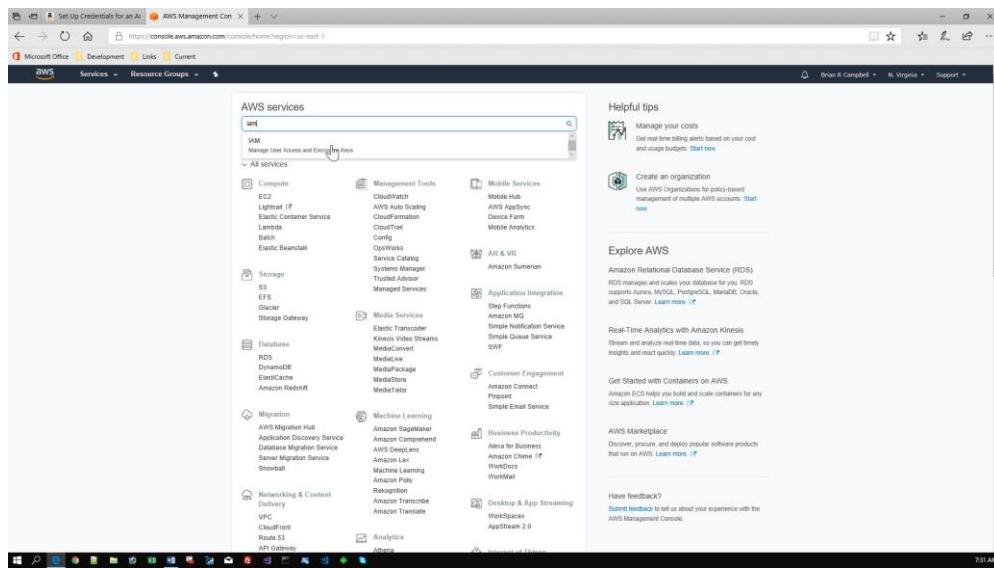




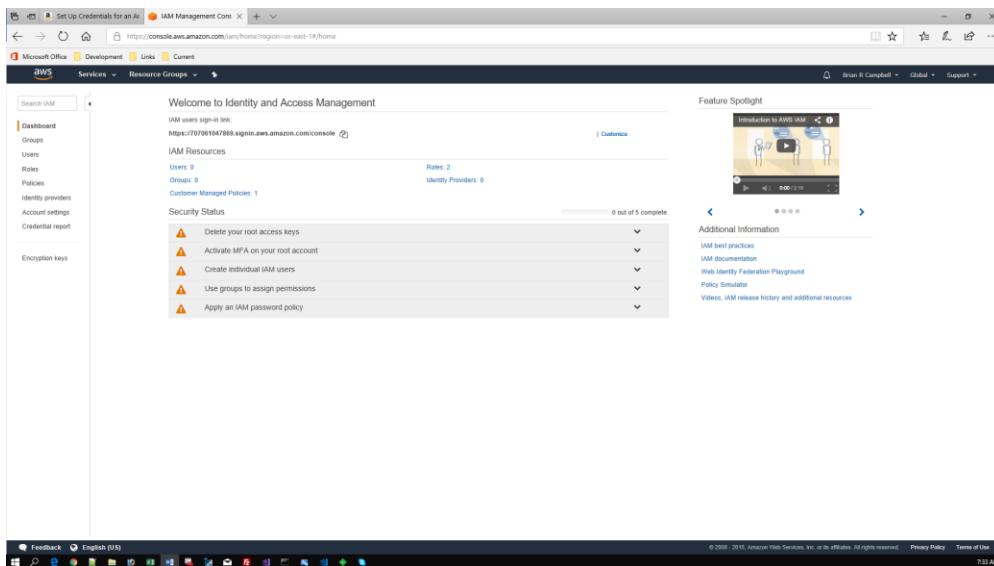
You will be shown the following screen



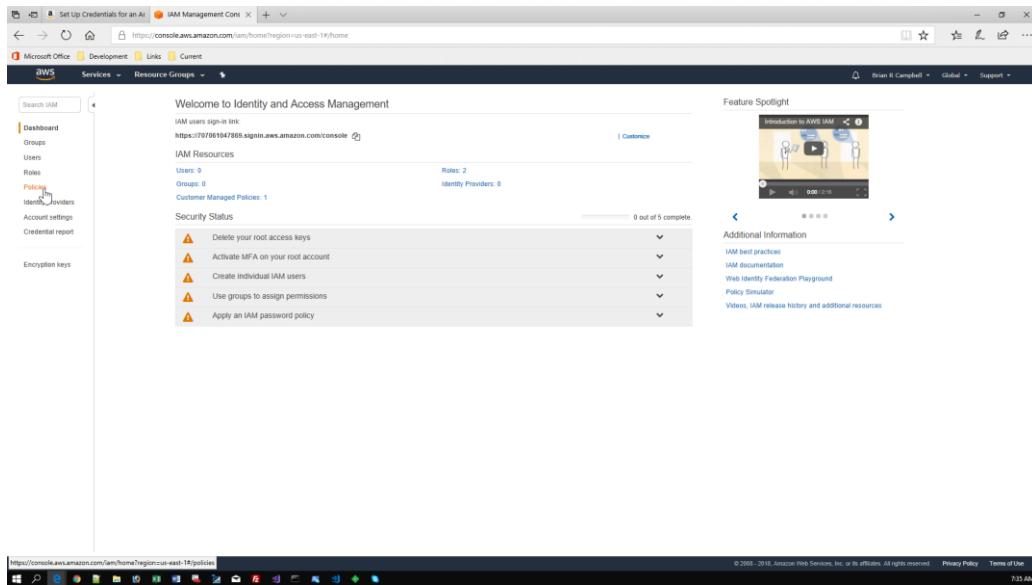
We are going to use the ‘Identity and Access Management’ (IAM) service to setup a new Policy and User. In the Search Bar, enter ‘IAM’ and as shown below



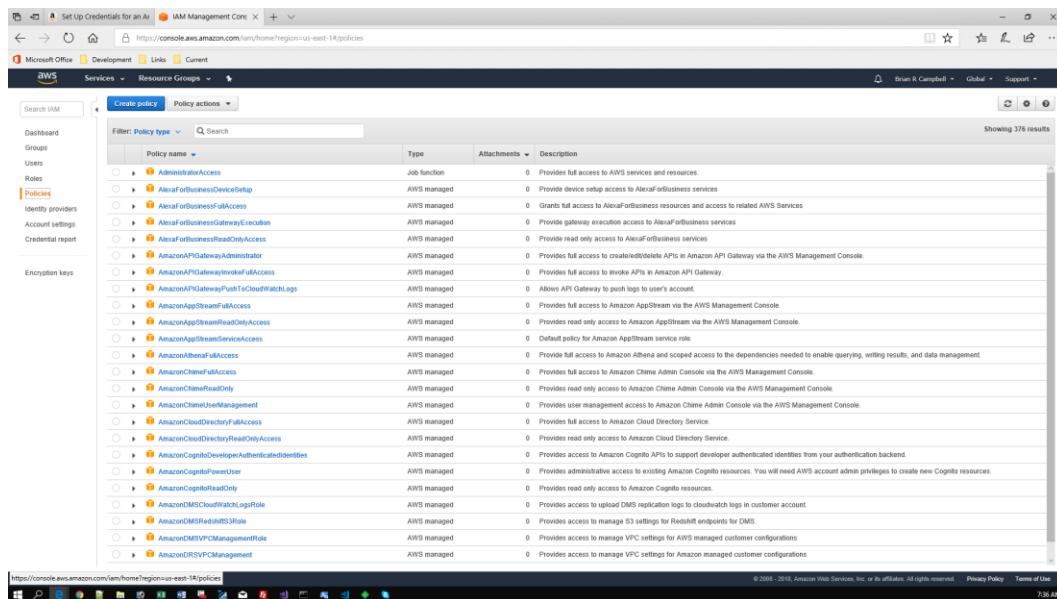
You will be taken to the following page



To setup a new Policy, click the ‘Policies’ link in the left-hand menu as shown below



You will be shown the following screen



Click on the ‘Create Policy’ button as shown below

The screenshot shows the AWS IAM Management Console with the URL <https://console.aws.amazon.com/iam/home?region=us-east-1#policies>. The left sidebar includes links for Dashboard, Groups, Users, Roles, Policies (which is selected), Identity providers, Account settings, Credential report, and Encryption keys. The main content area displays a table of policies with columns for Policy name, Type, Attachments, and Description. A search bar at the top has 'Create policy' highlighted. The status bar at the bottom shows the URL, copyright information, Privacy Policy, Terms of Use, and the time as 7:38 AM.

On the resulting screen, click on the ‘JSON’ tab as shown below

The screenshot shows the 'Create policy' screen with the URL <https://console.aws.amazon.com/iam/home?region=us-east-1#policies%7B%22tag%22%3D%22edit%22%7D%23tag-edit>. The 'Create policy' button is at the top. Below it is a note: 'A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more'. A validation error message is shown: 'This policy validation failed and might have errors converting to JSON - The policy must have at least one statement. For more information about the IAM policy grammar, see AWS IAM Policies.' The 'Visual editor' tab is active, but the 'JSON' tab is highlighted. The JSON code area contains the following:

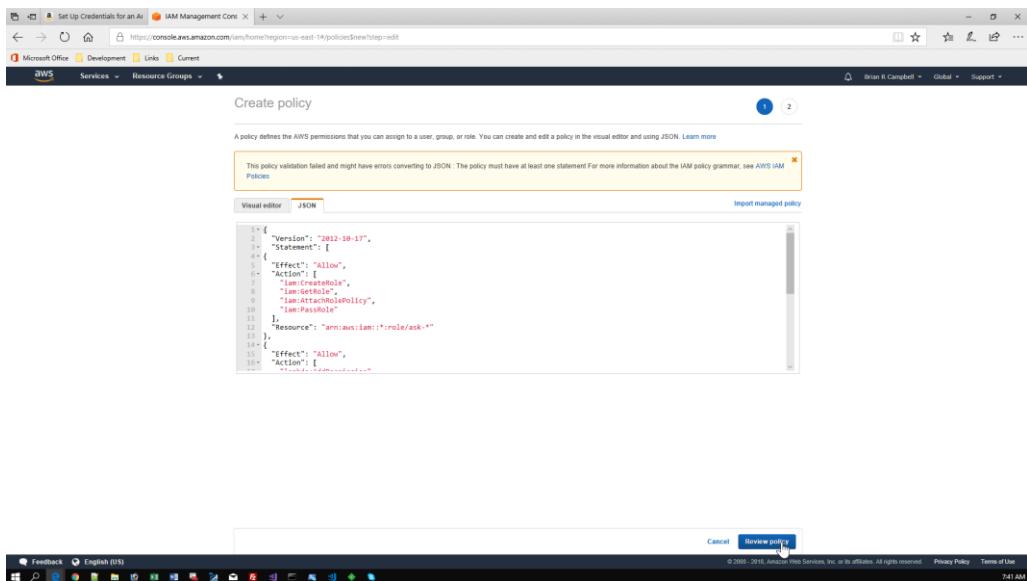
```

{
    "Version": "2012-10-17",
    "Statement": []
}

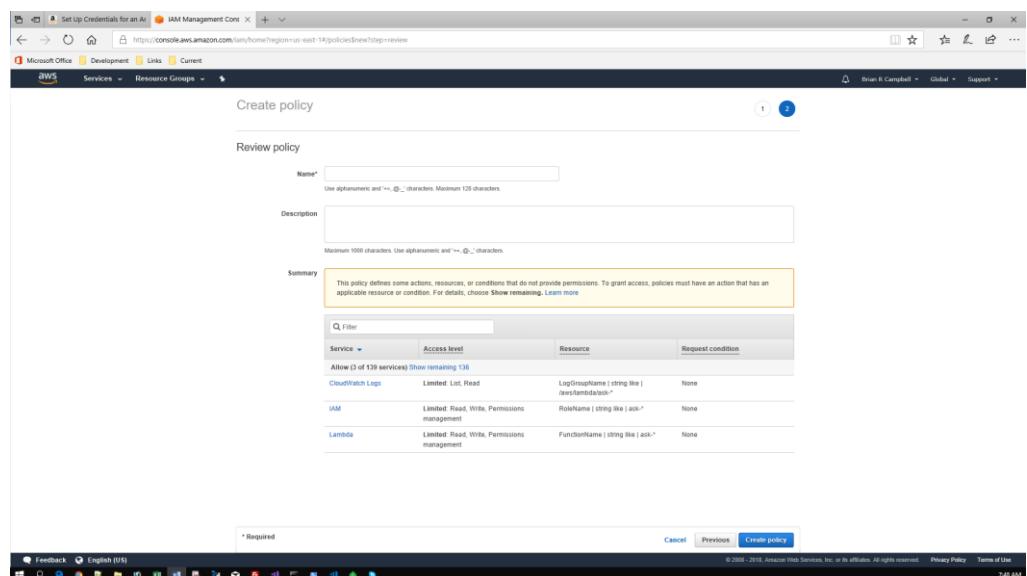
```

At the bottom right, there are 'Cancel' and 'Review policy' buttons. The status bar at the bottom shows the URL, copyright information, Privacy Policy, Terms of Use, and the time as 7:40 AM.

Delete the existing JSON and paste in the JSON that was copied earlier and then click the ‘Review Policy’ button as shown below



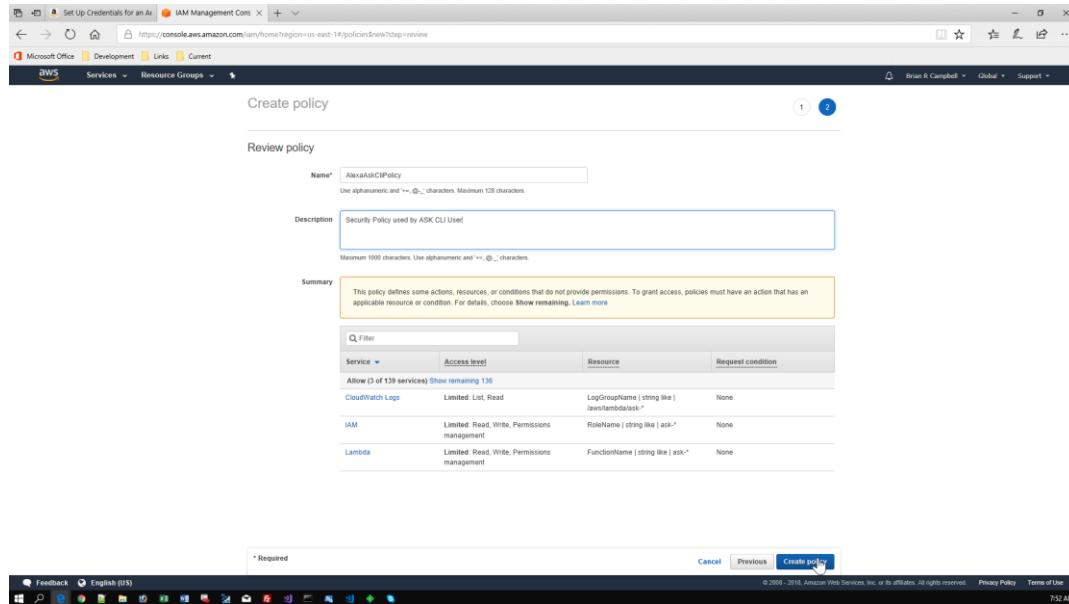
You will be presented with the following screen



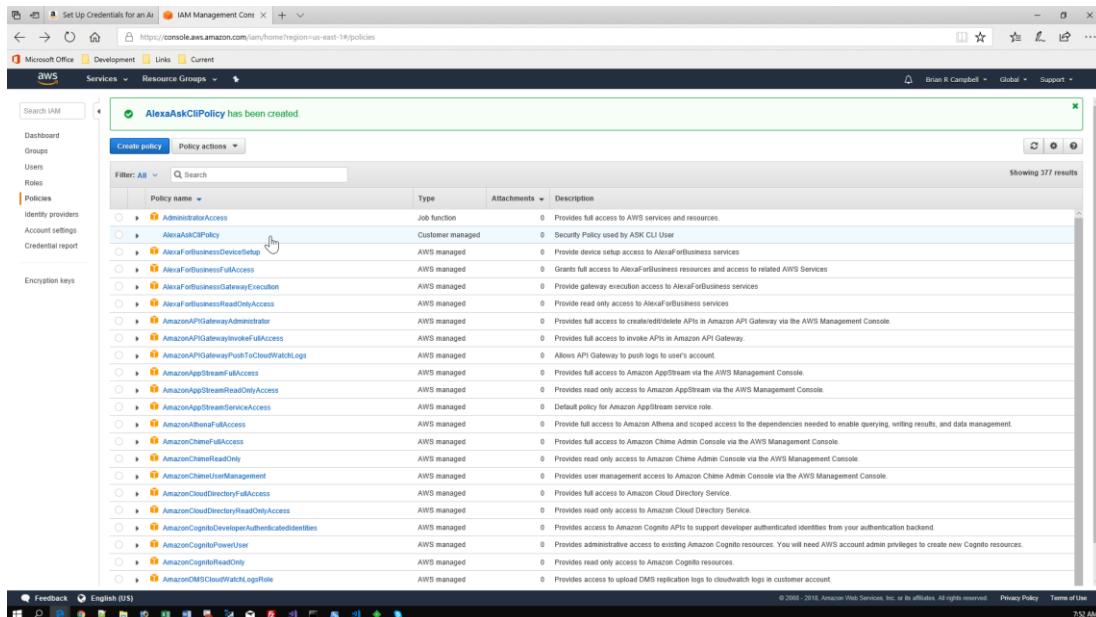
Fill in the Name and Description fields as follows

- Name - ‘AlexaAskCliPolicy’
- Description - ‘Security Policy used by ASK CLI User’

Finally, click the ‘Create Policy’ button as shown below

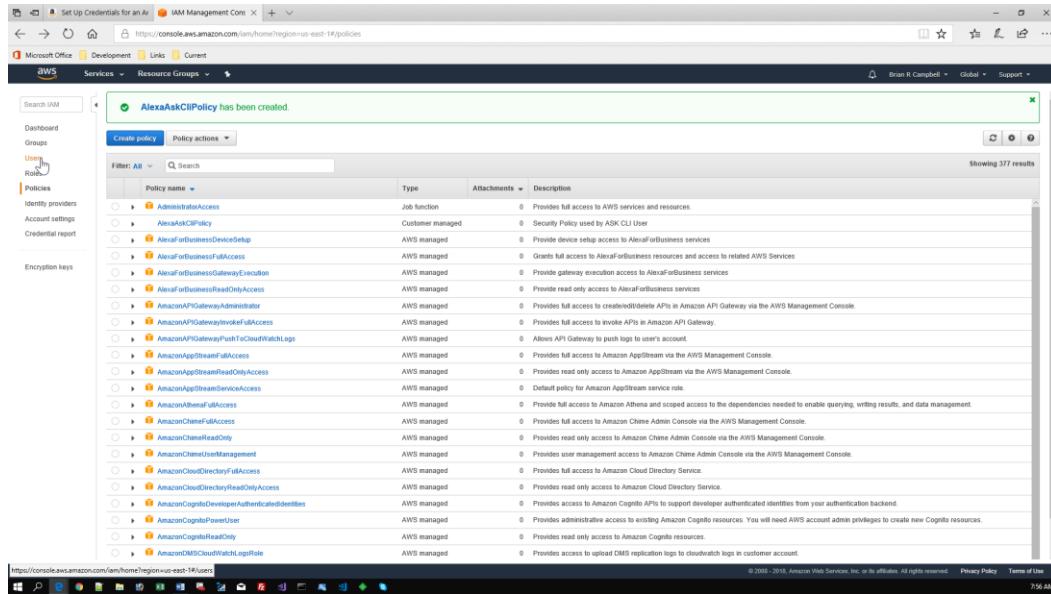


Back in the Policies screen you can see that the new Policy has been created



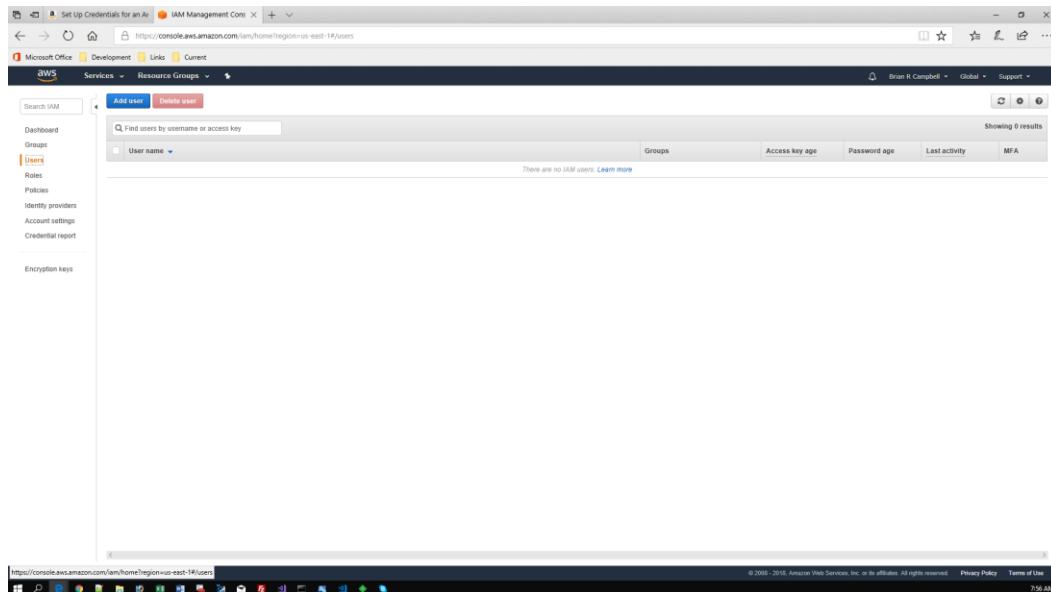
Step 2 - Create a new User and assign Security Policy

In the left-hand menu, click on the ‘Users’ link as shown below



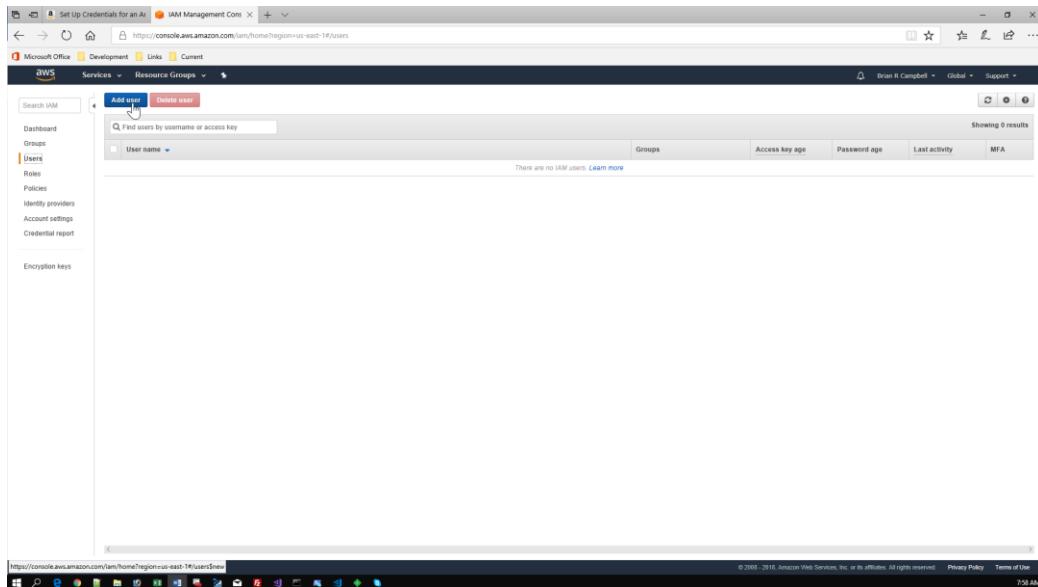
The screenshot shows the AWS IAM Management Console. The left sidebar is collapsed, and the main area displays a list of policies. At the top, a green banner says "AlexaAskCLIPolicy has been created." Below it, there are two tabs: "Create policy" and "Policy actions". A search bar is present. The list of policies is titled "Showing 377 results". The columns are "Policy name", "Type", "Attachments", and "Description". Some policies have icons indicating they are customer-managed or AWS managed.

You will be presented with the following screen



The screenshot shows the AWS IAM Management Console. The left sidebar is collapsed, and the main area displays a table for managing users. At the top, there are buttons for "Add user" and "Delete user". A search bar is present. The table has columns for "User name", "Groups", "Access key age", "Password age", "Last activity", and "MFA". A note at the bottom of the table says "There are no IAM users. Learn more". The status bar at the bottom indicates the URL as https://console.aws.amazon.com/iam/home?region=us-east-1#users.

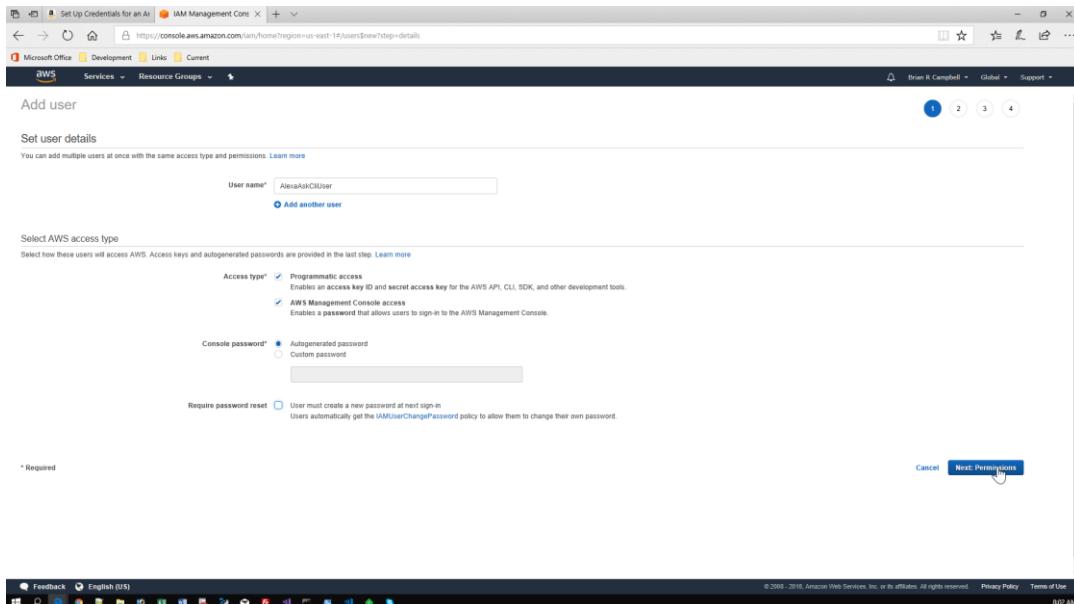
Click on the ‘Add User’ button as shown below



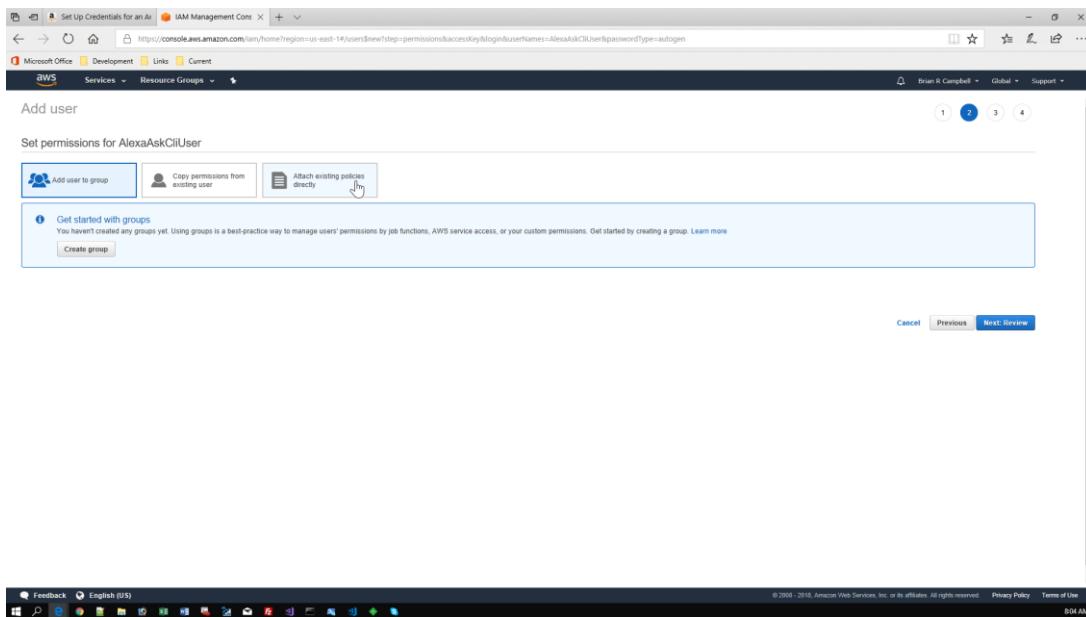
On the resulting screen, enter the following values

- User Name - ‘AlexaAskCliUser’
- Access Type - Programmatic Access - checked
- Access Type - AWS Management Console access - checked
- Console password - Autogenerate password
- Require password reset - unchecked

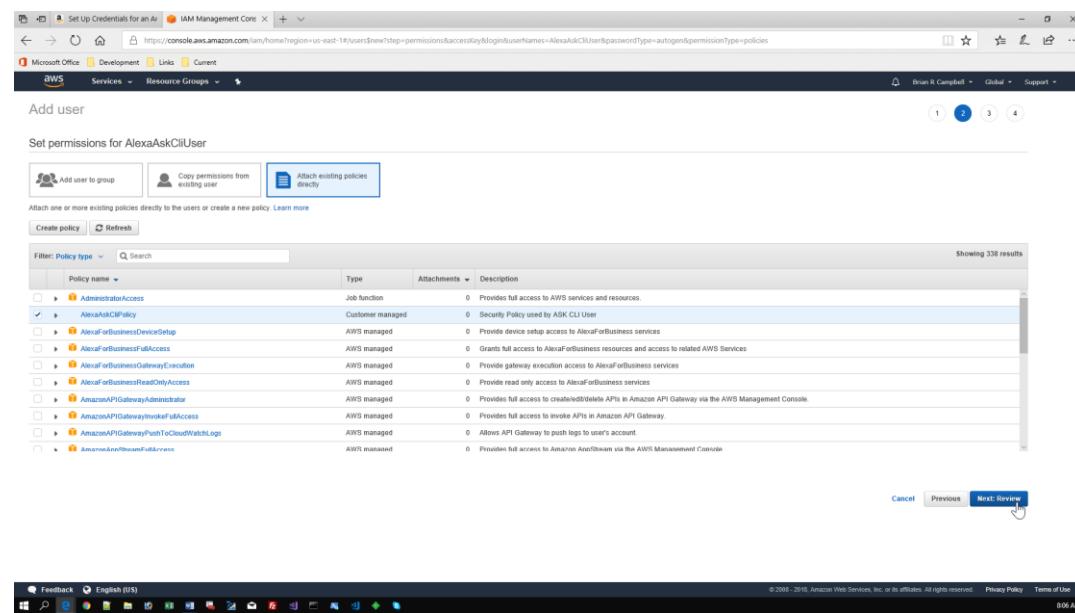
And then click on the ‘Next:Permissions’ button as shown below



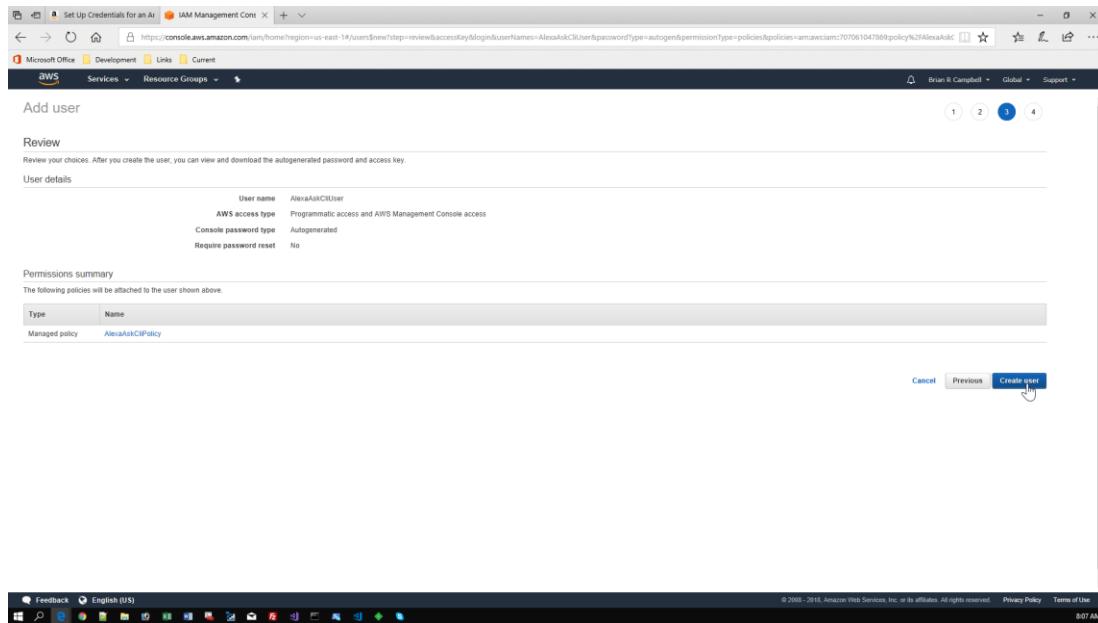
In the resulting screen, click on the large ‘Attach existing policies directly’ button as shown below



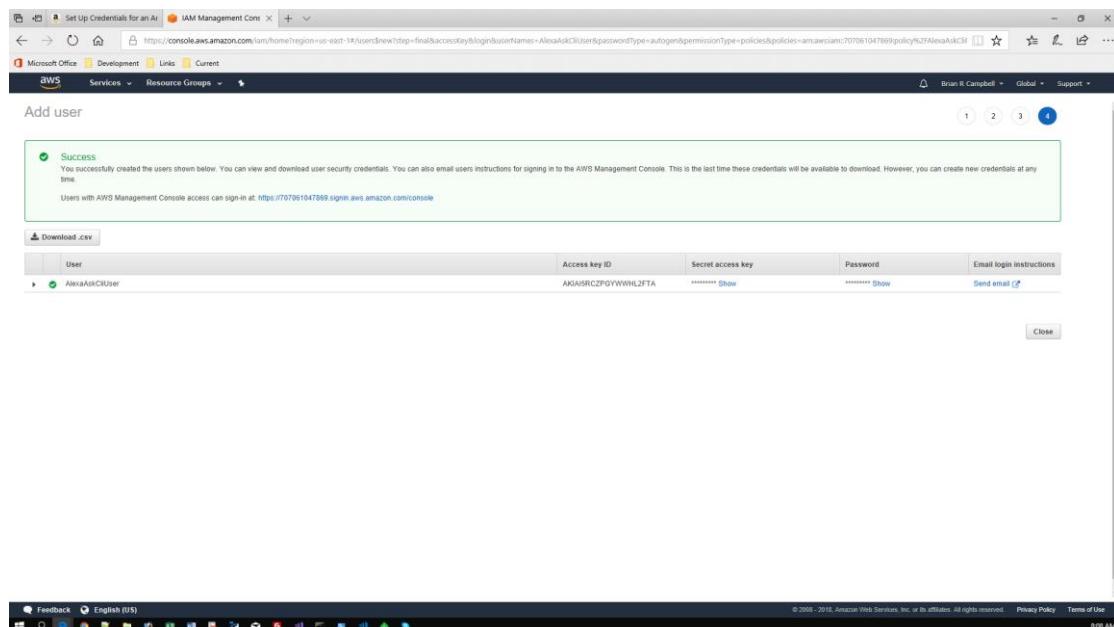
In the resulting screen, place a checkmark to the left of the ‘AlexaAskCliPolicy’ that we previously created and then click on the ‘Next:Review’ button as shown below



Finally, click the ‘Create User’ button as shown below

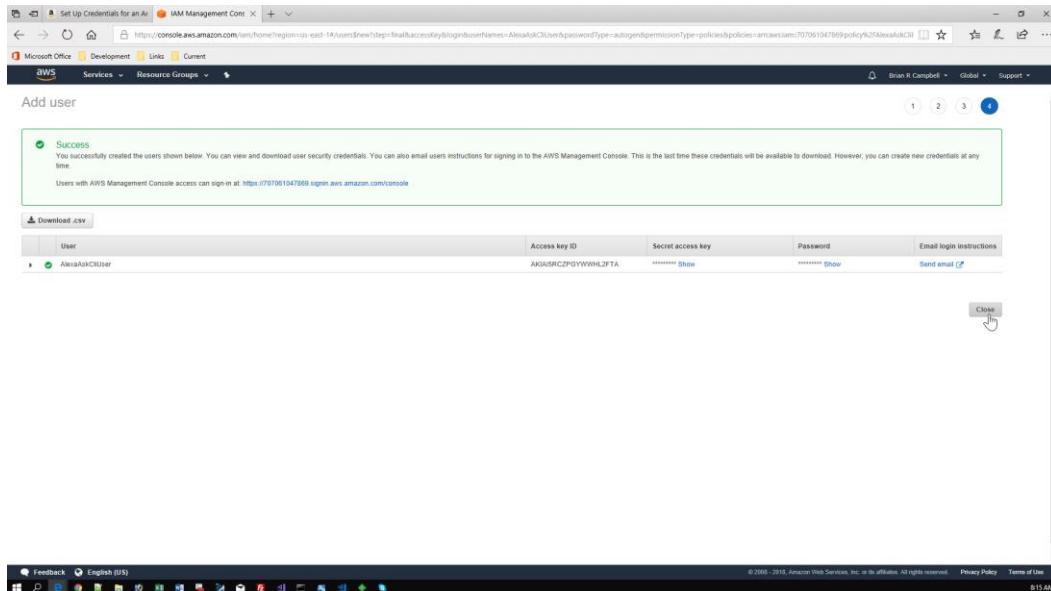


You will be shown the following screen which lists the new User that was created

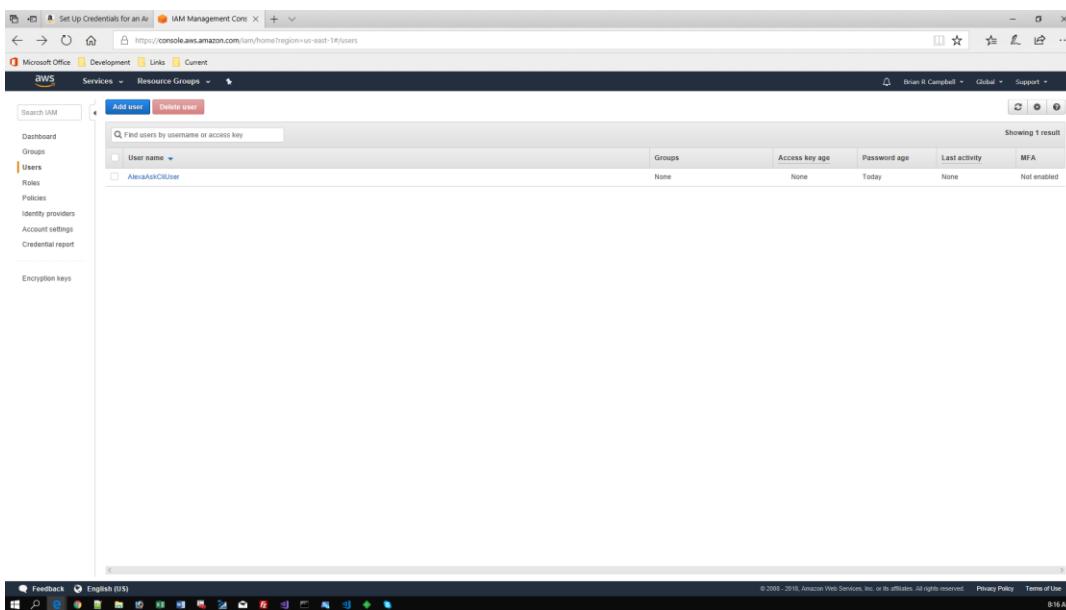


Now at this point we are going to need 2 pieces of information from this screen, the ‘Access key ID’ and the ‘Secret access key’. Copy these 2 values, as they will be needed in the next step

Once you have copied those two values, click on the ‘Close’ button as shown below



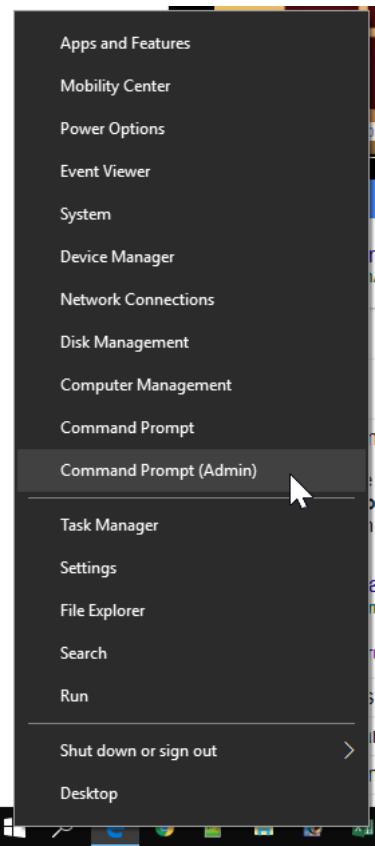
You will be taken back to the Users screen where you can see the newly created User as shown below



Step 3 - Installing the Alex Skills Kit CLI

For the next few steps we will need to open a command prompt. However, the command prompt must be opened as an Administrator.

To open a CMD Window as an Administrator right click on the Start Button and select CMD Prompt (Admin) as shown below



The next step is to install the Alexa Skills Kit Command Line Interface or ASK CLI for short. The ASK CLI is a tool that is used to manage Alexa Skills and related AWS Lambda functions.

With ASK CLI, you have access to the Skill Management API, which allows you to manage Alexa skills programmatically from the command line.

In the Command Prompt enter

```
npm install -g ask-cli
```

The above installs Alexa Skills Kit CLI (ASK CLI) globally onto the machine.

Note: if you get an error message that states

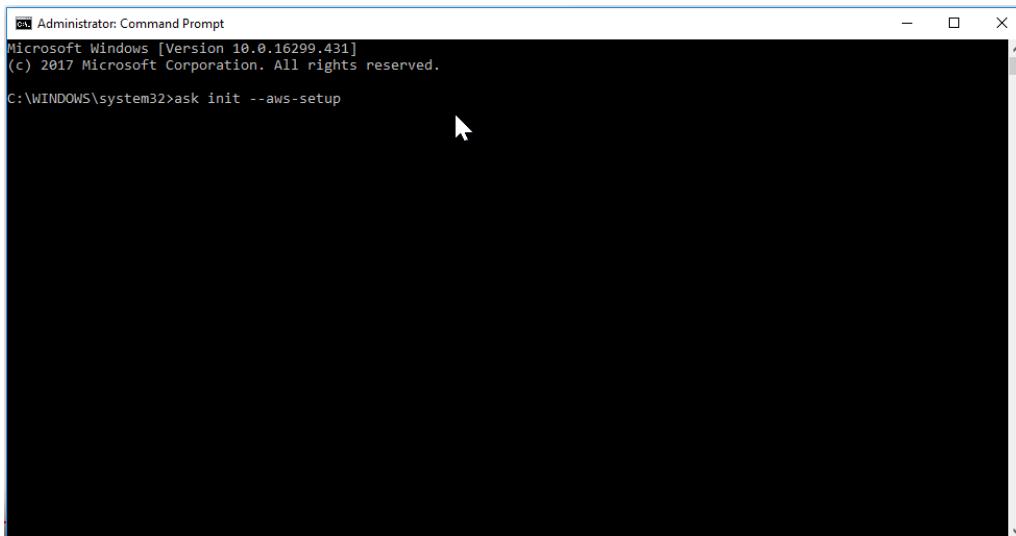
'Can't find Python executable "python", you can set the PYTHON env variable'

you did not Open the CMD Prompt as an Administrator. You will need to go back and redo the last process.

Step 4 - Setting up a Local AWS Profile

In the Command Prompt Window, enter the following and then press Enter as shown below

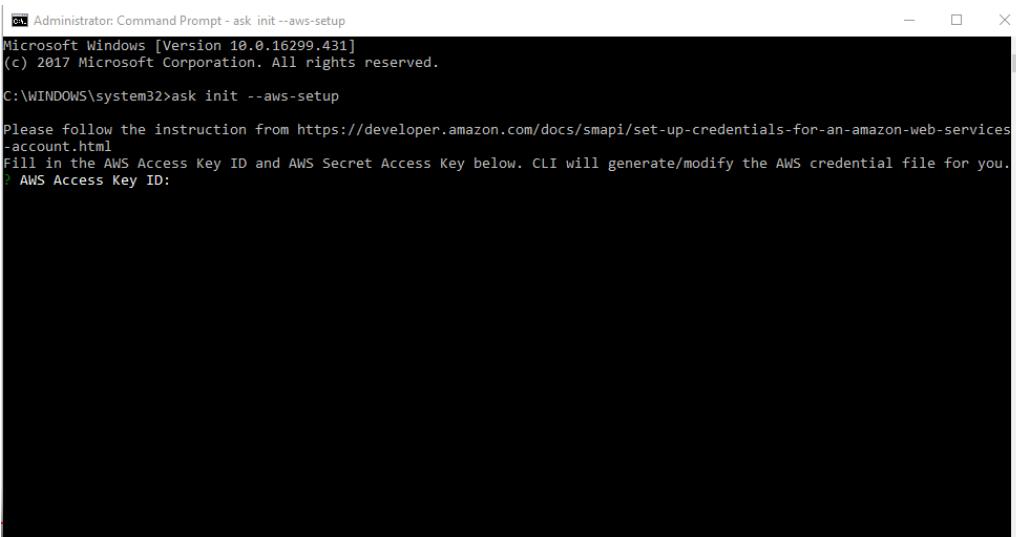
```
ask init --aws-setup
```



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup
```

You will be presented with the following

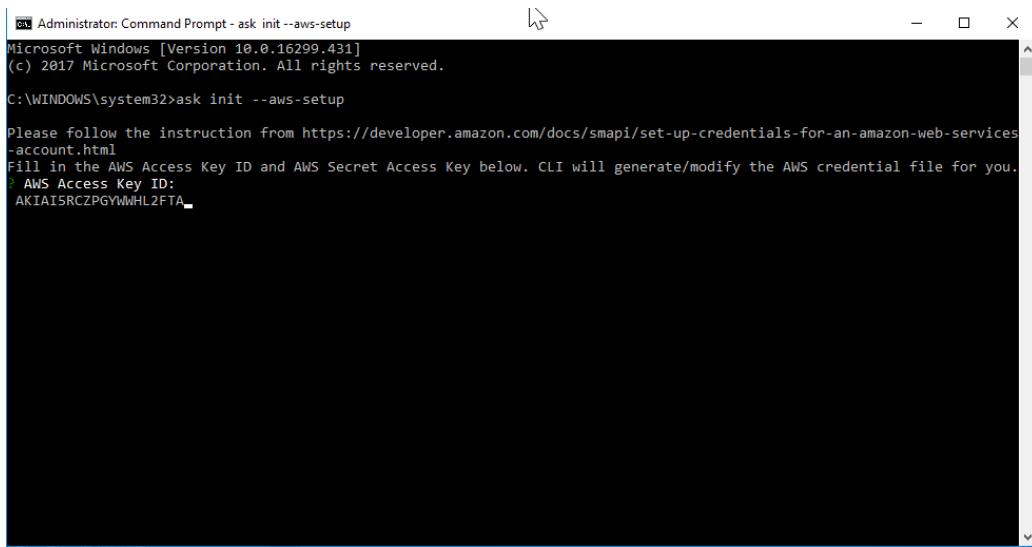


```
Administrator: Command Prompt - ask init --aws-setup
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup

Please follow the instruction from https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html
Fill in the AWS Access Key ID and AWS Secret Access Key below. CLI will generate/modify the AWS credential file for you.
? AWS Access Key ID:
```

It is asking for the 'AWS Access KEY ID' that we copied from a previous step. Paste that value in and then hit Enter as shown below

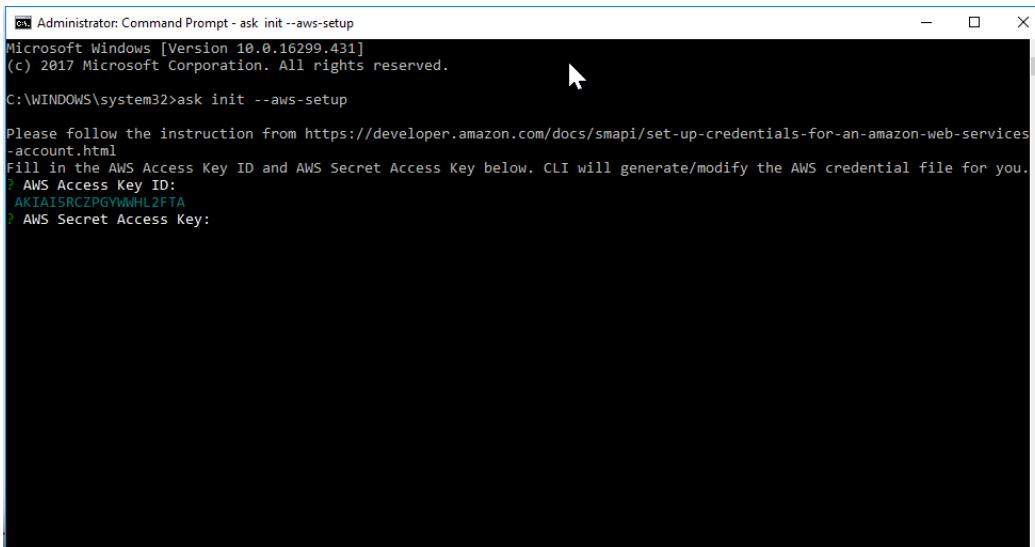


```
Administrator: Command Prompt - ask init --aws-setup
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup

Please follow the instruction from https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html
Fill in the AWS Access Key ID and AWS Secret Access Key below. CLI will generate/modify the AWS credential file for you.
? AWS Access Key ID:
AKIAISRCZPGYWhL2FTA
```

You will be presented with the following

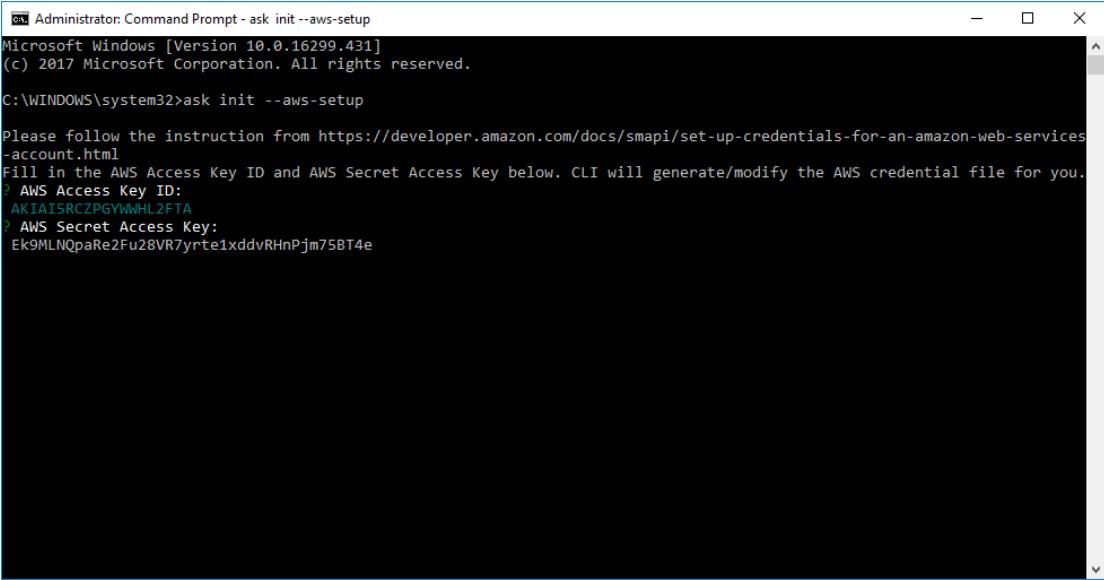


```
Administrator: Command Prompt - ask init --aws-setup
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup

Please follow the instruction from https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html
Fill in the AWS Access Key ID and AWS Secret Access Key below. CLI will generate/modify the AWS credential file for you.
? AWS Access Key ID:
AKIAISRCZPGYWhL2FTA
? AWS Secret Access Key:
```

It is now asking for the ‘AWS Secret Access Key’ that we copied from a previous step. Paste that value in and then hit Enter as shown below



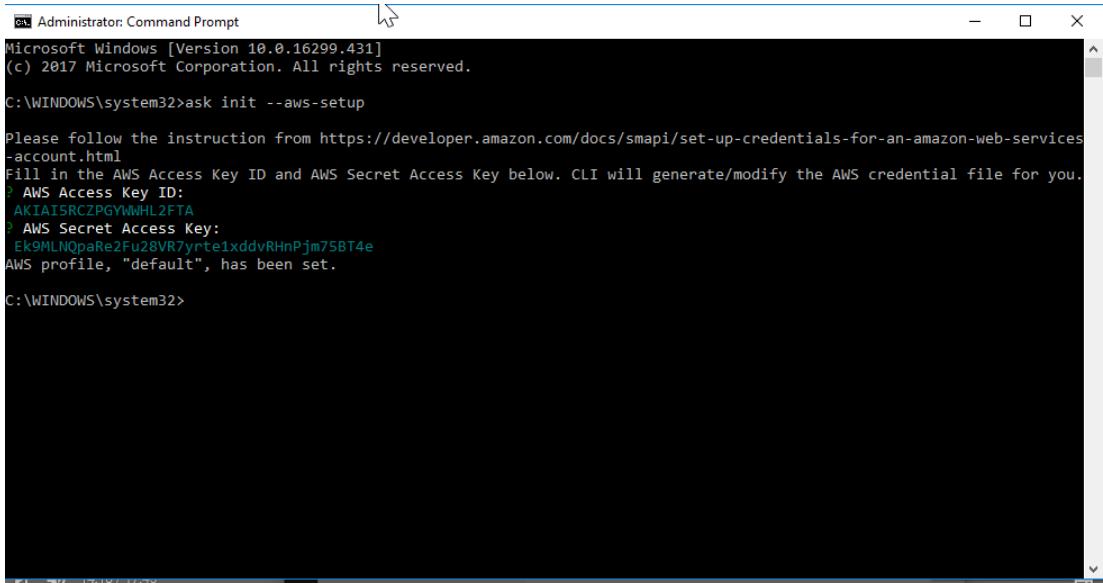
```
Administrator: Command Prompt - ask init --aws-setup
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup

Please follow the instruction from https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html
Fill in the AWS Access Key ID and AWS Secret Access Key below. CLI will generate/modify the AWS credential file for you.

? AWS Access Key ID:
AKIAISRCZPGYWHL2FTA
? AWS Secret Access Key:
Ek9MLNQpaRe2Fu28VR7yrte1xddvRHnPjm75BT4e
```

You will be shown the following screen



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ask init --aws-setup

Please follow the instruction from https://developer.amazon.com/docs/smapi/set-up-credentials-for-an-amazon-web-services-account.html
Fill in the AWS Access Key ID and AWS Secret Access Key below. CLI will generate/modify the AWS credential file for you.

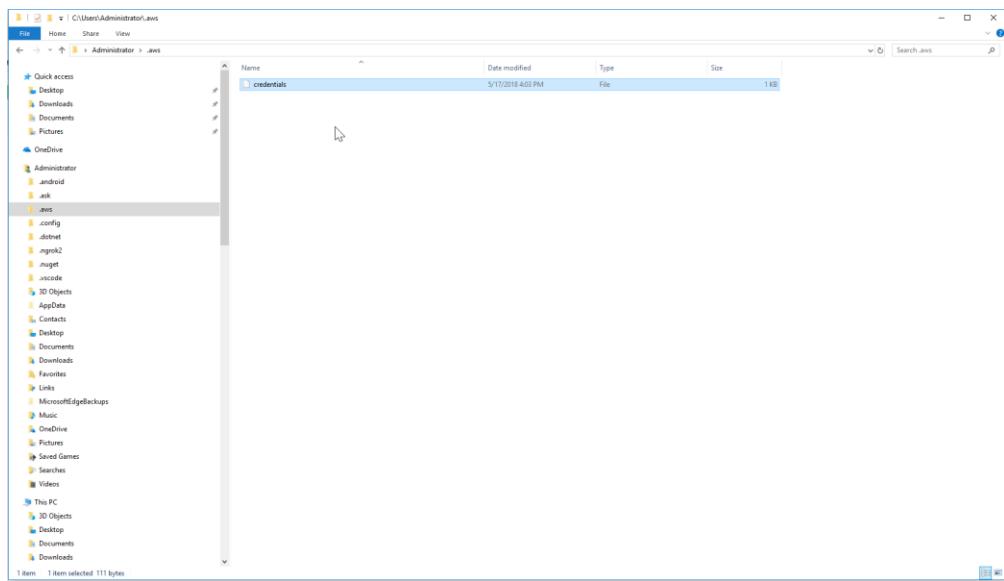
? AWS Access Key ID:
AKIAISRCZPGYWHL2FTA
? AWS Secret Access Key:
Ek9MLNQpaRe2Fu28VR7yrte1xddvRHnPjm75BT4e
AWS profile, "default", has been set.

C:\WINDOWS\system32>
```

The Local AWS Profile has now been setup. You can see the profile by opening Windows Explorer and navigating to

%USERNAME%/.aws/

Inside that folder you will find a file called ‘credentials’. If you open that file with a text editor, you will see the items that we just setup as shown below



A screenshot of a Notepad+ text editor window. The title bar says 'C:\Users\Administrator\aws\credentials - Notepad+'. The file contains the following text:

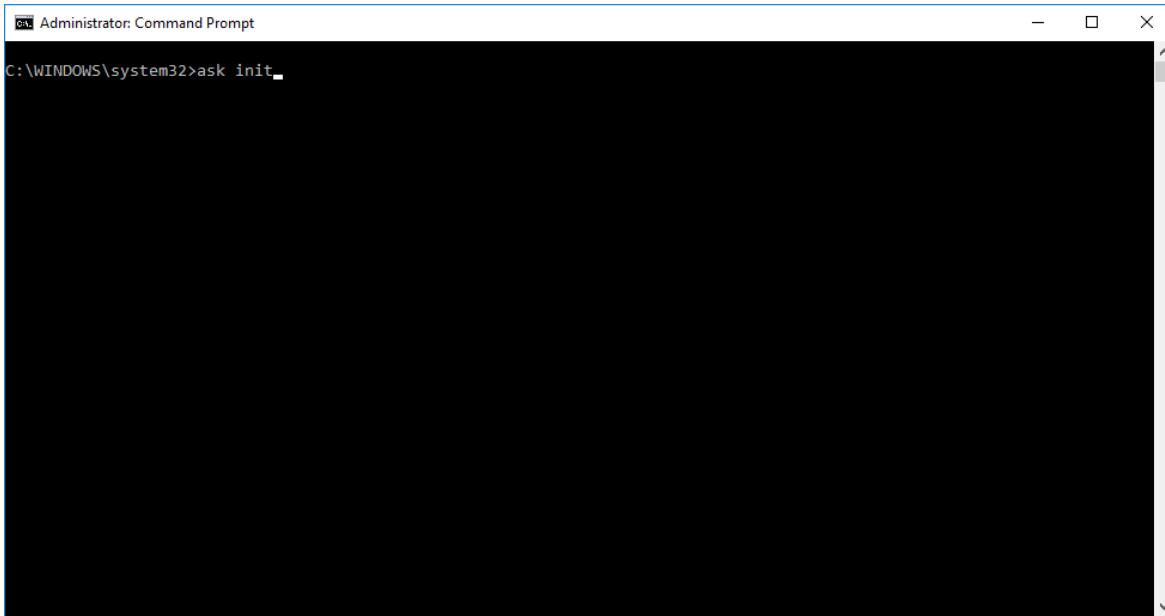
```
1 [default]
2 aws_access_key_id=AKIAI5RCZPGYWHL2FTA
3 aws_secret_access_key=Ek9MLNQpaRe2Fu28VR7yrte1xddvRHnPjm75BT4e
```

The status bar at the bottom shows: length: 111 lines: 3 Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 IN

Step 5 - Associate the User with the ASK CLI

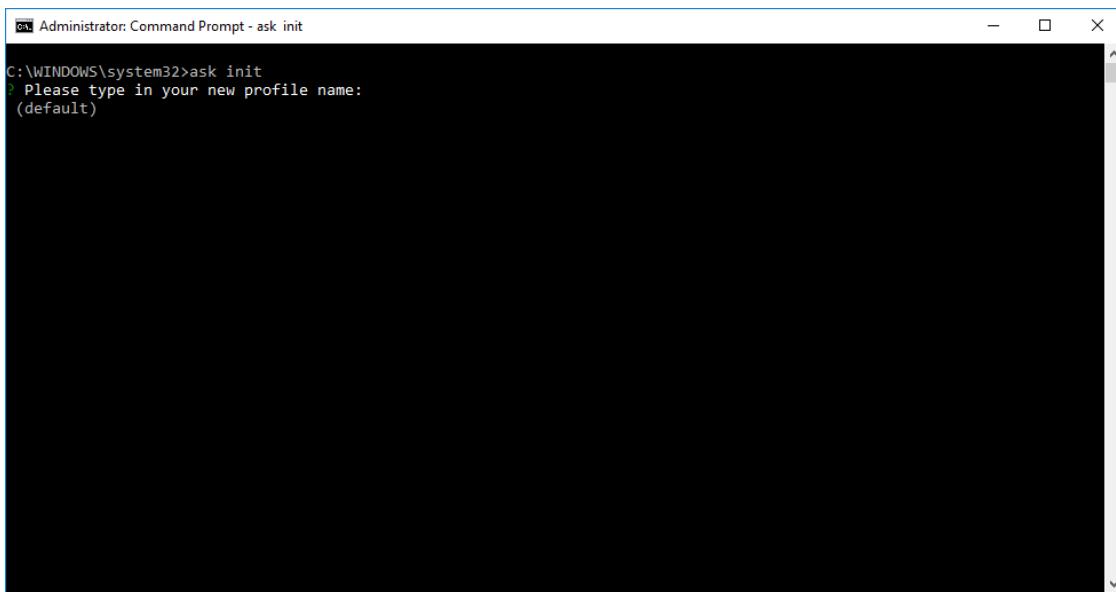
Now that the ASK CLI and the AWS Security Profile are both setup, we need to connect the two together

Enter ‘ask init’ as shown below



A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the command "C:\WINDOWS\system32>ask init" entered at the prompt. The rest of the window is blank, indicating no immediate feedback.

You will receive the following feedback



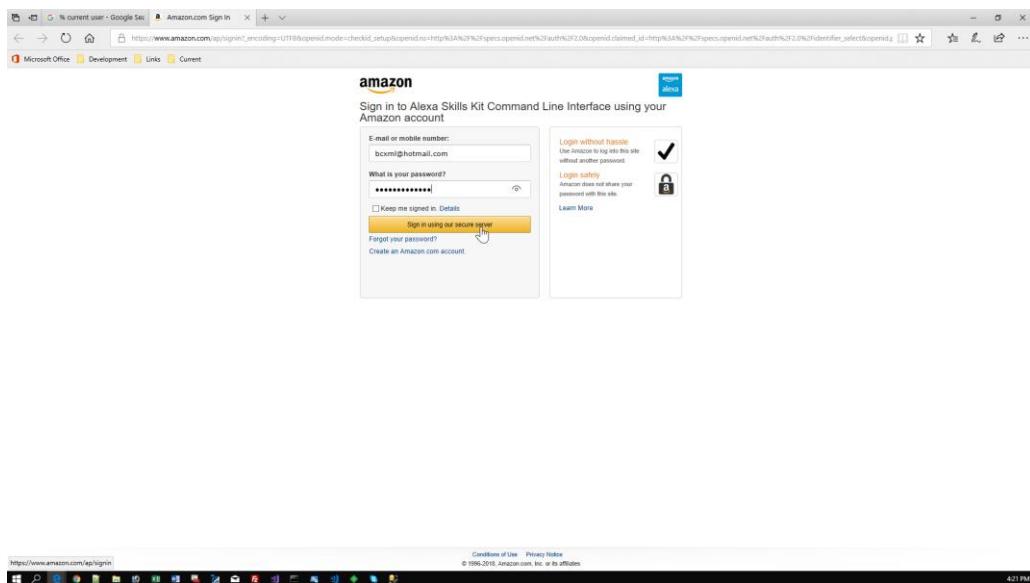
A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt - ask init". The window shows the command "C:\WINDOWS\system32>ask init" entered. Below it, a prompt reads: "Please type in your new profile name: (default)".

Just hit Enter to accept the Profile Name = default. You will be shown the following

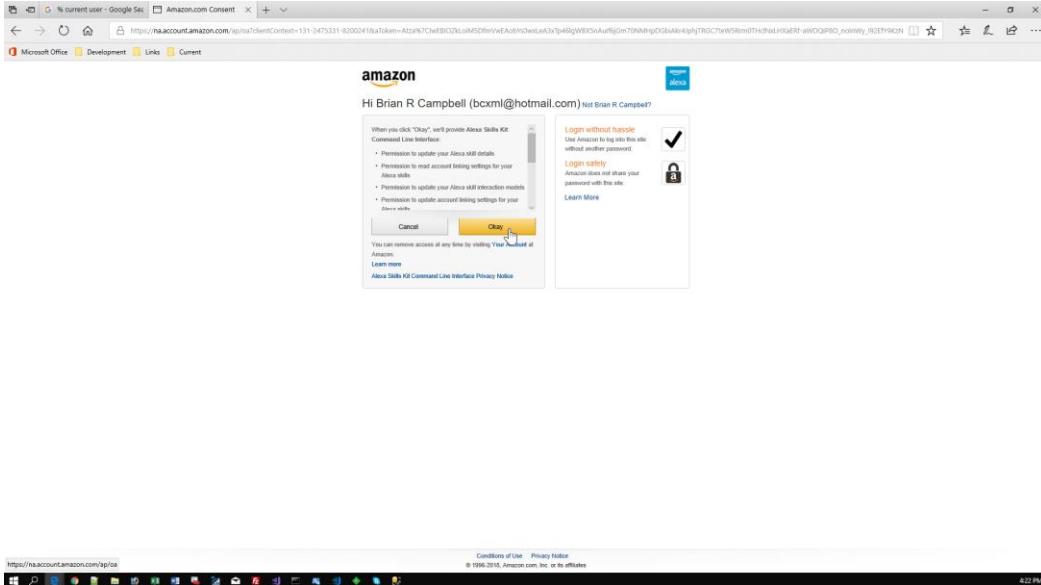
```
C:\WINDOWS\system32>ask init
/ Please type in your new profile name:
default
----- Initialize CLI -----
Setting up ask profile: [default]
/ Please choose one from the following AWS profiles for skill's Lambda function deployment.

> default
Skip AWS credential for ask-cli.
Use the AWS environment variables.
```

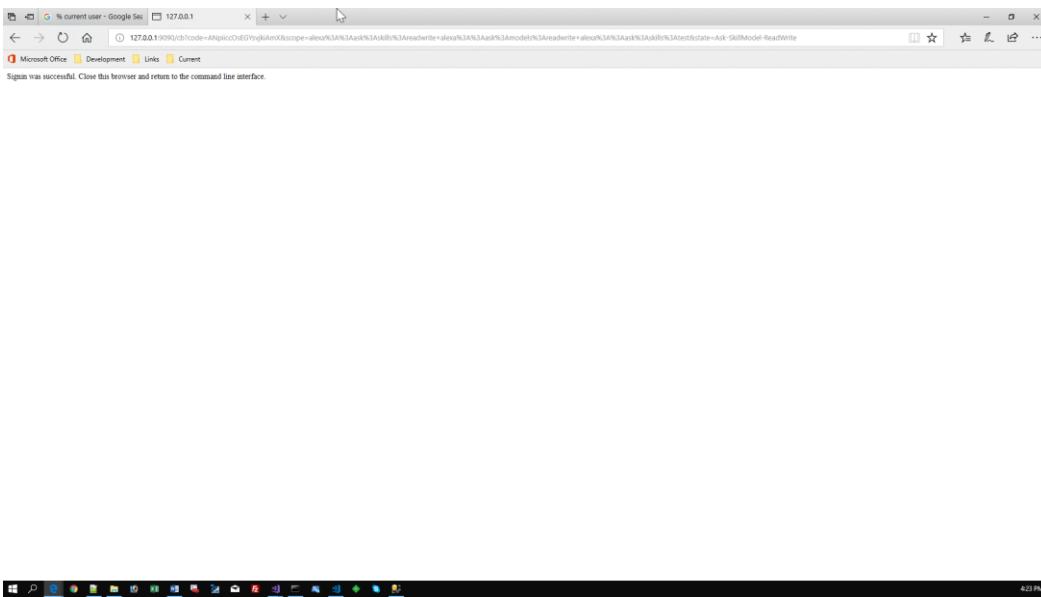
Just press Enter. You will be directed to Amazon so that you can login with your AWS credentials as shown below



Next, you will be asked if it is Ok to allow the locally installed ASK CLI to perform various operations in AWS. Click the Ok button as shown below

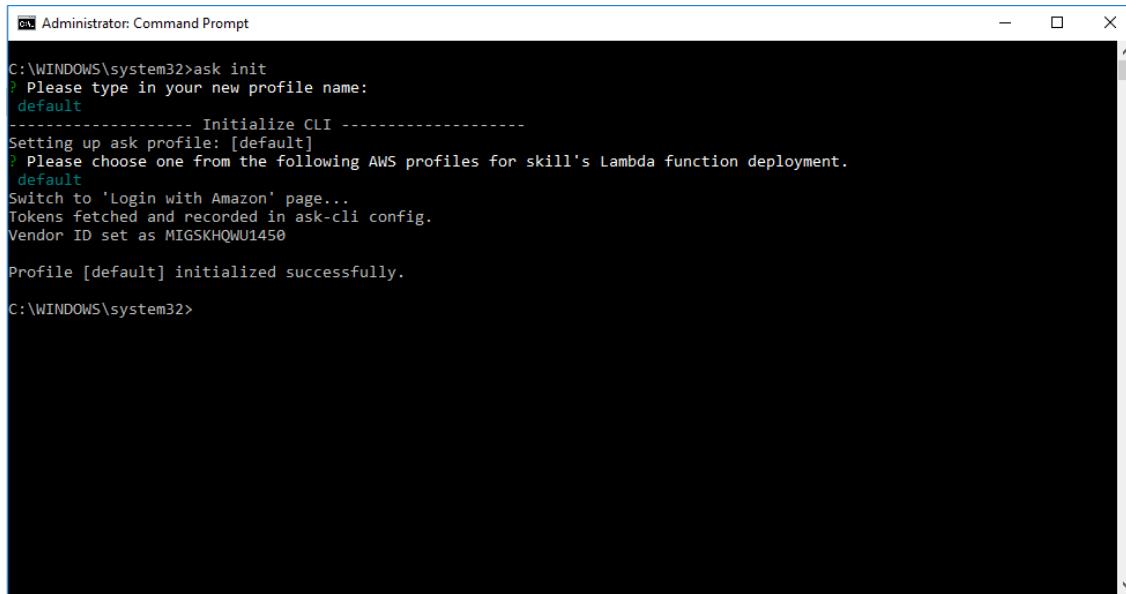


You will now be presented with a page telling you that Sign In was successful as shown below



You can close the browser at this point.

Back in the Command Prompt you will be shown that the initialization process completed successfully as shown below



The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The window has a dark background and light-colored text. It displays the following command-line session:

```
C:\WINDOWS\system32>ask init
? Please type in your new profile name:
default
----- Initialize CLI -----
Setting up ask profile: [default]
? Please choose one from the following AWS profiles for skill's Lambda function deployment.
default
Switch to 'Login with Amazon' page...
Tokens fetched and recorded in ask-cli config.
Vendor ID set as MIGSKHQWU1450

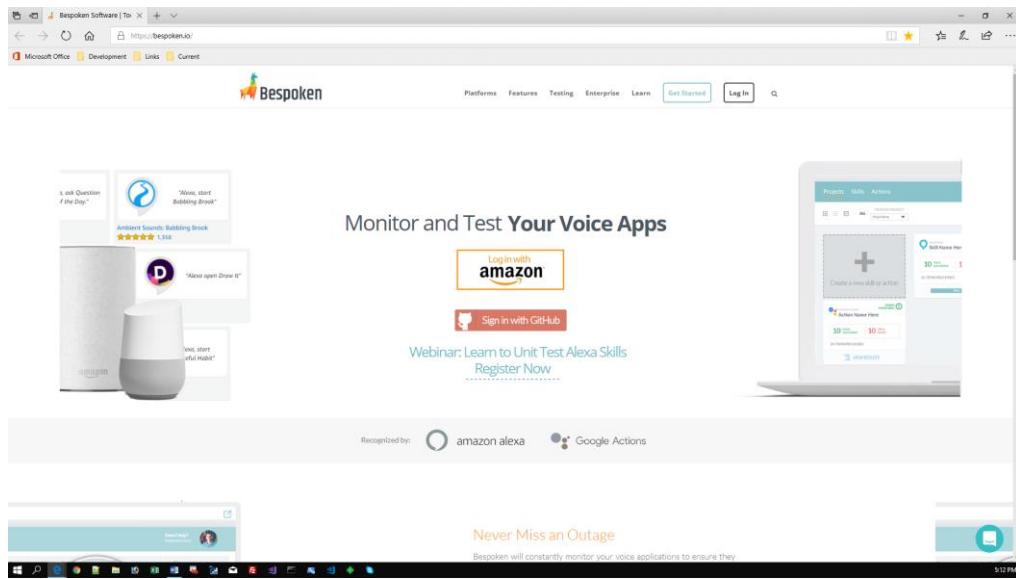
Profile [default] initialized successfully.

C:\WINDOWS\system32>
```

Bespoken Tools

The Bespoken Tools allow Alexa Developers to test and monitor the custom Alexa Skills they are developing. You can find the Bespoken Website at

<https://Bespoken.io/>



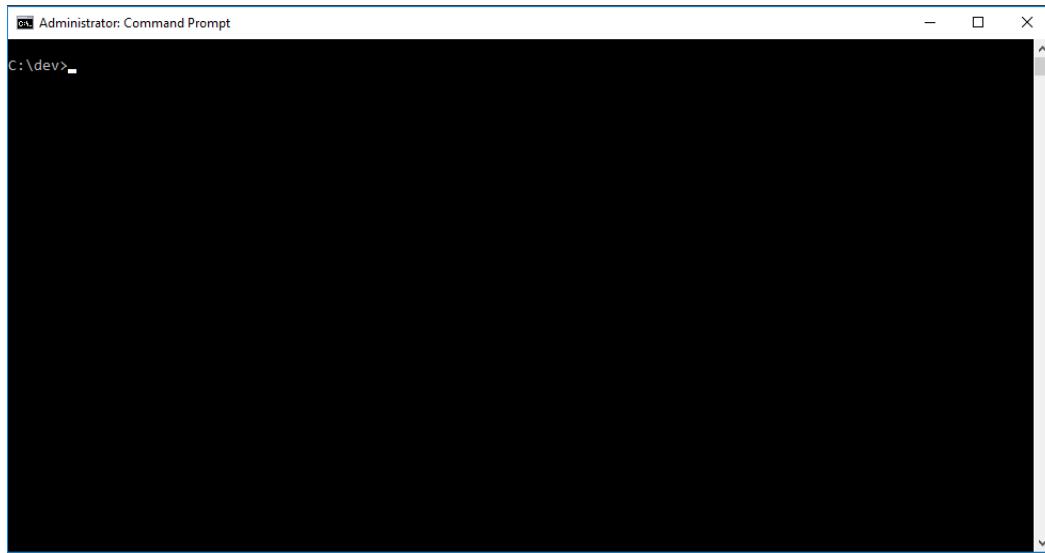
The Bespoken tools are installed using the Node Package Manager (npm). We will be installing these tools a little later in the process.

Creating an Alexa Skill locally

Now that all the tools have been installed we can go ahead and create an Alexa Skill to test out the Local Development Environment. I am going to call this Alexa Skill “Greeter”

Create the Alexa Skill

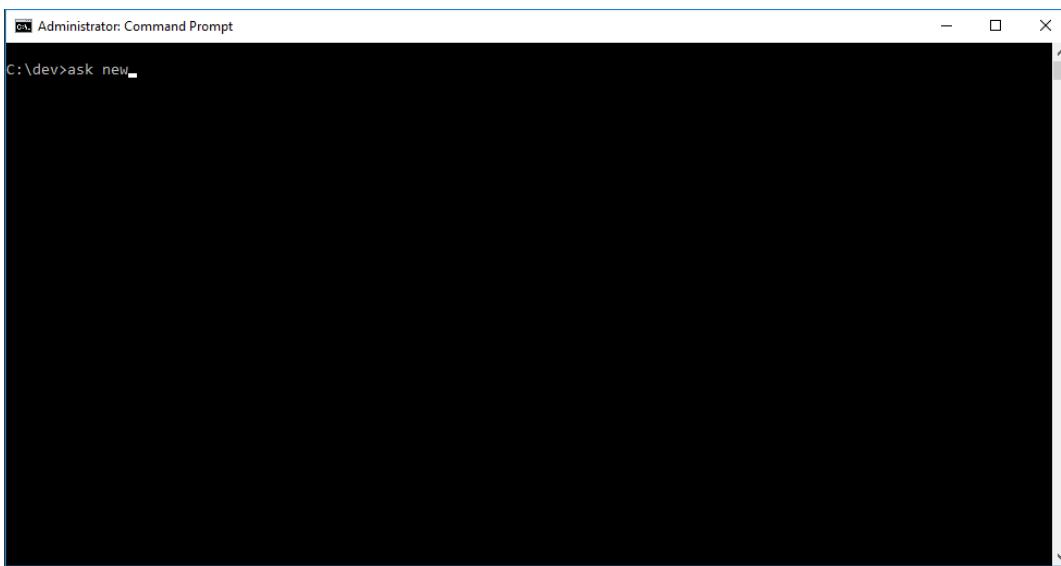
Using a command prompt, create a folder where this “Greeter” Skill can live. In my case I will be using C:\dev as shown below



```
C:\ Administrator: Command Prompt
C:\dev>
```

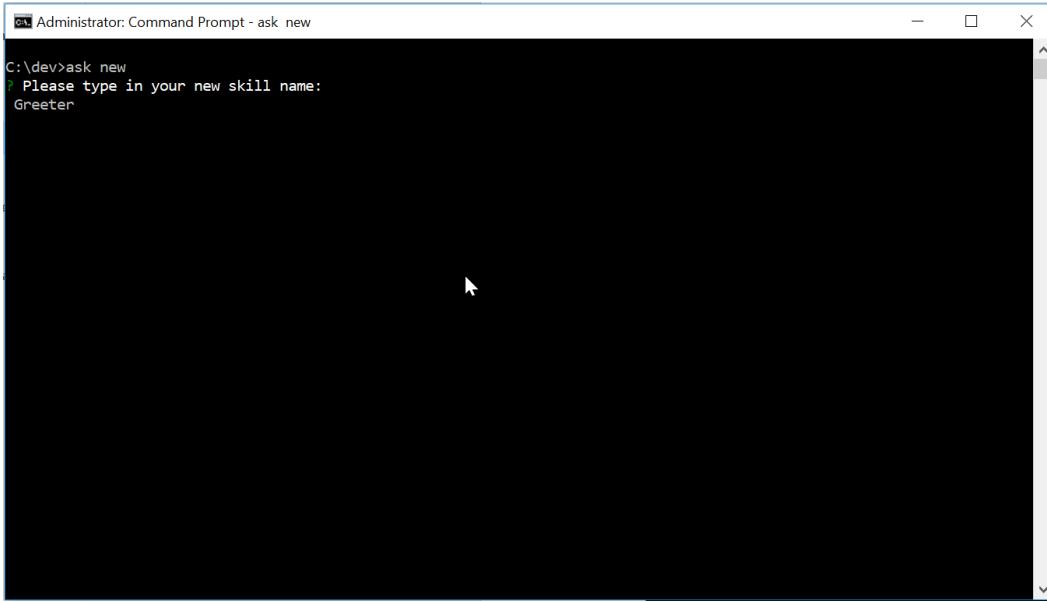
We will be using the ‘Alexa Skills Kit Command Line Interface’ (ASK CLI for short) to locally create the ‘Greeter Skill’. To create the ‘Greeter Skill’ enter

ask new



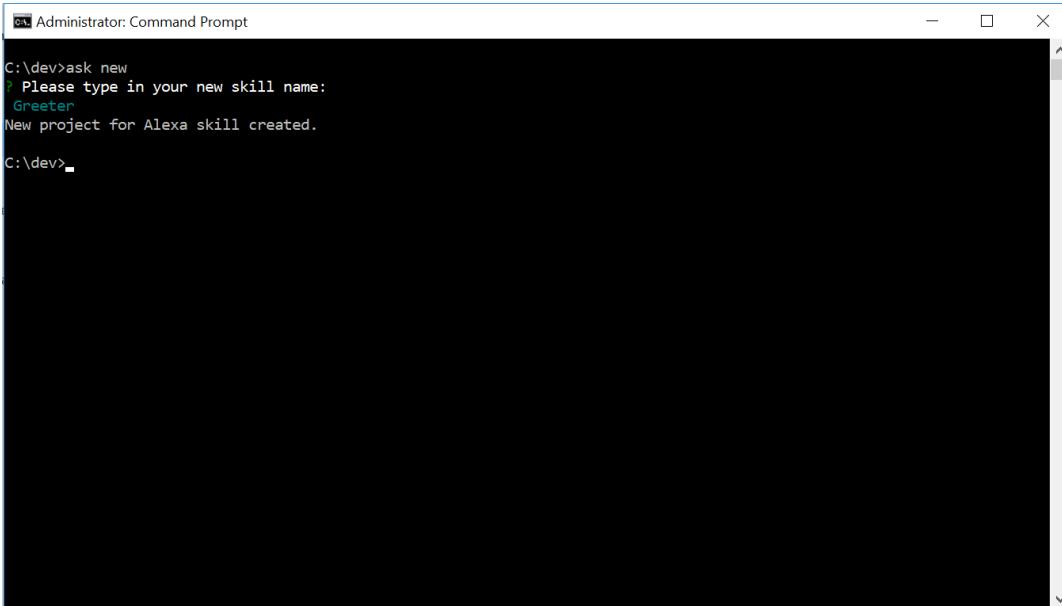
```
C:\ Administrator: Command Prompt
C:\dev>ask new
```

You will be asked for the name you want to give the Skill. Enter ‘Greeter’ and then hit Enter as shown below



```
C:\dev>ask new
? Please type in your new skill name:
Greeter
```

You will be notified once the Alexa Skill has been created as shown below

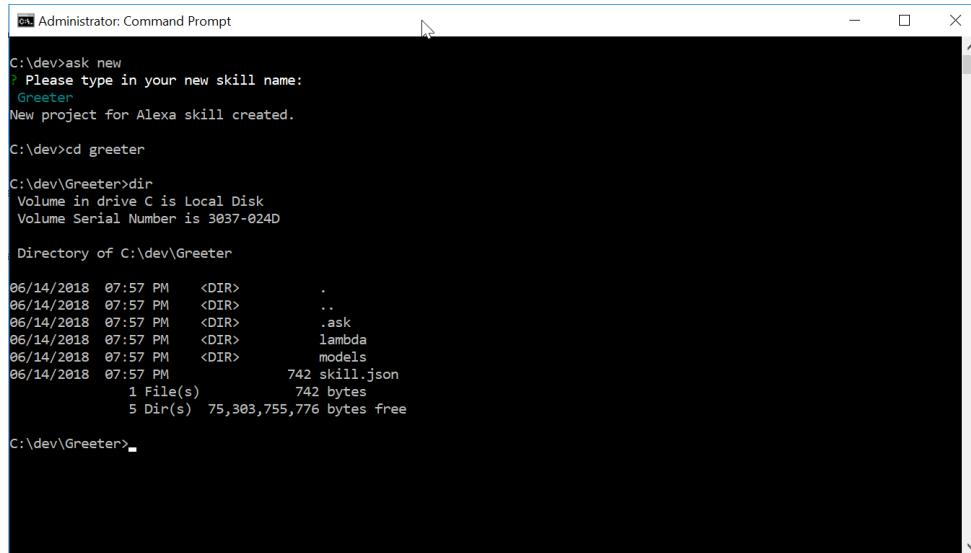


```
C:\dev>ask new
? Please type in your new skill name:
Greeter
New project for Alexa skill created.

C:\dev>
```

The Skill that was created is a fully working "Hello World" skill that can be enabled and invoked immediately. It is based on a sample that Amazon provides in the Alexa Skills Repo over at GitHub.

Navigate into the newly created ‘Greeter’ folder and have a look at the files that have been created as shown below



```
C:\dev>ask new
? Please type in your new skill name:
Greeter
New project for Alexa skill created.

C:\dev>cd greeter

C:\dev\Greeter>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter

06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM    <DIR>        .ask
06/14/2018  07:57 PM    <DIR>        lambda
06/14/2018  07:57 PM    <DIR>        models
06/14/2018  07:57 PM           742 skill.json
                           1 File(s)      742 bytes
                           5 Dir(s)  75,303,755,776 bytes free

C:\dev\Greeter>
```

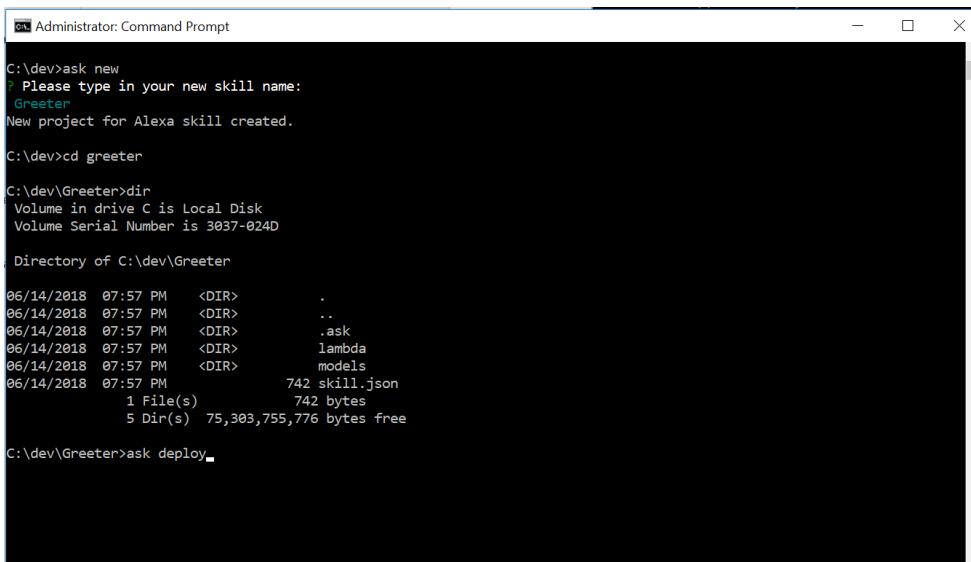
What has been created is the code for a complete Alex Skill. In the ‘models’ folder is a JSON file that represents the Skill Interface portion of the Skill and in the Lambda folder is the code that makes up the Skill Service.

Deploy the Alexa Skill

Of course, this code all needs to be deployed to Amazon before it can be used, so let’s do that right now. In the command prompt enter

```
ask deploy
```

as shown below



```
C:\dev>ask new
? Please type in your new skill name:
Greeter
New project for Alexa skill created.

C:\dev>cd greeter

C:\dev\Greeter>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter

06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM    <DIR>        .ask
06/14/2018  07:57 PM    <DIR>        lambda
06/14/2018  07:57 PM    <DIR>        models
06/14/2018  07:57 PM           742 skill.json
                           1 File(s)      742 bytes
                           5 Dir(s)  75,303,755,776 bytes free

C:\dev\Greeter>ask deploy
```

You will be shown some status messages as the deployment takes place

```
C:\dev>ask new
? Please type in your new skill name:
Greeter
New project for Alexa skill created.

C:\dev>cd greeter

C:\dev\Greeter>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter

06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM    <DIR>        .ask
06/14/2018  07:57 PM    <DIR>        lambda
06/14/2018  07:57 PM    <DIR>        models
06/14/2018  07:57 PM           742 skill.json
                           1 File(s)      742 bytes
                           5 Dir(s)  75,303,755,776 bytes free

C:\dev\Greeter>ask deploy
----- Create Skill Project -----
Profile for the deployment: [default]
Skill Id: amzn1.ask.skill.cb80cf41-d33e-4811-bd35-ff9e17e0d768
Skill deployment finished.
```

The whole process takes a couple of minutes. Once completed you will be shown the following

```
C:\dev>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter

06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM    <DIR>        .ask
06/14/2018  07:57 PM    <DIR>        lambda
06/14/2018  07:57 PM    <DIR>        models
06/14/2018  07:57 PM           742 skill.json
                           1 File(s)      742 bytes
                           5 Dir(s)  75,303,755,776 bytes free

C:\dev\Greeter>ask deploy
----- Create Skill Project -----
Profile for the deployment: [default]
Skill Id: amzn1.ask.skill.cb80cf41-d33e-4811-bd35-ff9e17e0d768
Skill deployment finished.
Model deployment finished.
Lambda deployment finished.
Lambda function(s) created:
  [URI] ask-custom-Greeter-default
No in-skill product to be deployed.
Your skill is now deployed and enabled in the development stage.
Try invoking the skill by saying "Alexa, open {your_skill_invocation_name}" or simulate an invocation via the `ask simulate` command.

C:\dev\Greeter>
```

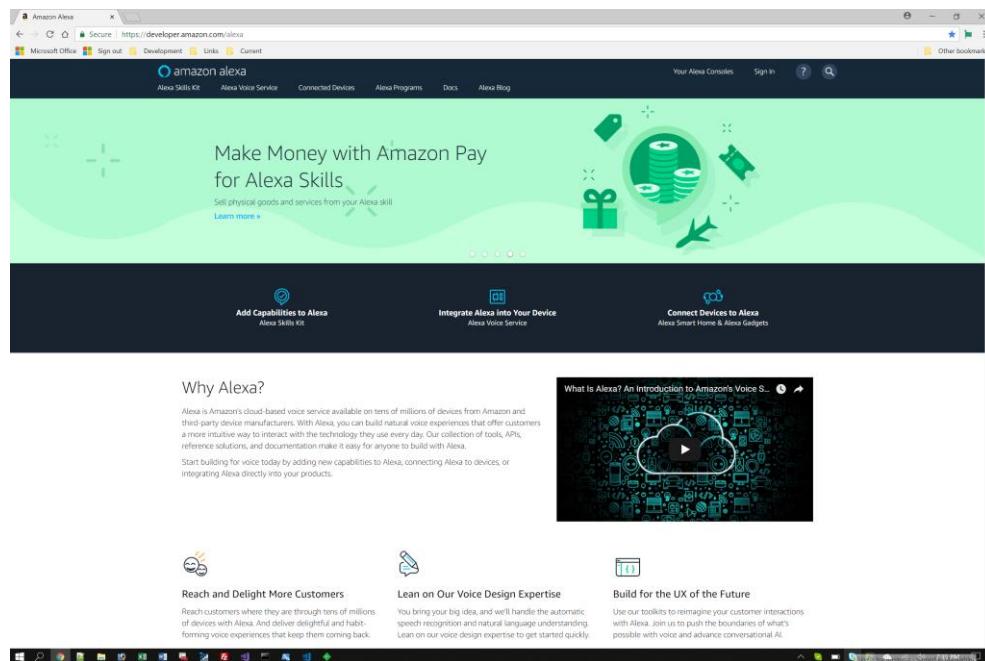
The local code has now been deployed to Amazon and setup as an Alexa Skill.

Verify that the Alexa Skill was Deployed

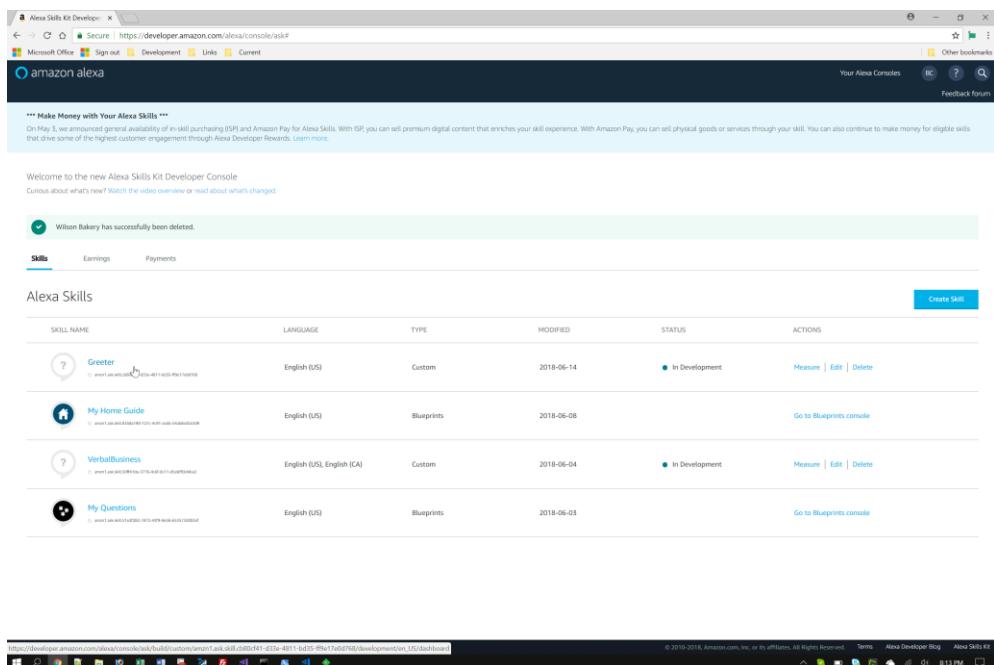
The above deployment process deployed the code for both the Alexa Skill Interface and the Alexa Skill Service. Let's have a look at what was deployed.

First for the Skill Interface, open a browser and head to the Alexa Skill Console at

<https://developer.amazon.com/alexa>



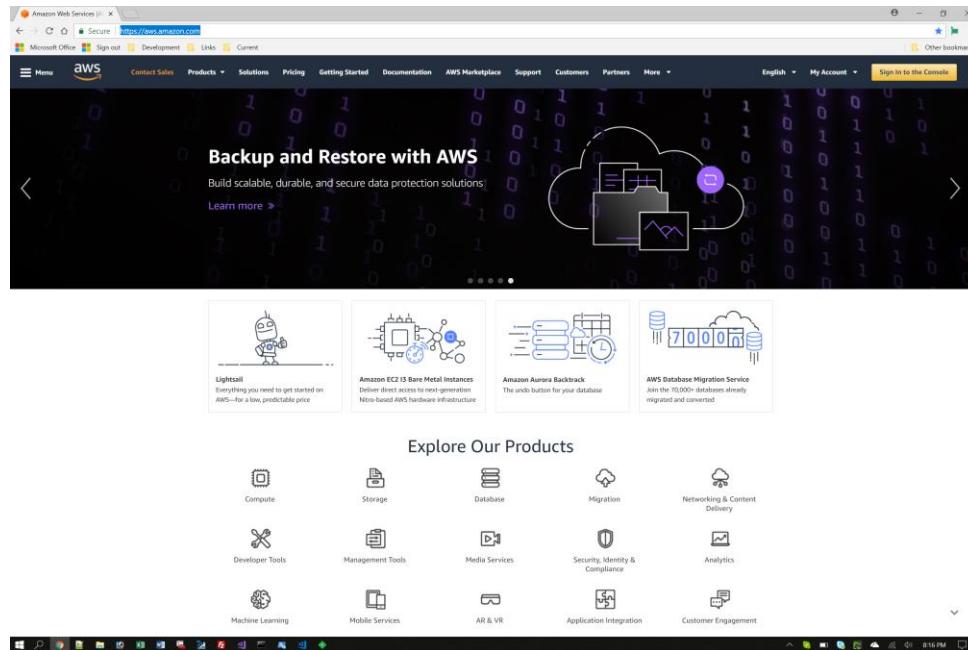
Login to the Skill Console and go to the Skill Console. You will be presented with the following



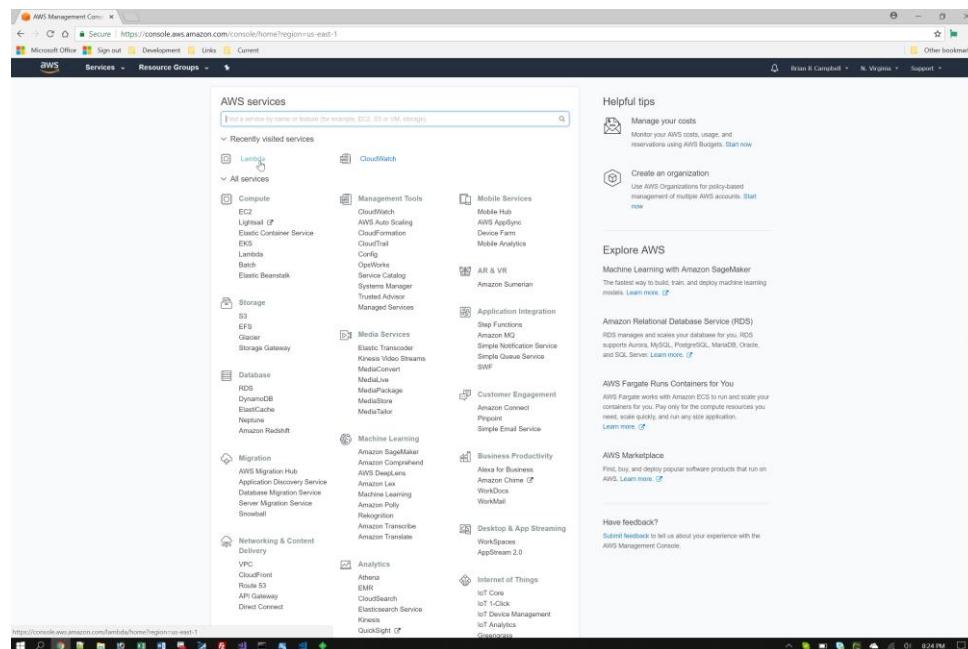
Notice that the ‘Greeter Skill’ that we previously deployed is indeed listed, indicating that the Skill Interface was successfully deployed.

Next for the Skill Service, open a browser and head to the AWS Console at

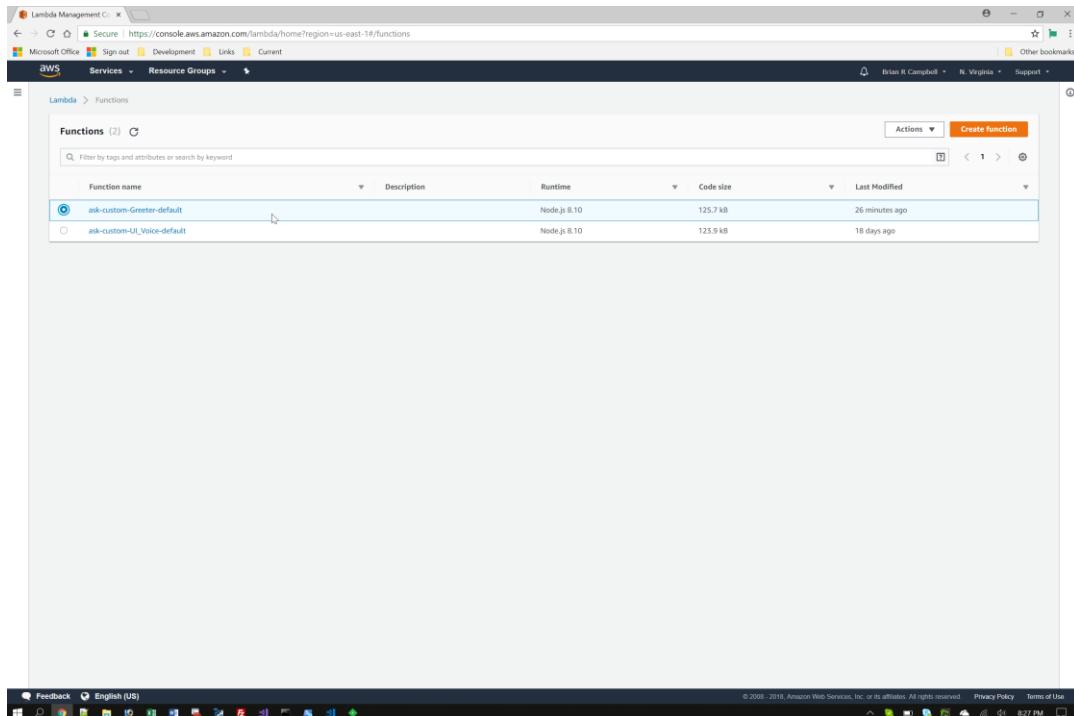
<https://aws.amazon.com/>



Login to the AWS Console. Once logged in, select ‘Lambda’ as shown below



Notice that the Skill Service for the “Greeter Skill” that we previously deployed is indeed listed as shown below



The screenshot shows the AWS Lambda Management Console interface. At the top, there's a navigation bar with links for Microsoft Office, Sign out, Development, Links, Current, Brian R Campbell, N Virginia, and Support. Below the navigation bar, the main title is "Lambda > Functions". A search bar says "Filter by tags and attributes or search by keyword". There's a "Create function" button. A table lists two functions:

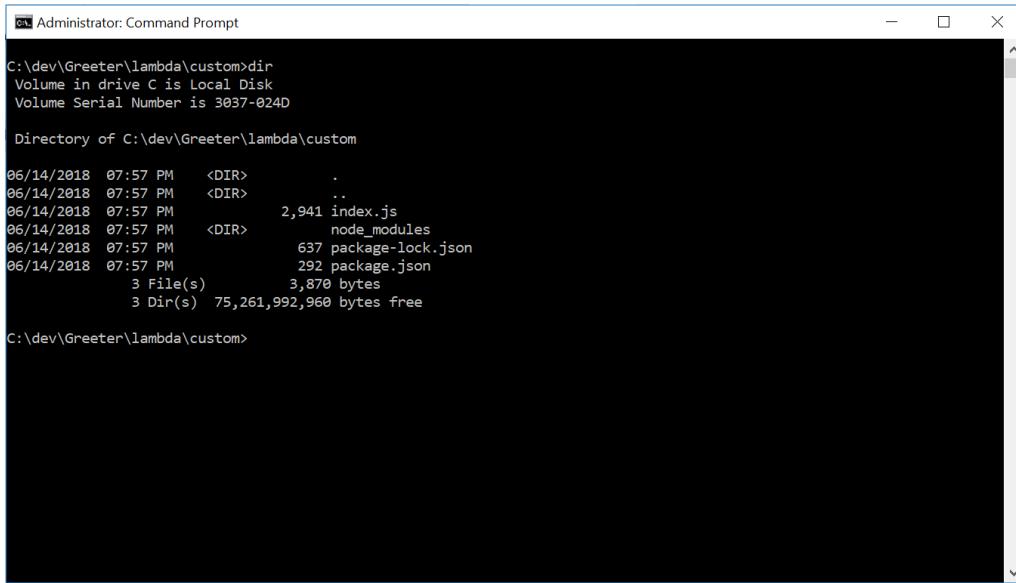
| Function name | Description | Runtime | Code size | Last Modified |
|-----------------------------|-------------|--------------|-----------|----------------|
| ask-custom-Greeter-default | | Node.js 8.10 | 125.7 kB | 26 minutes ago |
| ask-custom-UI_Voice-default | | Node.js 8.10 | 123.9 kB | 18 days ago |

Debugging the local Alexa Skill with Bespoken Tools

We will be debugging the code that makes up the Skill Service using the tools provided by Bespoken.

Starting Visual Studio Code

Back in the Command Prompt, navigate to the ‘lambda\custom’ folder as shown below



```
Administrator: Command Prompt
C:\dev\Greeter\lambda\custom>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter\lambda\custom

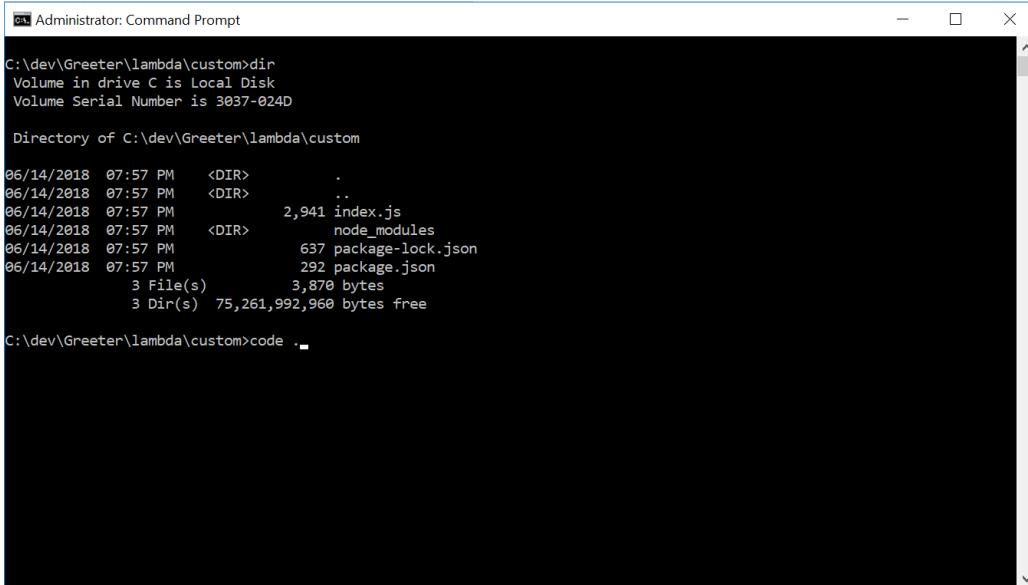
06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM           2,941 index.js
06/14/2018  07:57 PM    <DIR>        node_modules
06/14/2018  07:57 PM           637 package-lock.json
06/14/2018  07:57 PM           292 package.json
               3 File(s)       3,870 bytes
               3 Dir(s)   75,261,992,960 bytes free

C:\dev\Greeter\lambda\custom>
```

Start Visual Studio Code by entering

```
code .
```

as shown below



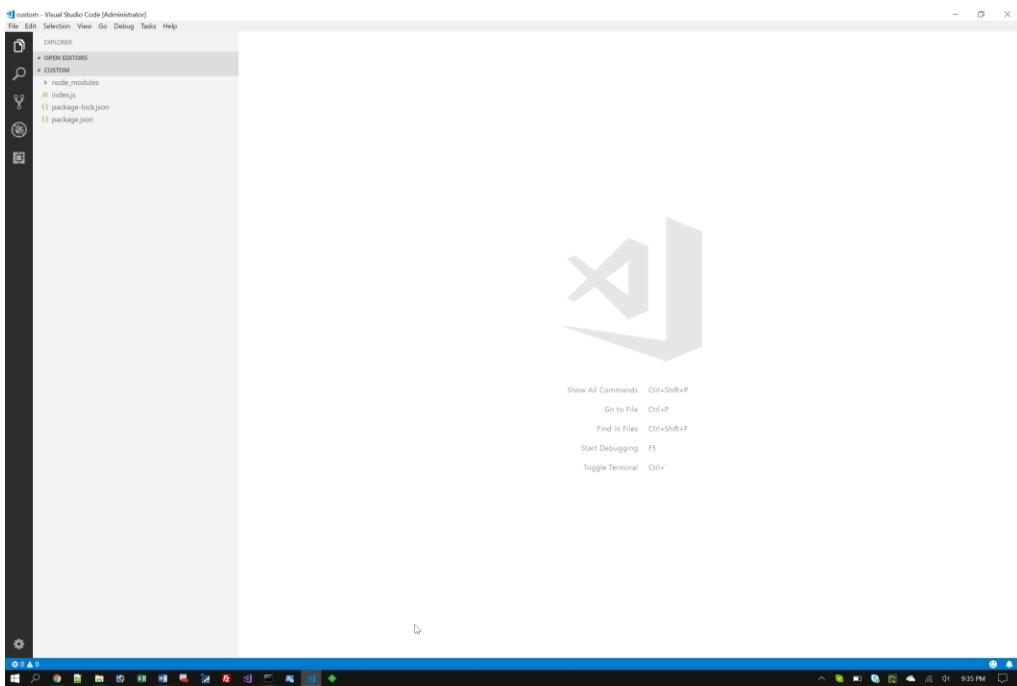
```
Administrator: Command Prompt
C:\dev\Greeter\lambda\custom>dir
Volume in drive C is Local Disk
Volume Serial Number is 3037-024D

Directory of C:\dev\Greeter\lambda\custom

06/14/2018  07:57 PM    <DIR>        .
06/14/2018  07:57 PM    <DIR>        ..
06/14/2018  07:57 PM           2,941 index.js
06/14/2018  07:57 PM    <DIR>        node_modules
06/14/2018  07:57 PM           637 package-lock.json
06/14/2018  07:57 PM           292 package.json
               3 File(s)       3,870 bytes
               3 Dir(s)   75,261,992,960 bytes free

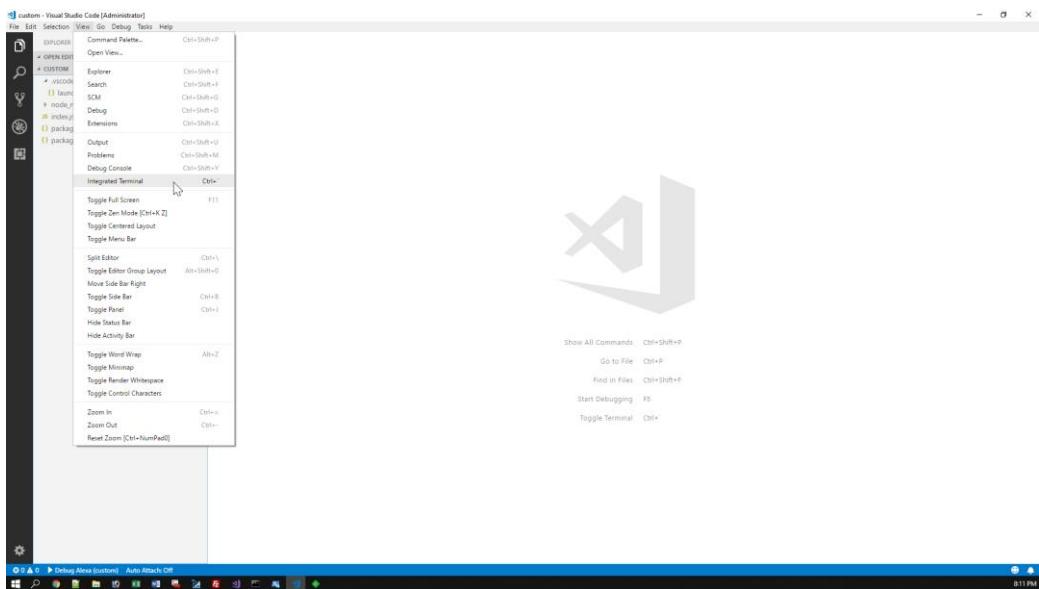
C:\dev\Greeter\lambda\custom>code .
```

Visual Studio Code will now start up and you will be shown the following

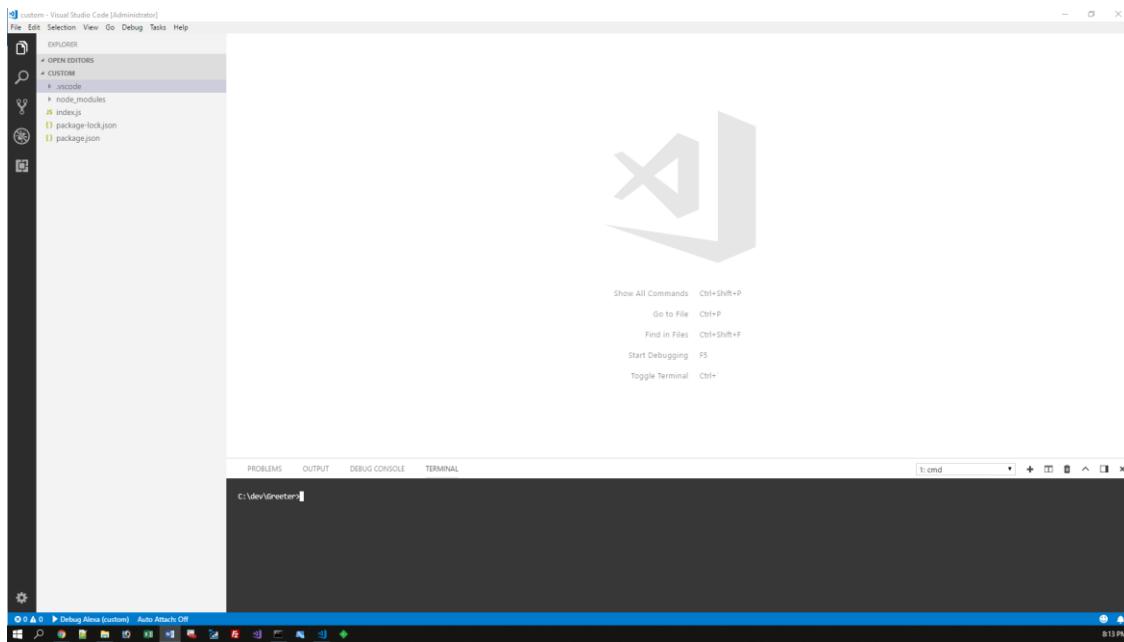


Installing the Bespoke Tools

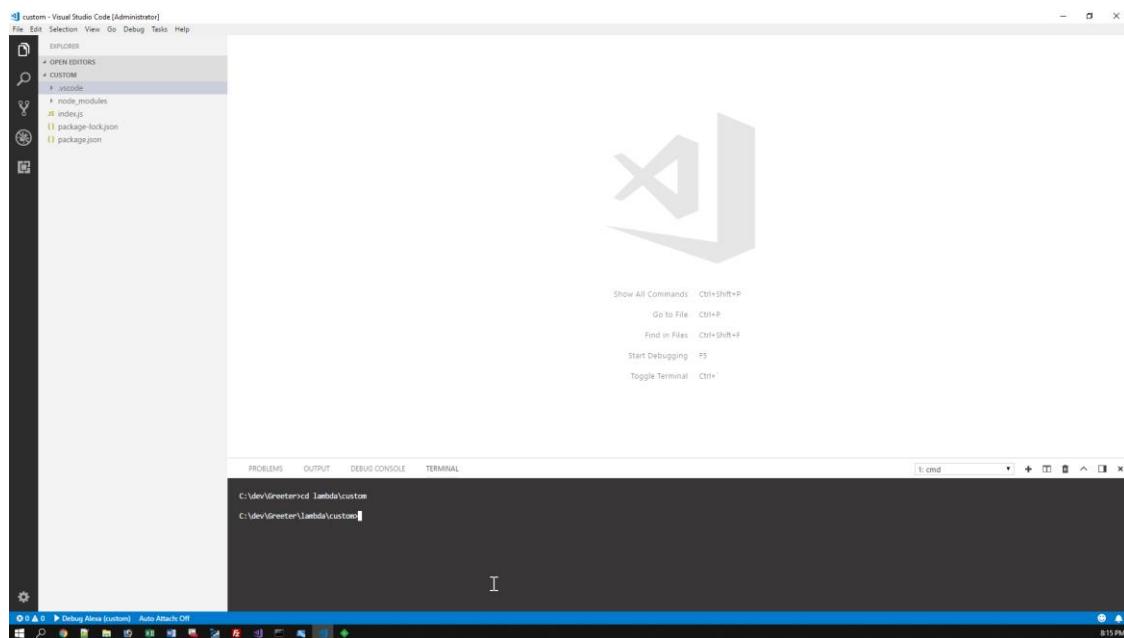
In Visual Studio Code, from the top menu select 'View' and then select 'Integrated Terminal' as shown below



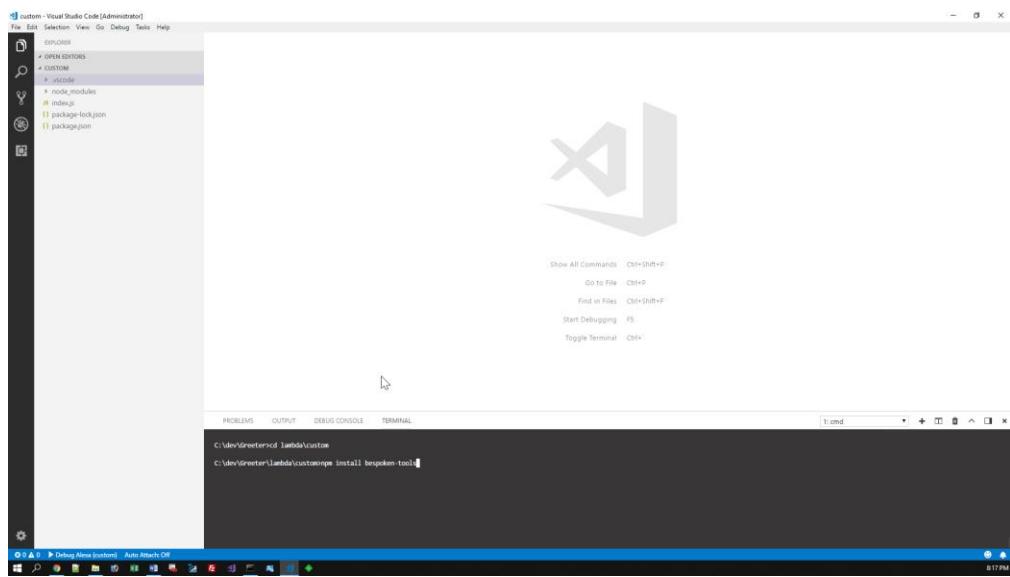
You will be shown the 'Integrated Terminal' at the bottom of the screen



Navigate to the lambda\custom folder as shown below



Enter ‘npm i bespoken-tools’ as shown below



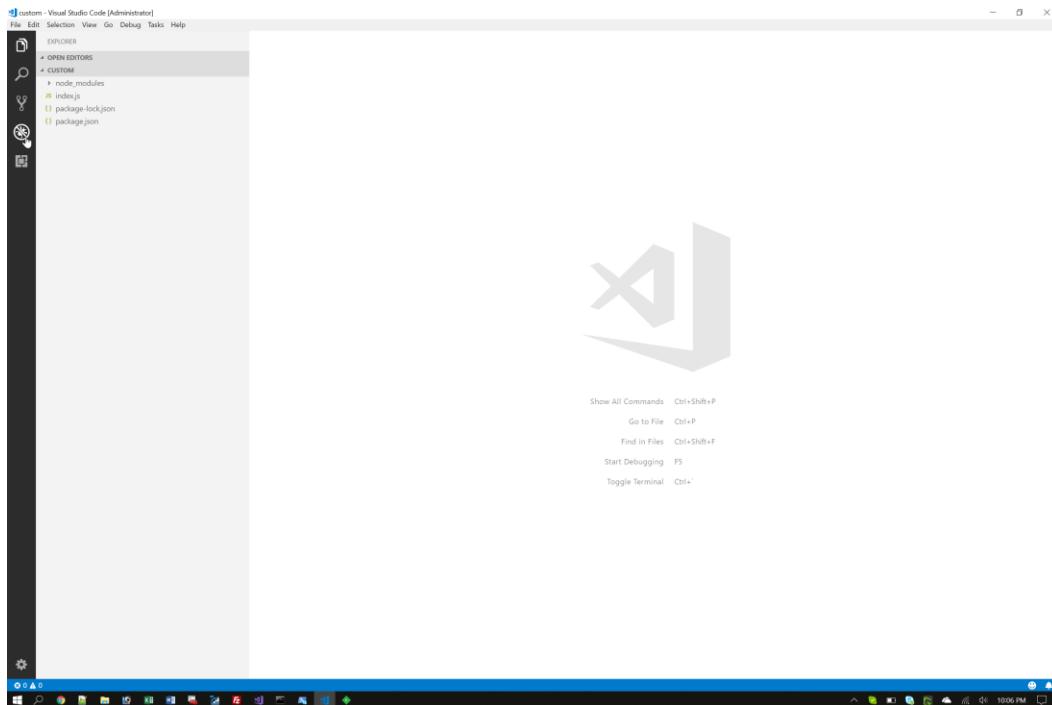
A screenshot of the Visual Studio Code interface. The title bar says "custom - Visual Studio Code [Administrator]". The left sidebar shows an "EXPLORER" view with a tree structure: "OPEN EDITORS" (empty), "CUSTOM" (empty), and "node_modules" (containing "index.js", "package-lock.json", and "package.json"). The main area is a terminal window titled "cmd" with the command "C:\dev\lambdatest\lambda\lambda\custom> npm install bespoken-tools" entered. Below the terminal are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The status bar at the bottom shows "8:17PM".

The Bespoken Tools will now be installed into the ‘node_modules’ folder.

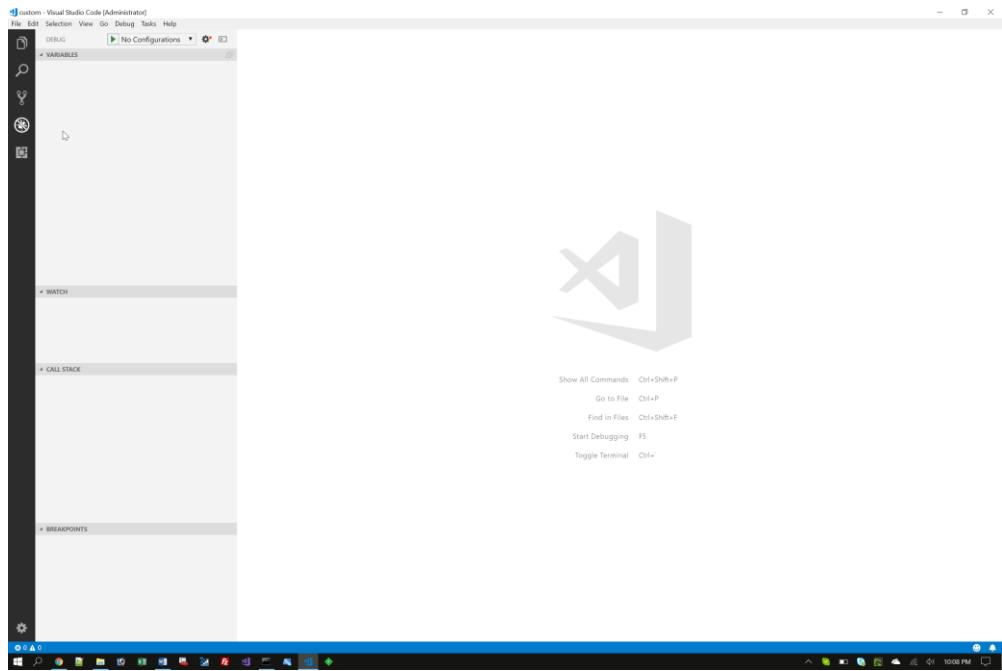
Updating Visual Studio Code Configuration

Now before we can begin debugging code using the Bespoken Tools that we just installed, we need to make a configuration change in Visual Studio Code.

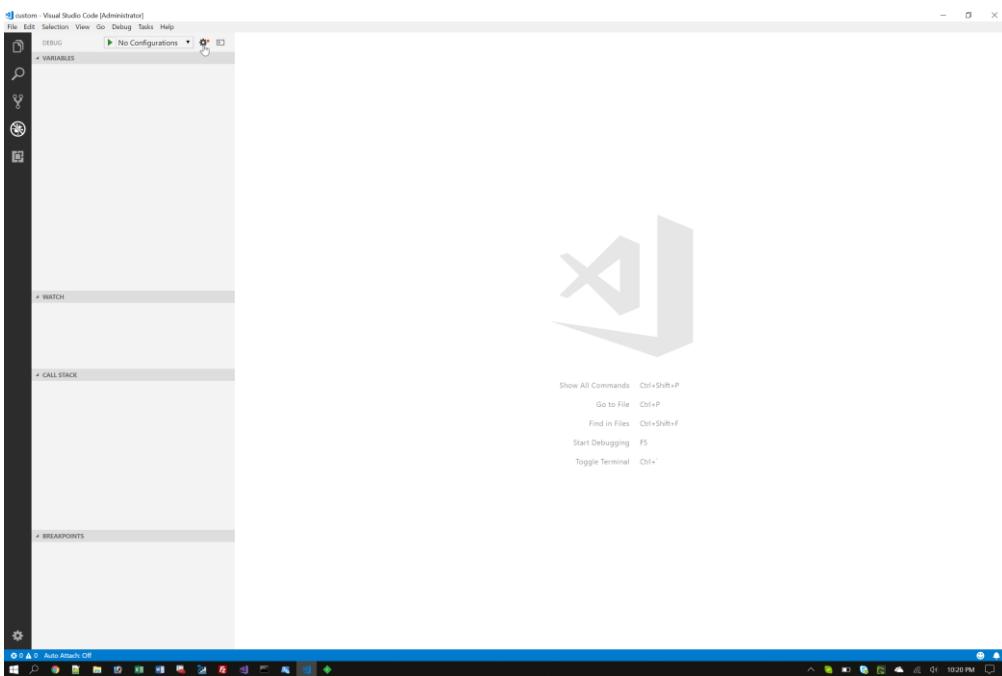
Click on the ‘Debug’ icon in the left-hand menu as shown below



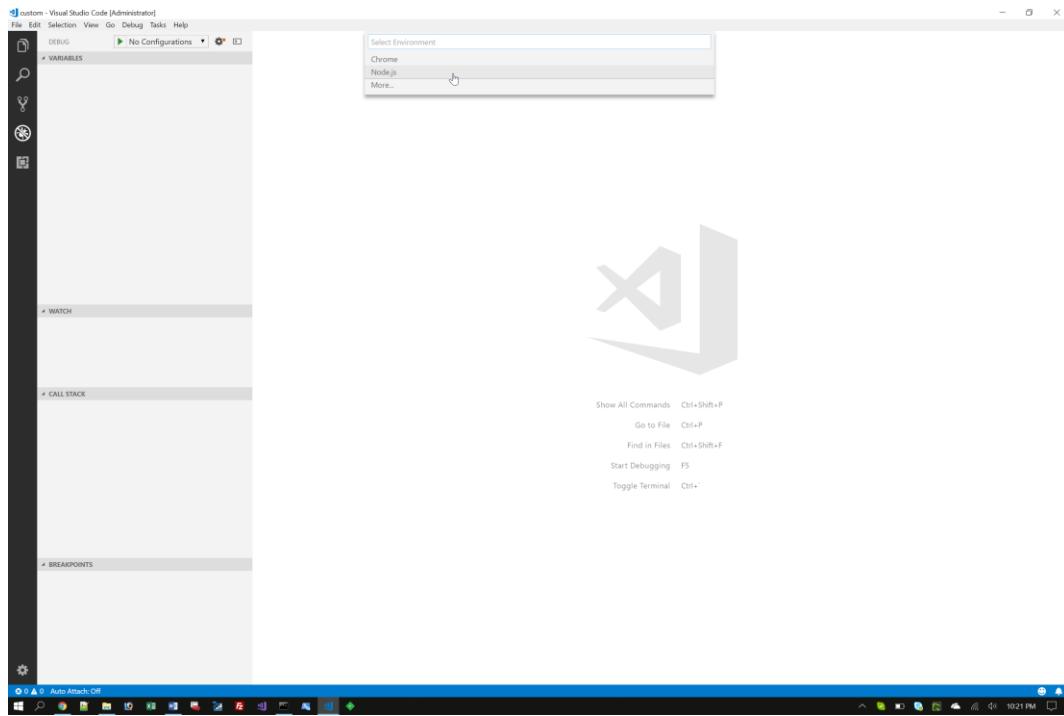
You will be shown the following screen



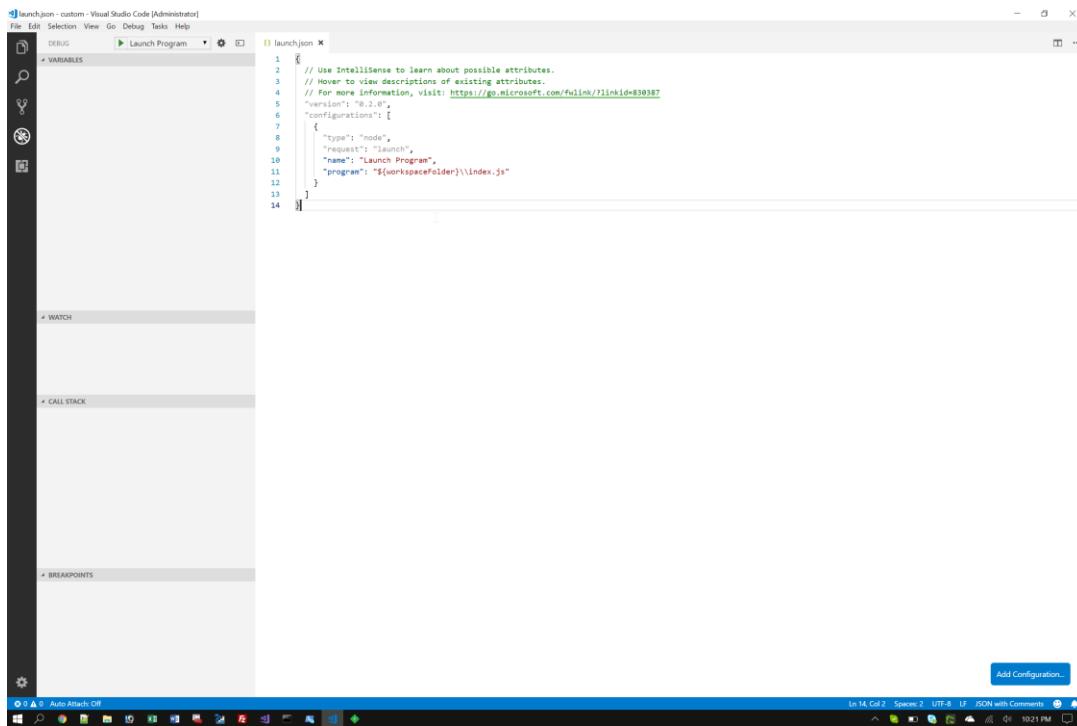
Click on the ‘Gear’ icon as shown below



Right away you will be presented with a drop-down list. Select ‘Node.js’ from that list



The launch.json configuration file will be opened.



The contents of the launch.json file are shown below

```
{  
    // Use IntelliSense to learn about possible attributes.  
    // Hover to view descriptions of existing attributes.  
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
    "version": "0.2.0",  
    "configurations": [  
        {  
            "type": "node",  
            "request": "launch",  
            "name": "Launch Program",  
            "program": "${workspaceFolder}\\index.js"  
        }  
    ]  
}
```

We will need to add a new element to the configurations array with the following contents

```
{  
    "type": "node",  
    "request": "launch",  
    "name": "Debug Alexa With Bespoken Tools",  
    "program": "${workspaceRoot}/node_modules/Bespoken-tools/bin/bst-proxy.js",  
    "args": [  
        "lambda",  
        "index.js"  
    ],  
    "cwd": "${workspaceRoot}"  
}
```

This new element instructs Visual Studio to start the Bespoken Tool ‘bst-proxy.js’ when a new debugging session starts.

Once the new element is added the launch.json file should look like the following

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Program",
            "program": "${workspaceFolder}\\.\\index.js"
        },
        {
            "type": "node",
            "request": "launch",
            "name": "Debug Alexa",
            "program": "${workspaceRoot}/node_modules/Bespoken-tools/bin/bst-proxy.js",
            "args": [
                "lambda",
                "index.js"
            ],
            "cwd": "${workspaceRoot}"
        }
    ]
}
```

At this point you can save the launch.json file.

Visual Studio Code is now completely setup for Alexa debugging with the Bespoken Tools.

Debugging an Alexa Skill using the Bespoken Tools

The Bespoken Tools provide a variety of items to assist Alexa Skill Developers with debugging, testing and monitoring

- Command Line Interface (CLI)
- Monitoring
- Automated Testing
- Logging
- Virtual Device

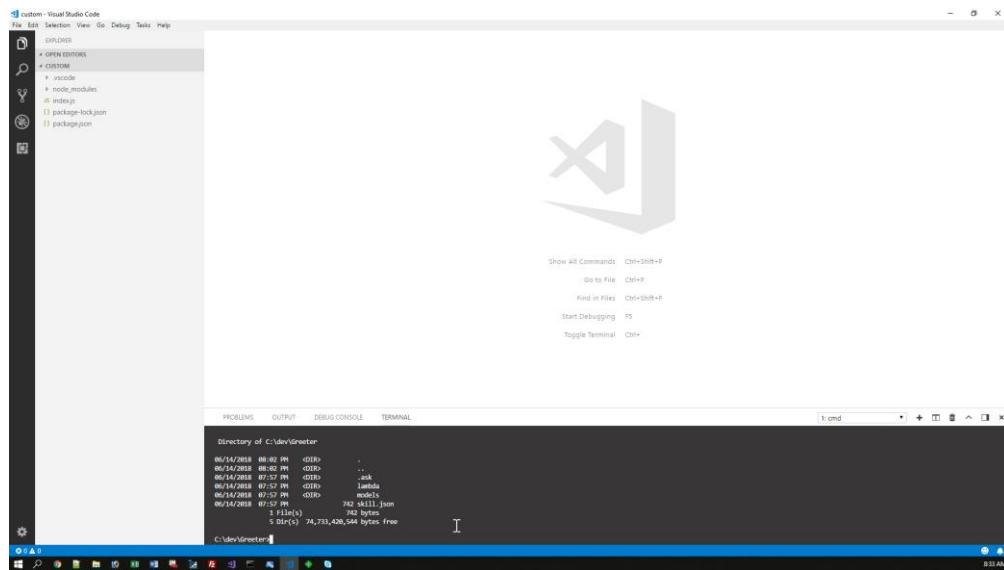
We won't be covering all of these items here; however, we will be looking at the features that specifically assist with the debugging of code.

Using the Bespoken Tools, we have two ways that we can debug our Skill Service code

- Send commands directly to our code using the Bespoken CLI
- Send commands from an Alexa Device using the Bespoken Proxy

Debugging Alexa Skill Service code using the Bespoken CLI

Back in Visual Studio Code, use the 'Integrated Terminal' to navigate to the root of the Skill which is where the skill.json file is located



Next, open the index.js file as shown below

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `index.js`, `package.json`, and `node_modules`.
- Code Editor:** Displays the `index.js` file content, which is a Node.js script for an Alexa skill. It includes imports for `ask-sdk-core`, defines launch and intent handlers, and handles speech text responses.
- Terminal:** Shows the command line interface with the path `C:\dev\Greeter` and the output of a command that lists files and their sizes.
- Status Bar:** Shows the current file is `index.js`, the terminal tab is active, and the status bar indicates `Line 31, Col 49`, `Spaces 2`, `UTF-8`, and `JavaScript`.

Locate the code for the ‘LaunchRequestHandler’ and place a break point on the line that reads

```
const speechText = 'Welcome to the Alexa Skills Kit, you can say hello!';
```

```
index.js  x
1 //silent-disable func-name */
2 /*silent-disable no-console*/
3
4 const Alexa = require('ask-sdk-core');
5
6 const LaunchRequestHandler = {
7   canHandle(handlerInput) {
8     return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
9   },
10  handle(handlerInput) {
11    const speechText = 'Welcome to the Alexa Skills Kit, you can say hello!';
12
13    return handlerInput.responseBuilder
14      .speak(speechText)
15      .reprompt(speechText)
16      .withSimpleCard('Hello World', speechText)
17      .getResponse();
18  },
19}
20
21 const HelloIntentHandler = {
22  canHandle(handlerInput) {
23    return handlerInput.requestEnvelope.request.type === 'IntentRequest'
24      && handlerInput.requestEnvelope.request.intent.name === 'HelloWorldIntent';
25  },
26  handle(handlerInput) {
27    const speechText = 'Hello world!';
28
29    return handlerInput.responseBuilder
30      .speak(speechText)
31      .withSimpleCard('Hello World', speechText)
32      .getResponse();
33  },
34}
35
36 const HelpIntentHandler = {
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Directory of C:\dev\viewreeter

| | | |
|---------------------|-----------------|---------------------------|
| 06/14/2018 08:02 PM | <DIR> | . |
| 06/14/2018 08:02 PM | <DIR> | .. |
| 06/14/2018 08:02 PM | ask | |
| 06/14/2018 07:57 PM | lambda | |
| 06/14/2018 07:57 PM | nodejs | |
| 06/14/2018 07:57 PM | 242_201411.json | |
| | 1 File(s) | 742 bytes |
| | 5 Dir(s) | 24,733,439,544 bytes free |

C:\dev\viewreeter[]

Line 11, Col 78 (24 selected) Spaces 2 LF-F 1F NewLineScript 84JS

At this point it is time to start the Visual Studio Code Debugger. Click on the ‘Debug’ icon in the left-hand menu and make sure that ‘Debug Alexa’ is chosen as the configuration. Click on the green arrow or press F5 to start the Debugger

```

1  /* eslint-disable func-names */
2  /* eslint-disable no-console */
3
4  const Alexa = require('ask-sdk-core');
5
6  const LaunchRequestHandler = {
7    canHandle(handlerInput) {
8      return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
9    },
10   handle(handlerInput) {
11     const speechText = 'Welcome to the Alexa Skills Kit, you can say hello!';
12
13     return handlerInput.responseBuilder
14       .speak(speechText)
15       .reprompt(speechText)
16       .withSimpleCard('Hello World', speechText)
17       .getResponse();
18   },
19 };
20
21 const HelloWorldIntentHandler = {
22   canHandle(handlerInput) {
23     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
24       && handlerInput.requestEnvelope.request.intent.name === 'HelloWorldIntent';
25   },
26   handle(handlerInput) {
27     const speechText = 'Hello World!';
28
29     return handlerInput.responseBuilder
30       .speak(speechText)
31       .withSimpleCard('Hello World', speechText)
32       .getResponse();
33   },
34 };
35
36 const HelpIntentHandler = {
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1
```

The screenshot shows the Visual Studio Code interface with the Alexa Skills Kit debugger extension open. The left sidebar includes tabs for 'index.js' (selected), 'WATCH', 'CALL STACK', and 'BREAKPOINTS'. The main editor area displays the code for 'index.js', which handles LaunchRequest and Intent requests for a skill named 'HelloWorldIntent'. The 'WATCH' tab shows variables like 'this' (Object), 'handlerInput' (Object), and 'speechText' (undefined). The 'CALL STACK' tab lists frames from 'node.js' and 'DefaultHandlerAdapter'. The 'BREAKPOINTS' tab shows breakpoints at line 16 ('const HelpIntentHandler = {'). The bottom status bar indicates the file is 'index.js', the current line is '1: node', and the terminal shows 'C:\Dev\Directive\bst launch' and 'BST: v1.2.11 Node: v8.6.0'.

```
index.js:1
1 // eslint-disable func-names
2 // eslint-disable no-console
3
4 const Alexa = require('ask-sdk-core');
5
6 const LaunchRequestHandler = (
7   commandHandlerInput =>
8     return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
9   );
10
11 const HelpIntentHandler = (
12   commandHandlerInput =>
13   const speechText = 'Welcome to the Alexa Skills Kit, you can say hello!';
14   return handlerInput.responseBuilder
15     .speak(speechText)
16     .reprompt(speechText)
17     .withSimpleCard('Hello World', speechText)
18     .getResponse();
19   );
20
21 const HelloWorldIntentHandler = (
22   commandHandlerInput =>
23   if (handlerInput.requestEnvelope.request.type === 'IntentRequest'
24     && handlerInput.requestEnvelope.request.intent.name === 'HelloWorldIntent');
25   );
26
27 handle(handlerInput) {
28   const speechText = 'Hello World!';
29   const HelpIntentHandler = {
30     handle(handlerInput) {
31       return handlerInput.responseBuilder
32         .speak(speechText)
33         .withSimpleCard('Hello world', speechText)
34         .getResponse();
35     };
36   }
37   const HelpIntentHandler = {
```

We can now Step Into, Step Over, inspect variables etc. and fully debug the ‘LaunchRequestHandler’ portion of this Skill Service Code.

Press F5 to let the code continue. In the Terminal window you will see the actual JSON Request (red) that was sent to the ‘LaunchRequestHandler’ and the JSON Response (blue) that it returned.

```
C:\dev\Greeter>bst launch
BST: v1.2.11 Node: v8.6.0

Request:
{
  "context": {
    "System": {
      "application": {
        "applicationId": "amzn1.echo-sdk-ams.app.9a57d520-371a-4541-9b71-39897290a667"
      },
      "device": {
        "supportedInterfaces": {
          "AudioPlayer": {}
        }
      },
      "user": {
        "userId": "amzn1.ask.account.67e0245b-bd9c-4c47-9178-dd792b7b5f18"
      }
    },
    "AudioPlayer": {
      "playerActivity": "IDLE"
    }
  }
}
```

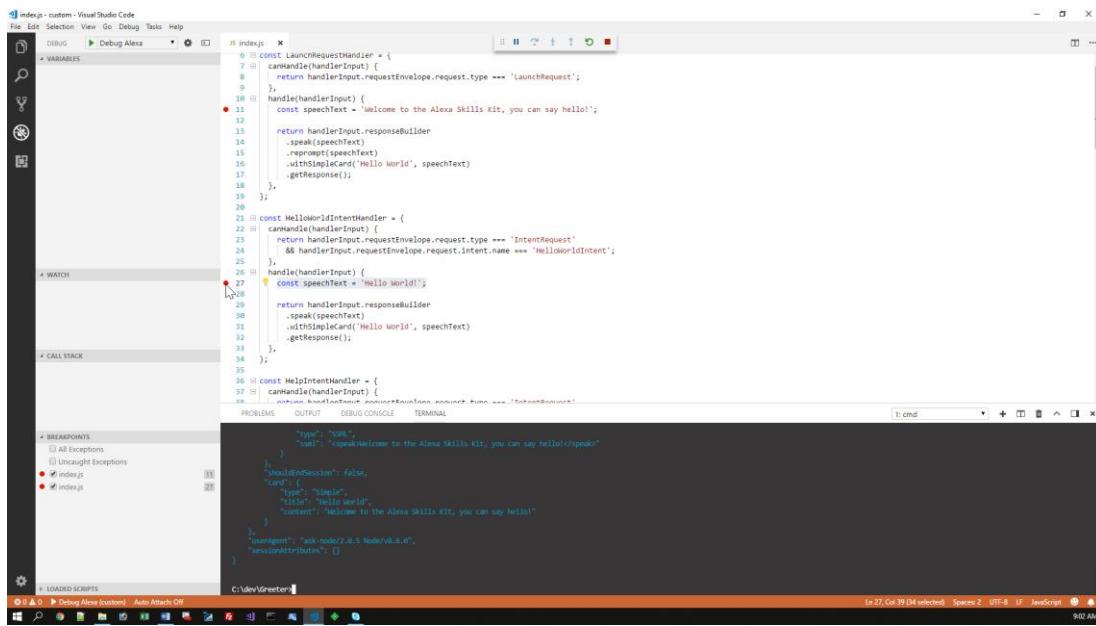
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: cmd + - ×

Response:
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "type": "SSML",
      "ssml": "<speak>Welcome to the Alexa Skills Kit, you can say hello!</speak>"
    },
    "reprompt": {
      "outputSpeech": {
        "type": "SSML",
        "ssml": "<speak>Welcome to the Alexa Skills Kit, you can say hello!</speak>"
      }
    },
    "shouldEndSession": false,
    "card": {
      "type": "Simple",
      "title": "Hello World",
      "content": "Welcome to the Alexa Skills Kit, you can say hello!"
    }
  },
  "userAgent": "ask-node/2.0.5 Node/v8.6.0",
}
```

That works great for the Launch Request, but we can also do the same kind of testing with the Intent Handlers in the code.

Add a break point to the following ‘HelloWorldIntentHandler’ code

```
const speechText = 'Hello World!';
```

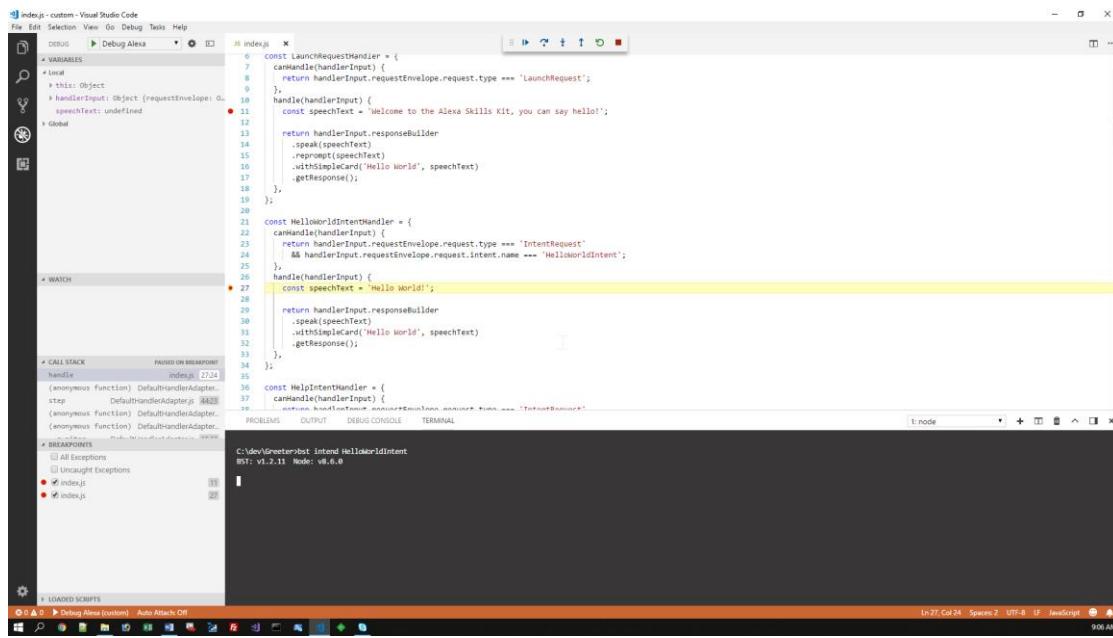


In the ‘Integrated Terminal enter the following

```
bst intend HelloWorldIntent
```



After you hit ‘Enter’, you will notice that the ‘HelloWorldIntentHandler’ code has been fired and Visual Studio Code has stopped on the break point that was previously set.



Again, we can now Step Into, Step Over, inspect variables etc. and fully debug the ‘HelloWorldIntentHandler’ portion of this Skill Service Code.

Press F5 to let the code continue. In the Terminal window you will see the actual JSON Request (red) that was sent to the ‘HelloWorldIntentHandler’ and the JSON Response (blue) that it returned.

The terminal window displays the following JSON output in two parts:

```

Request:
{
  "context": {
    "System": {
      "application": {
        "applicationId": "amzn1.echo-sdk-ams.app.c6e4de29-2ad9-4ef6-af9e-a269e2eb720c"
      },
      "device": {
        "supportedInterfaces": {
          "AudioPlayer": {}
        }
      }
    },
    "user": {
      "userId": "amzn1.ask.account.b35db91d-be1b-4a24-a659-3123d1ce1859"
    }
  }
}

Response:
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "type": "SSML",
      "ssml": "<speak>Hello World!</speak>"
    },
    "card": {
      "type": "Simple",
      "title": "Hello World",
      "content": "Hello World!"
    }
  },
  "userAgent": "ask-node/2.0.5 Node/v8.6.0",
  "sessionAttributes": {}
}

```

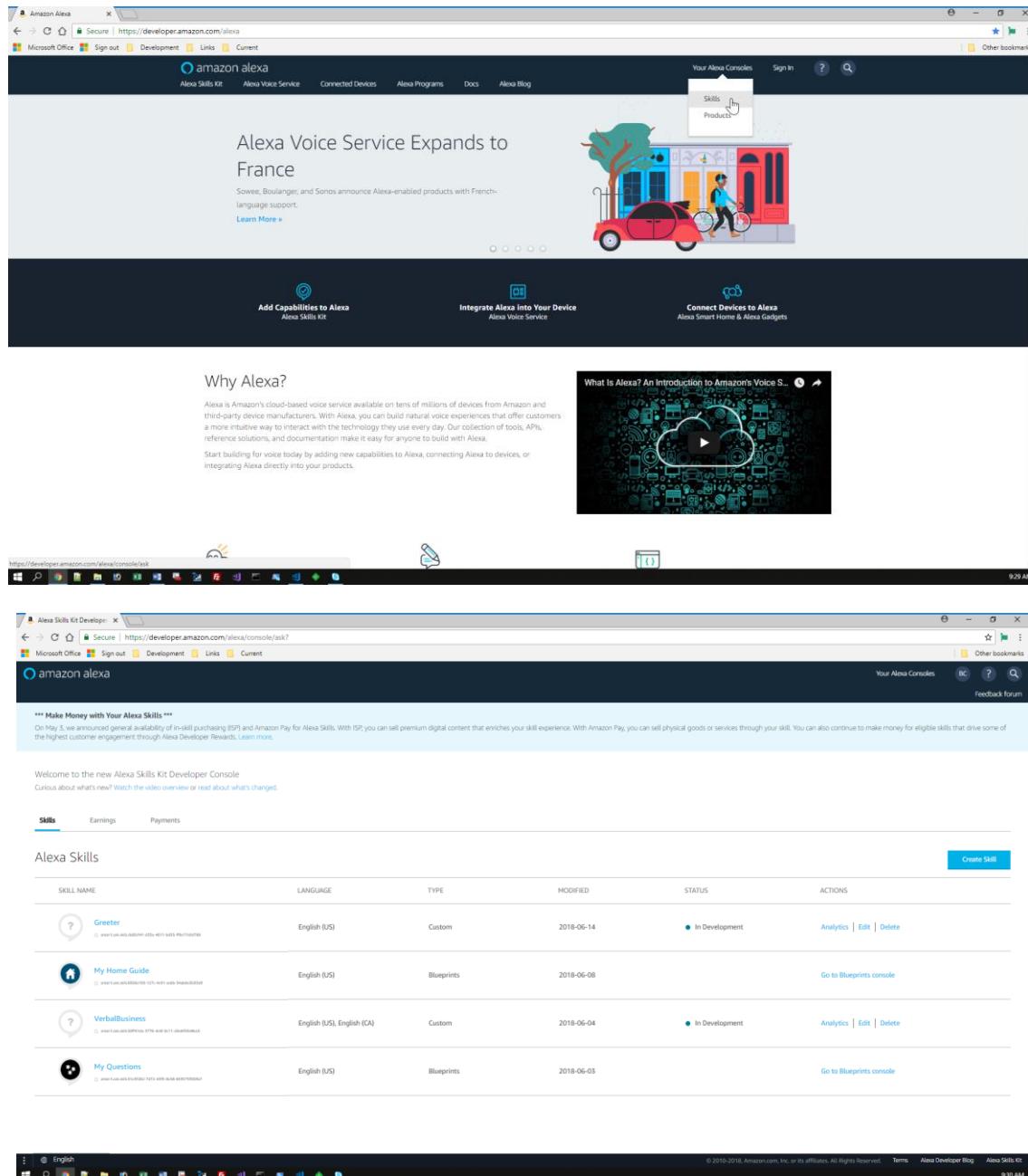
The terminal status bar indicates the file is `index.js`, line `Ln 32 Col 22`, character `Spaces: 2`, encoding `UTF-8`, and language `JavaScript`.

Debug an Alexa Skill using the Bespoken Proxy

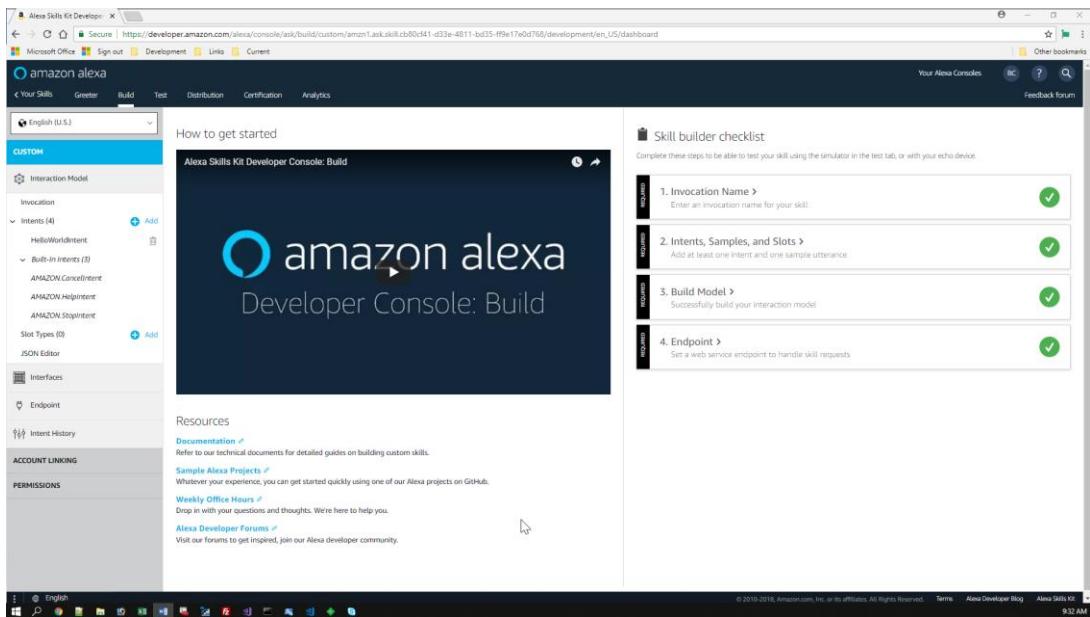
Using the Bespoken Proxy, we can actually do end to end testing of our Alexa Skill. To start off with, open a browser and navigate to the Alexa Skill Console at

<https://developer.amazon.com>

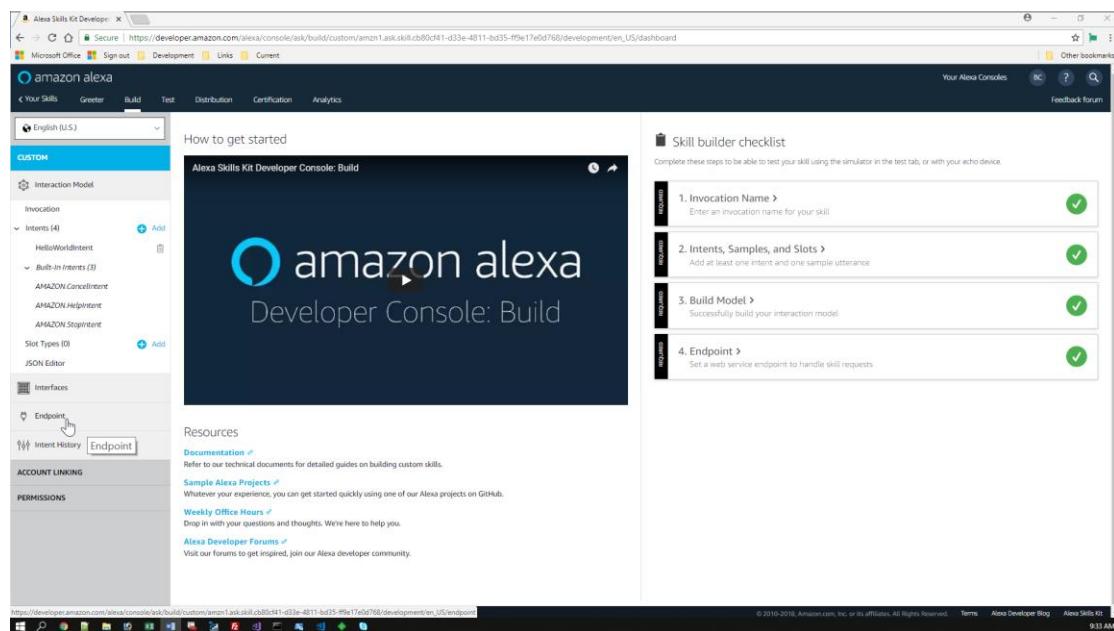
as shown below



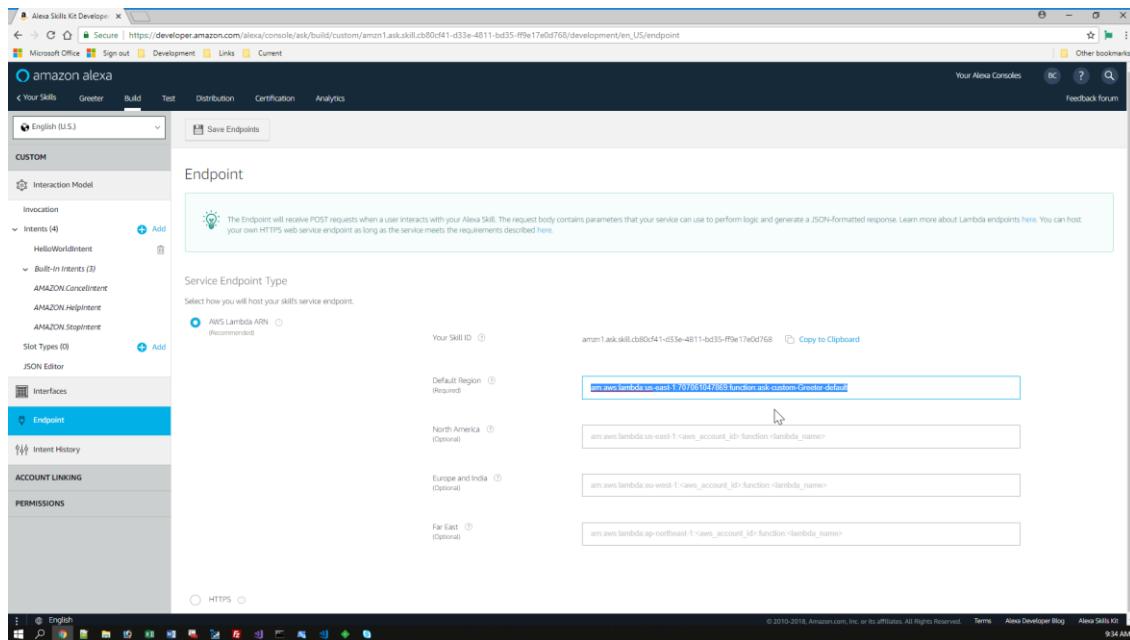
Click on the ‘Greeter Skill’ and you will be taken to the following page



Click on the ‘Endpoint’ item in the left-hand menu



In the resulting page, notice that the Endpoint is currently set to a value that points to the cloud version of the Skill Service. This was set when we initially deployed the Alexa Skill using the ASK CLI.



Now to debug the Skill Service Code locally, we will need to divert the Requests coming from the Skill Interface portion of the Skill to our local machine.

We can do that using the Bespoken Proxy.

Back in Visual Studio Code, the debugger should still be running. If not, you will need to start it again.

Next to the Integrated Terminal' you will find a 'Debug Console' tab. Click on that tab and you will be shown the following

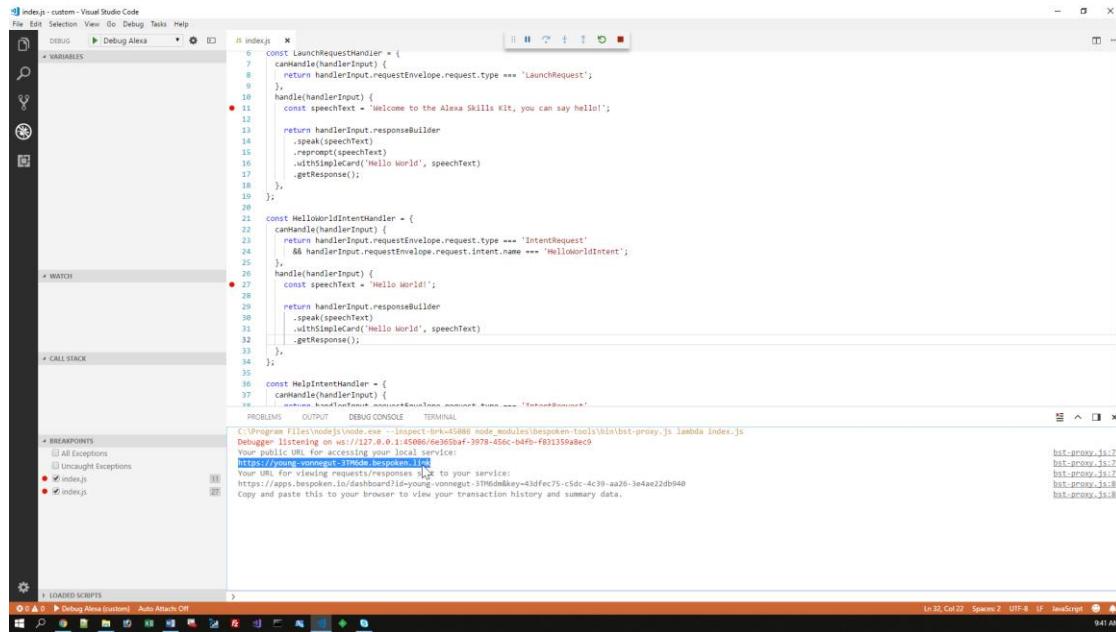
The screenshot shows the Visual Studio Code interface with the following details:

- Editor:** The main area displays the `index.js` file for an Alexa Skill. It contains several function definitions for handling different types of requests (LaunchRequest, IntentRequest, HelpIntent) and their corresponding responses.
- Breakpoints:** Breakpoints are set at various points in the code, indicated by red dots. One breakpoint is explicitly labeled "Break on Response".
- Terminal:** The bottom right terminal window shows the deployment process:
 - It starts with the command: `C:\Users\jason\source\repos\HelloWorld\lambda\bin\lambda deploy --stage test`.
 - It shows the AWS Lambda function being deployed: `Uploading /var/task/handler.js (111 B)`.
 - It displays the deployment ID: `Deployment ID: 3098-455c-04f9-ff81359a8e49`.
 - It shows the public URL for the skill: `Your public URL for accessing your local service: https://appp.bespoken.io/dashboard?tid=young-vonnegot-3T96de&key=A3dfec75-5dC-4c39-a226-3edae22d0940`.
 - It concludes with the instruction: `Copy and paste this to your browser to view your transaction history and summary data.`
- Bottom Status Bar:** Shows the current file is `index.js`, the status bar indicates "In 32, Col 22", and the bottom right corner shows "Java Script".

Because of the configuration changes that we made earlier to Visual Studio Code, the Bespoken Proxy starts each time the debugger is invoked. As you can see from the screen shot, we are given a public URL for accessing our local code. (yours of course will be different than what is show below)

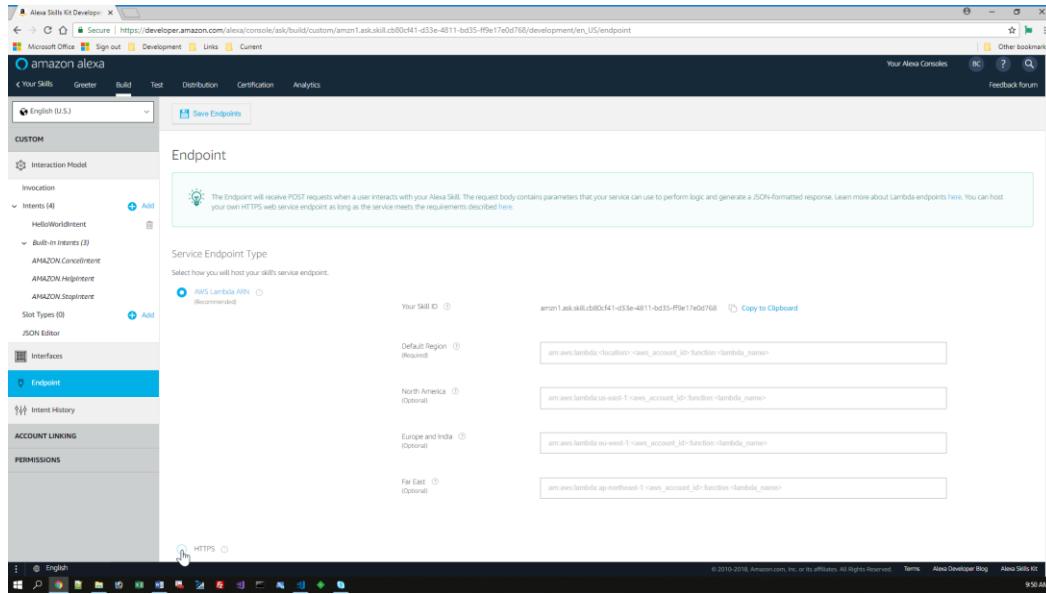
In my case the public URL is

<https://young-vonnegut-3TM6dm.Bespoken.link>

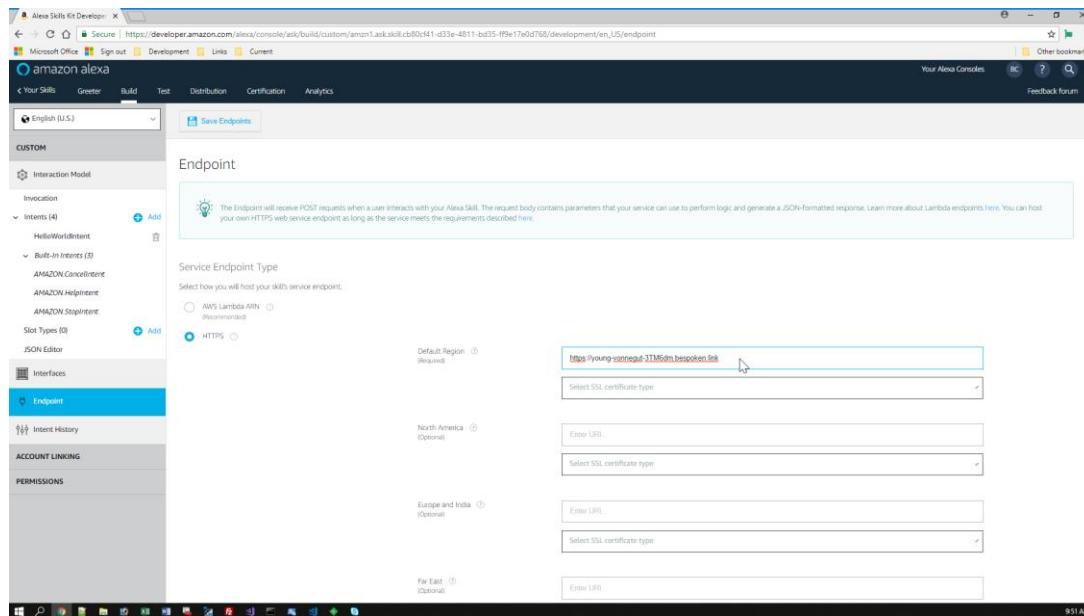


This Public URL was created when the Bespoken Tools were initially installed, and it will not change. So, we can use this Public URL to divert the Requests from the Alexa Skill Interface to our local Alexa Skill Service Code.

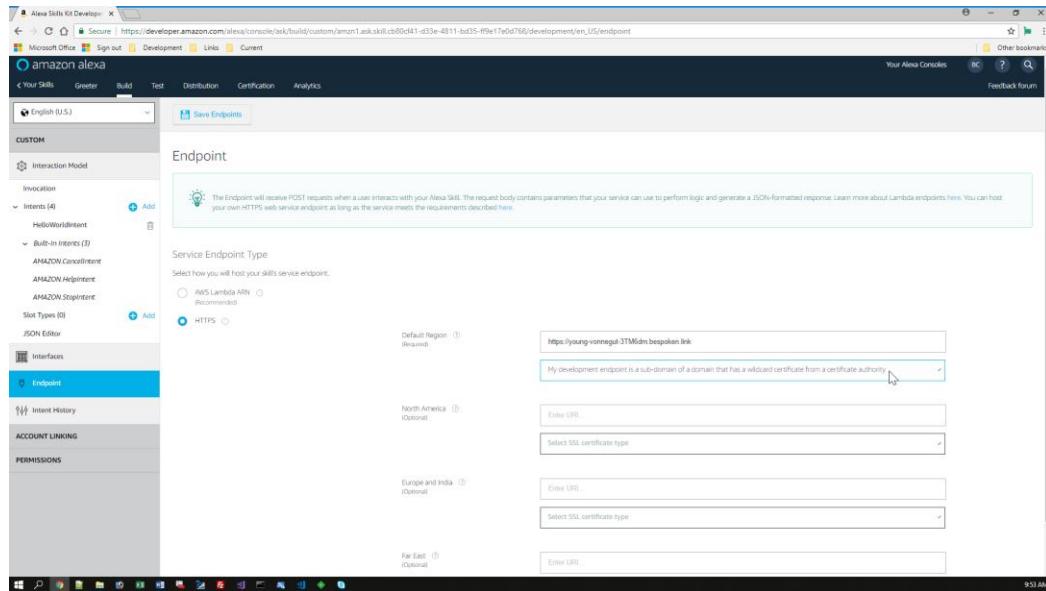
Returning to the Alexa Skill Console, change the Endpoint to a URL by clicking on the URL Radio Button



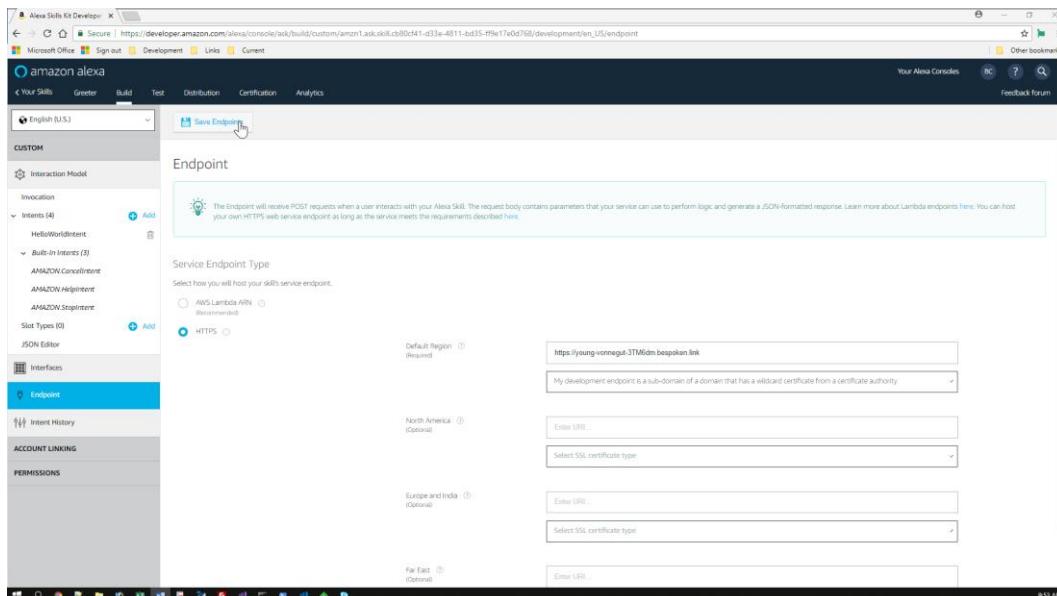
Enter the Public URL value that we were assign by the Bespoken Proxy. (make sure you use the one that was assigned to you and not the one that was assigned to me)



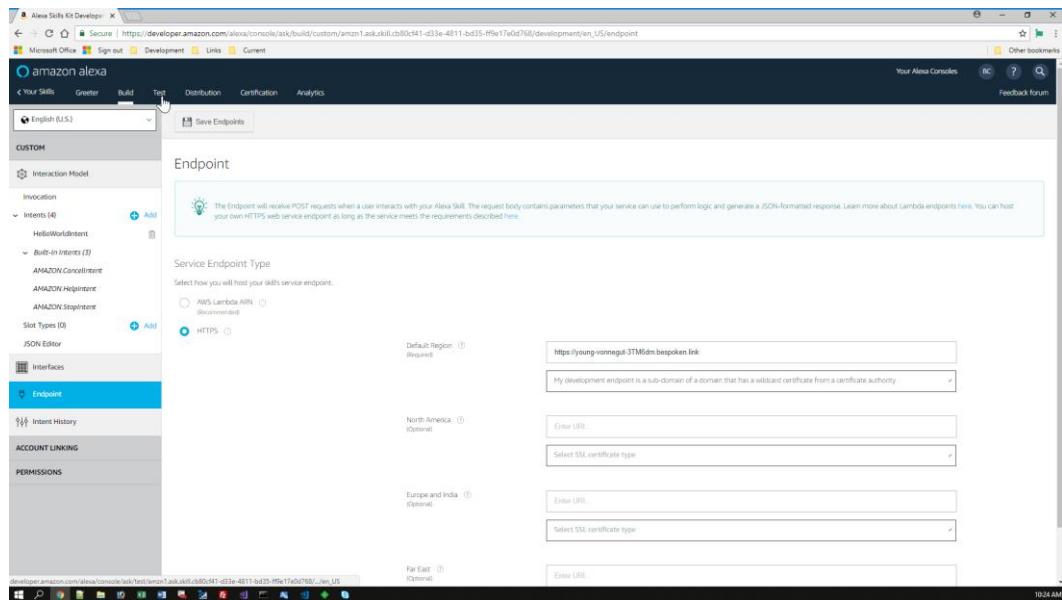
Select ‘My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority’ as the Certificate Type.



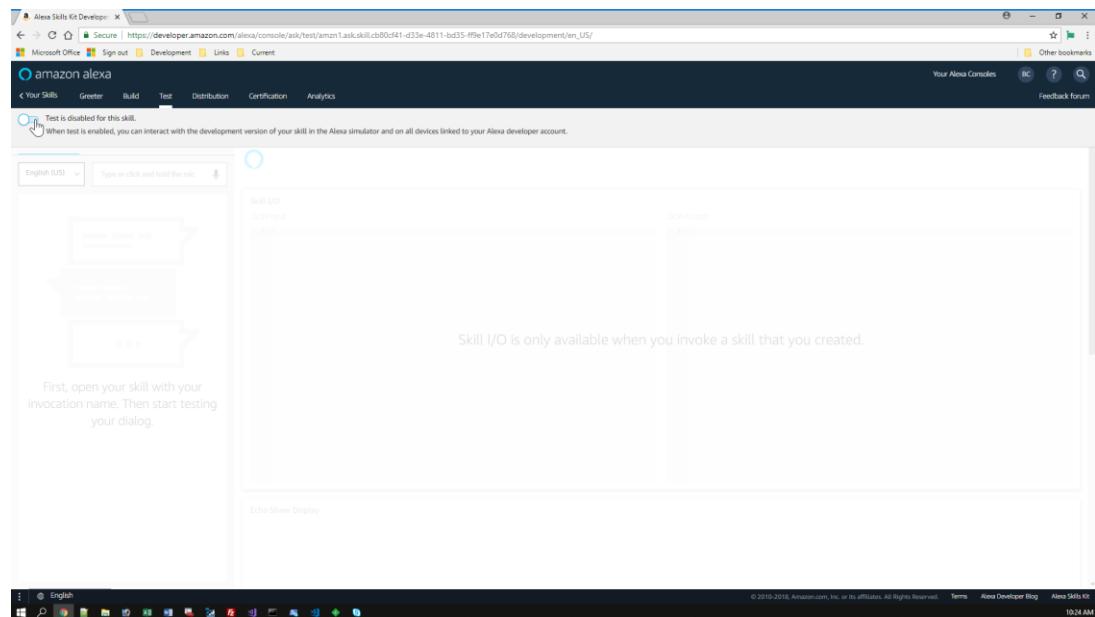
Finally click the ‘Save Endpoints’ button as shown below



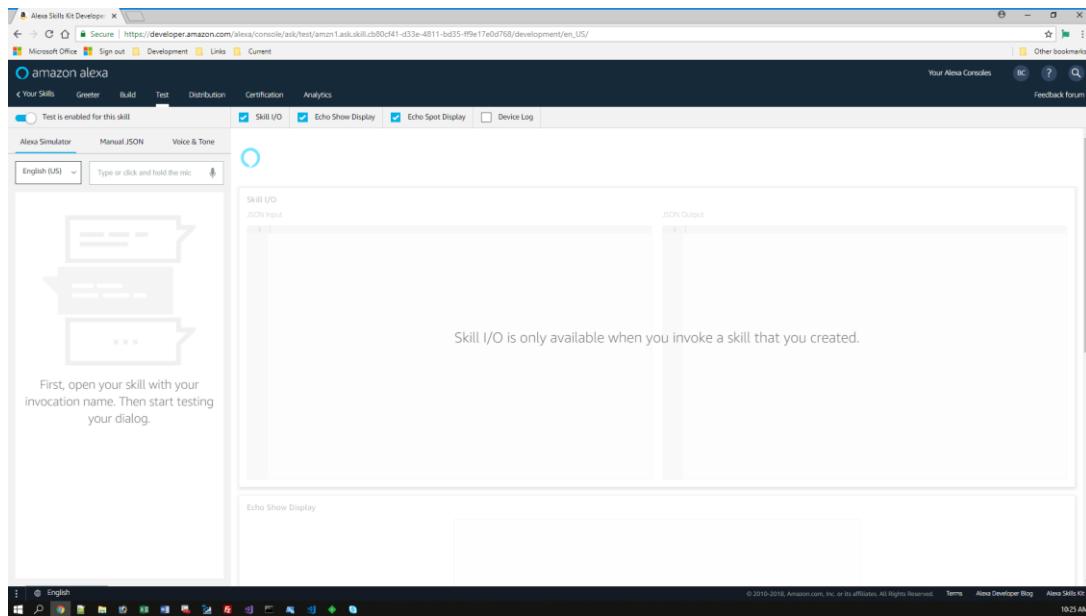
Next, click on the ‘Test’ tab as shown below



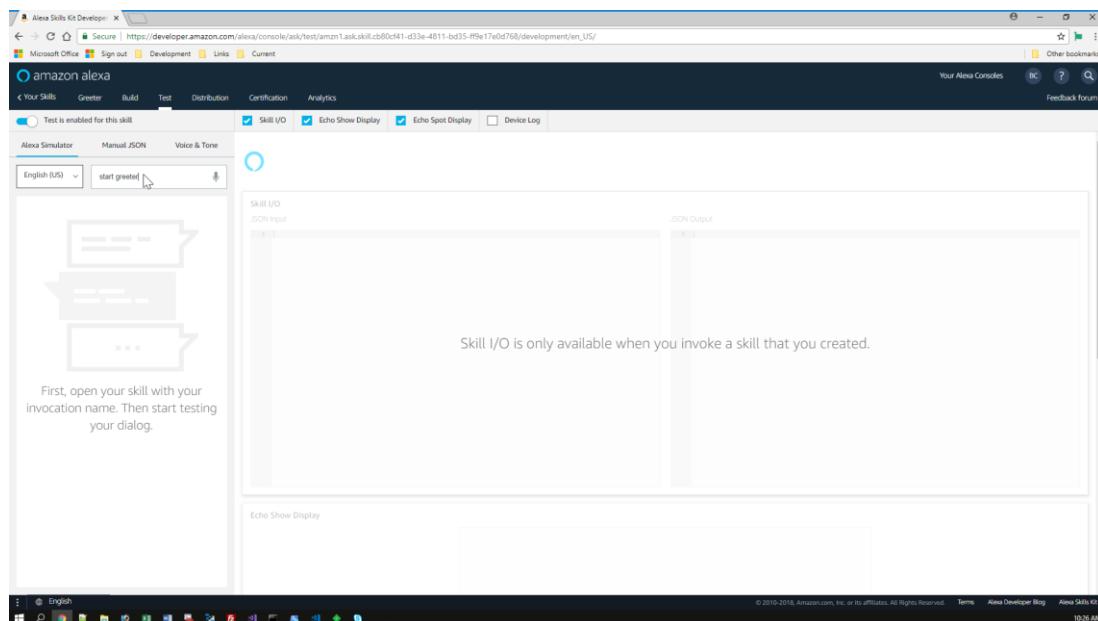
Enable testing for the Skill by clicking on the enable testing button as shown below



You will be shown the following



In the speech input box, enter 'start greeter' as shown below



Now press Enter.

Back in Visual Studio Code you will see that the ‘LaunchRequest’ handler in our local code has been fired and our break point has been hit.

The screenshot shows the Visual Studio Code interface with the 'Debug' tool bar at the top. The 'Debug' tab is selected, and the 'Debug Alexa' configuration is chosen. The main editor area displays the 'index.js' file for the Alexa Skill. The code implements a LaunchRequestHandler and an IntentRequestHandler for the 'HelloWorldIntent'. A yellow highlight is on line 11, which contains the speechText assignment. The bottom status bar indicates the current file is 'index.js'.

```
/* eslint-disable func-names */
/* eslint-disable no-console */

this: Object
handlerInput: Object {requestEnvelope: Object, speechlets: undefined}
Global

 1 const Alexa = require('ask-sdk-core');
 2
 3 const LaunchRequestHandler = {
 4   canHandle(handlerInput) {
 5     return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
 6   },
 7   handle(handlerInput) {
 8     const speechText = ['Welcome to the Alexa Skills Kit, you can say hello!'];
 9
10     return handlerInput.responseBuilder
11       .speak(speechText)
12       .reprompt(speechText)
13       .withSimpleCard('Hello World', speechText)
14       .getResponse();
15   },
16 }
17
18
19
20
21 const HelloWorldIntentHandler = {
22   canHandle(handlerInput) {
23     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
24       && handlerInput.requestEnvelope.request.intent.name === 'HelloWorldIntent';
25   },
26   handle(handlerInput) {
27     const speechText = 'Hello World!';
28
29     return handlerInput.responseBuilder
30       .speak(speechText)
31       .withSimpleCard('Hello World', speechText)
32       .getResponse();
33   },
34 }
35

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

C:\Program Files\Nodejs\node.exe --inspect-brk=670 node_modules\bespoken-tools\bin\bst-proxy.js lambda index.js
Debugger listening on ws://127.0.0.1:6700/bst-proxy-7999-4000-c63ebc4dc37e
Your public URL for access to your local service:
<https://young-vonnegut-37966.bespoken.link>
Your URL for viewing requests/responses sent to your service:
<https://app.bespoken.io/dashboard?tid=young-vonnegut-37966&mkkey=43dfec75-c5dc-4c39-aa26-3e4ae22d948>
Copy and paste this to your browser to view your transaction history and summary data.

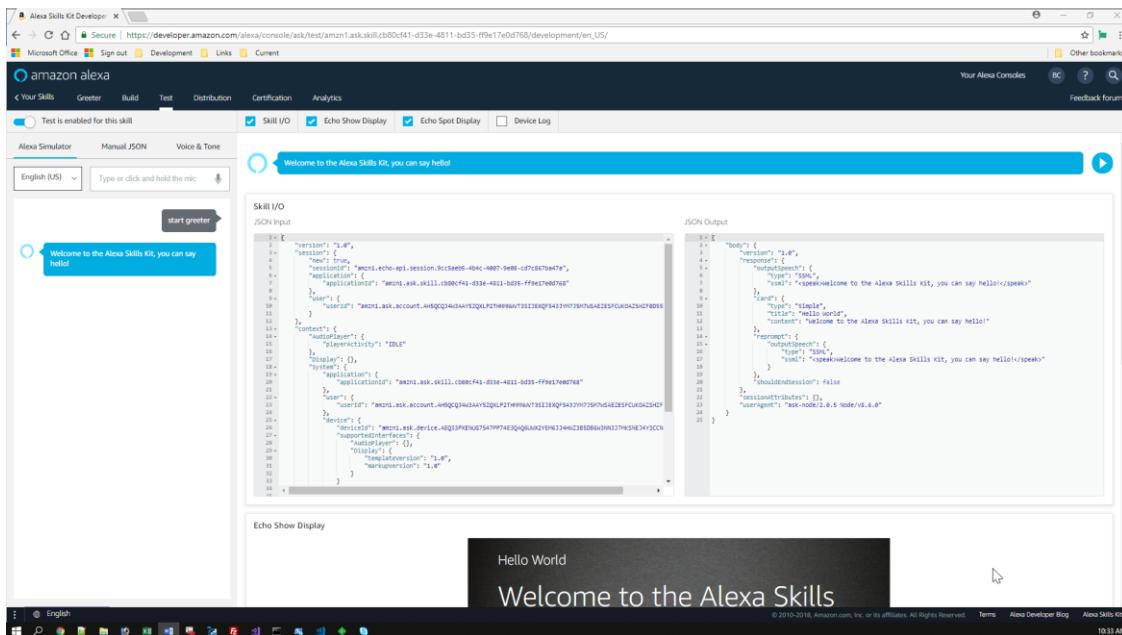
LOADED SCRIPTS

Debug Alexa (extension) Auto Attach Off

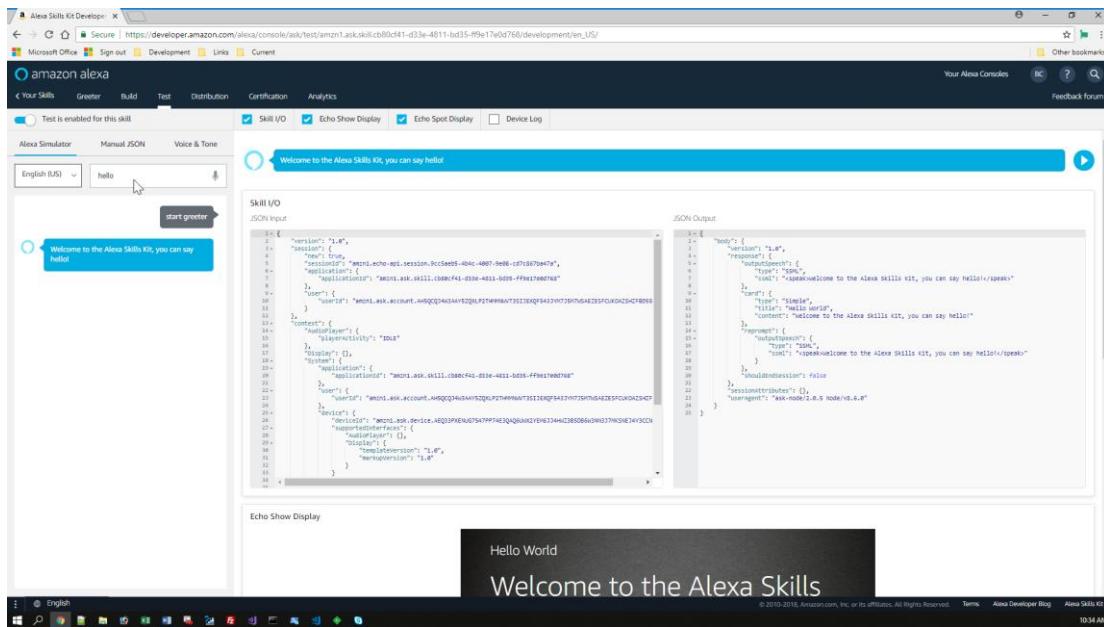
In 11, Col 24 Specs 2 /UIT-8 /F JavaScript

Press F5 to allow the code to continue past our break point

Back in the Alexa Skill Console, you can see the Request and Response that were exchanged between the Alexa Skill Interface hosted in the cloud and our local Alexa Skill Service. You will also see a representation of what would be shown on an Echo Show and Echo Spot device.

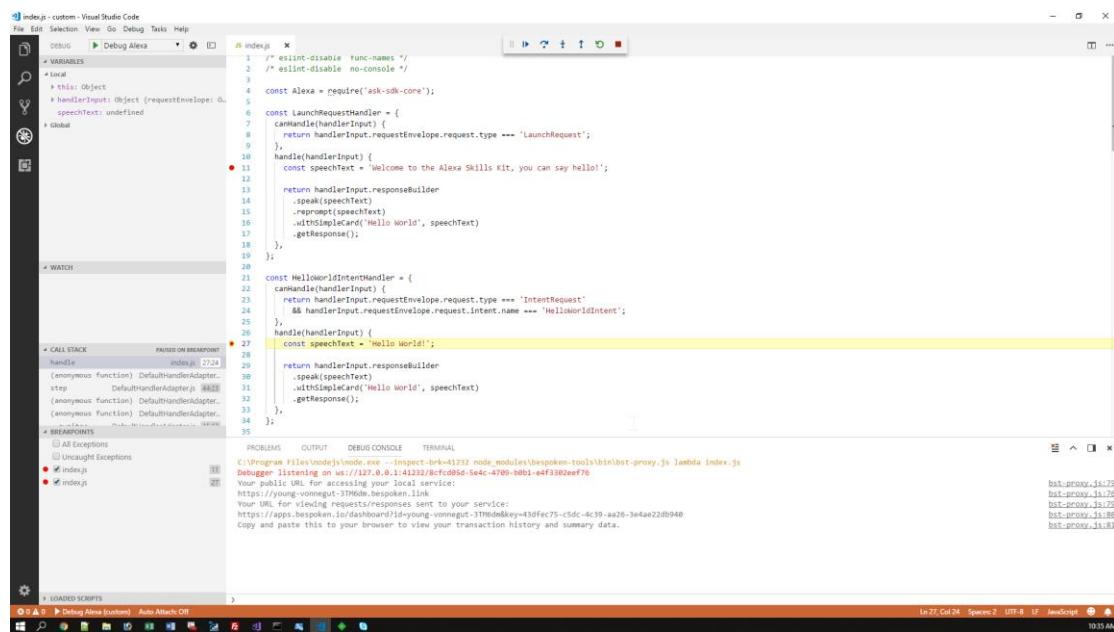


In the speech input box, enter ‘Hello’ as shown below



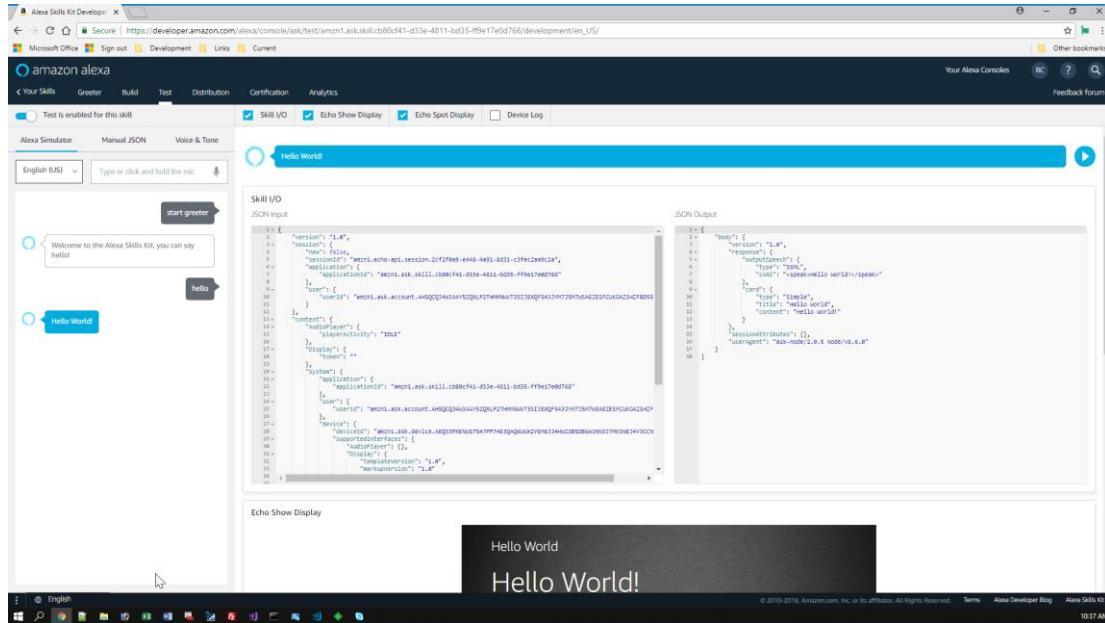
Now press Enter.

Back in Visual Studio Code you will see that the ‘HelloWorldIntent’ handler in our local code has been fired and our break point has been hit.



Press F5 to allow the code to continue past our break point

Back in the Alexa Skill Console, you can again see the Request and Response that were exchanged between the Alexa Skill Interface hosted in the cloud and our local Alexa Skill Service. You will also see a representation of what would be shown on an Echo Show and Echo Spot device.



Taking this one step further, we can make a change in the local Skill Service code and it will be immediately available for testing.

In Visual Studio Code, stop the debugger and then update the ‘LaunchRequestHandler’ code from

```
const speechText = 'Welcome to the Alexa Skills Kit, you can say hello!';
```

to

```
const speechText = 'Welcome to the Alexa Skills Kit, we modified this line';
```

Remove all of the breakpoints in Visual Studio Code and restart the debugger.

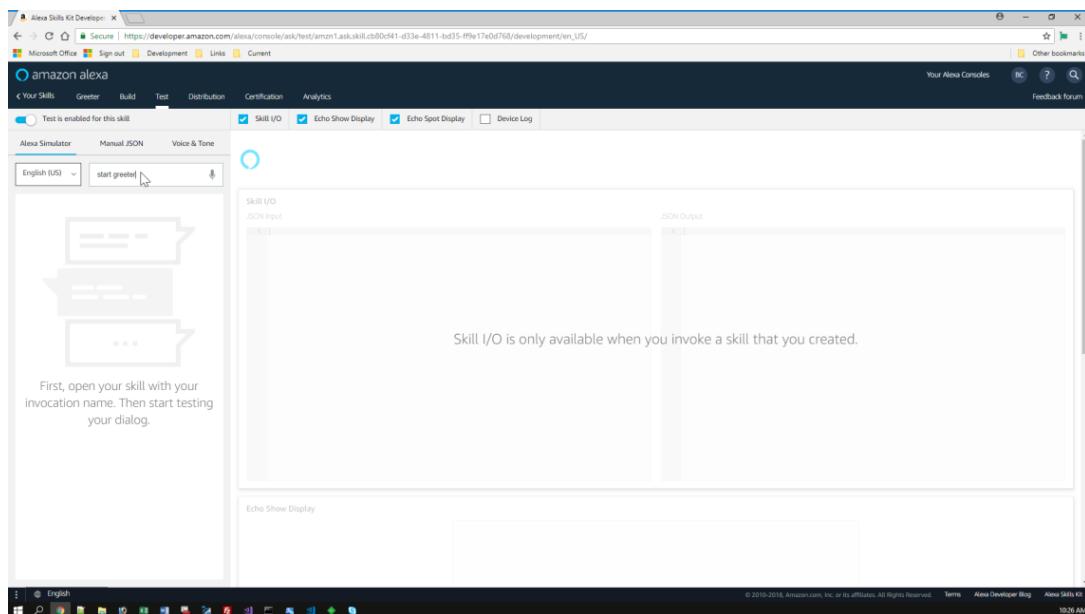
```

1  /* eslint-disable func-names */
2  /* eslint-disable no-console */
3
4  const Alexa = require('ask-sdk-core');
5
6  const LaunchRequestHandler = {
7    canHandle(handlerInput) {
8      return handlerInput.requestEnvelope.request.type === 'LaunchRequest';
9    },
10   handle(handlerInput) {
11     const speechText = "Welcome to the Alexa Skills Kit, we modified this line."
12
13     return handlerInput.responseBuilder
14       .speak(speechText)
15       .reprompt(speechText)
16       .withSimpleCard("Hello World", speechText)
17       .getResponse();
18   },
19 };
20
21 const HelloWorldIntentHandler = {
22   canHandle(handlerInput) {
23     return handlerInput.requestEnvelope.request.type === 'IntentRequest'
24     && handlerInput.requestEnvelope.request.intent.name === 'HelloWorldIntent';
25   },
26   handle(handlerInput) {
27     const speechText = "Hello World!";
28
29     return handlerInput.responseBuilder
30       .speak(speechText)
31       .reprompt("Hello world", speechText)
32       .getResponse();
33   },
34 };
35

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
C:\Program Files\nodejs\node.exe -inspect-brk:37713 node modules\awsspoken-tools\bin\bst-proxy.js lambda index.js
Debugger listening on ws://127.0.0.1:37713/c000030d-91d0-4aa8-80c9-d1b845f9d227
Your URL for accessing your local service:
http://127.0.0.1:37713/c000030d-91d0-4aa8-80c9-d1b845f9d227
Your URL for viewing requests/responses sent to your service:
https://apis.bespooken.io/dashboard?l=yong-vnngcgt-3THmdmKey-43fec75-c5dc-4c39-aa26-3e4aa22db948
Copy and paste this to your browser to view your transaction history and summary data.

In the Alexa Skill Console, in the speech input box, enter ‘start greeter’ as shown below



Now press Enter.

You will see that the Response that we get back from the local Alexa Skill Service does indeed reflect the changes that we just made in our local environment.

The screenshot shows the Alexa Skills Kit Developer Console. In the top navigation bar, there are tabs for Your Skills, Greeter, Build, Test, Distribution, Certification, and Analytics. The 'Test' tab is selected. Below the tabs, there are checkboxes for Test is enabled for this skill, Skill I/O, Echo Show Display, Echo Spot Display, Device Log, and Device Log. The 'Alexa Simulator' tab is selected. Under 'Skill I/O', there are tabs for English (US) and Type or click and hold the mic. A 'start greeter' button is visible. The main area shows 'Welcome to the Alexa Skills Kit, we modified this line'. The 'JSON Input' pane shows the request JSON, and the 'JSON Output' pane shows the response JSON. The response JSON includes a 'body' object with a 'text' key containing the modified welcome message. Below the JSON panes, there is a preview window showing the Echo Show Display with the text 'Hello World' and 'Welcome to the Alexa Skills'. At the bottom of the screen, there is a Windows taskbar with icons for Start, Task View, File Explorer, Edge, and File Explorer. The status bar at the bottom right shows the date and time: © 2010-2018, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms Alexa Developer Blog Alexa Skills Kit 10:32 AM

```
JSON Input
1 <?
2   "version": "1.0",
3   "session": {
4     "new": true,
5     "sessionId": "amzn.echo-api.session.0e911a9-ced7-419b-8c34-d079196211f3",
6     "application": {
7       "applicationId": "amzn.ask.skill.0e911a9-ced7-419b-8c34-d079196211f3"
8     },
9     "user": {
10       "userId": "amzn.ask.account.AE9Q403AAYT5QPLPZT4H9HAA7T211E0QF5437H73PH6AE2ESFLUHQ2H07FB055
11     }
12   },
13   "context": {
14     "user": {
15       "playActivity": "IDLE"
16     },
17     "display": {
18       "styling": {
19         "application": {
20           "applicationId": "amzn.ask.skill.0e911a9-ced7-419b-8c34-d079196211f3"
21         },
22         "userId": "amzn.ask.account.AE9Q403AAYT5QPLPZT4H9HAA7T211E0QF5437H73PH6AE2ESFLUHQ2H07FB055
23       }
24     },
25     "device": {
26       "capabilities": [
27         {
28           "name": "AMAZON_AUDIO_PLAYER"
29         },
30         {
31           "name": "AMAZON_AV_REMOTE"
32         }
33       ]
34     }
35   }
36 >
37
38 <?
39   "body": {
40     "version": "1.0",
41     "response": {
42       "outputSpeech": {
43         "type": "SSML",
44         "text": "Welcome to the Alexa Skills Kit, we modified this line:speak"
45       },
46       "card": {
47         "type": "Simple",
48         "title": "Hello World",
49         "content": "Welcome to the Alexa Skills Kit, we modified this line:speak"
50       },
51       "reprompt": {
52         "outputSpeech": {
53           "type": "SSML",
54           "text": "Welcome to the Alexa Skills Kit, we modified this line:speak"
55         }
56       }
57     },
58     "shouldEndSession": false
59   },
60   "sessionAttributes": {}
61 },
62 "semantics": "ask-node/2.0.5 node/v8.4.8"
63
64 >
```

Finally, if you don't feel like typing commands into the speech input box in the Alexa Skill Console to test the code, you can verbally issue commands to your Alexa Echo Device. The breakpoints in Visual Studio Code will also be hit using this method.

Summary

That completes the setup of our Local Development Environment for Alexa Skills. This setup now gives us the ability to do complete end to end testing and debugging of Alexa Skill code using the Alexa Skills Console, Visual Studio Code and the Bespoken Tools.

The setup of course takes a bit of time, but the time that can be saved during the testing and debugging phase is well worth the effort.