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

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

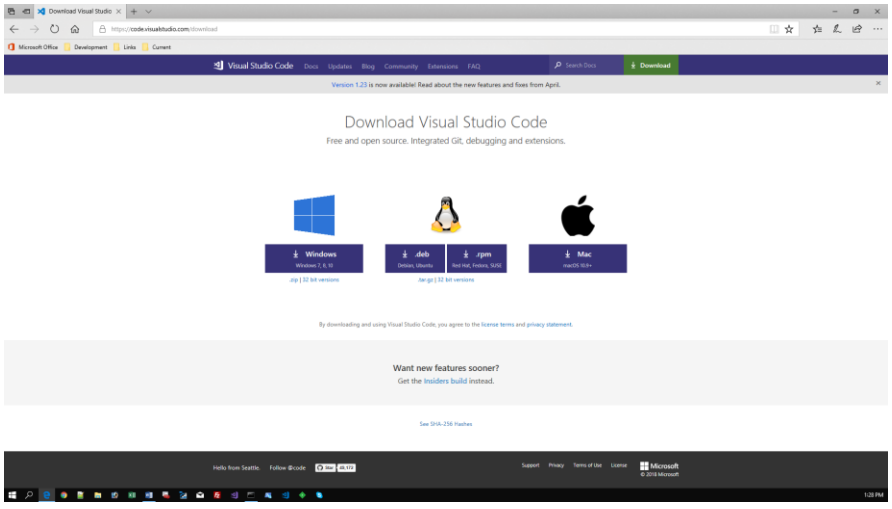
Installing Visual Studio Code

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.

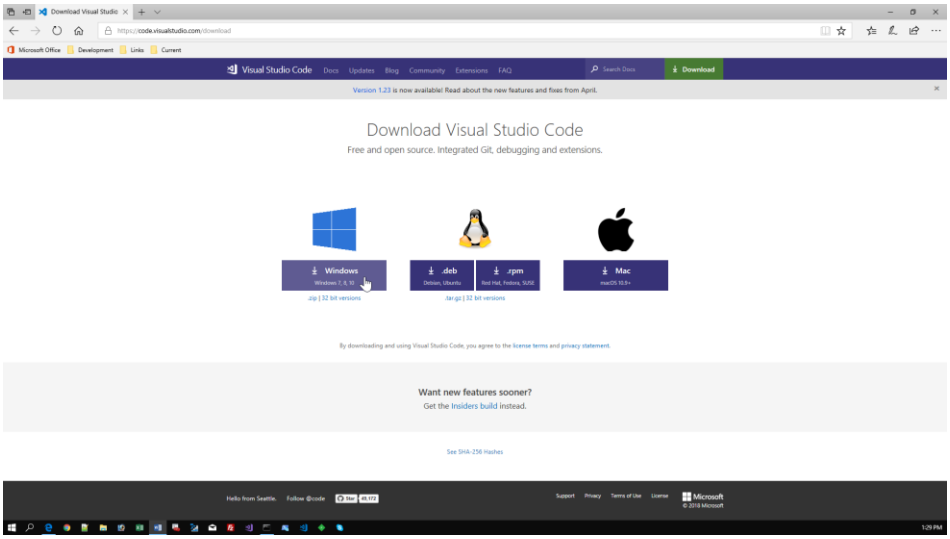
Open a browser and navigate to

https://code.visualstudio.com/download

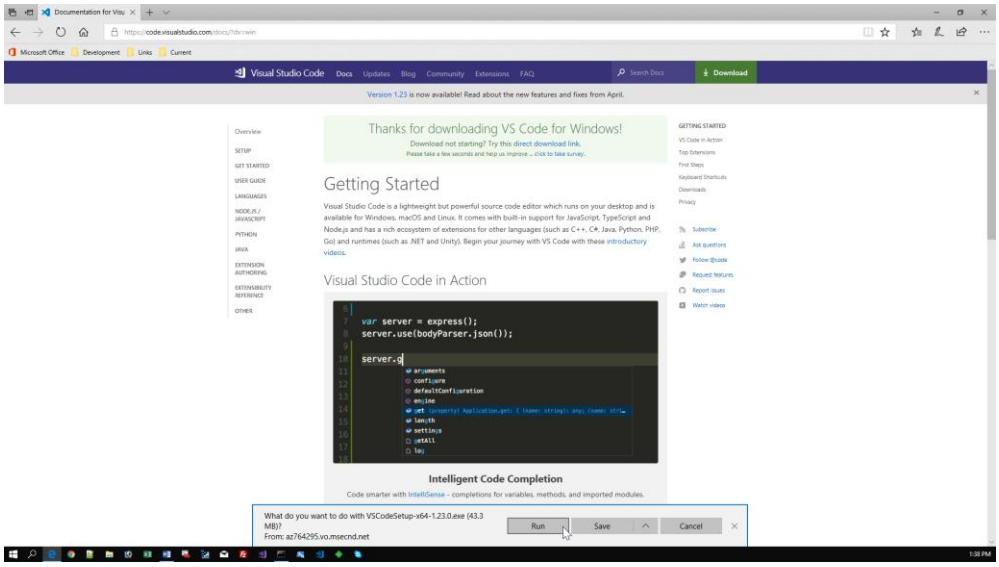
You will be taken to the following page



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

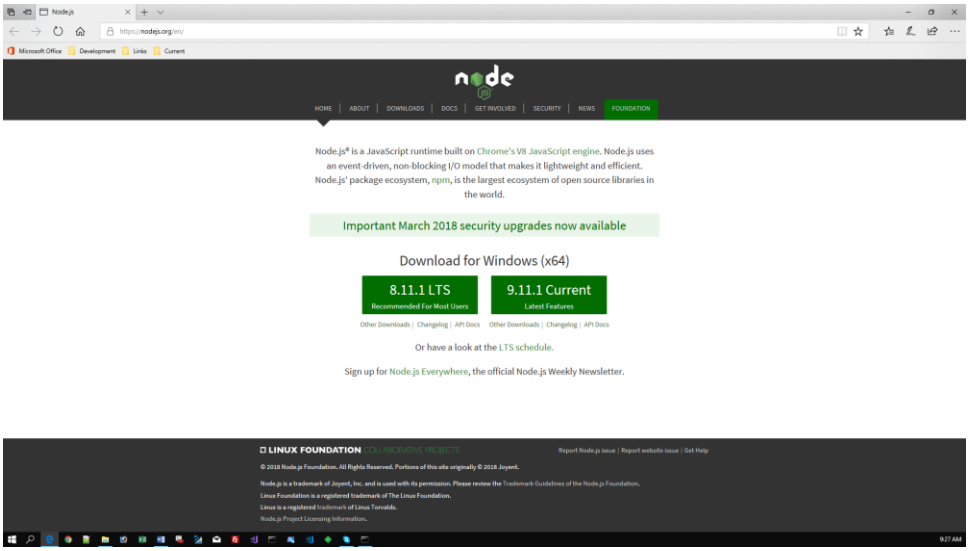


You will be presented with the Visual Studio Code Setup Wizard. Follow along with the steps in the Wizard and setup for Visual Studio Code will take place.

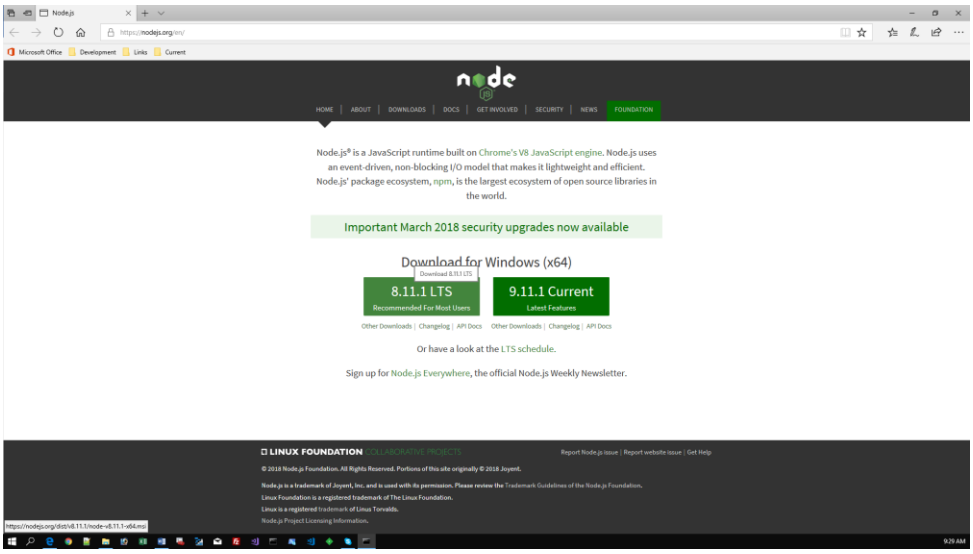
Installing nodejs

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

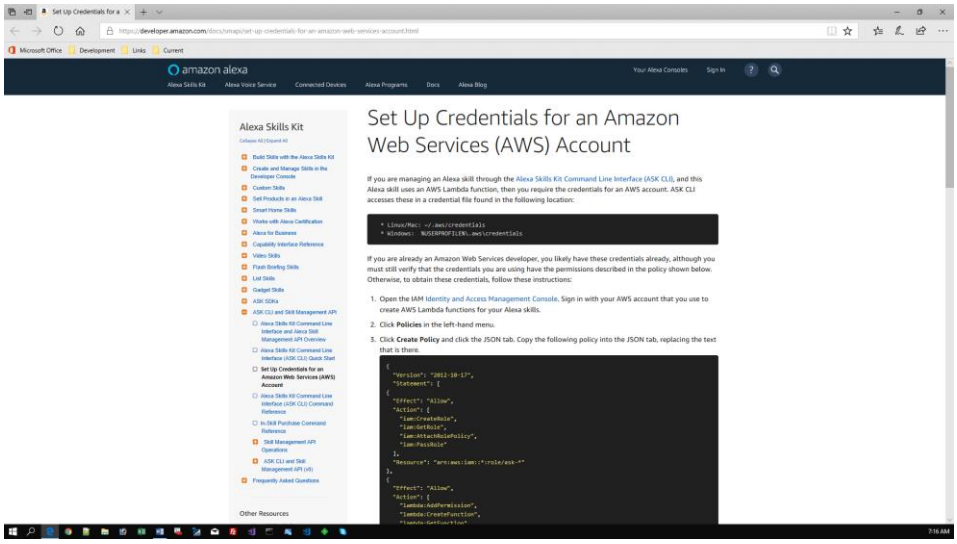
- Create a new Security Policy
- Create a new User and assign Security Policy
- Setting up a Local AWS Profile
- Install the ASK CLI
- Associate the User to the ASK CLI

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

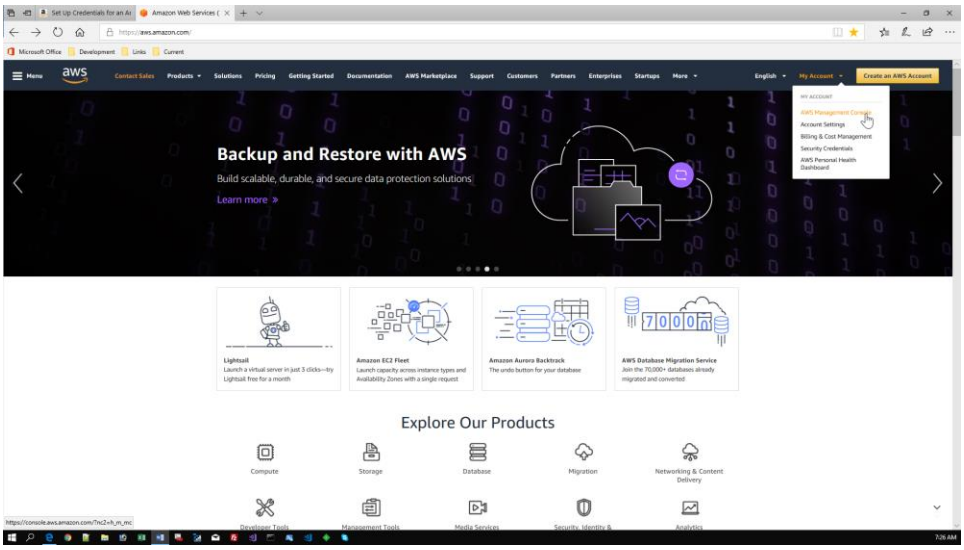


On this page you will find some JSON code that is the specifications for the new Policy that we will be creating. Make a copy of this JSON as we will need it in the next step.

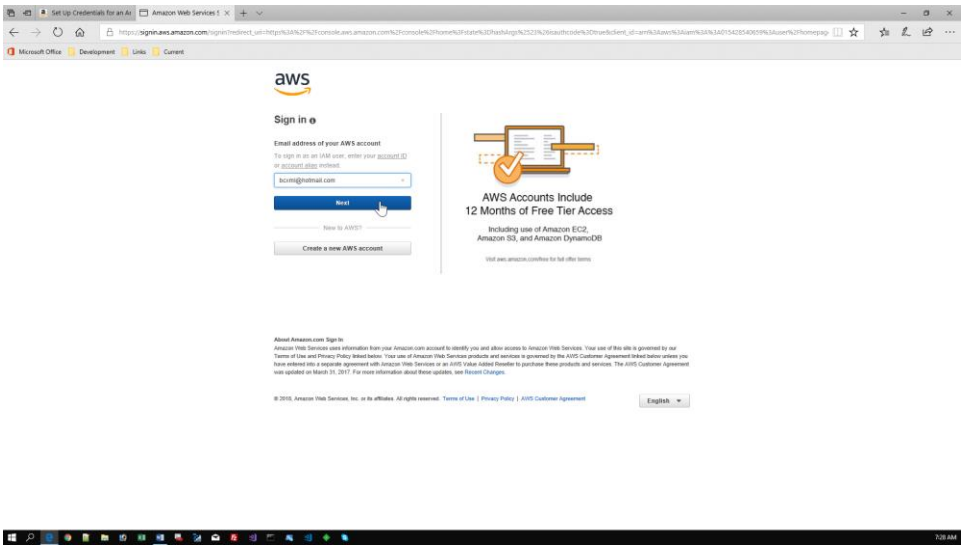
Next, navigate to the AWS home page located at

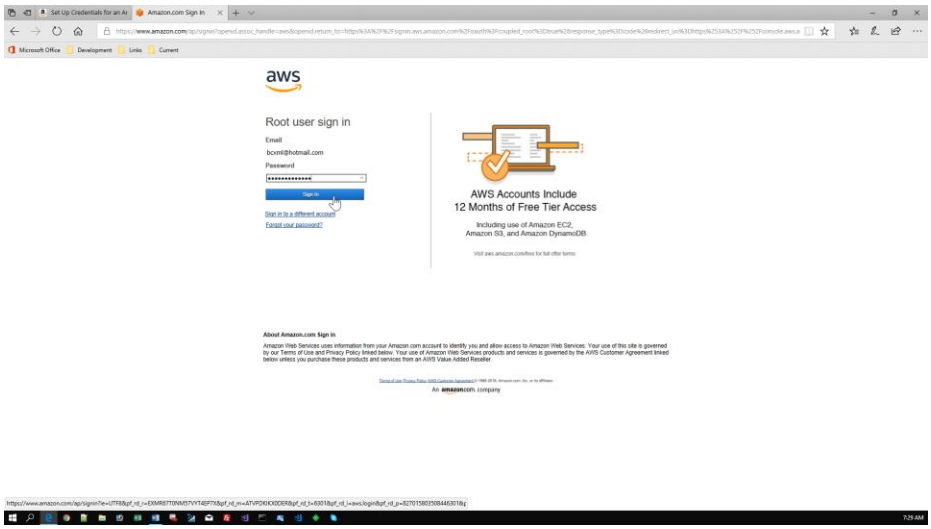
https://aws.amazon.com

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

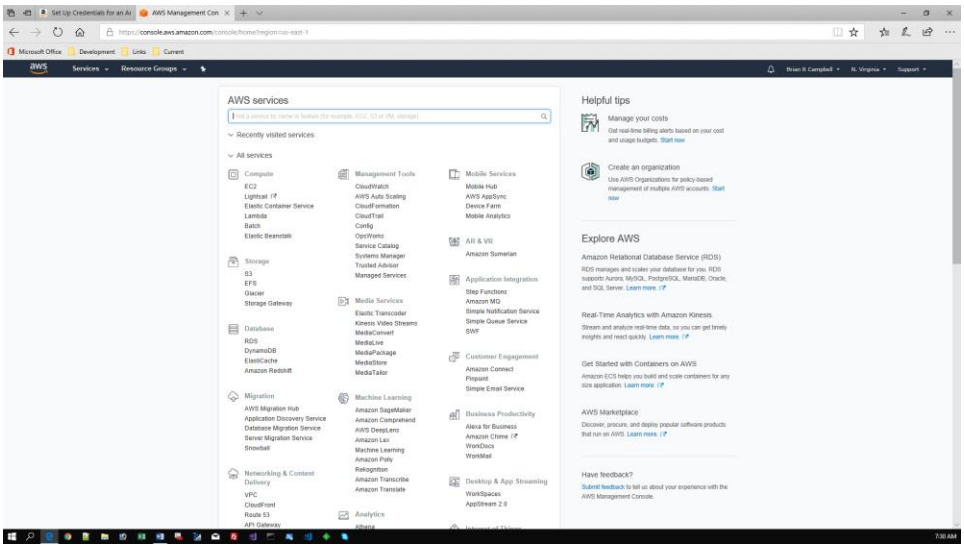


Enter the email address of your AWS Account and password as shown below

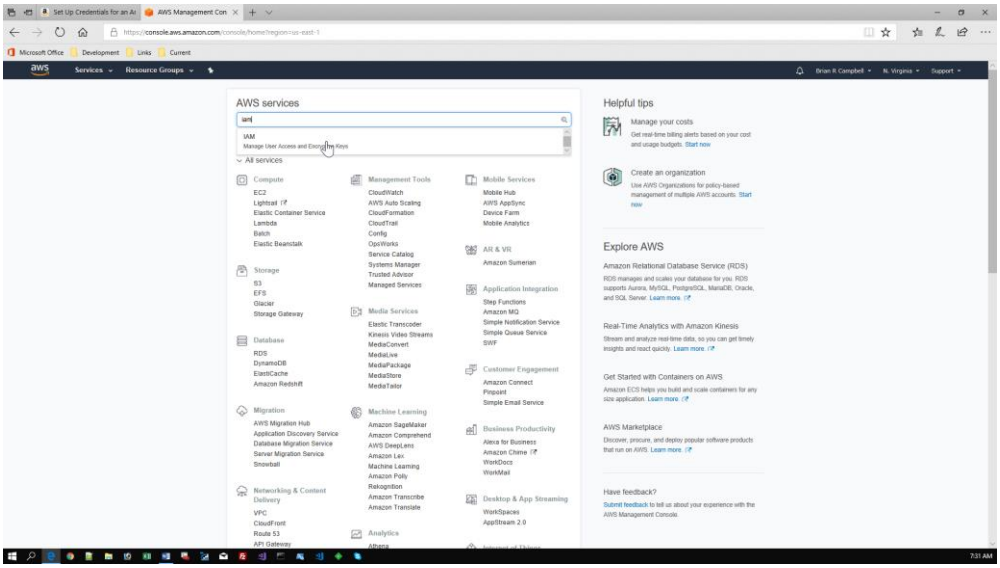




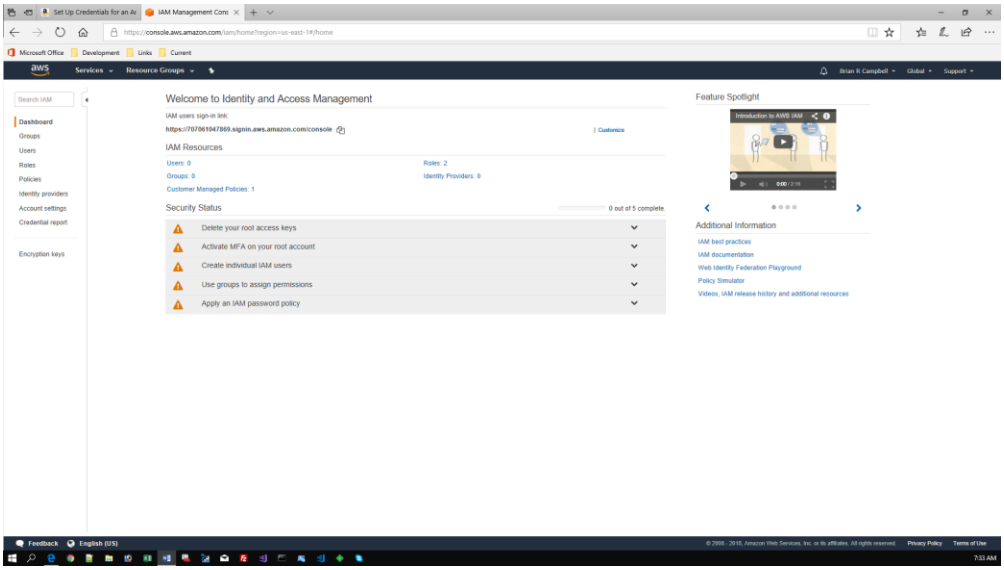
You will be shown the following screen



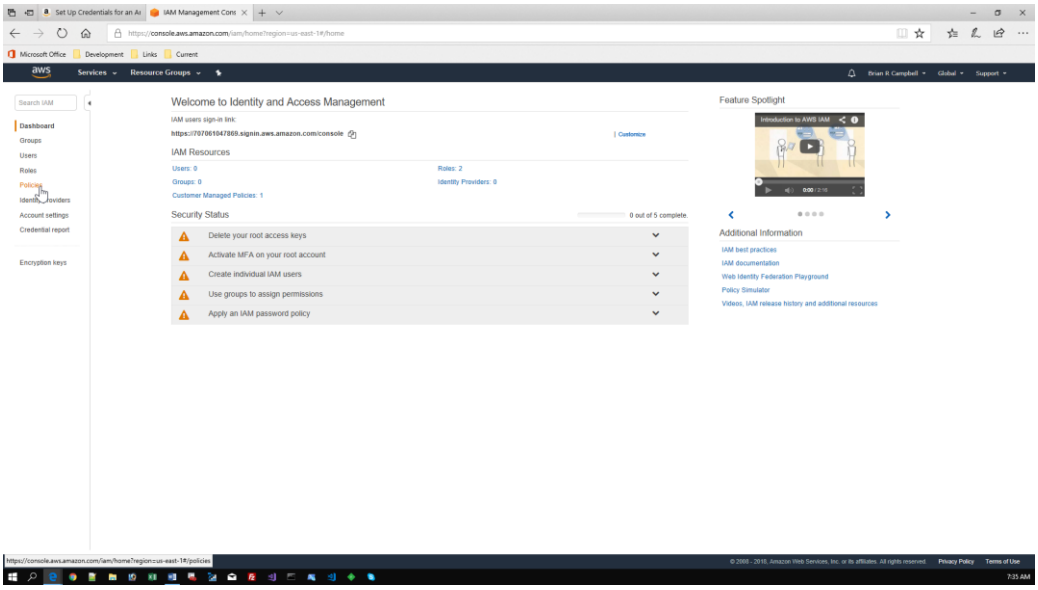
We are going to use the ‘Identity and Access Management’ (IAM) service to setup a new Policy and User. In the Search Bar, enter ‘IAM’ and as shown below



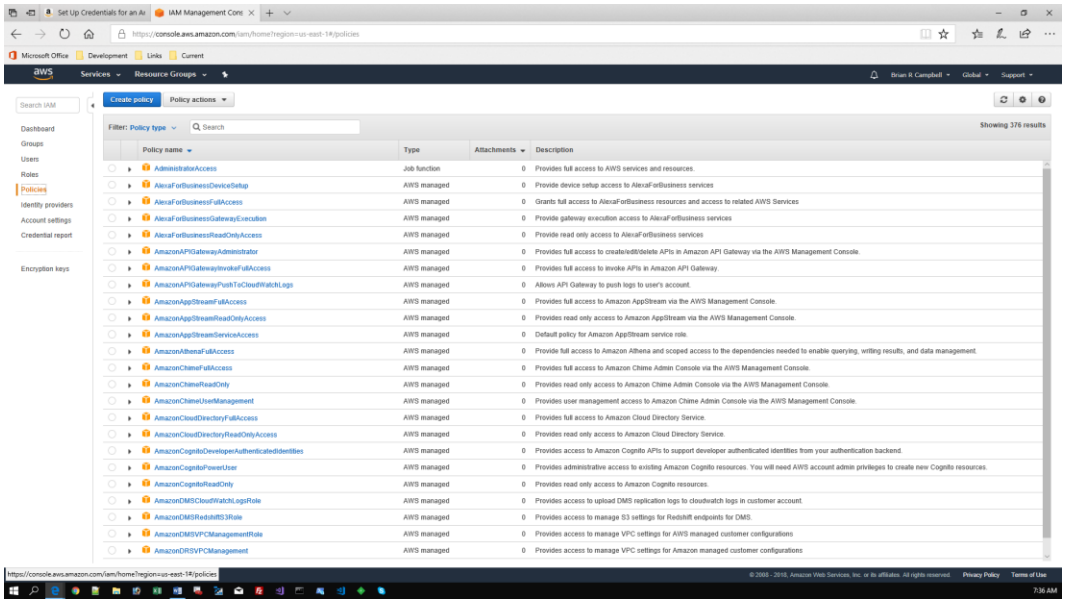
You will be taken to the following page



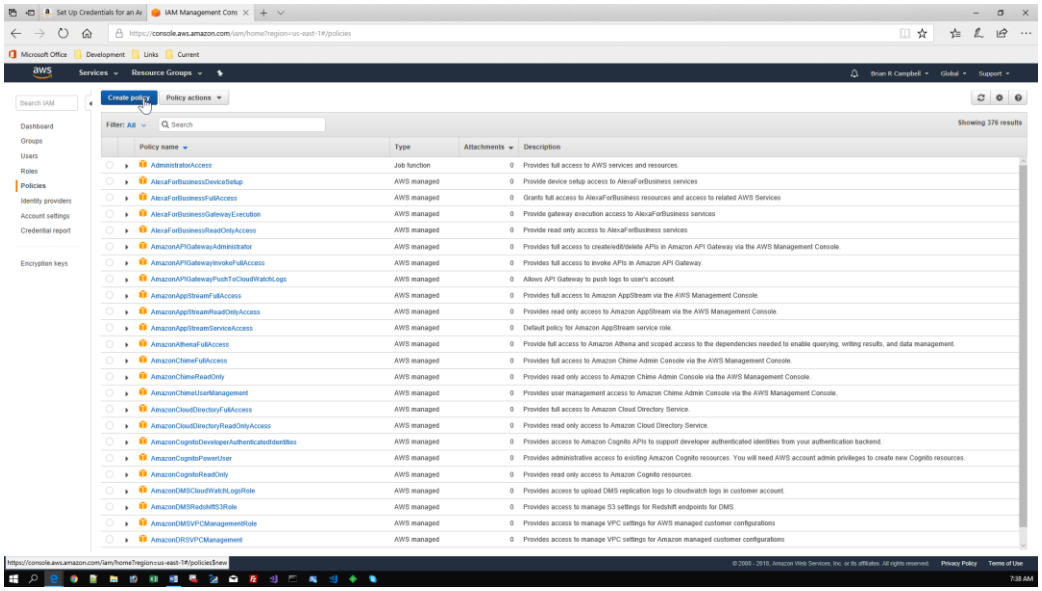
To setup a new Policy, click the ‘Policies’ link in the left-hand menu as shown below



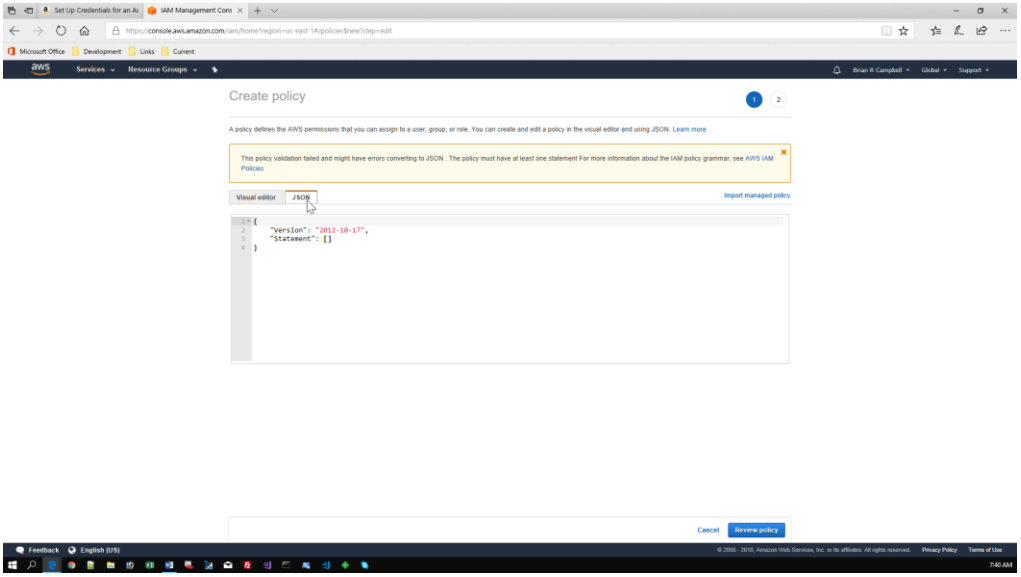
You will be shown the following screen



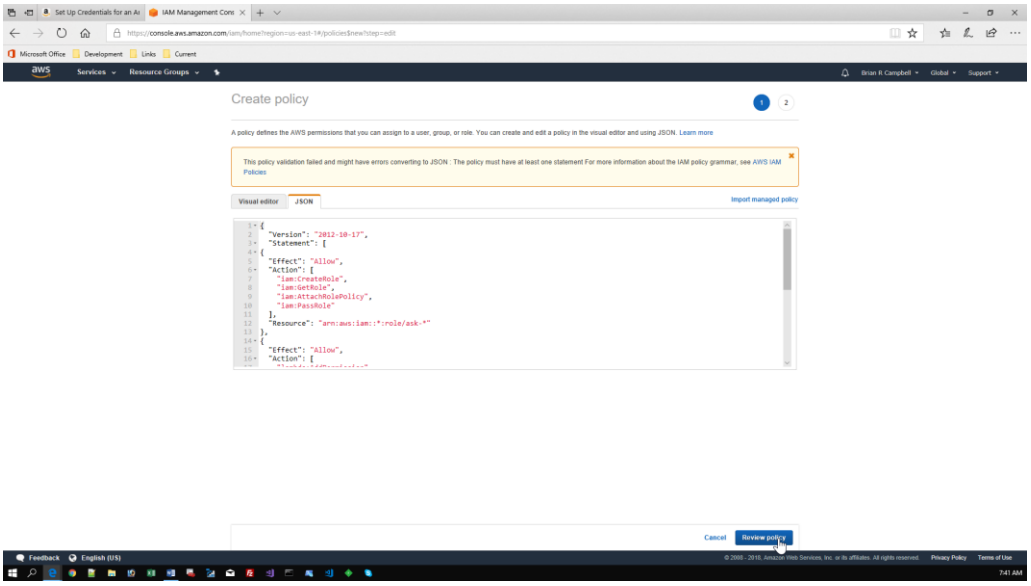
Click on the ‘Create Policy’ button as shown below



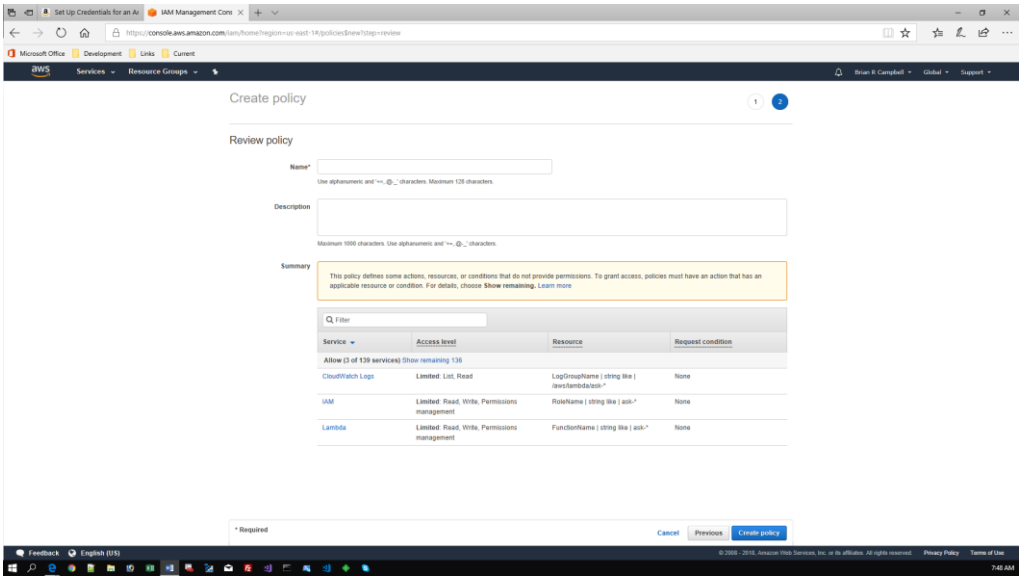
On the resulting screen, click on the ‘JSON’ tab as shown below



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



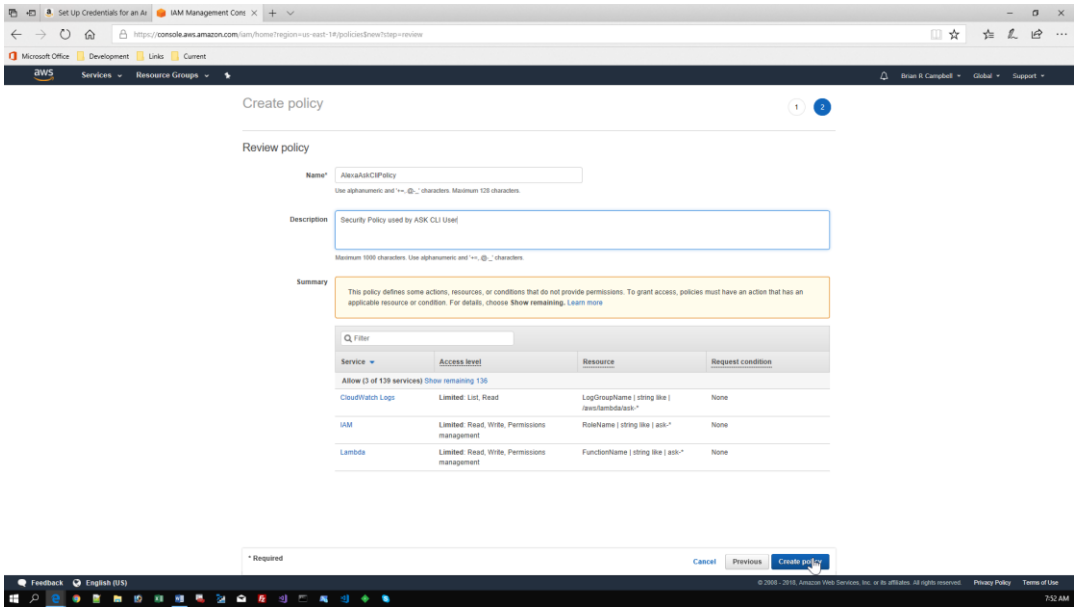
You will be presented with the following screen



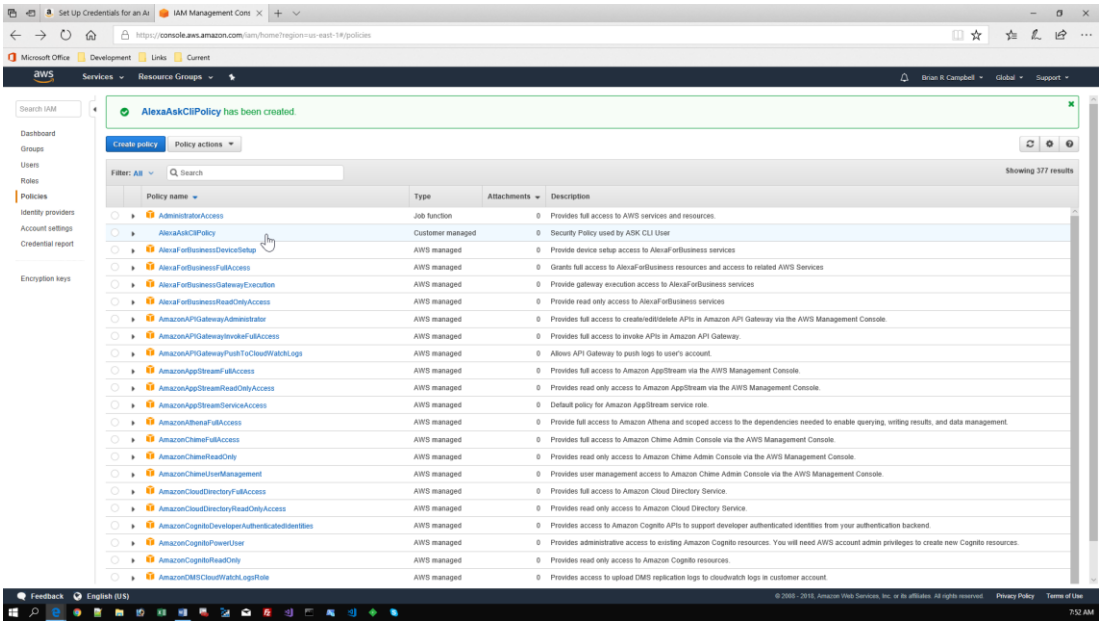
Fill in the Name and Description fields as follows

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

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

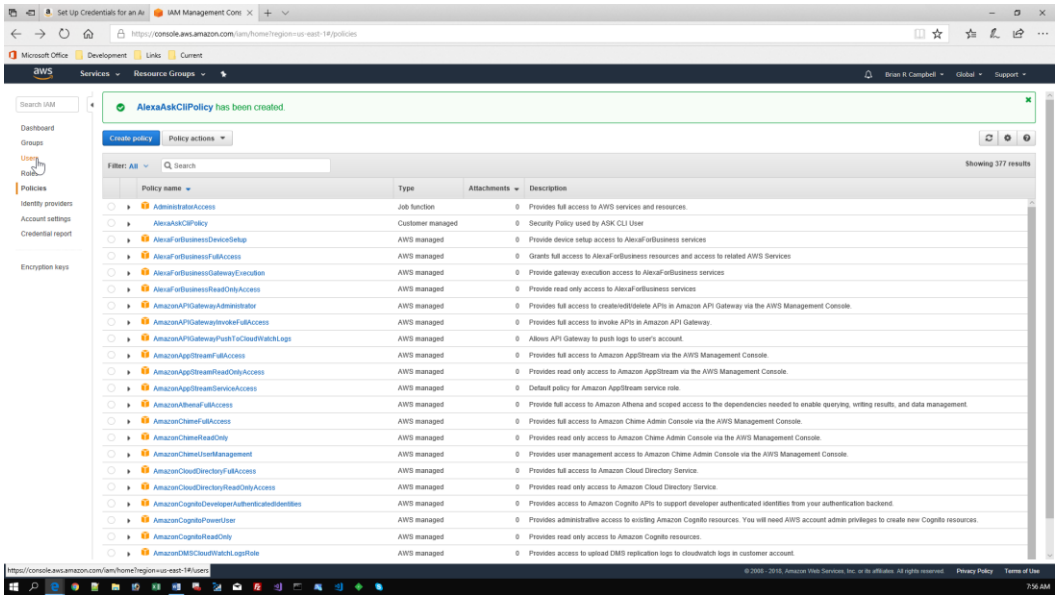


Back in the Policies screen you can see that the new Policy has been created

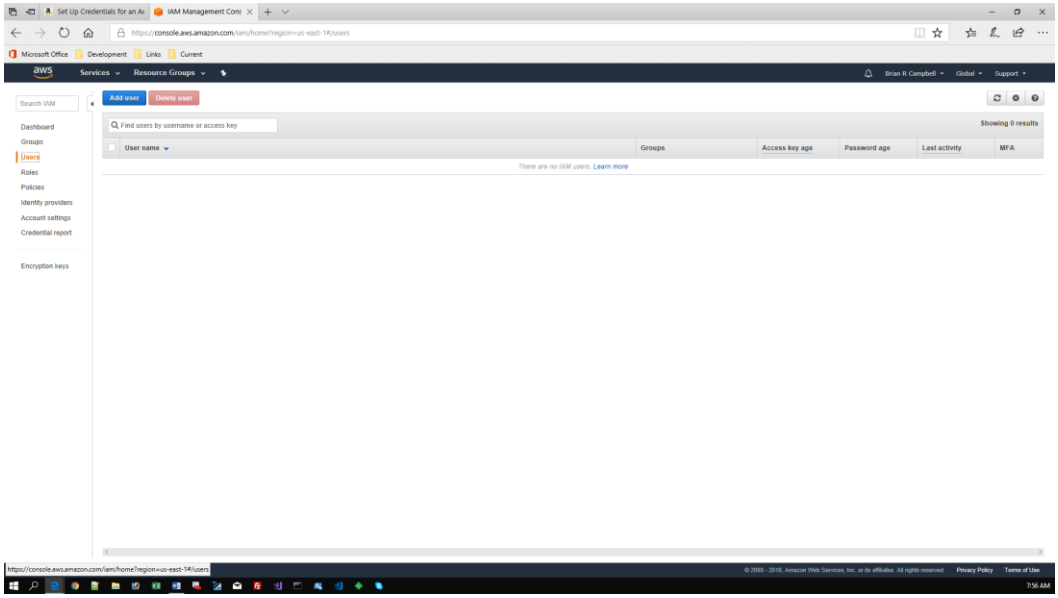


Step 2 - Create a new User and assign Security Policy

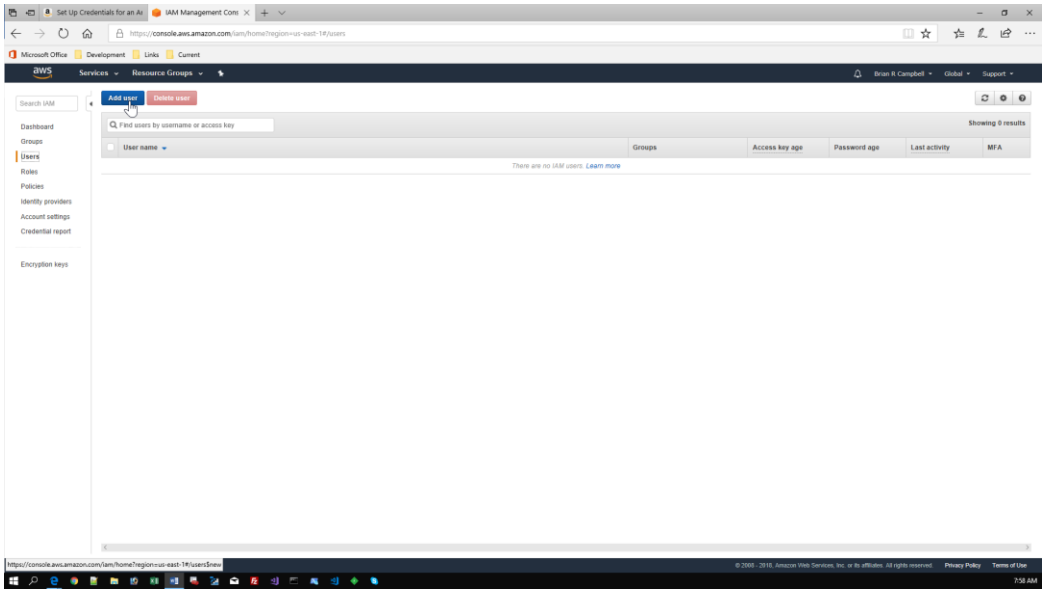
In the left-hand menu, click on the ‘Users’ link as shown below



You will be presented with the following screen



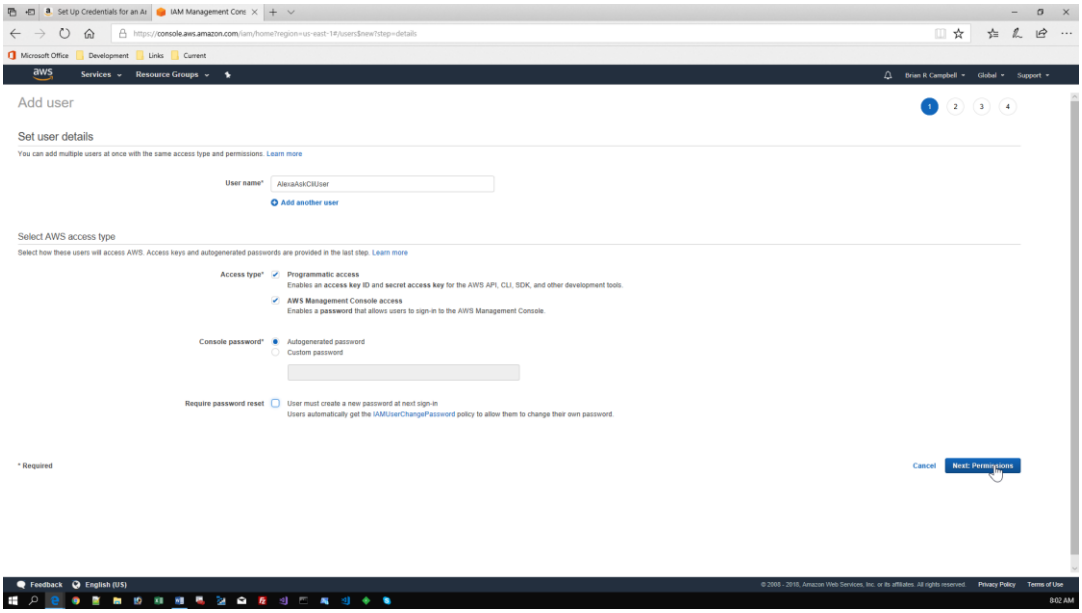
Click on the ‘Add User’ button as shown below



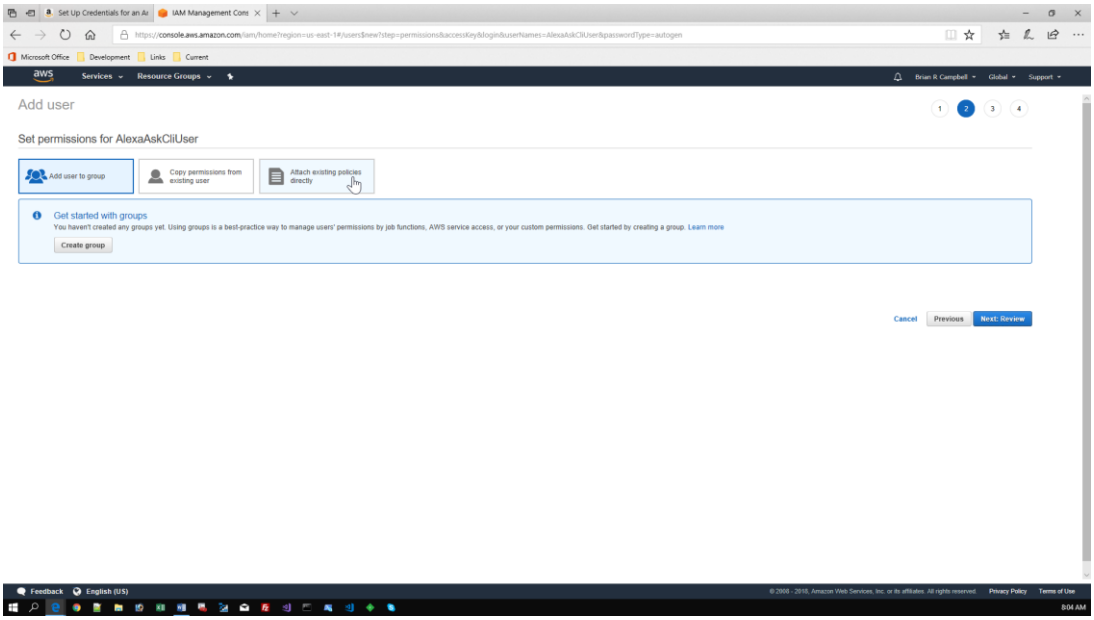
On the resulting screen, enter the following values

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

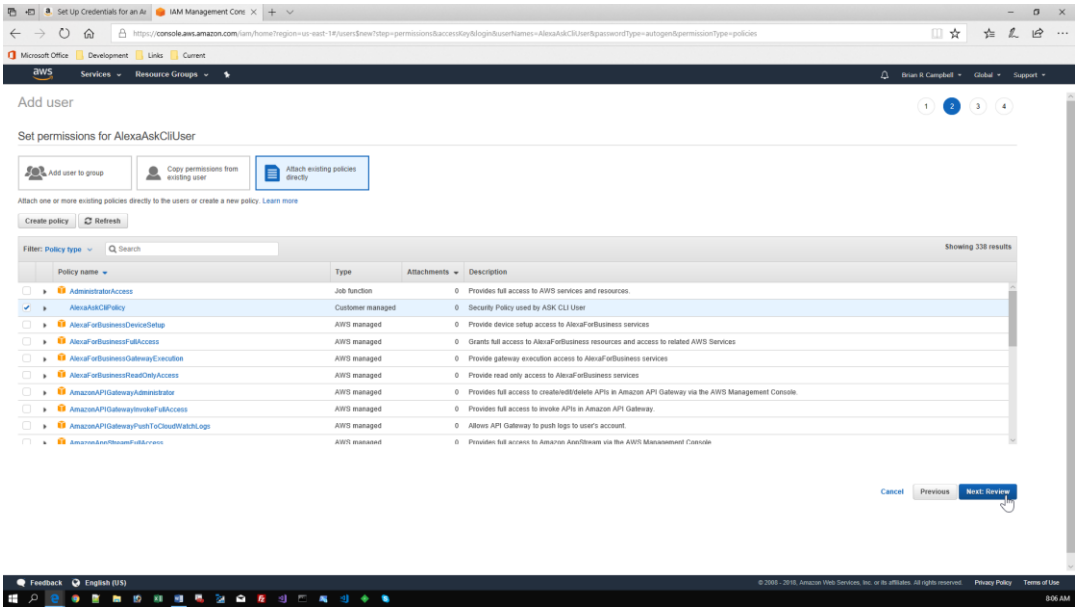
And then click on the ‘Next:Permissions’ button as shown below



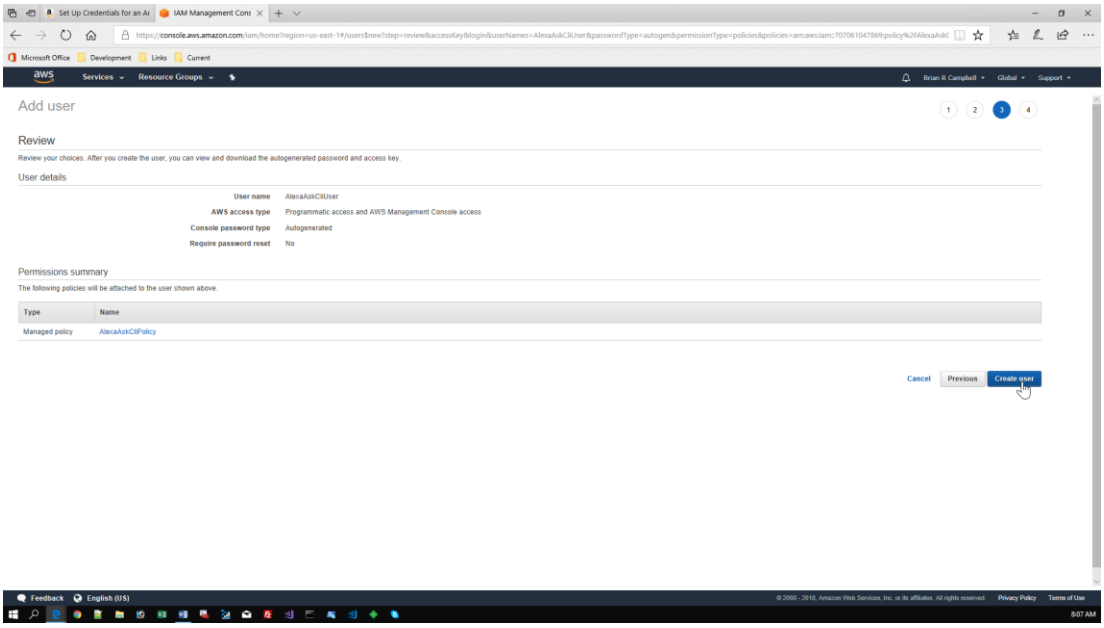
In the resulting screen, click on the large ‘Attach existing policies directly’ button as shown below



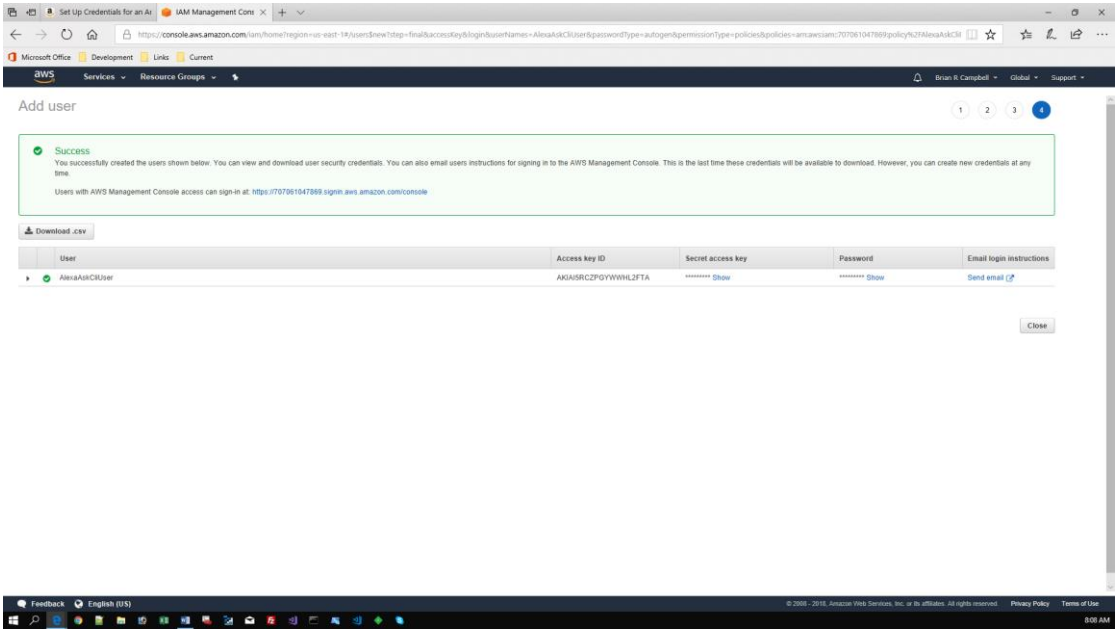
In the resulting screen, place a checkmark to the left of the ‘AlexaAskCliPolicy’ that we previously created and then click on the ‘Next:Review’ button as shown below



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

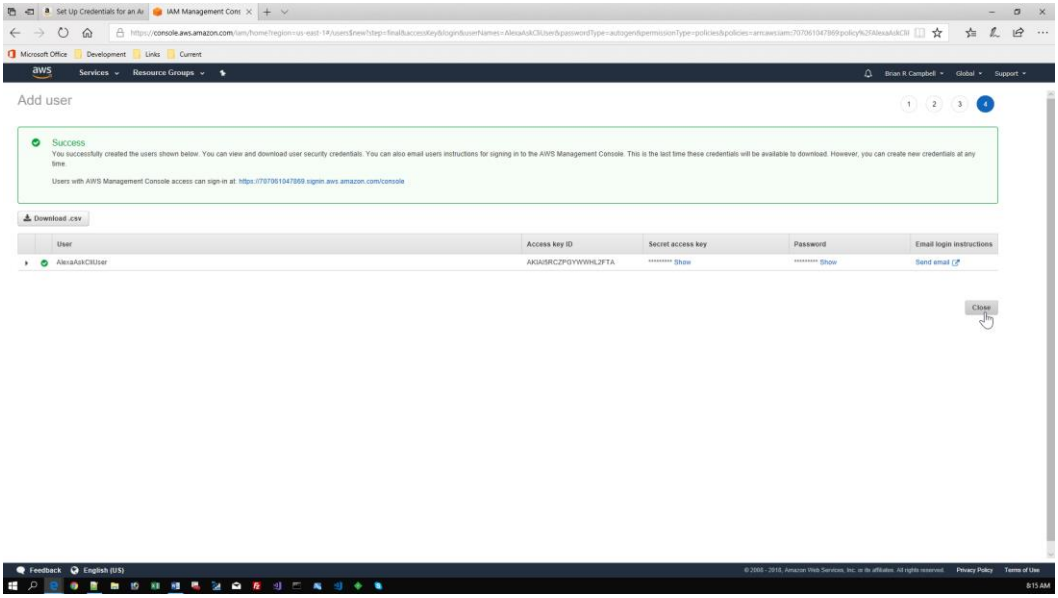


You will be shown the following screen which lists the new User that was created

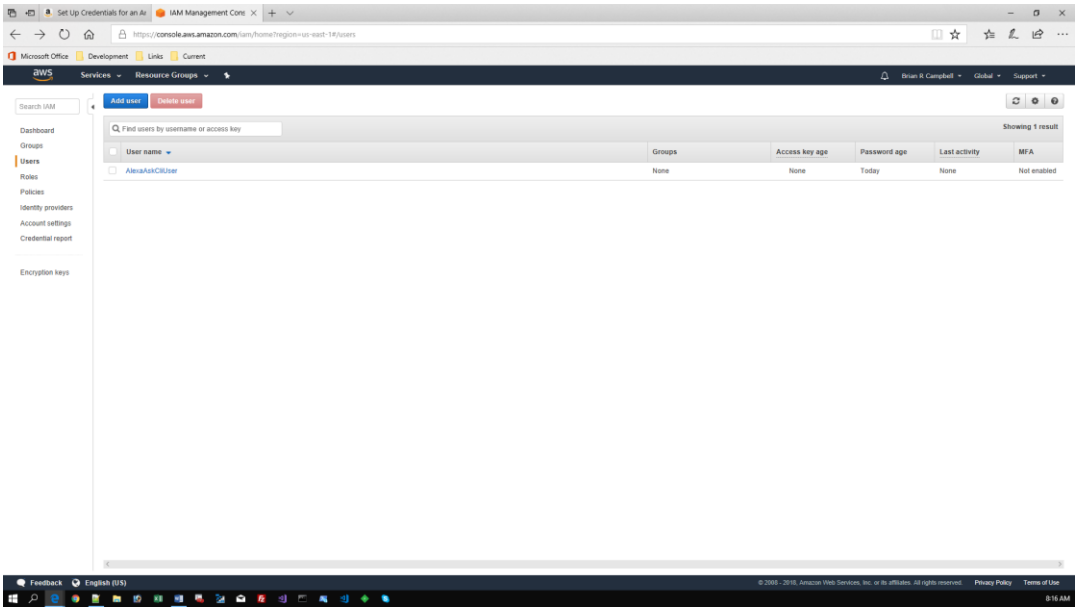


Now at this point we are going to need 2 pieces of information from this screen, the ‘Access key ID’ and the ‘Secret access key’. Copy these 2 values, as they will be needed in the next step

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



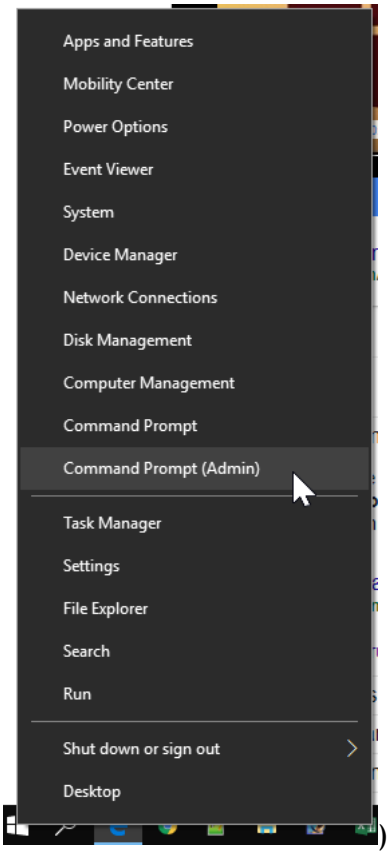
You will be taken back to the Users screen where you can see the newly created User as shown below



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

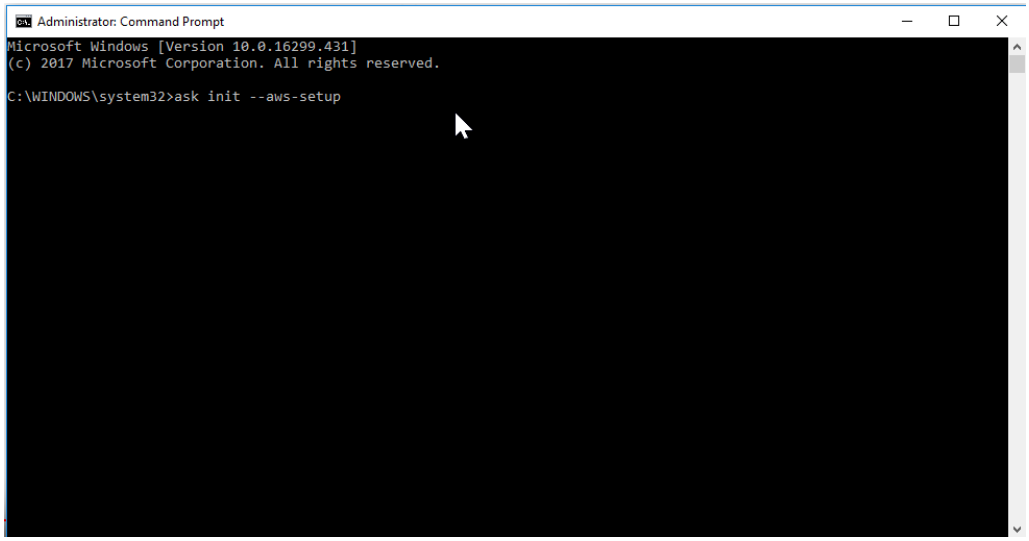
‘Can't find Python executable "python", you can set the PYTHON env variable’

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

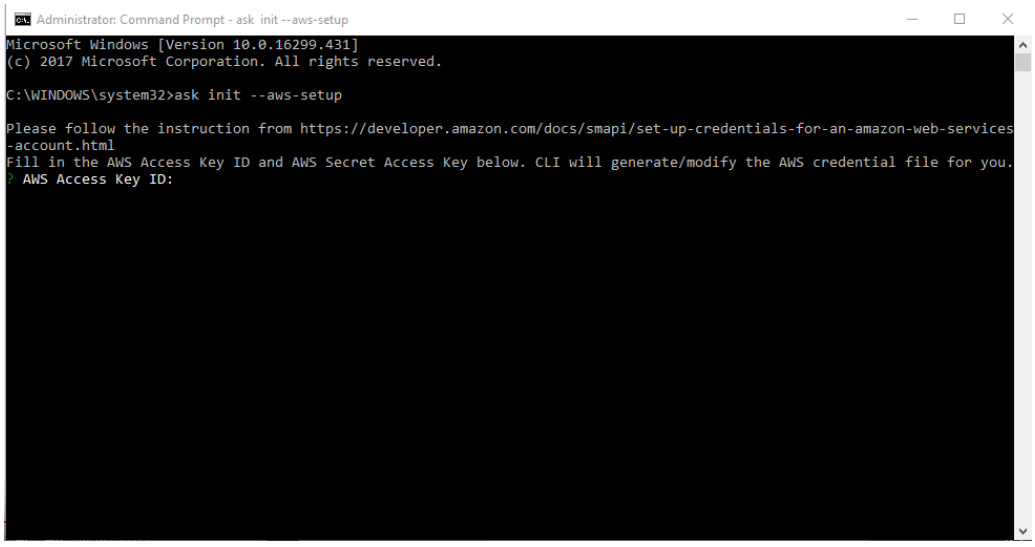
Step 4 - Setting up a Local AWS Profile

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

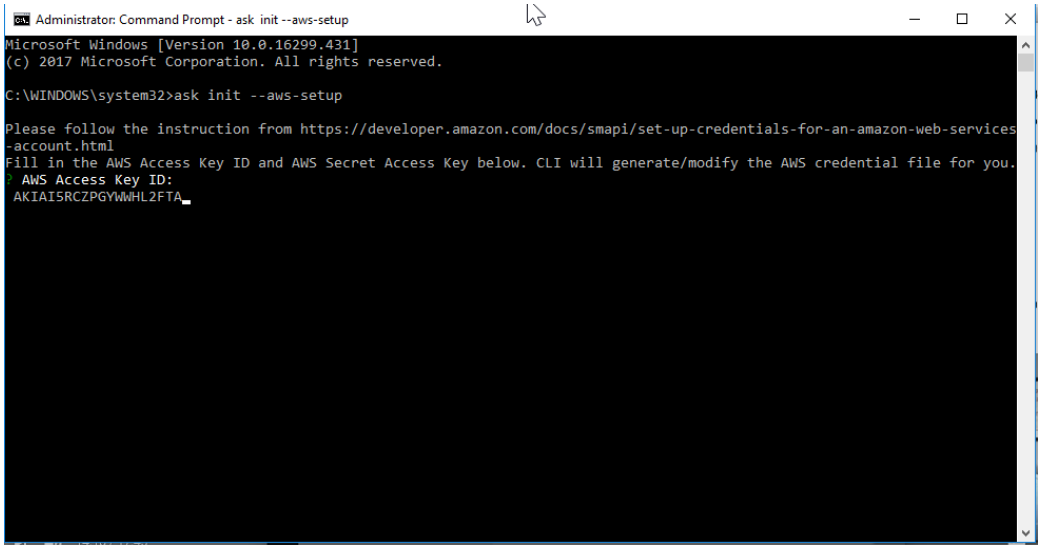
ask init --aws-setup



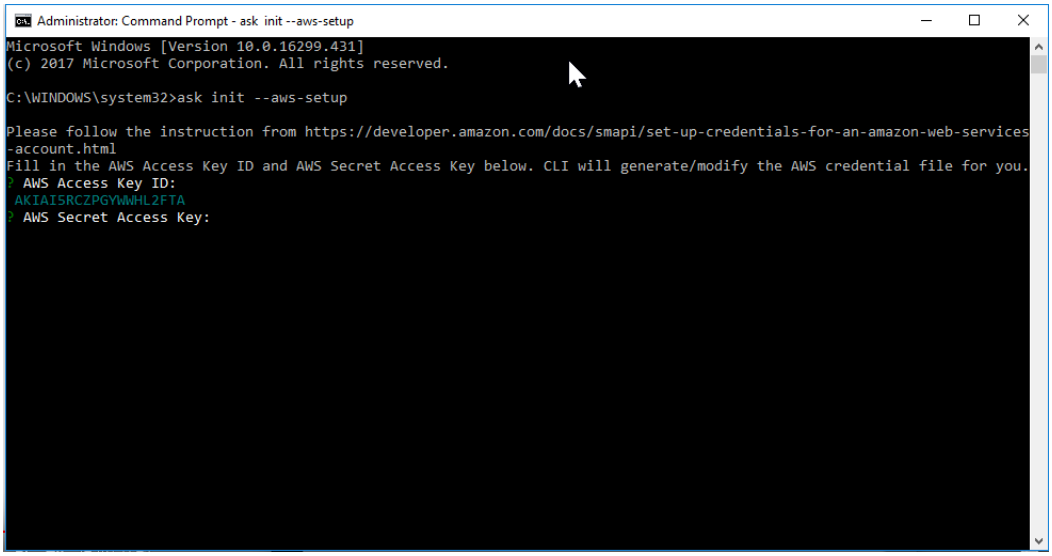
You will be presented with the following



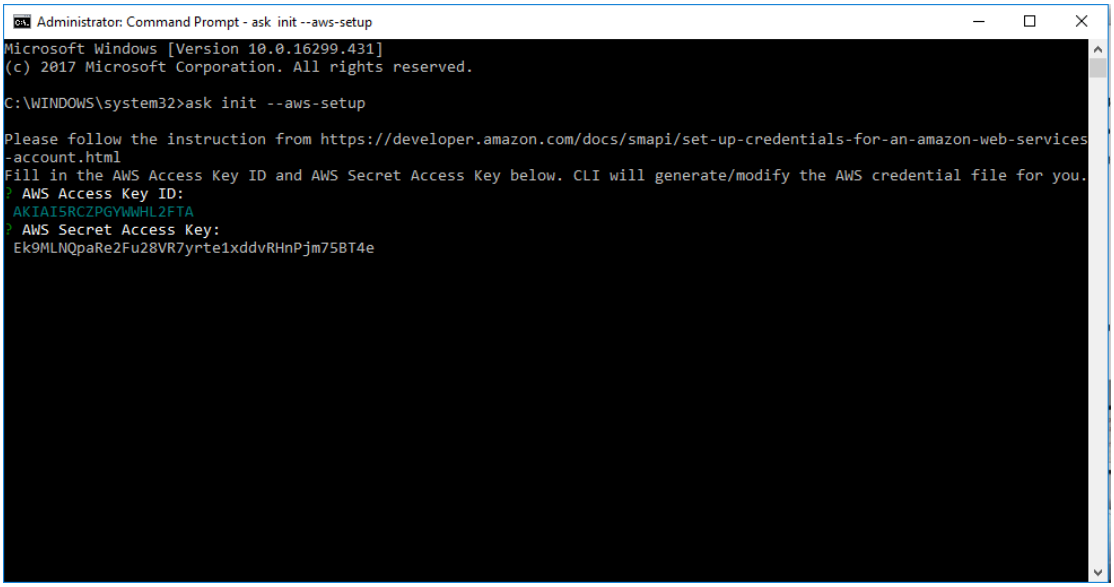
It is asking for the ‘AWS Access KEY ID’ that we copied from a previous step. Paste that value in and then hit Enter as shown below



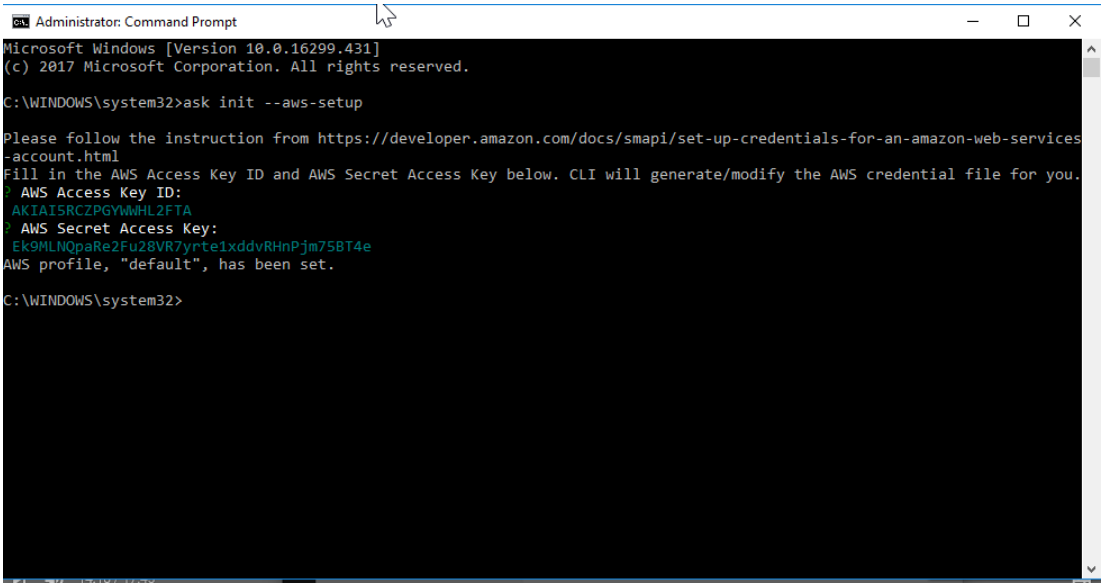
You will be presented with the following



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



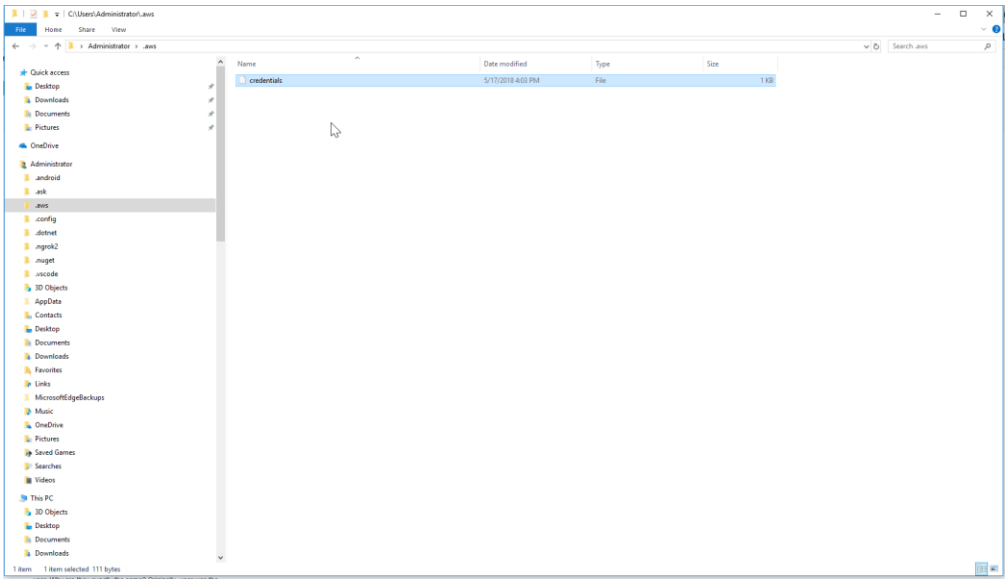
You will be shown the following screen

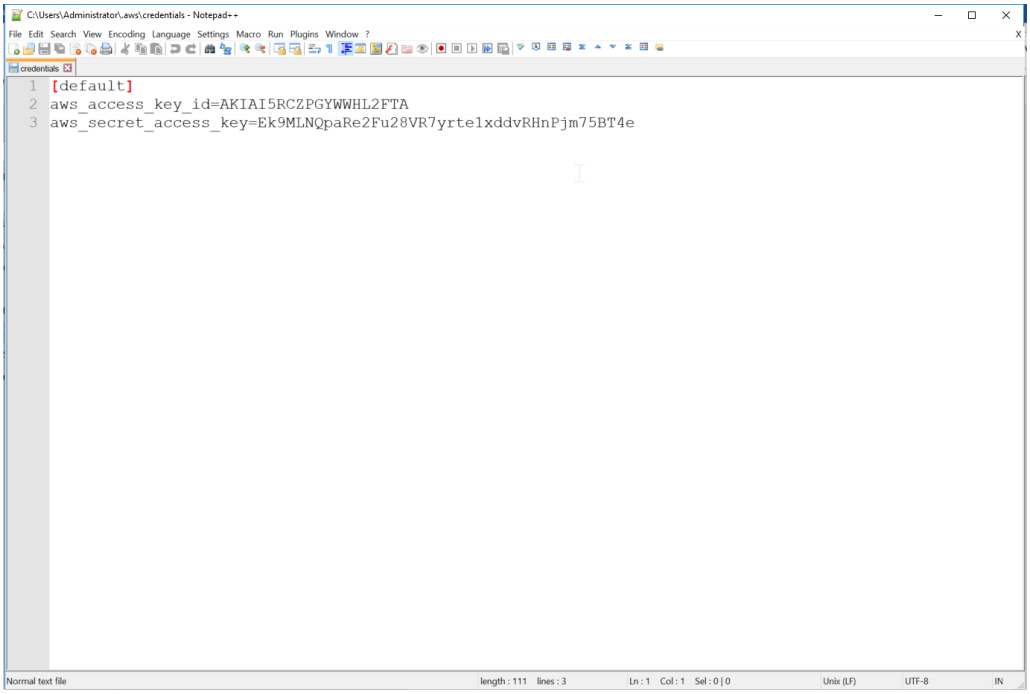


The Local AWS Profile has now been setup. You can see the profile by opening Windows Explorer and navigating to

%**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

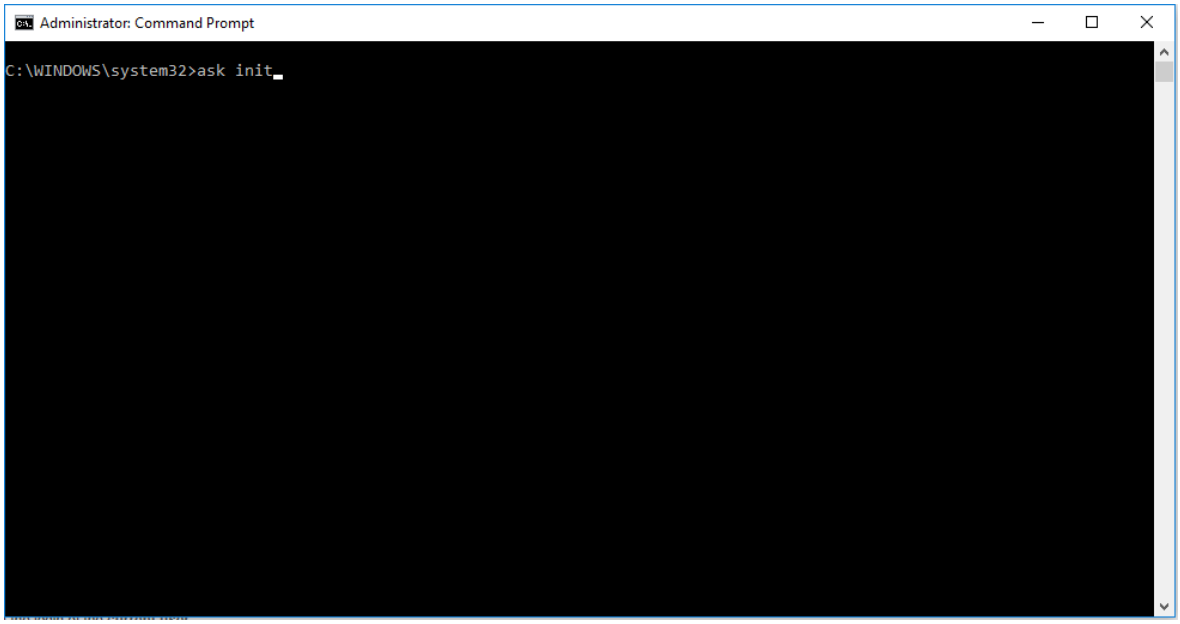

C:\Users\Administrator\.aws
File
Home
Share
View
Administrator > .aws
Search .aws
Quick access
Desktop
Downloads
Documents
Pictures
OneDrive
Administrator
.android
.ask
.aws
.config
.dotnet
.ngrok2
.nuget
.vscode
3D Objects
AppData
Contacts
Desktop
Documents
Downloads
Favorites
Links
MicrosoftEdgeBackups
Music
OneDrive
Pictures
Saved Games
Searches
Videos
This PC
3D Objects
Desktop
Documents
Downloads
Name
Date modified
Type
Size
credentials
5/17/2018 4:03 PM
File
1 KB
1 item
1 item selected 111 bytes


C:\Users\Administrator\.aws\credentials - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
credentials
1 [default]
2 aws_access_key_id=AKIAI5RCZPGYWWHL2FTA
3 aws_secret_access_key=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
Normal text file
length : 111 lines : 3
Ln : 1 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
IN

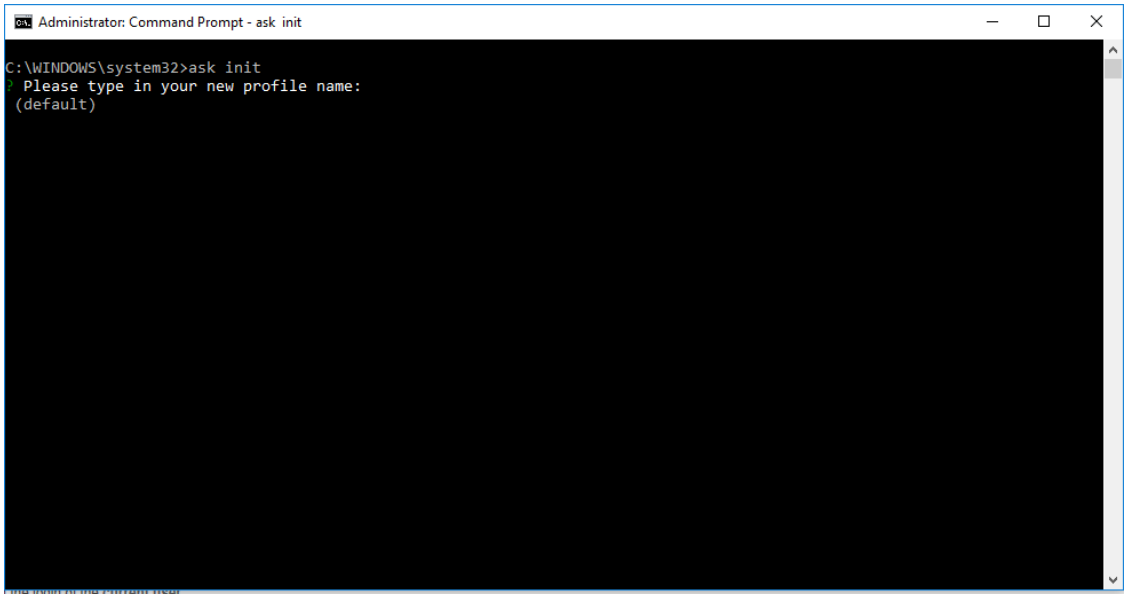
Step 5 – Associate the User with the ASK CLI

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

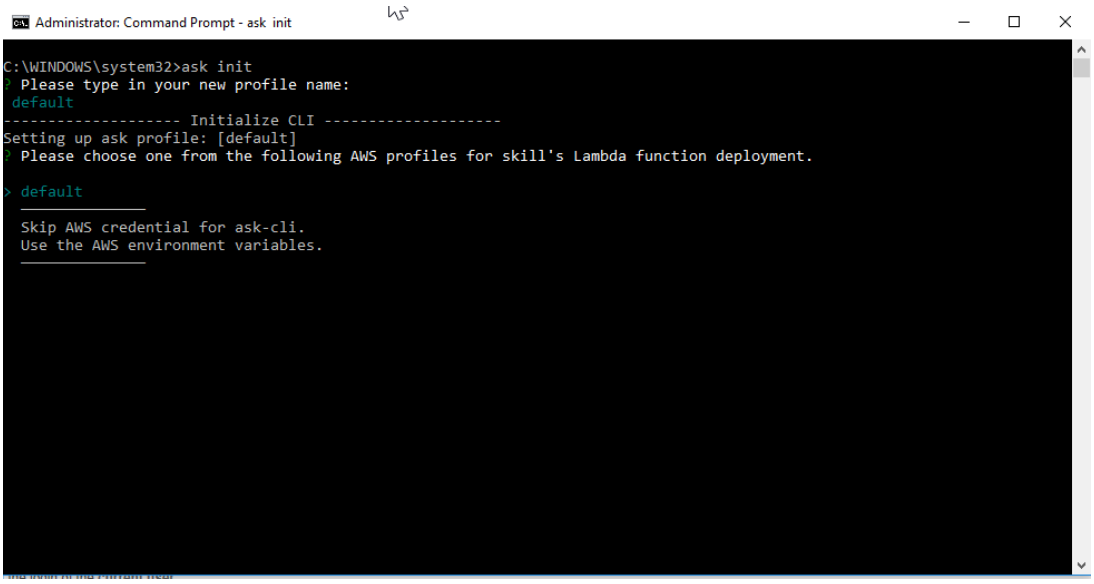
Enter ‘ask init’ as shown below



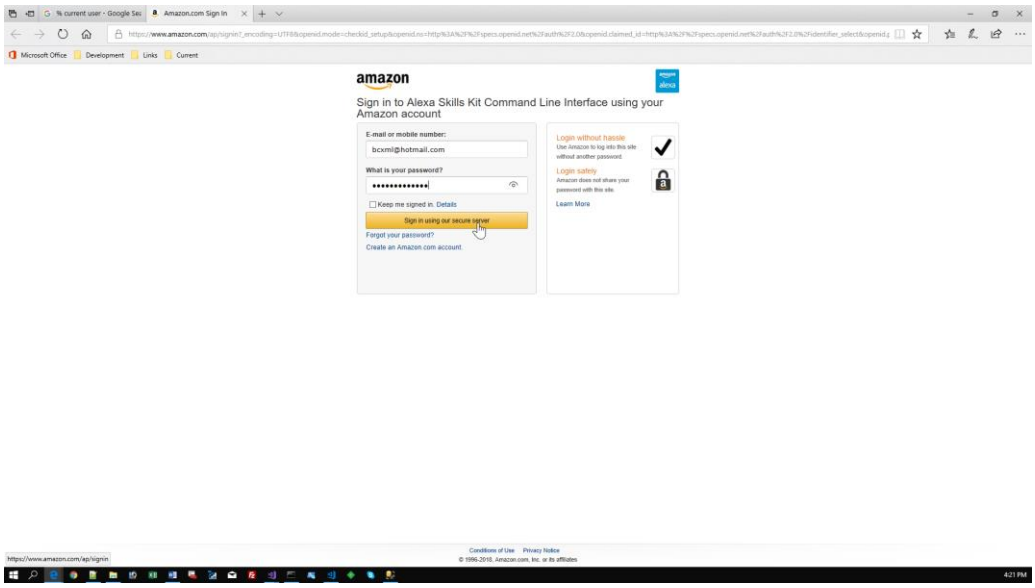
You will receive the following feedback



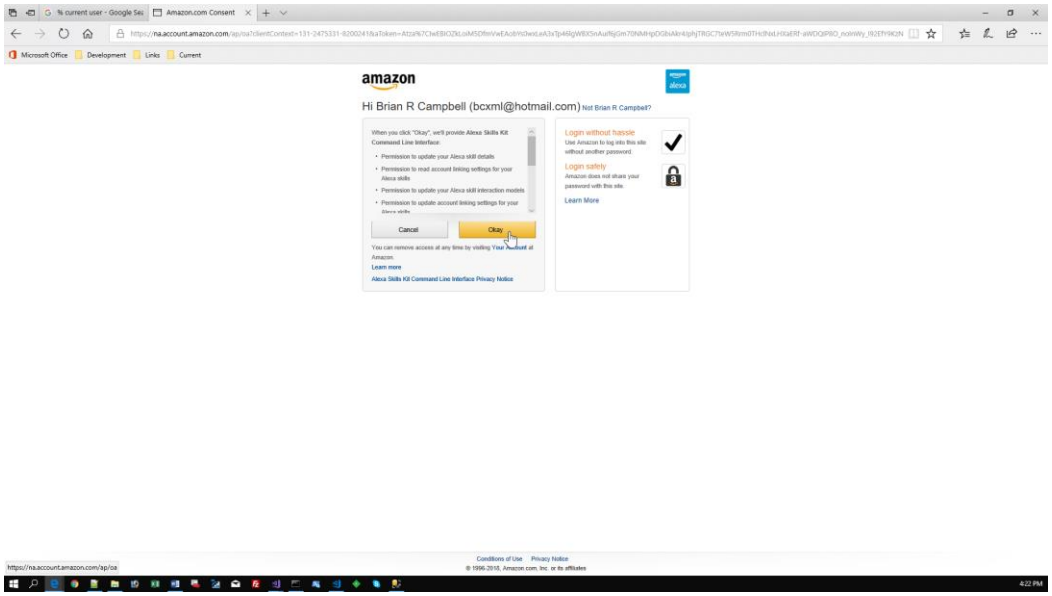
Just hit Enter to accept the Profile Name = default. You will be shown the following



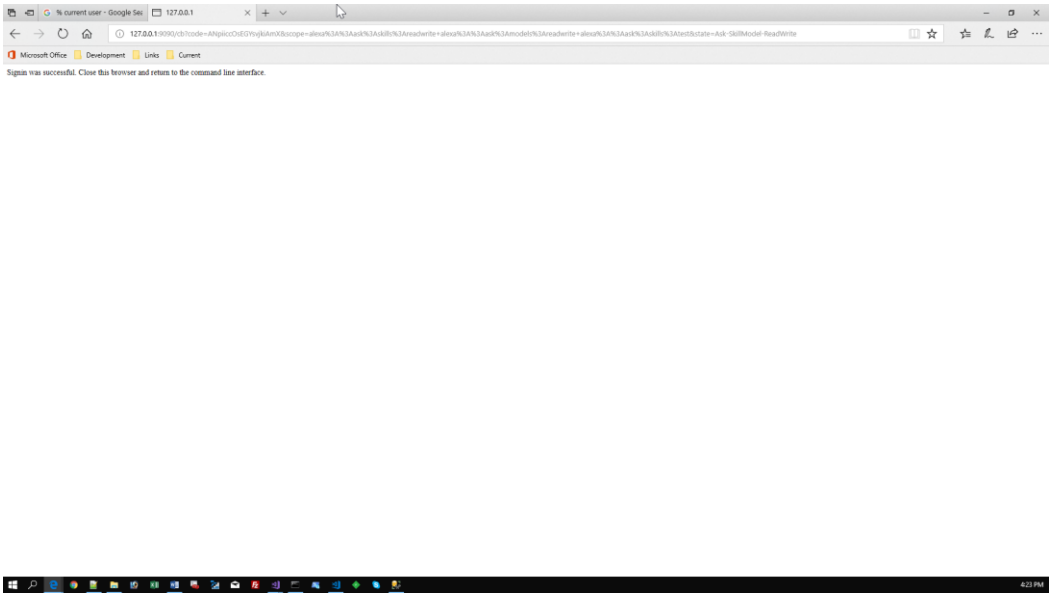
Just press Enter. You will be directed to Amazon so that you can login with your AWS credentials as shown below



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

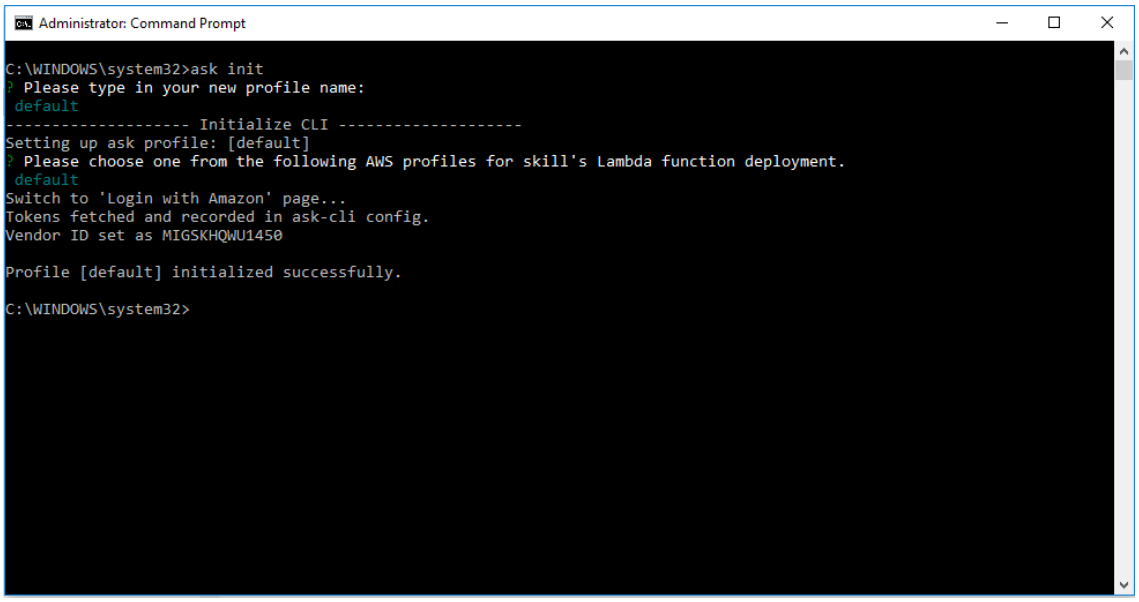


You will now be presented with a page telling you that Sign In was successful as shown below



You can close the browser at this point.

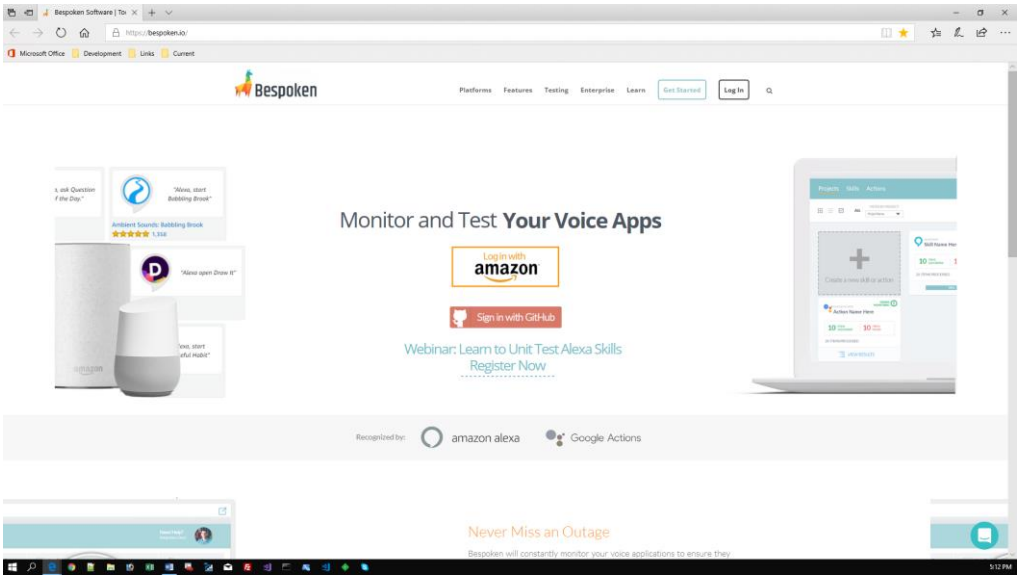
Back in the Command Prompt you will be shown that the initialization process completed successfully as shown below


Administrator: Command Prompt
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

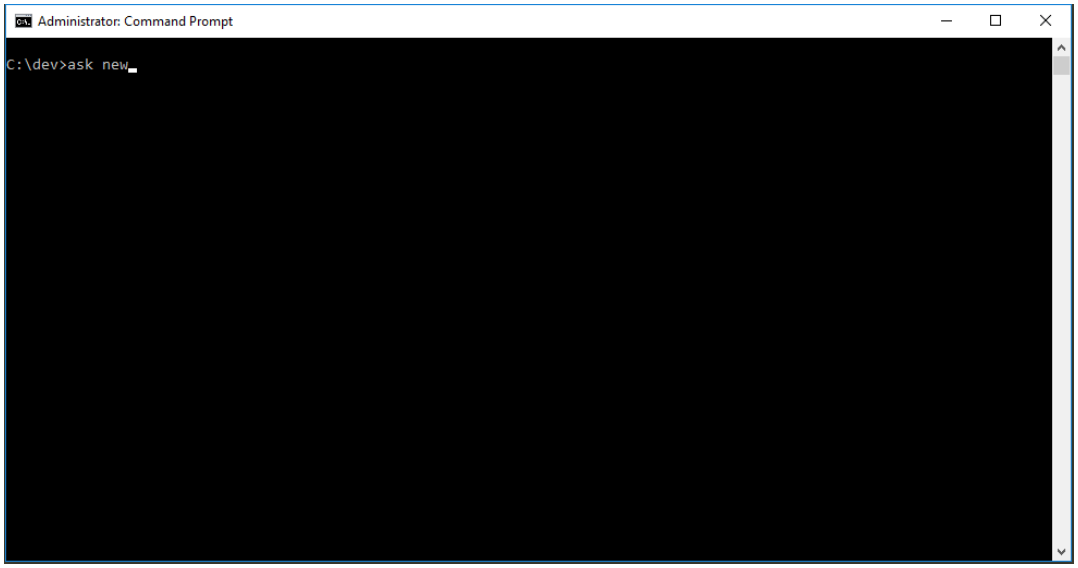
Now that all the tools have been installed we can go ahead and create an Alexa Skill to test out the Local Development Environment. I am going to call this Alexa Skill “Greeter”

Create the Alexa Skill

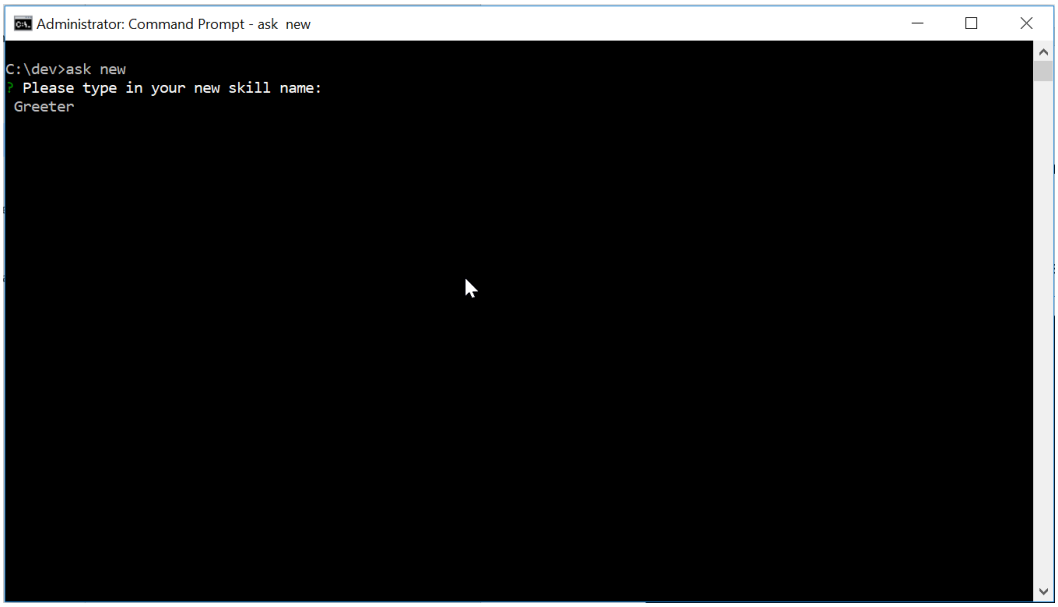
Using a command prompt, create a folder where this “Greeter” Skill can live. In my case I will be using C:\dev as shown below

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

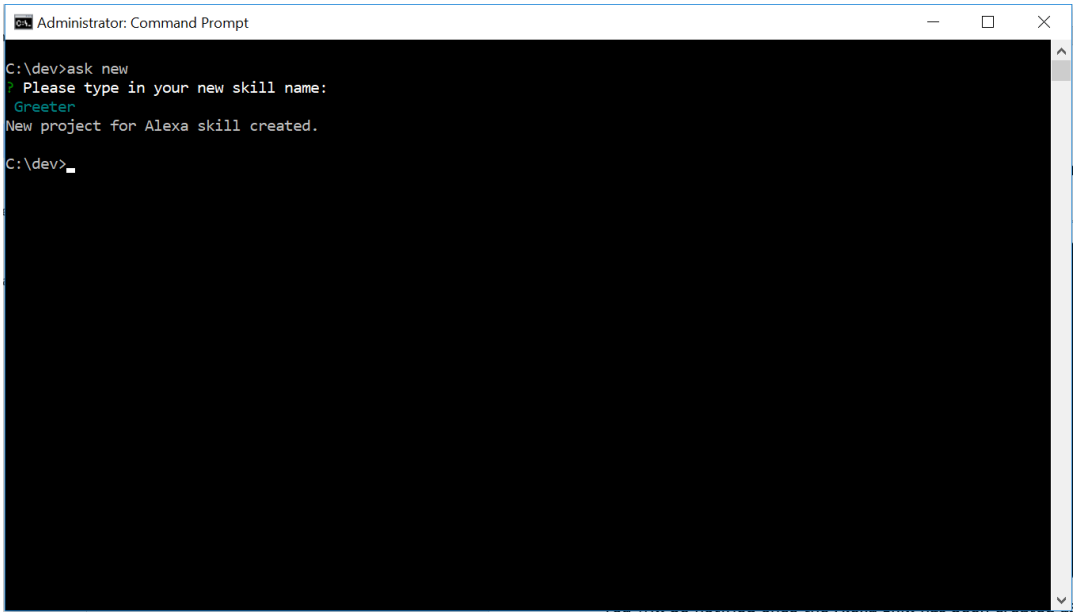
ask new



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

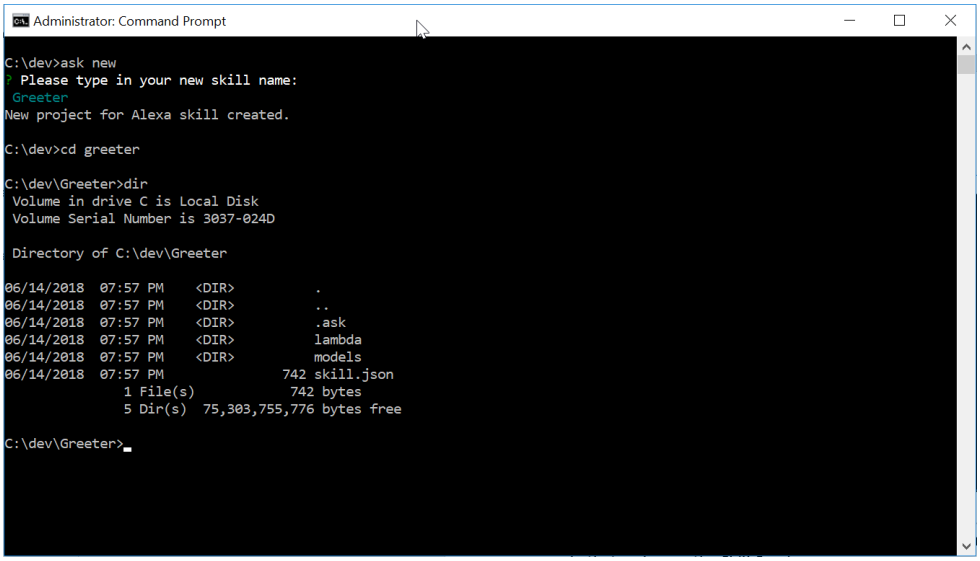


You will be notified once the Alexa Skill has been created as shown below



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



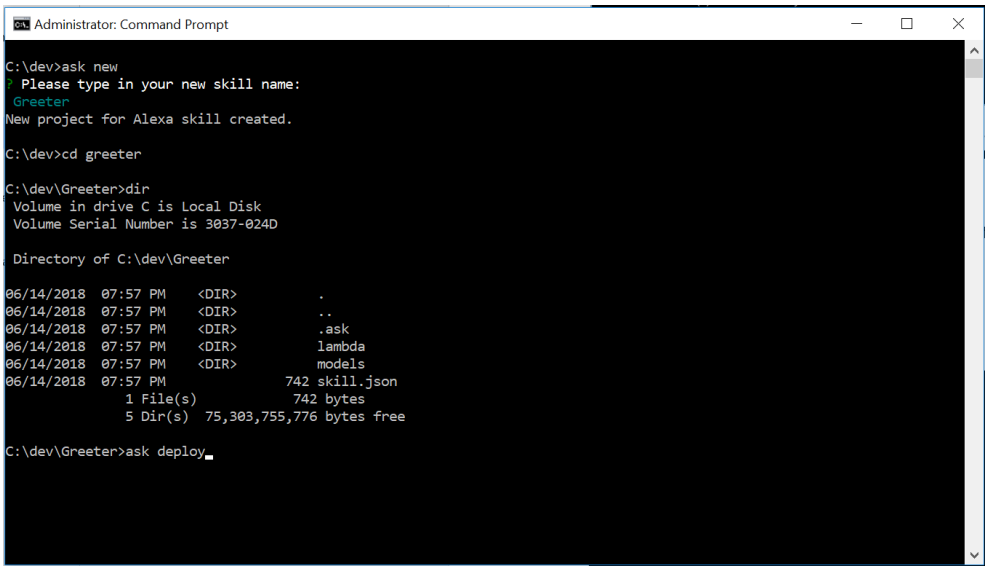
What has been created is the code for a complete Alex Skill. In the ‘models’ folder is a JSON file that represents the Skill Interface portion of the Skill and in the Lambda folder is the code that makes up the Skill Service.

Deploy the Alexa Skill

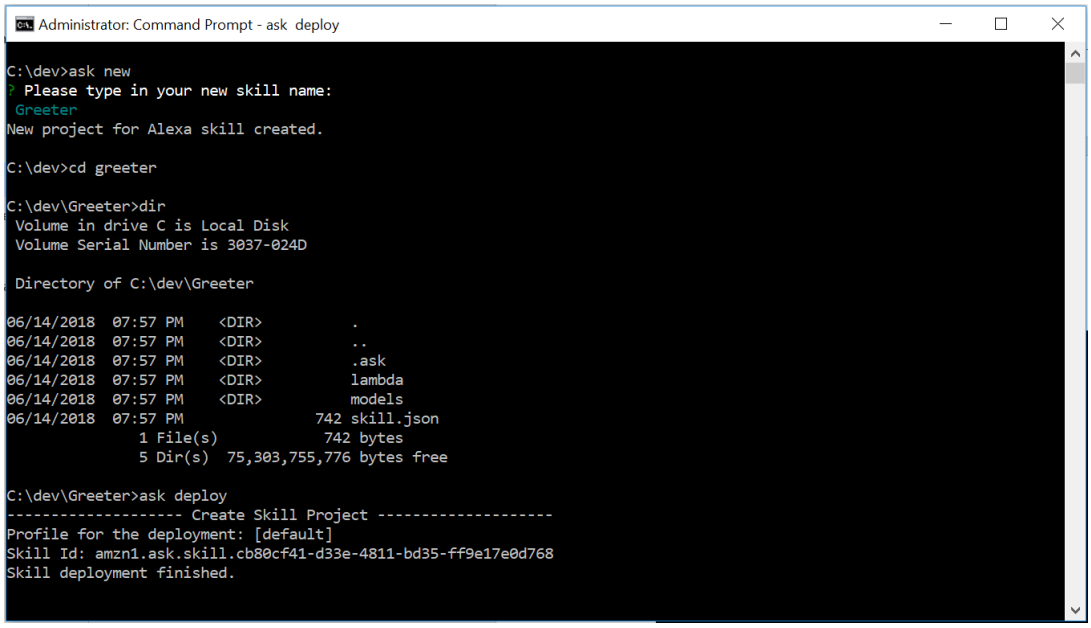
Of course, this code all needs to be deployed to Amazon before it can be used, so let’s do that right now. In the command prompt enter

ask deploy

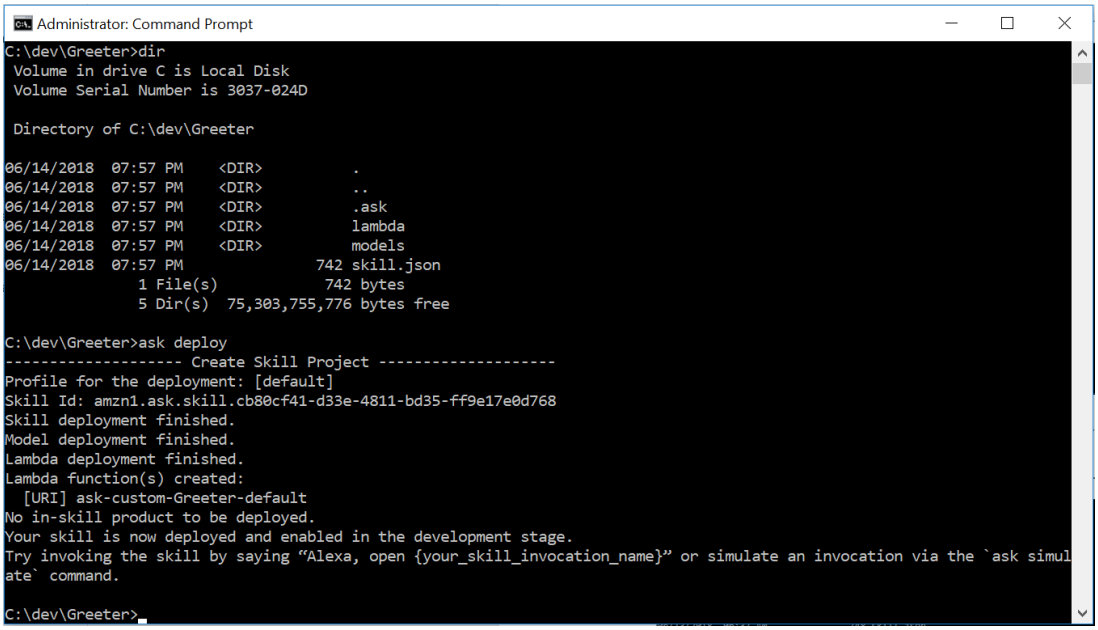
as shown below



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



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



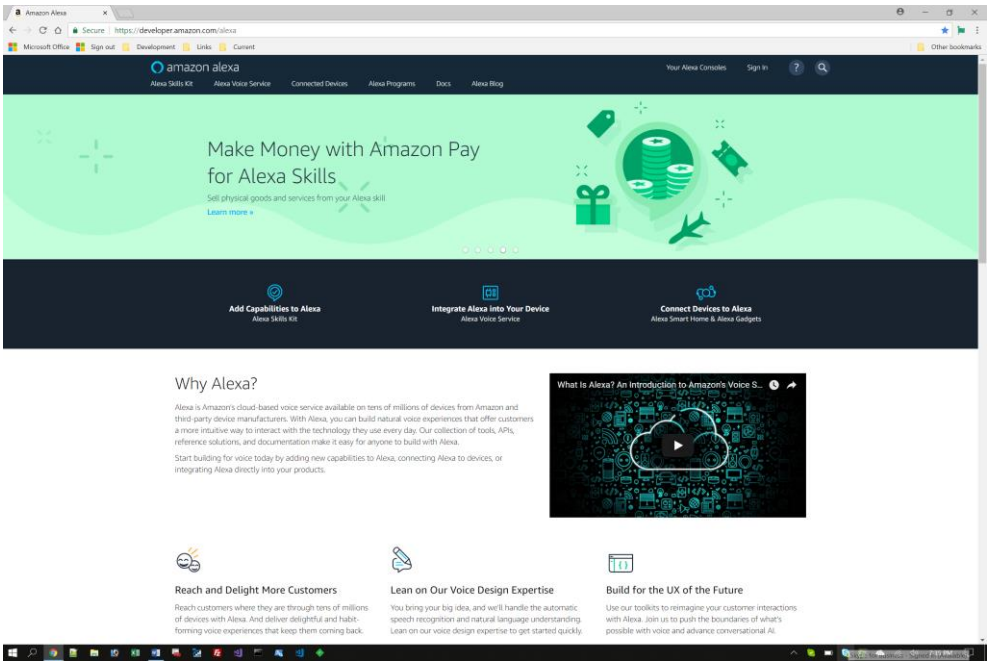
The local code has now been deployed to Amazon and setup as an Alexa Skill.

Verify that the Alexa Skill was Deployed

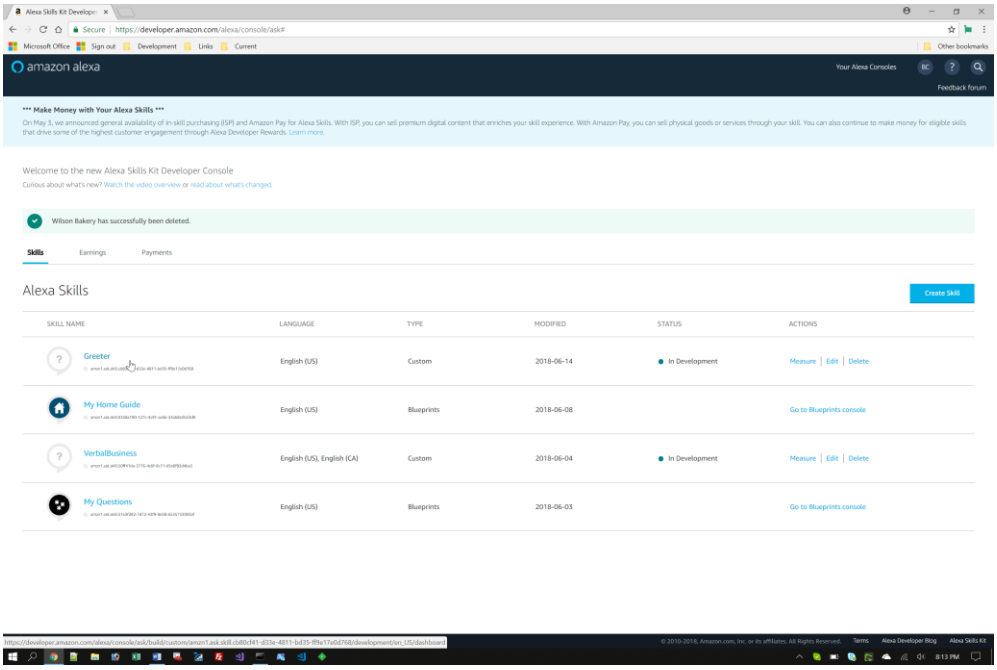
The above deployment process deployed the code for both the Alexa Skill Interface and the Alexa Skill Service. Let's have a look at what was deployed.

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

https://developer.amazon.com/alexa



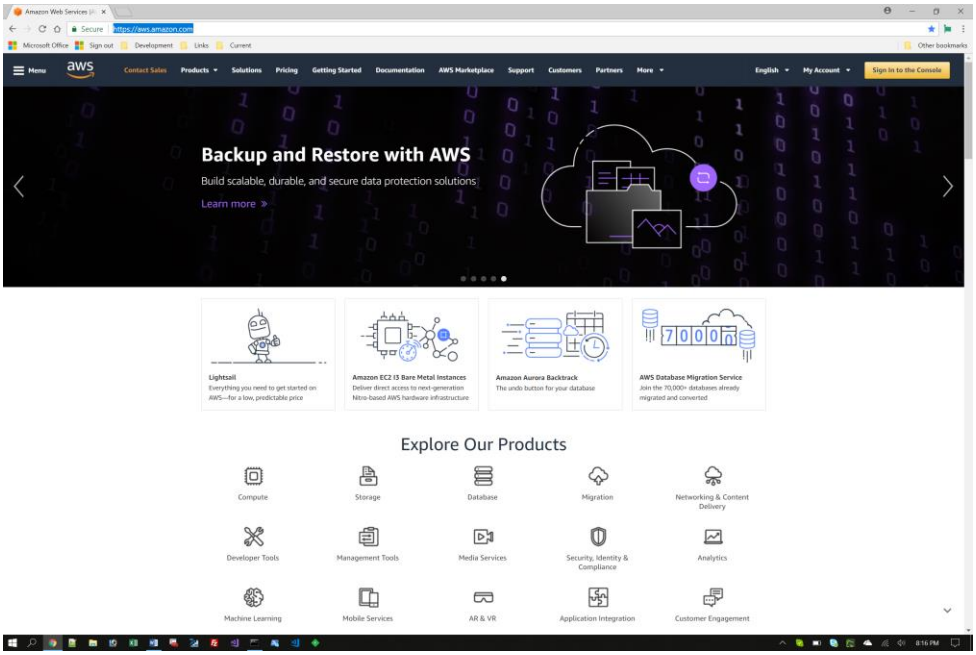
Login to the Skill Console and go to the Skill Console. You will be presented with the following



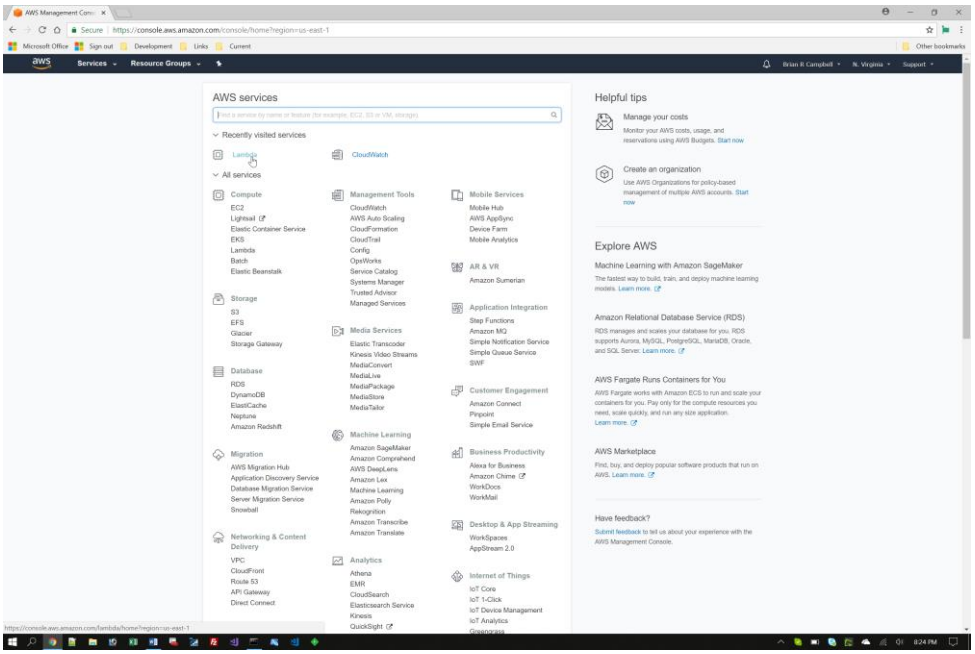
Notice that the ‘Greeter Skill’ that we previously deployed is indeed listed, indicating that the Skill Interface was successfully deployed.

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

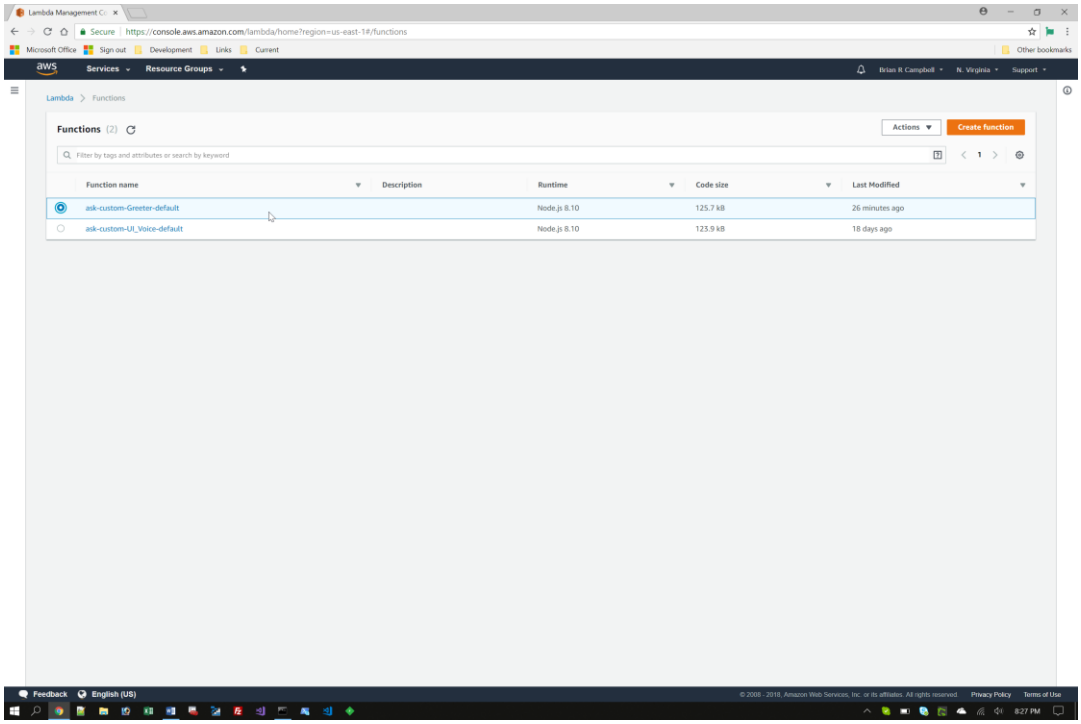
https://aws.amazon.com/



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

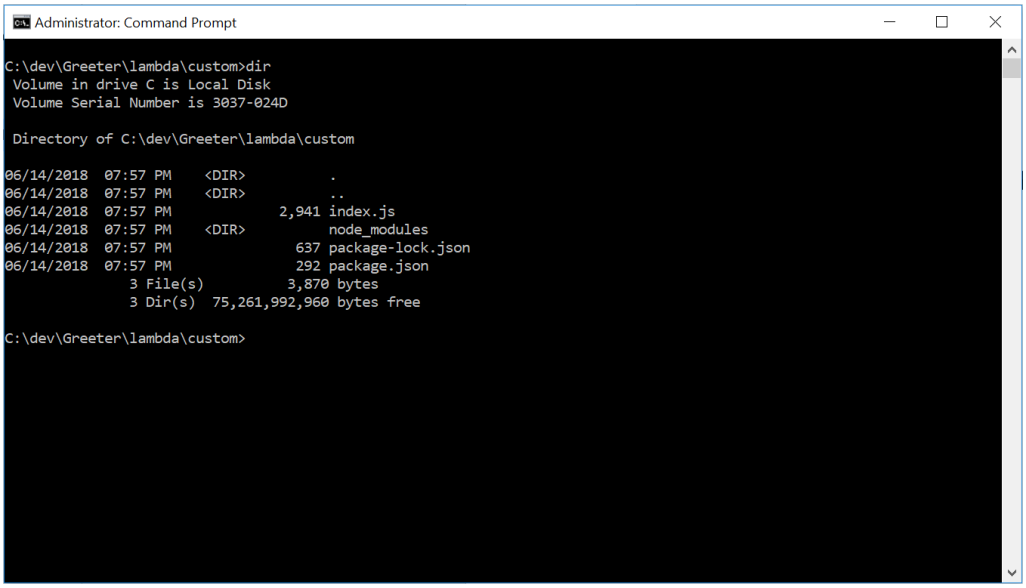


Debugging the local Alexa Skill with Bespoken Tools

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

Starting Visual Studio Code

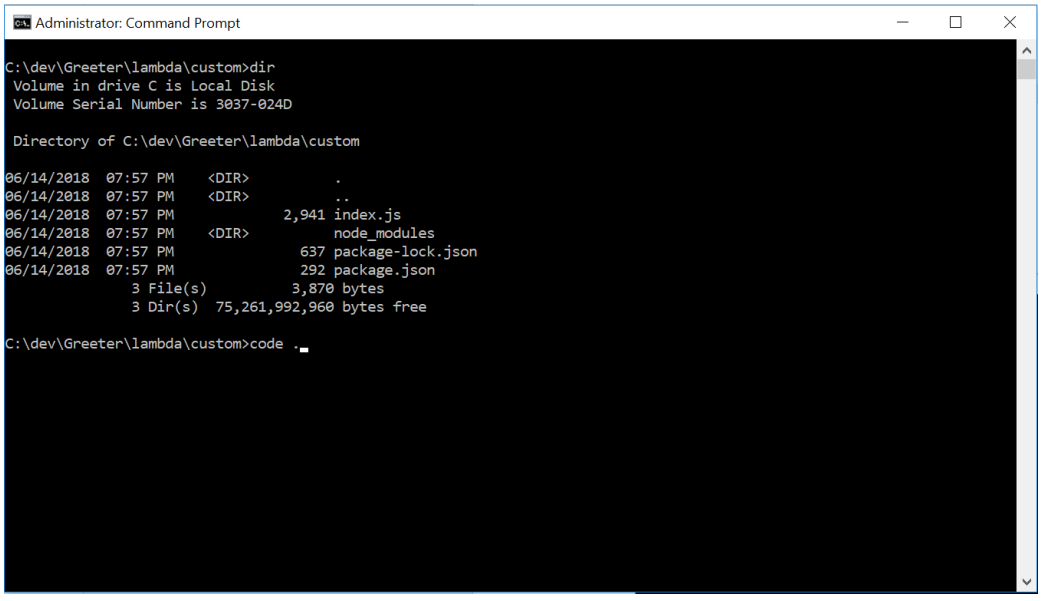
Back in the Command Prompt, navigate to the 'lambda\custom' folder as shown below



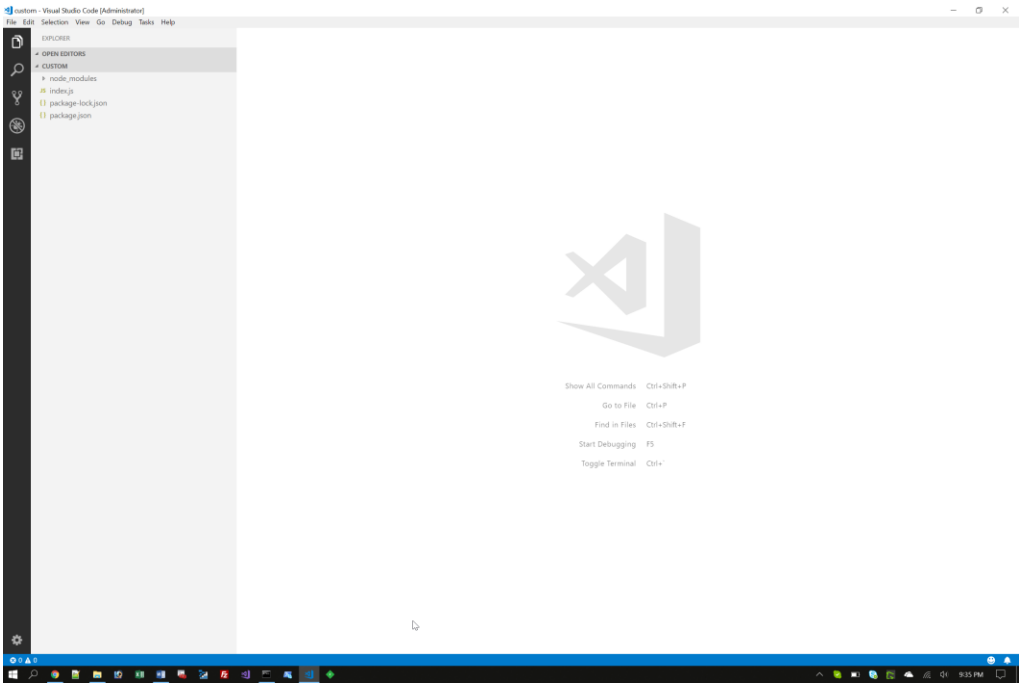
Start Visual Studio Code by entering

code .

as shown below

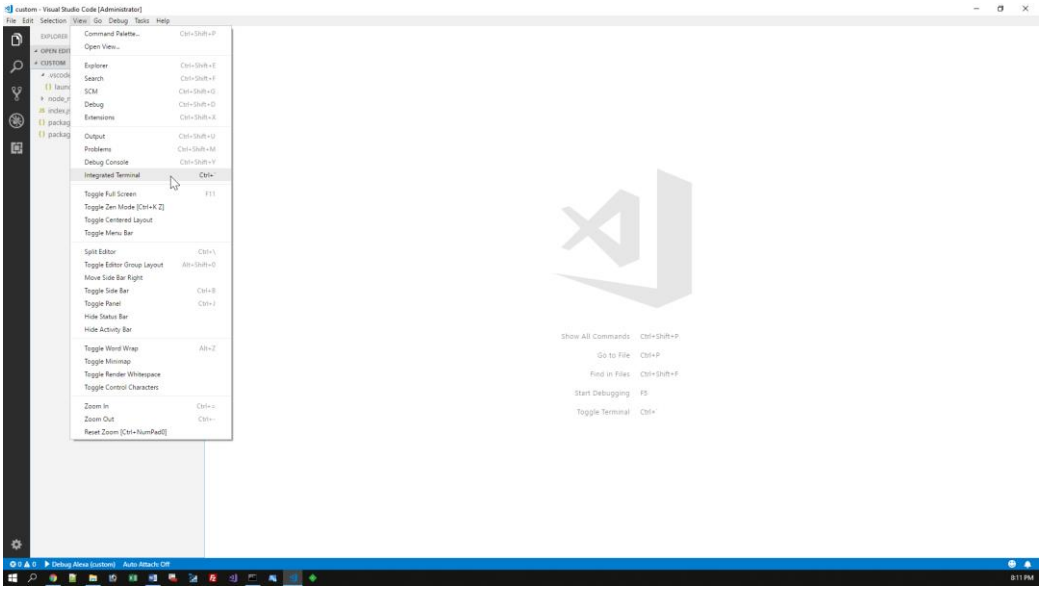


Visual Studio Code will now start up and you will be shown the following

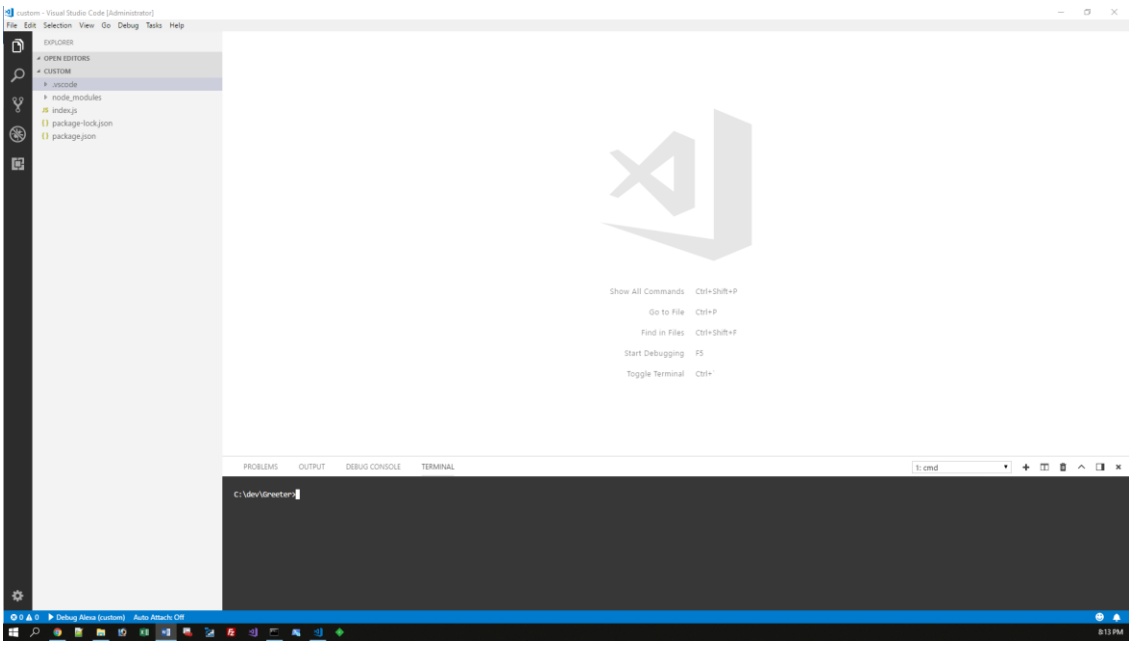


Installing the Bespoken Tools

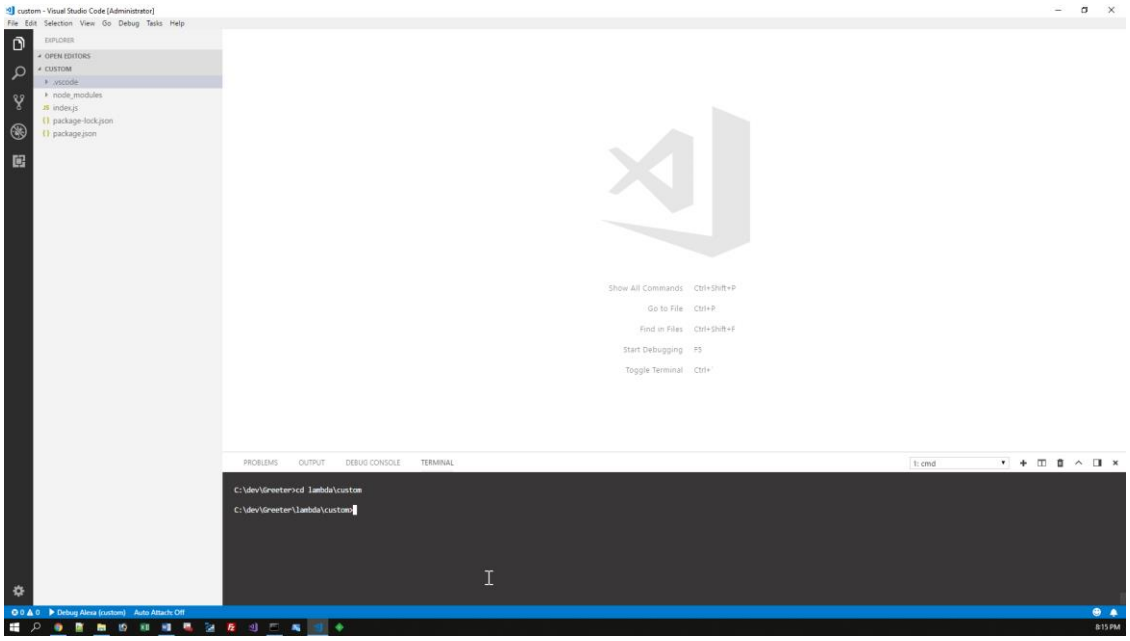
In Visual Studio Code, from the top menu select ‘View’ and then select ‘Integrated Terminal’ as shown below



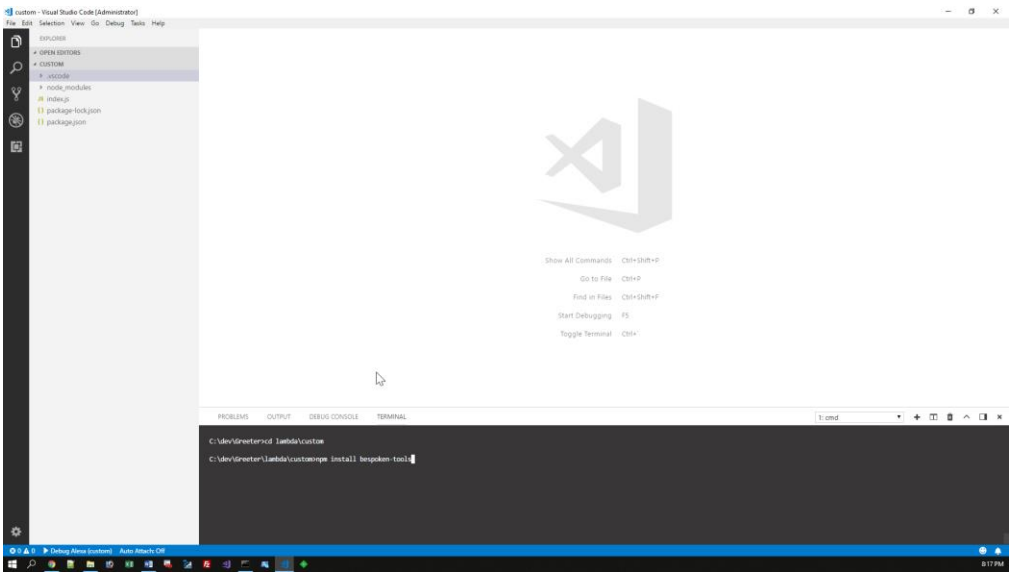
You will be shown the ‘Integrated Terminal’ at the bottom of the screen



Navigate to the lambda\custom folder as shown below



Enter ‘`npm i bespoken-tools`’ as shown below

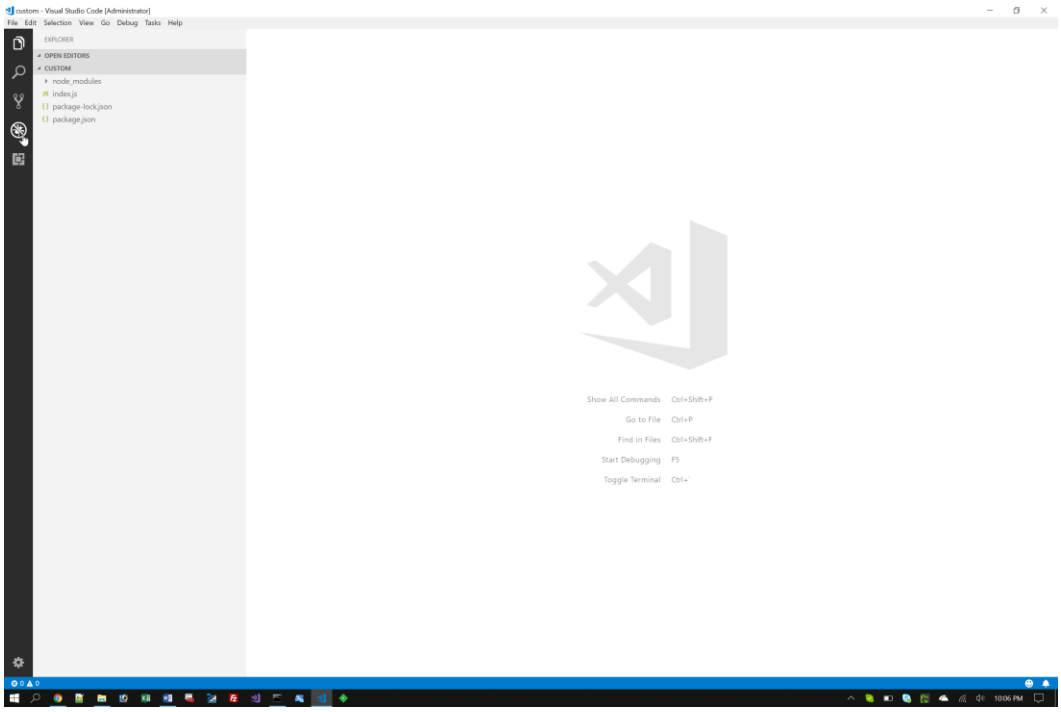


The Bespoken Tools will now be installed into the ‘node_modules’ folder.

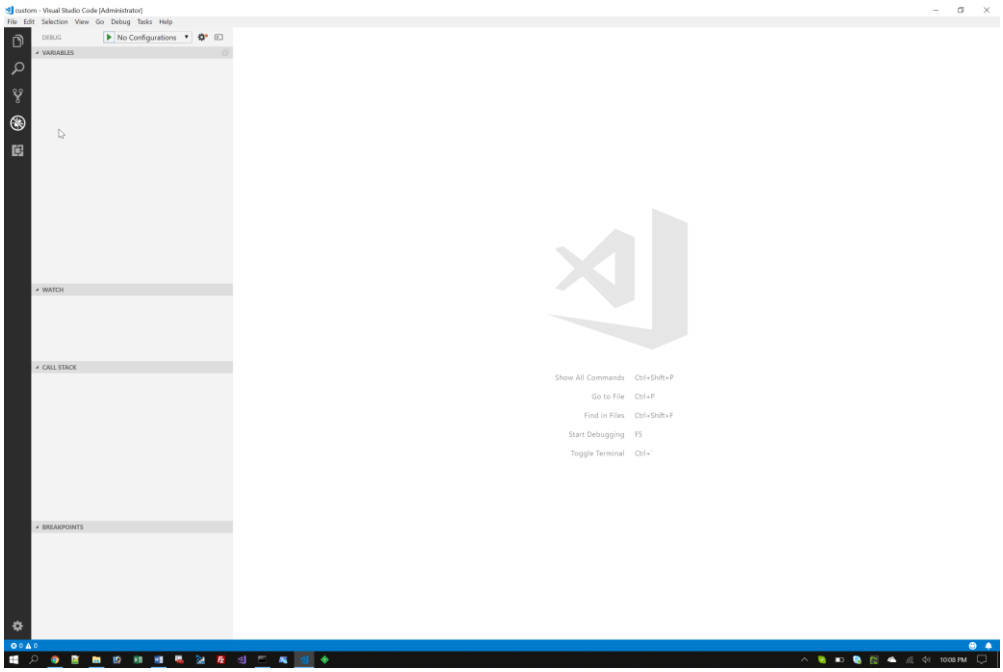
Updating Visual Studio Code Configuration

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.

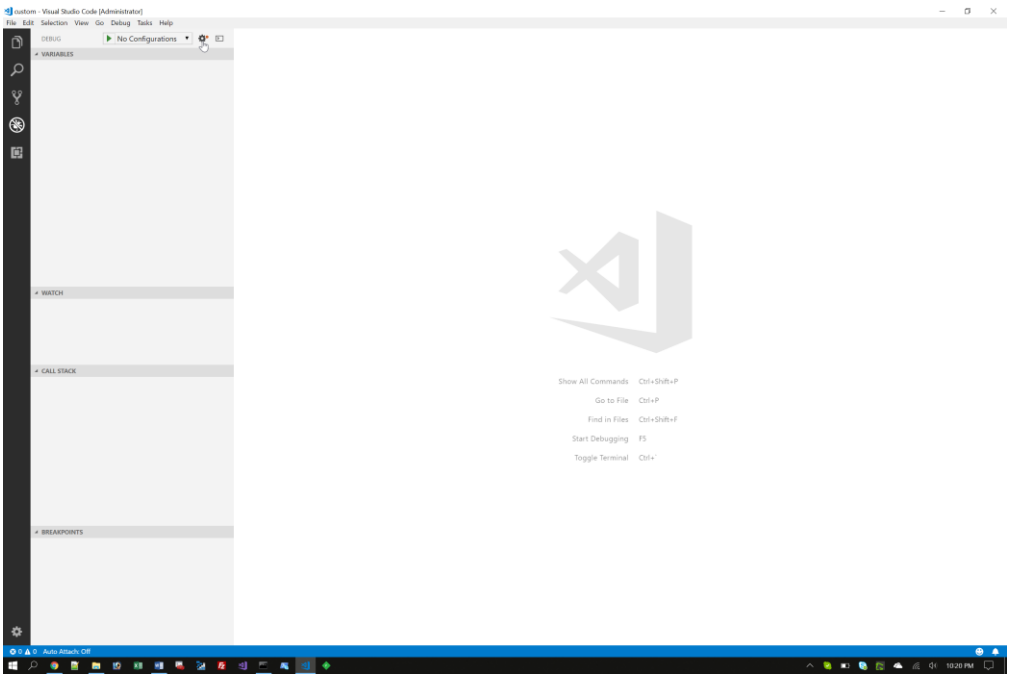
Click on the ‘Debug’ icon in the left-hand menu as shown below



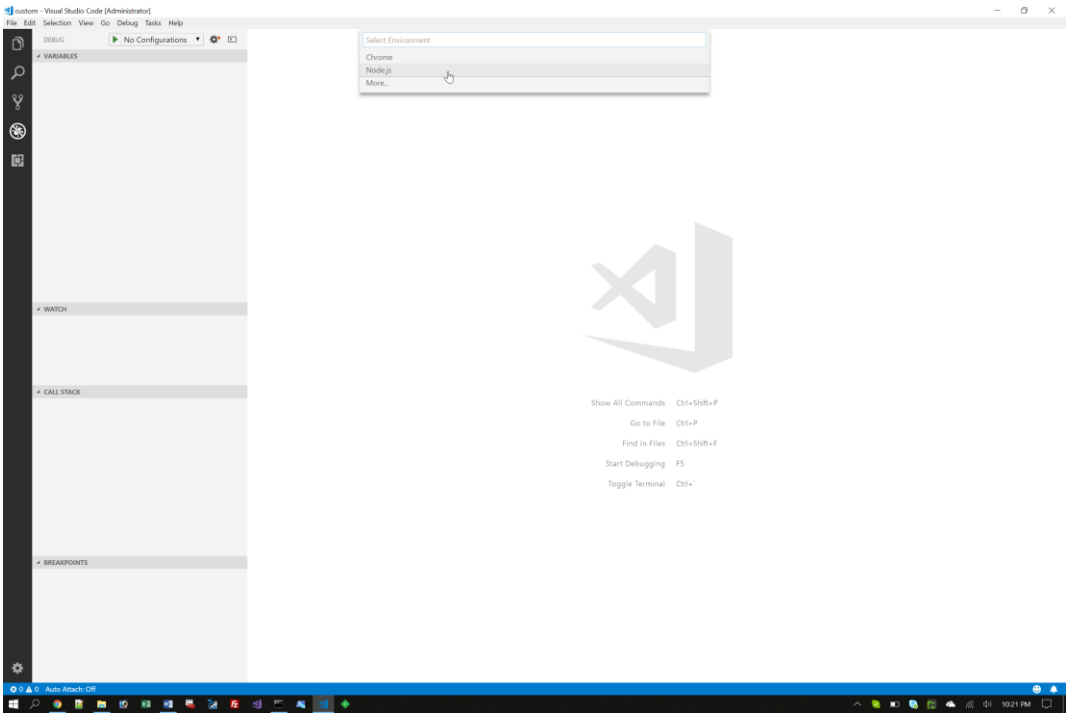
You will be shown the following screen



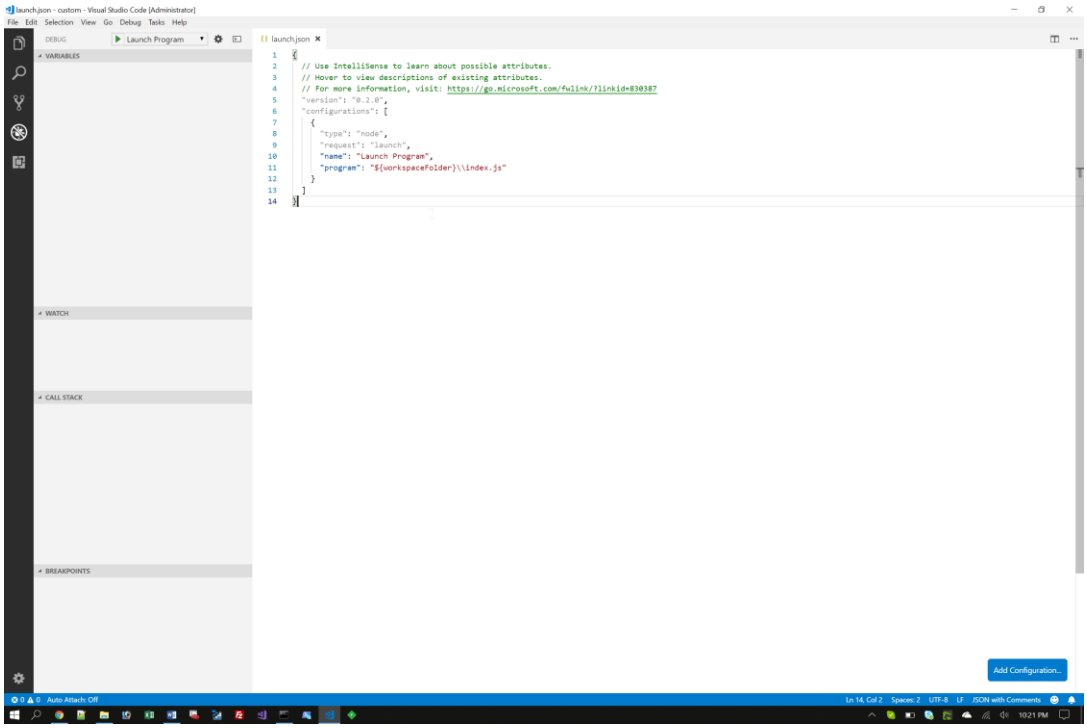
Click on the ‘Gear’ icon as shown below



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

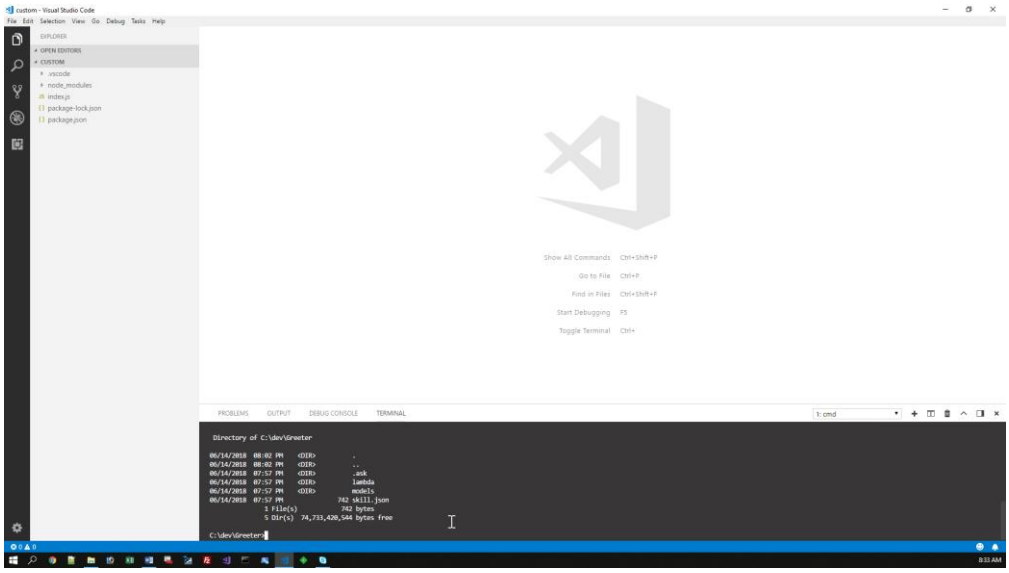
We won't be covering all of these items here; however, we will be looking at the features that specifically assist with the debugging of code.

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

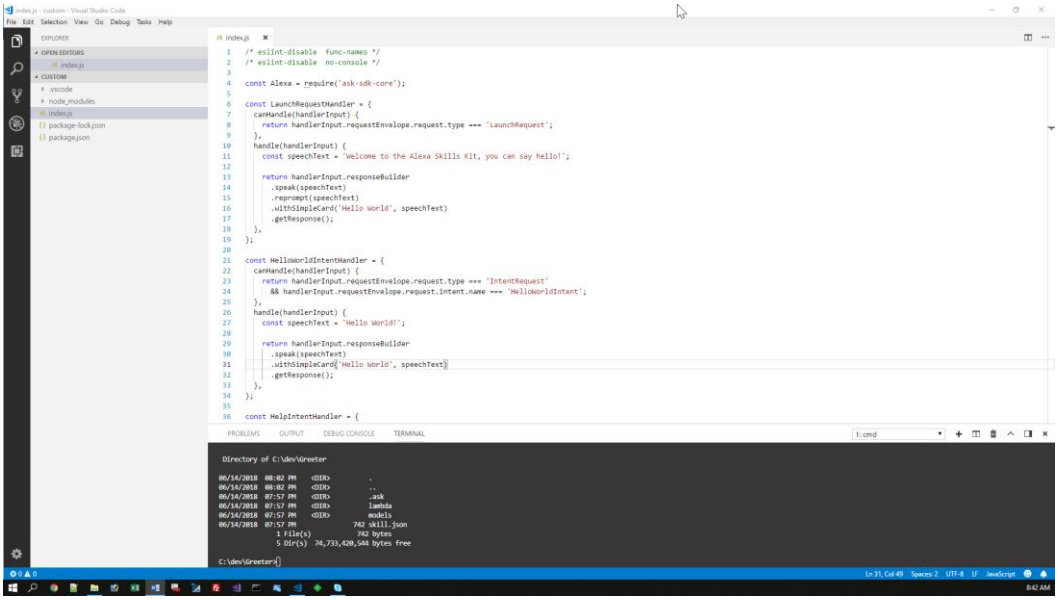
- Send commands directly to our code using the Bespoken CLI
- Send commands from an Alexa Device using the Bespoken Proxy

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

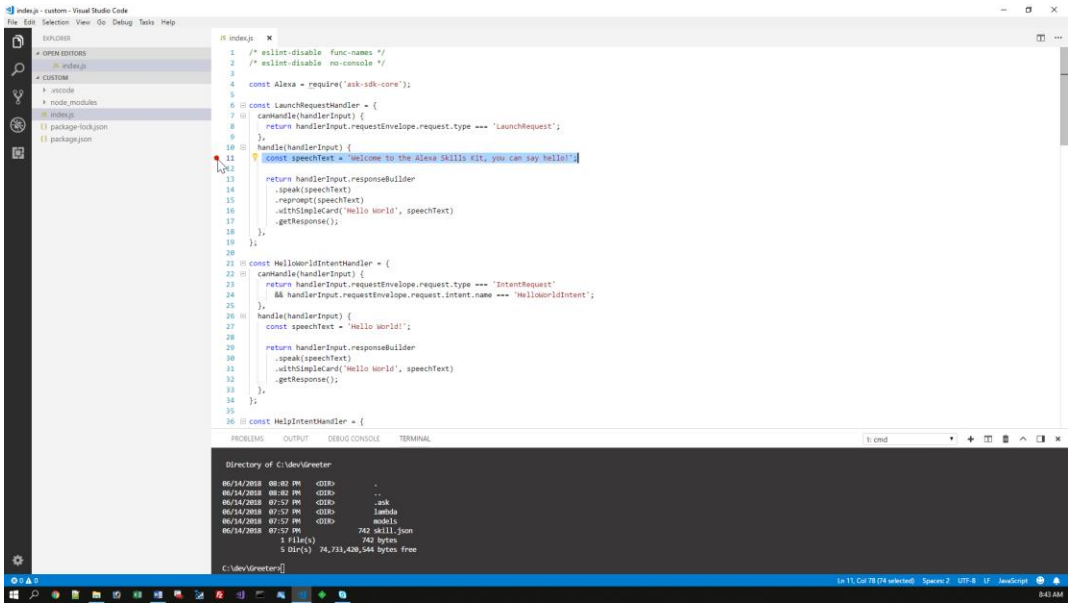


Next, open the index.js file as shown below

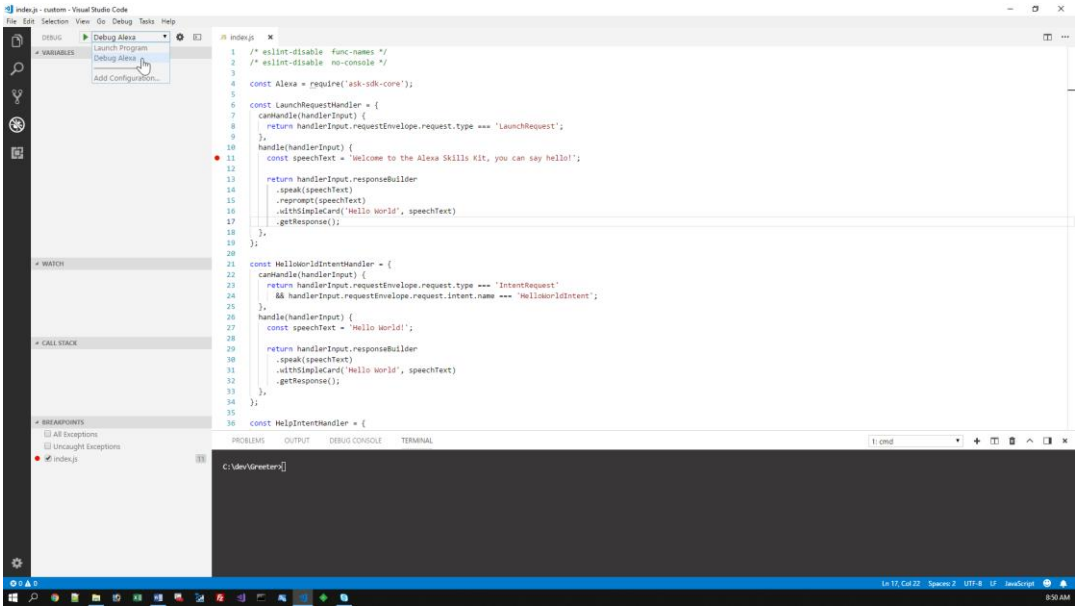


Locate the code for the ‘LaunchRequestHandler’ and place a break point on the line that reads

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

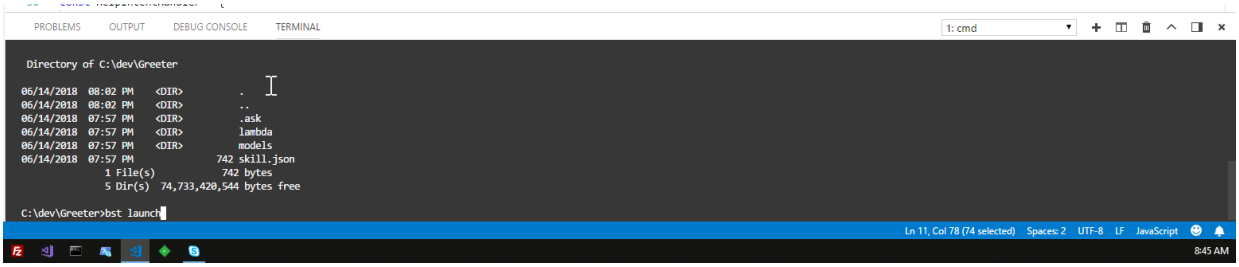


At this point it is time to start the Visual Studio Code Debugger. Click on the ‘Debug’ icon in the left-hand menu and make sure that ‘Debug Alexa’ is chosen as the configuration. Click on the green arrow or press F5 to start the Debugger

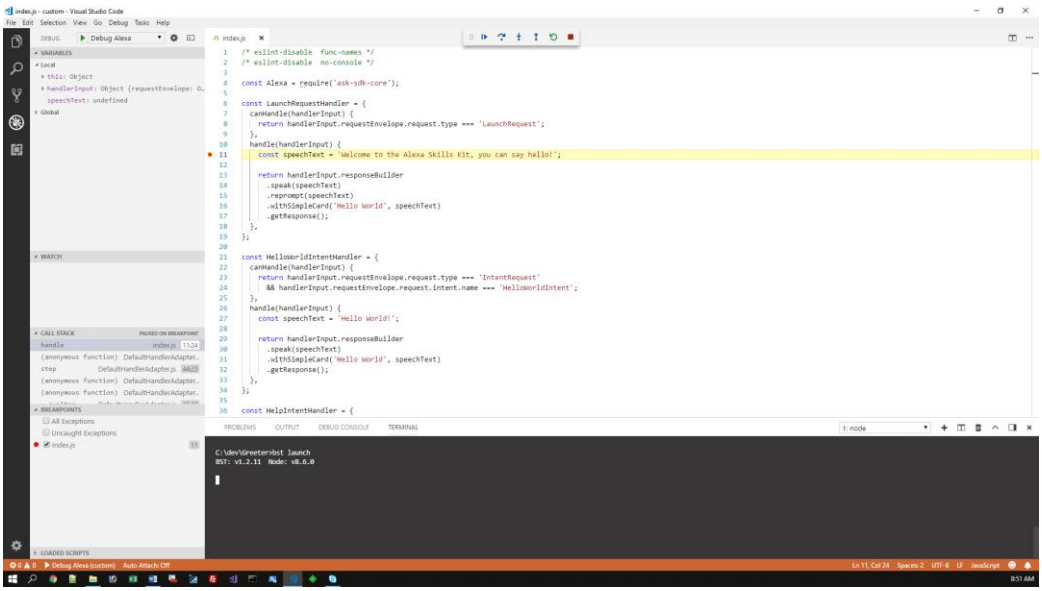


Now back in the ‘Integrated Terminal’ enter the following

bst launch

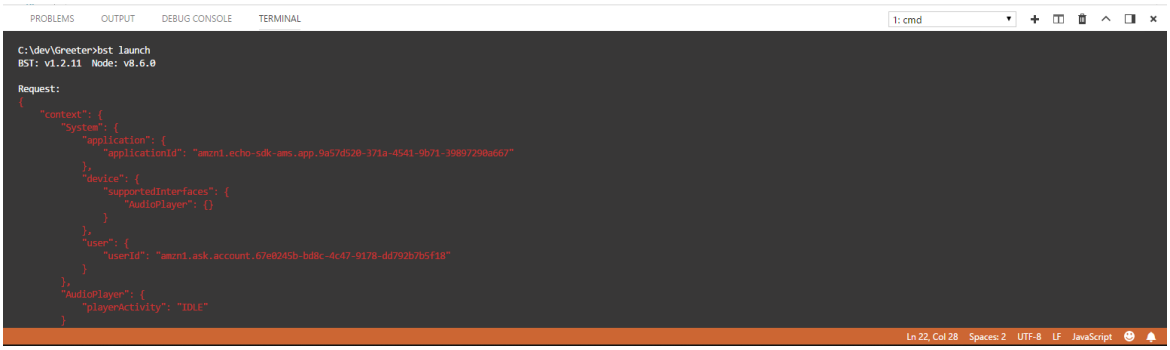


After you hit ‘Enter’, you will notice that the ‘LaunchRequestHandler’ code has been fired and Visual Studio Code has stooped on the break point that was previously set.



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

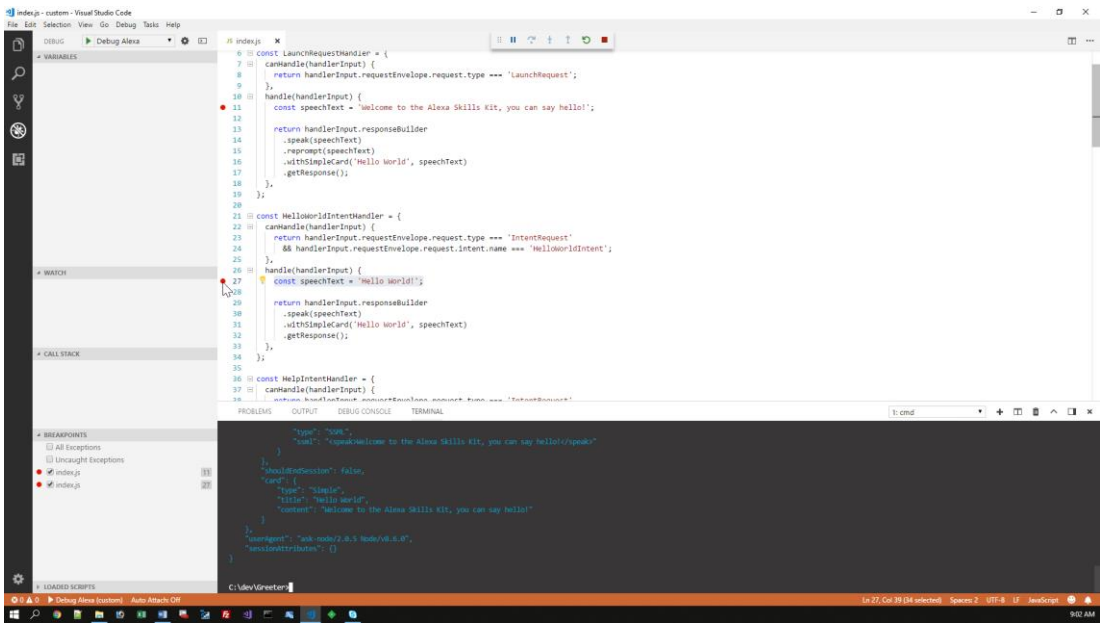
Press F5 to let the code continue. In the Terminal window you will see the actual JSON Request (red) that was sent to the ‘LaunchRequestHandler’ and the JSON Response (blue) that it returned.



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

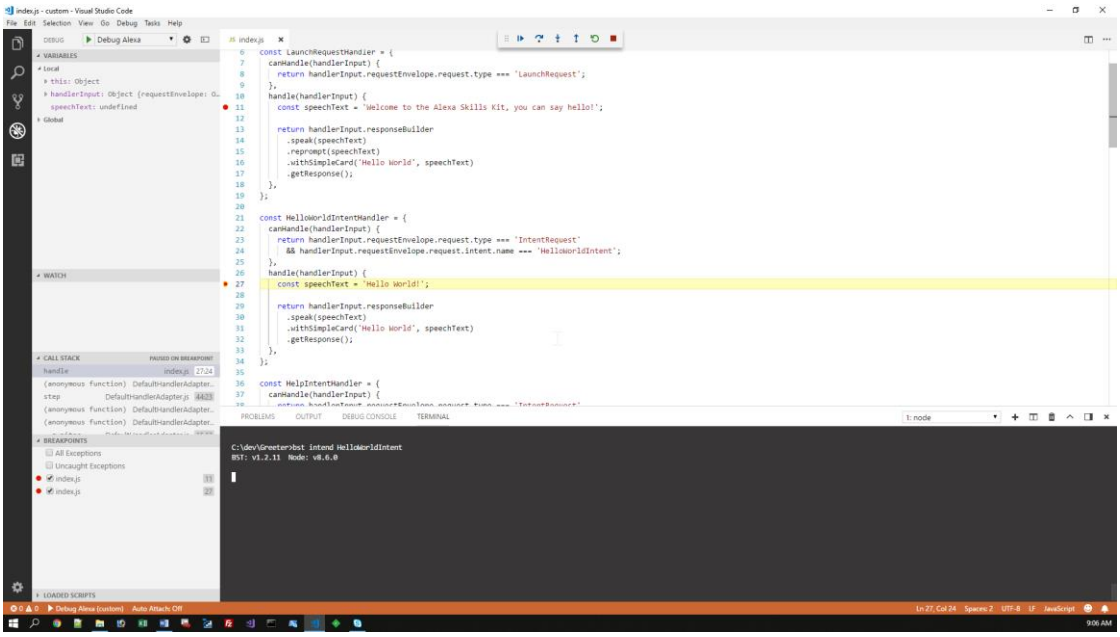
```
const speechText = 'Hello World!';
```



In the ‘Integrated Terminal enter the following

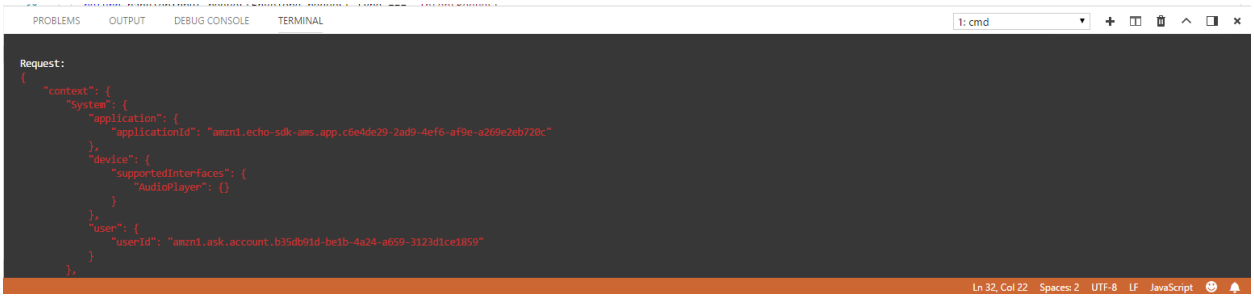
```
bst intend HelloWorldIntent
```

After you hit ‘Enter’, you will notice that the ‘HelloWorldIntentHandler’ code has been fired and Visual Studio Code has stooped 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.

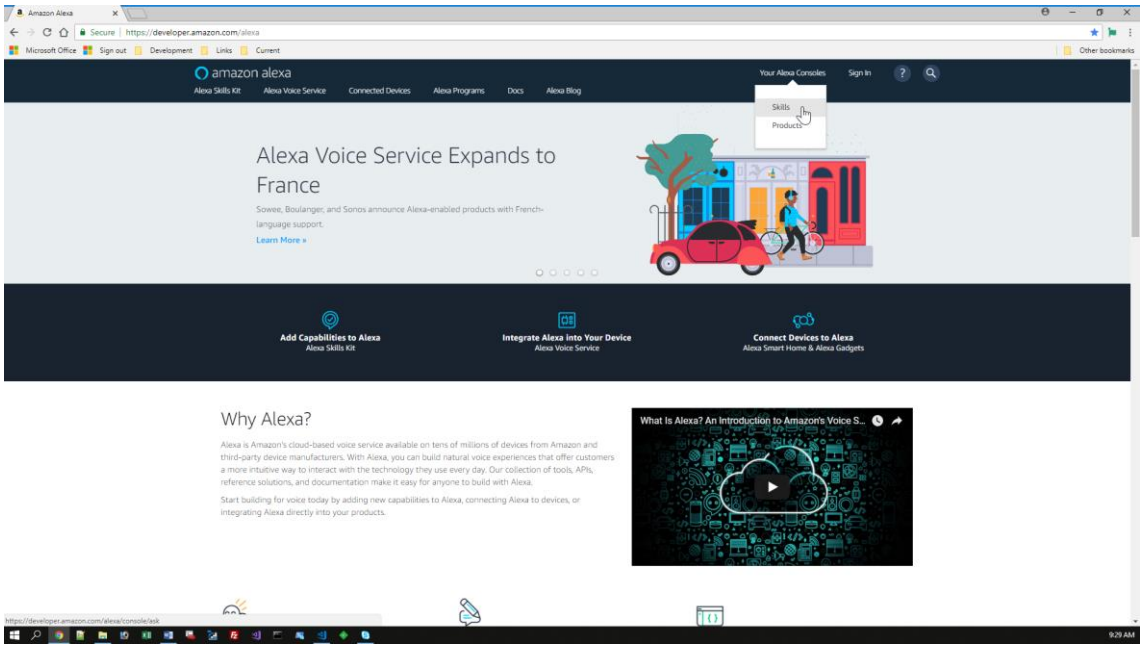


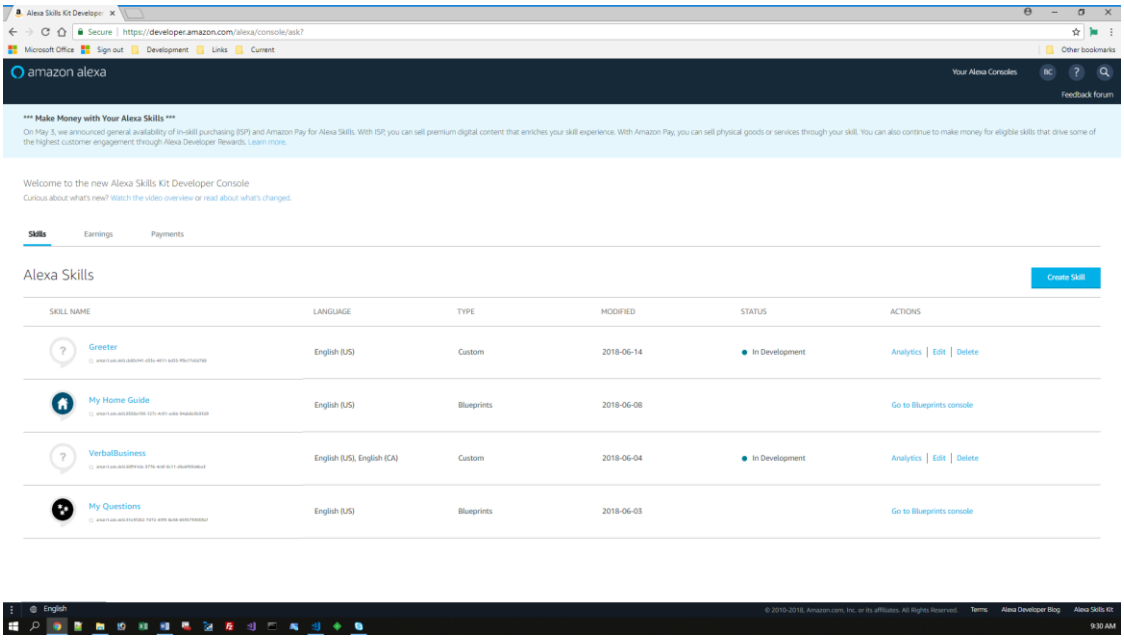
Debug an Alexa Skill using the Bespoken Proxy

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

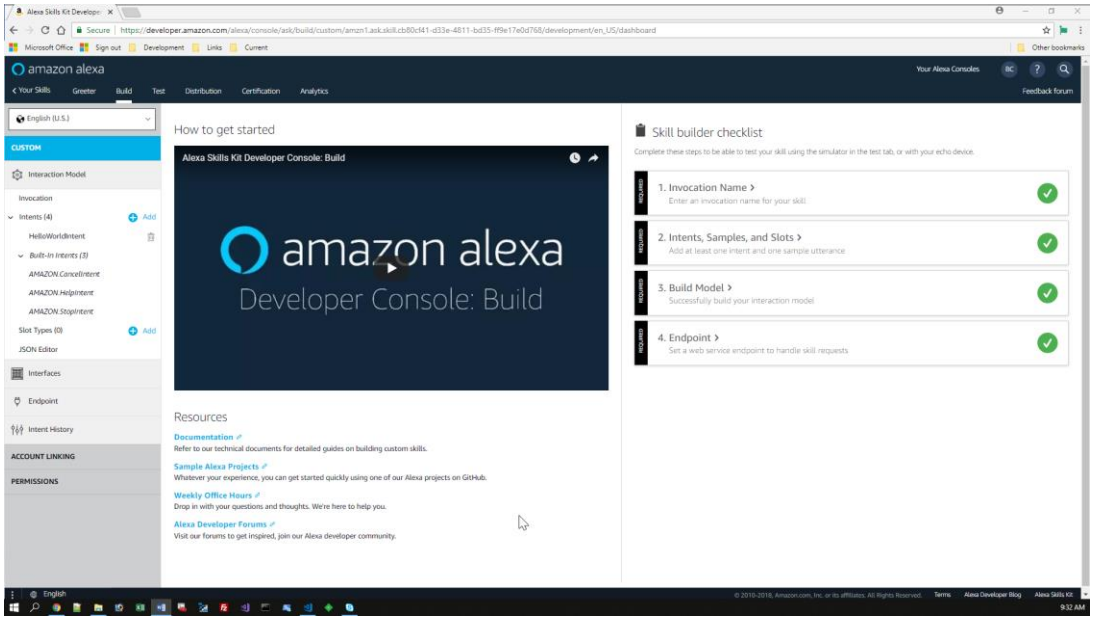
https://developer.amazon.com

as shown below

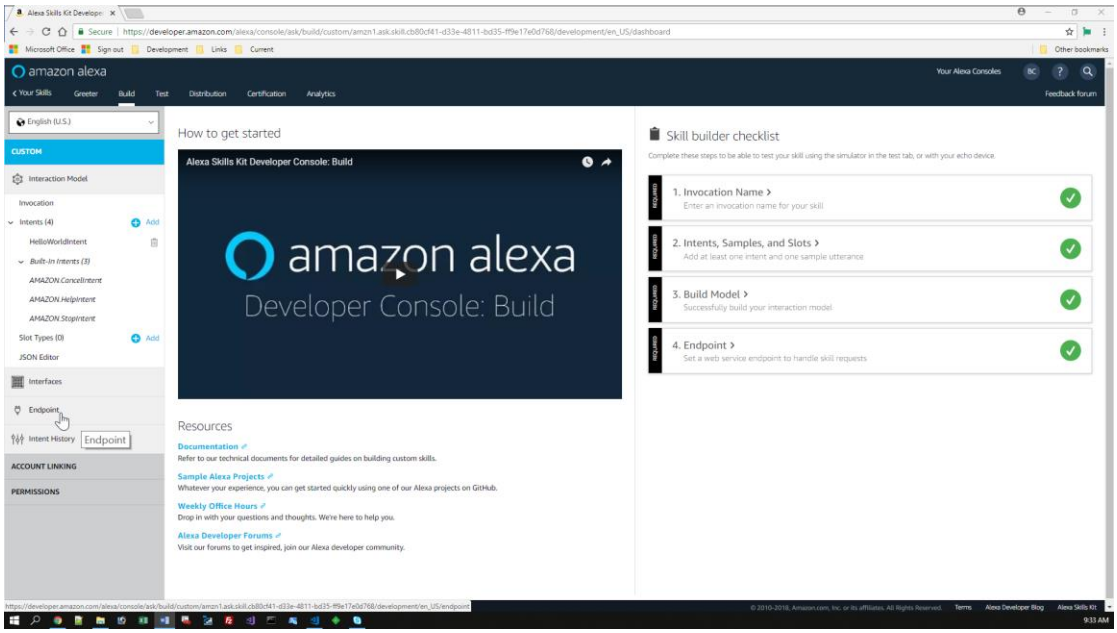




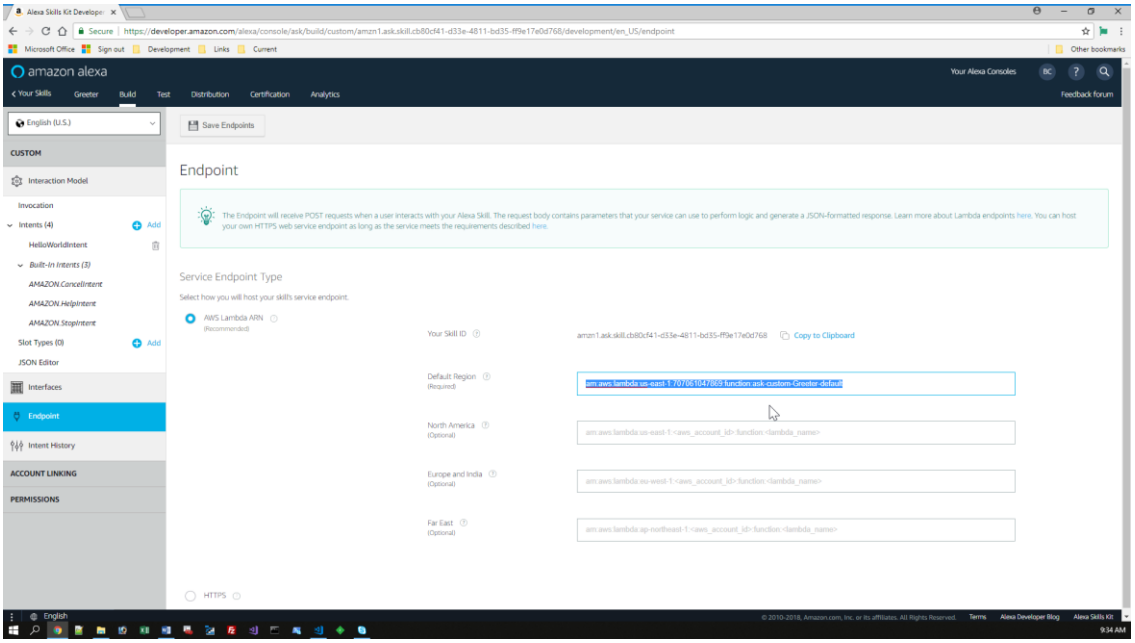
Click on the 'Greeter Skill' and you will be taken to the following page



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.

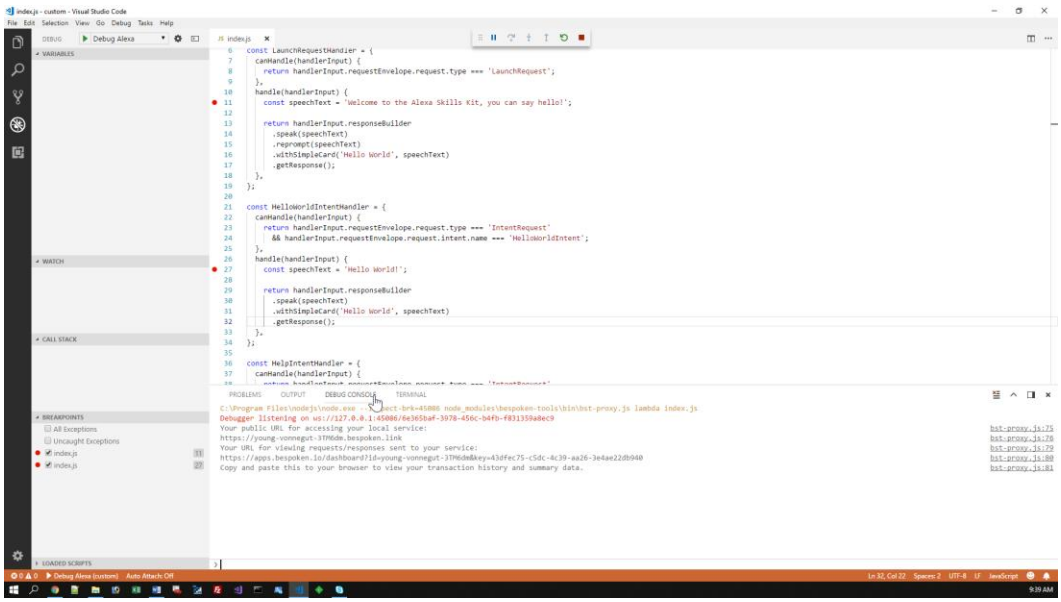


Now to debug the Skill Service Code locally, we will need to divert the Requests coming from the Skill Interface portion of the Skill to our local machine.

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.

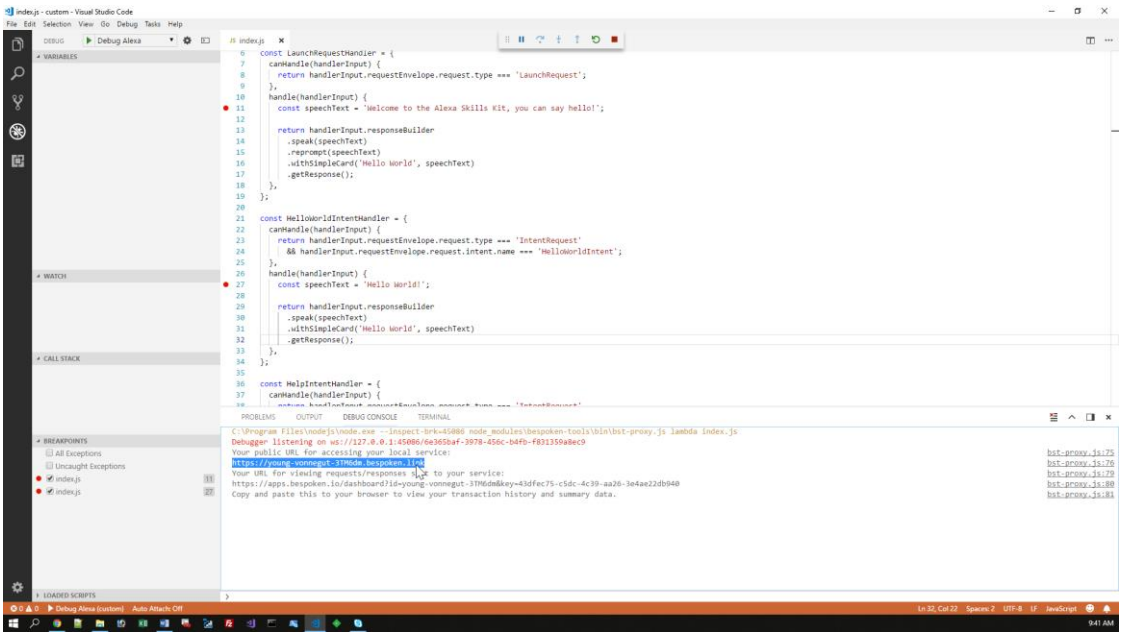
Next to the Integrated Terminal' you will find a 'Debug Console' tab. Click on that tab and you will be shown the following



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)

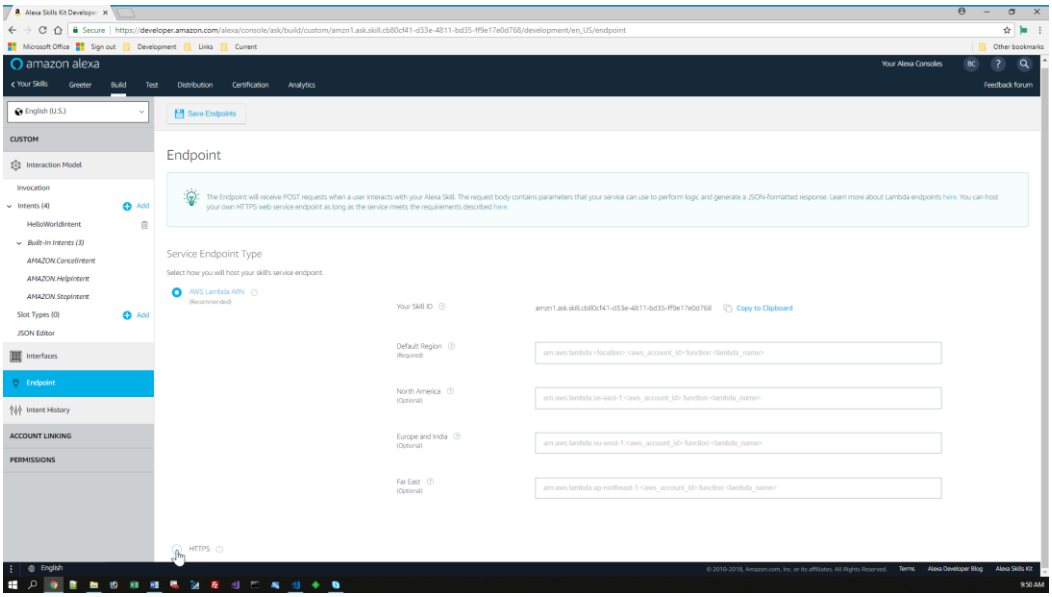
In my case the public URL is

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

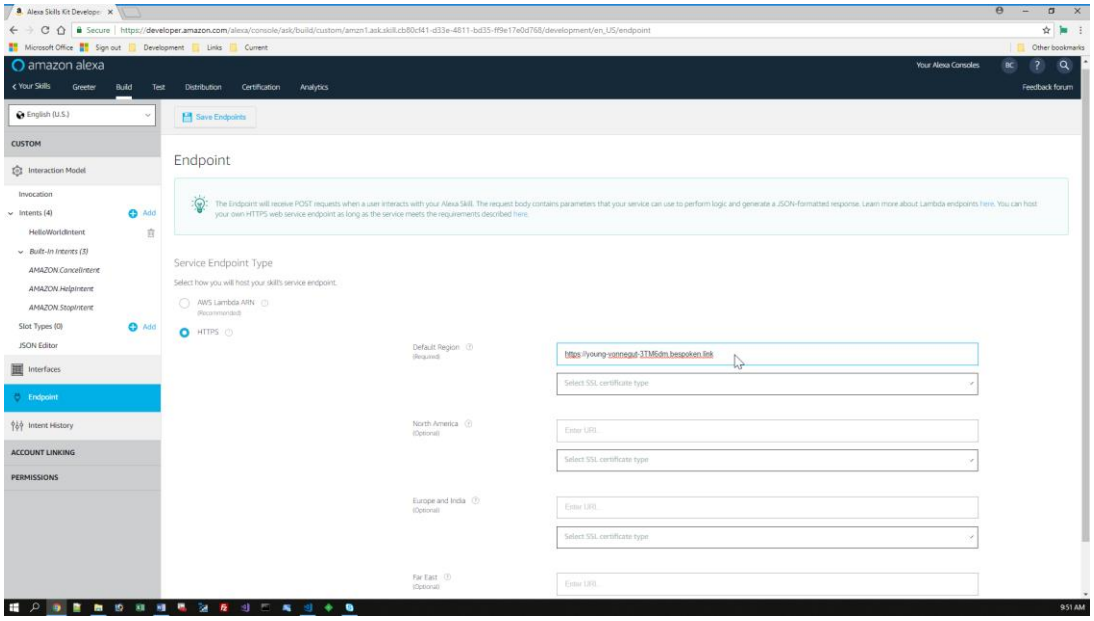


This Public URL was created when the Bespoken Tools were initially installed, and it will not change. So, we can use this Public URL to divert the Requests from the Alexa Skill Interface to our local Alexa Skill Service Code.

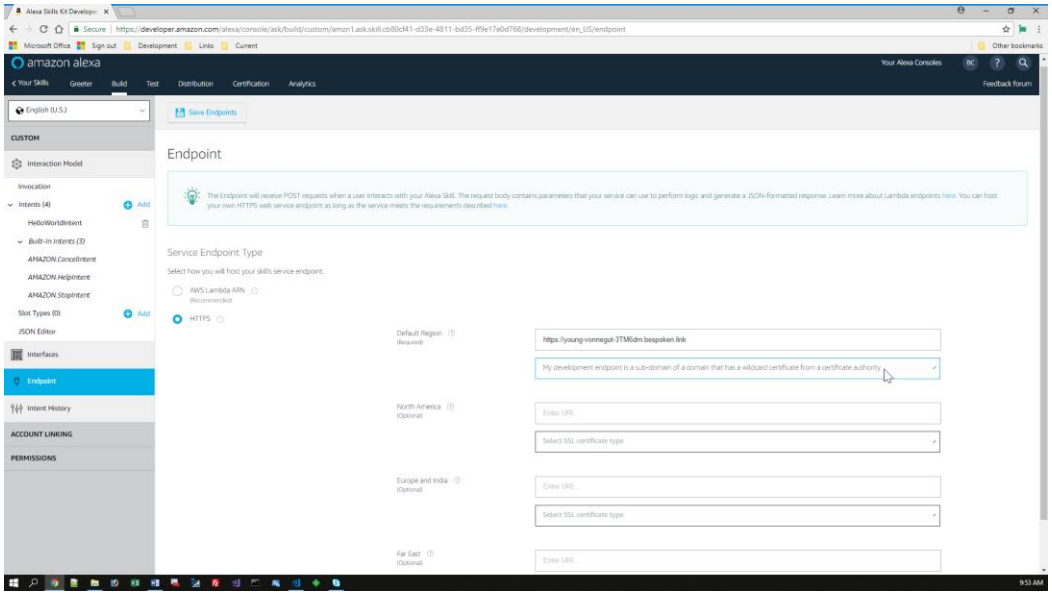
Returning to the Alexa Skill Console, change the Endpoint to a URL by clicking on the URL Radio Button



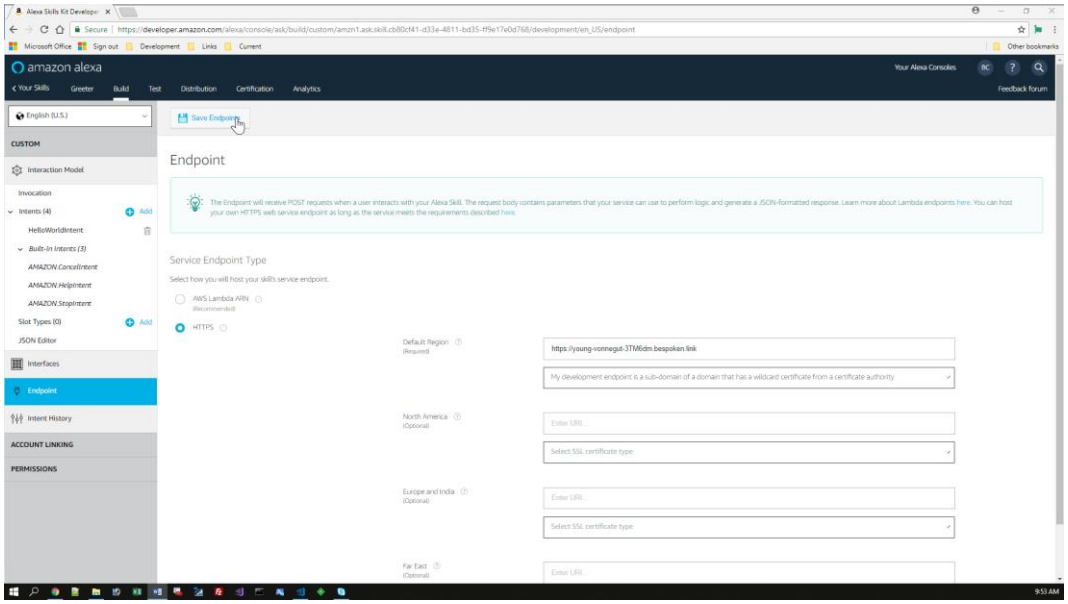
Enter the Public URL value that we were assign by the Bespoken Proxy. (make sure you use the one that was assigned to you and not the one that was assigned to me)



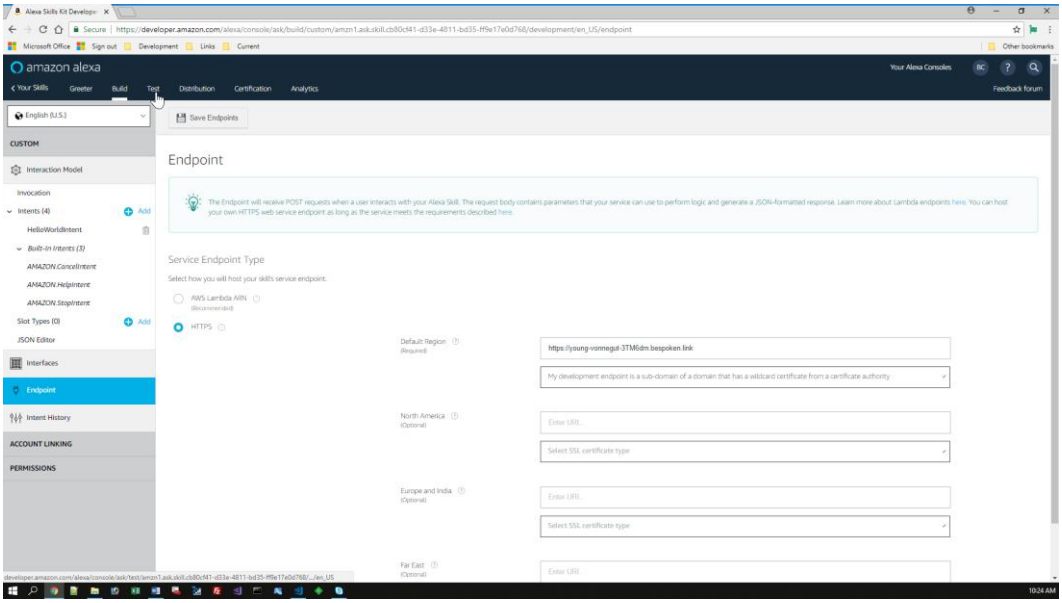
Select ‘My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority’ as the Certificate Type.



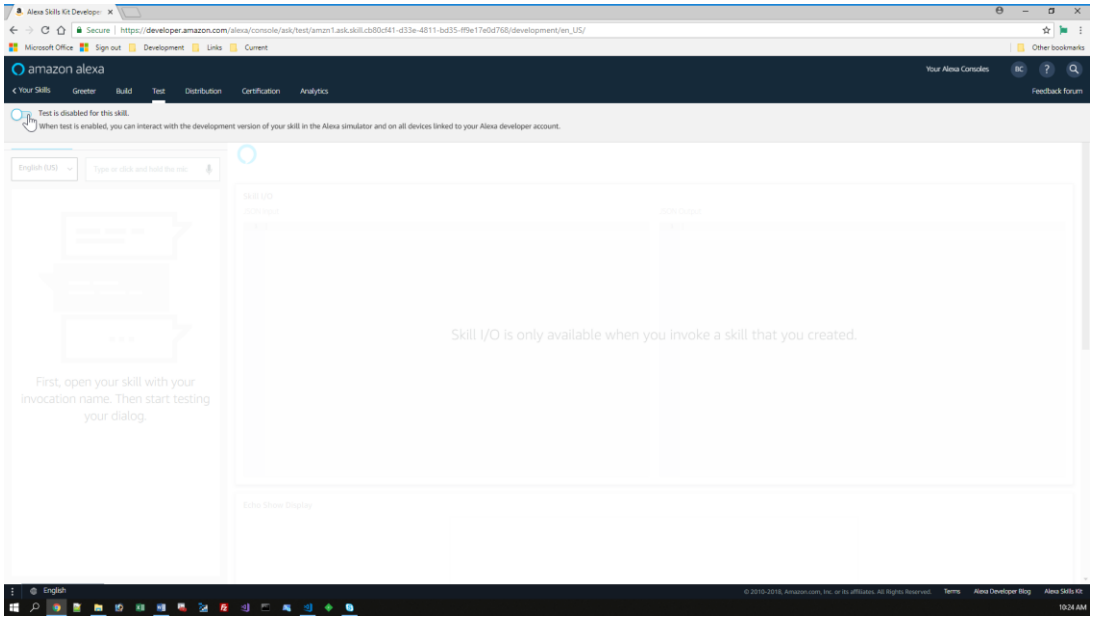
Finally click the ‘Save Endpoints’ button as shown below



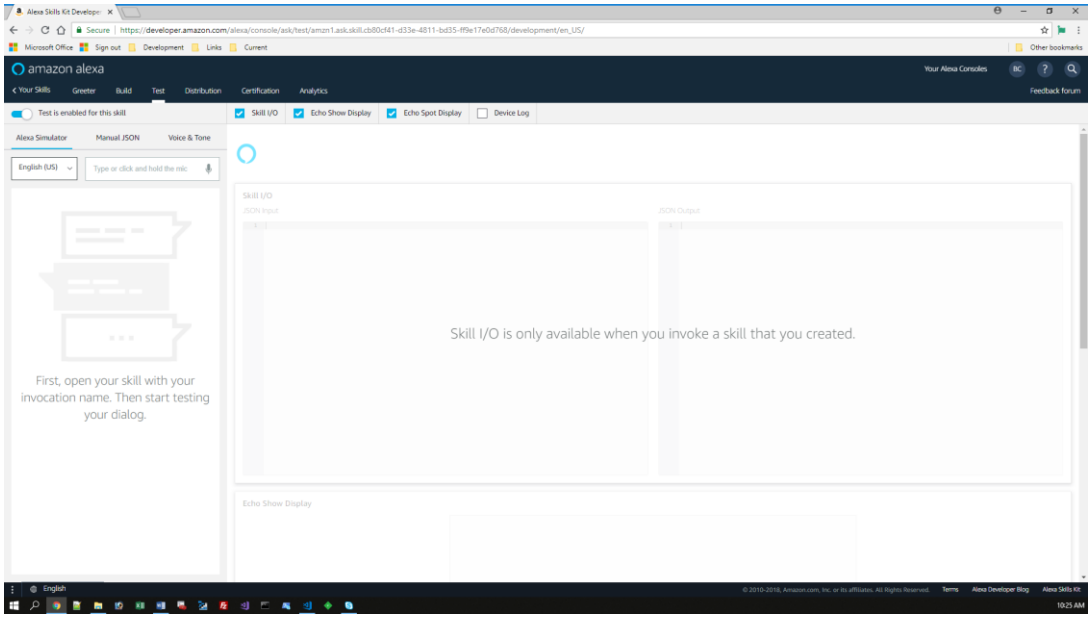
Next, click on the ‘Test’ tab as shown below



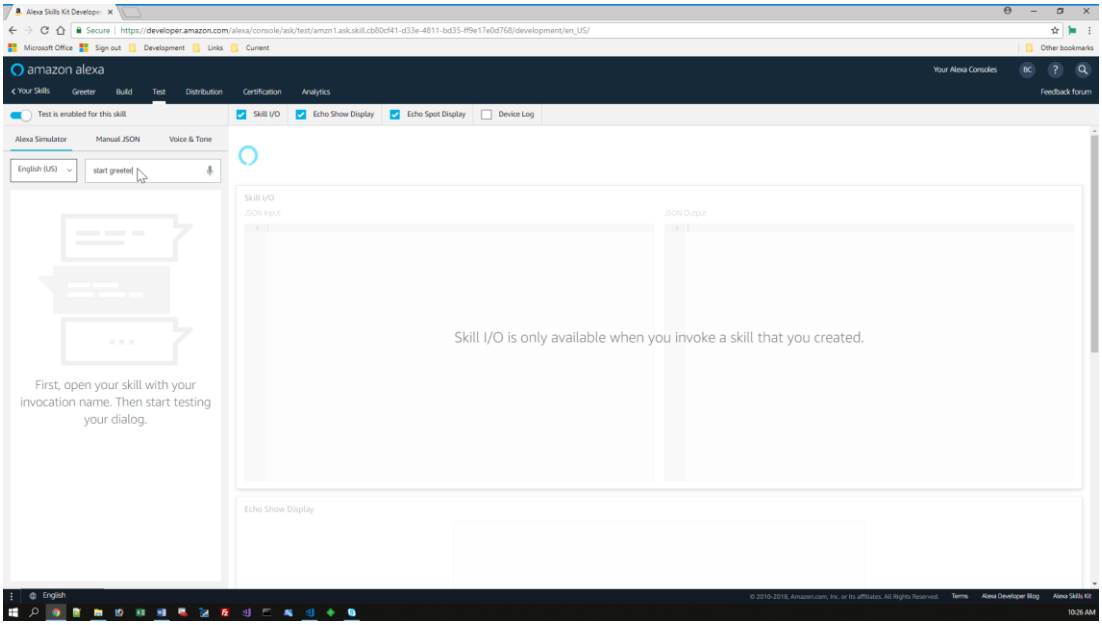
Enable testing for the Skill by clicking on the enable testing button as shown below



You will be shown the following

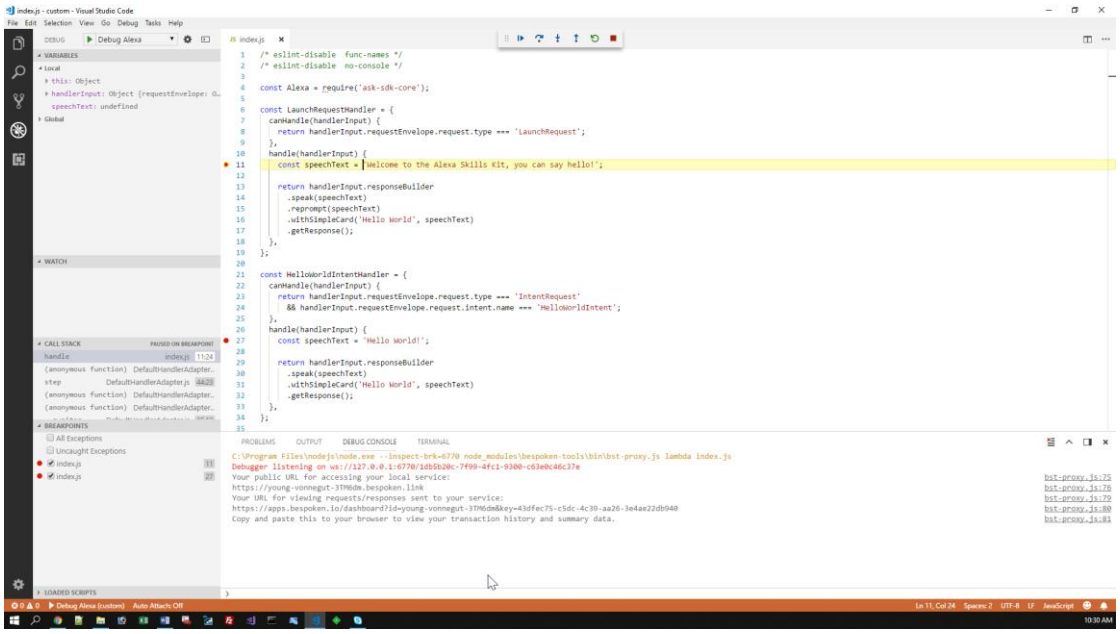


In the speech input box, enter ‘start greeter’ as shown below



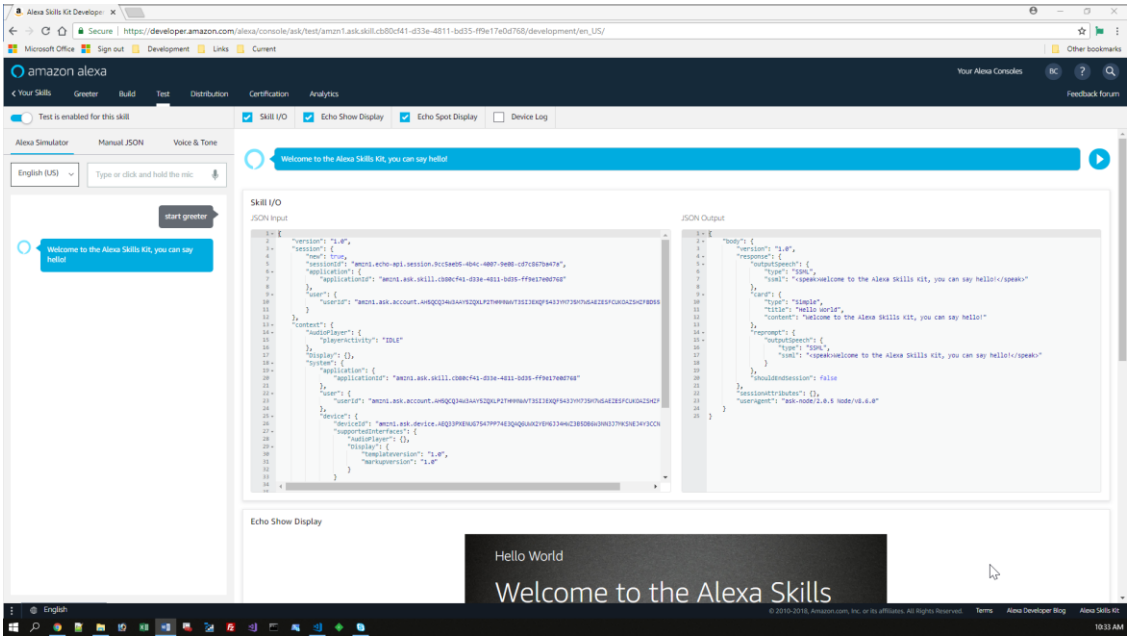
Now press Enter.

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.

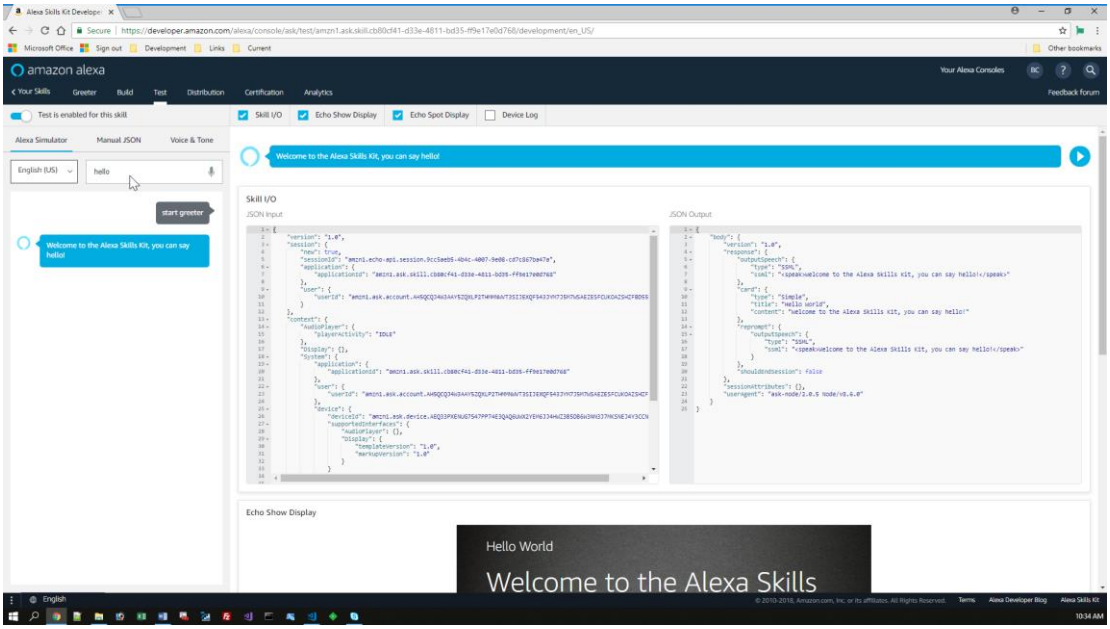


Press F5 to allow the code to continue past our break point

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.

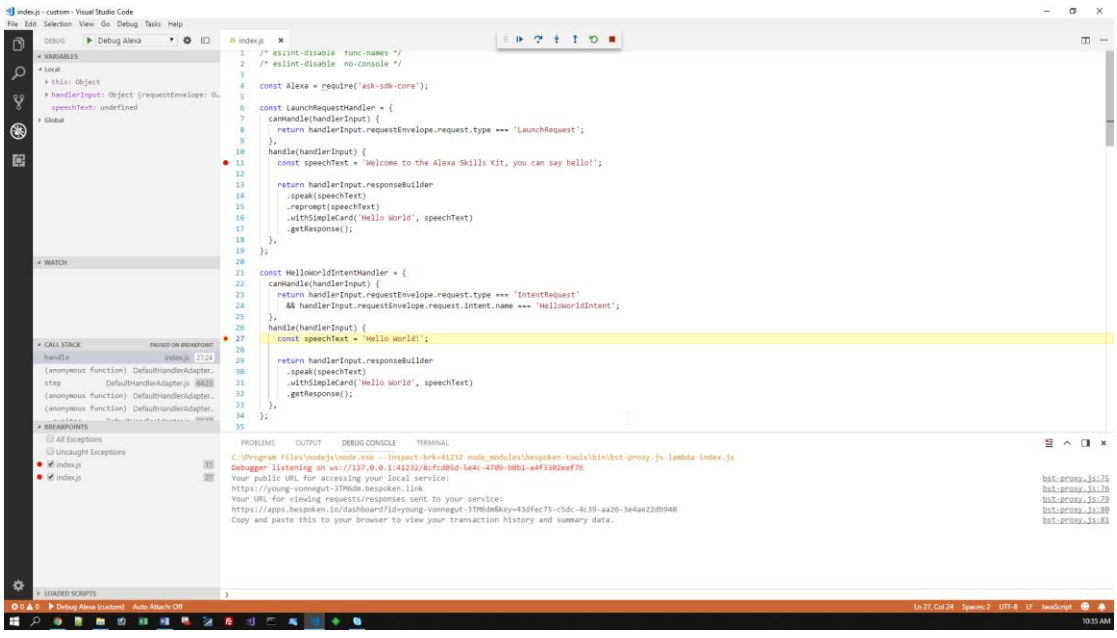


In the speech input box, enter ‘Hello’ as shown below



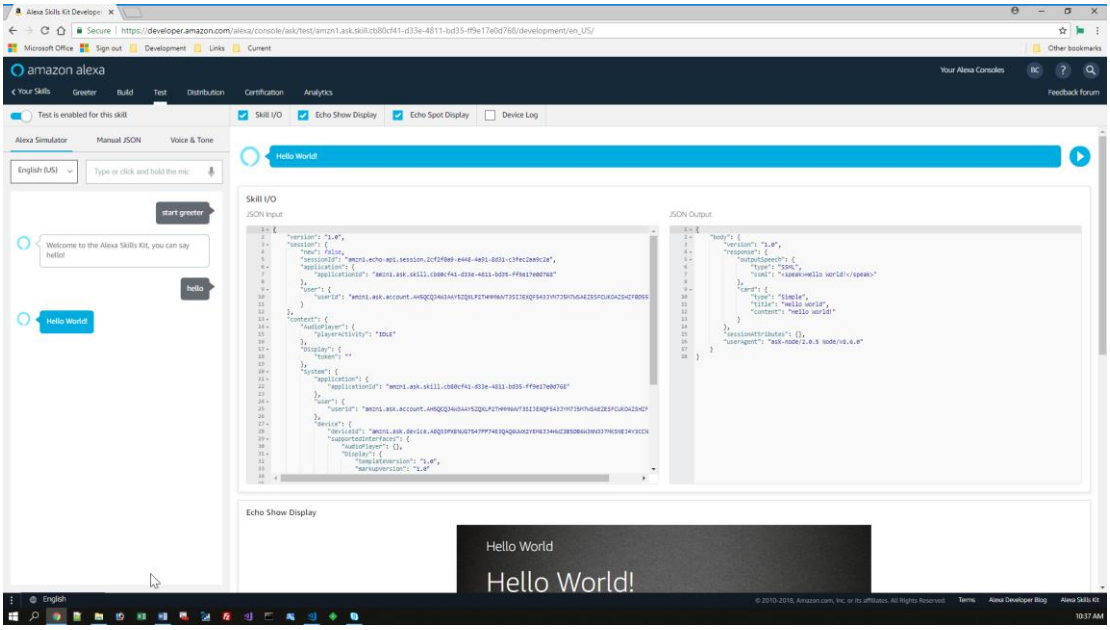
Now press Enter.

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.

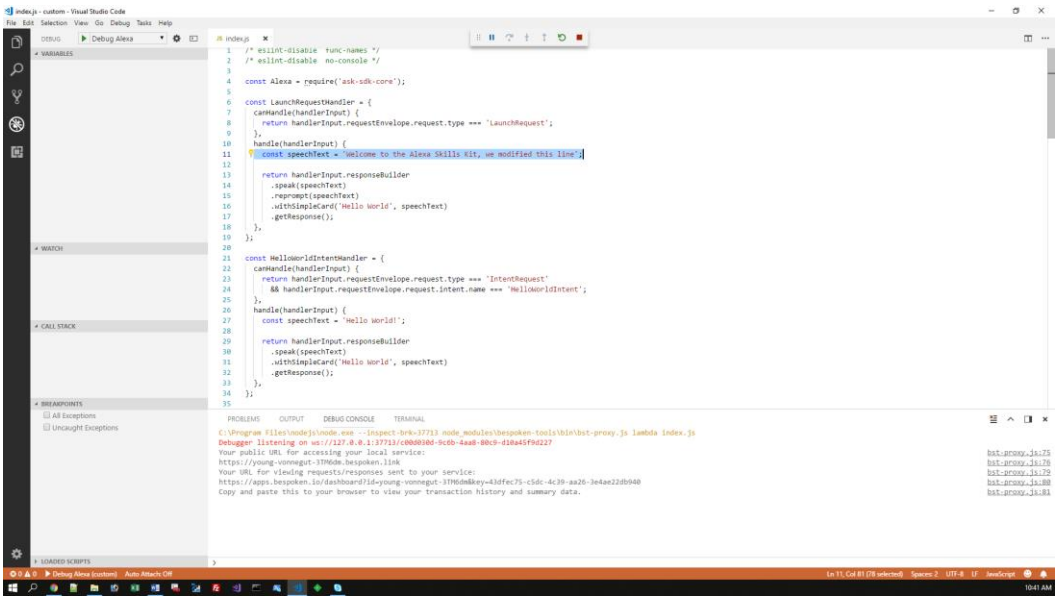
In Visual Studio Code, stop the debugger and then update the 'LaunchRequestHandler' code from

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

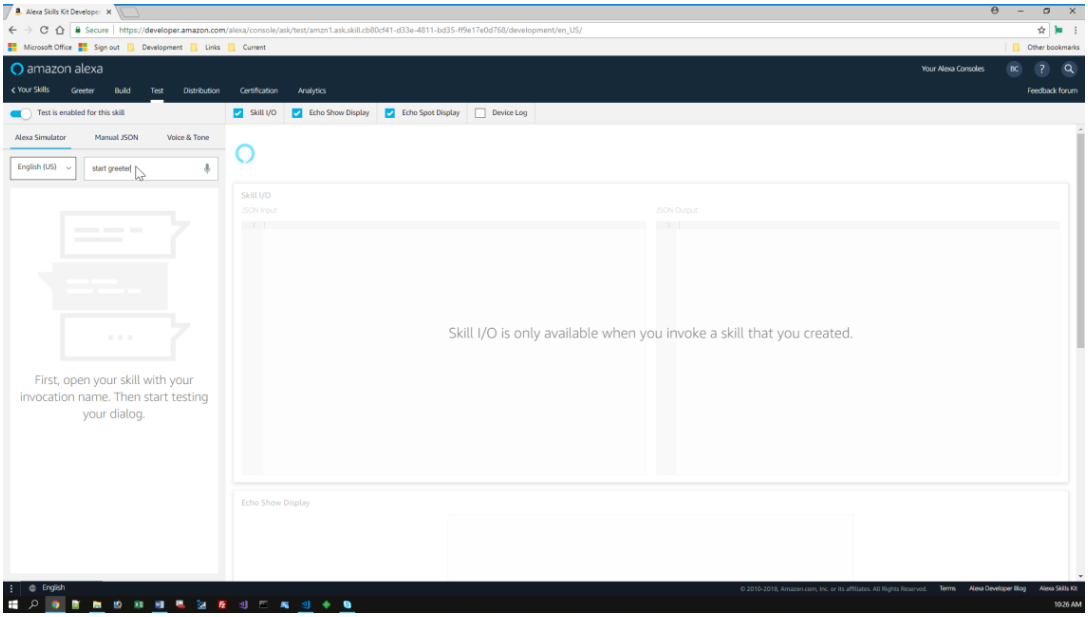
to

```
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.

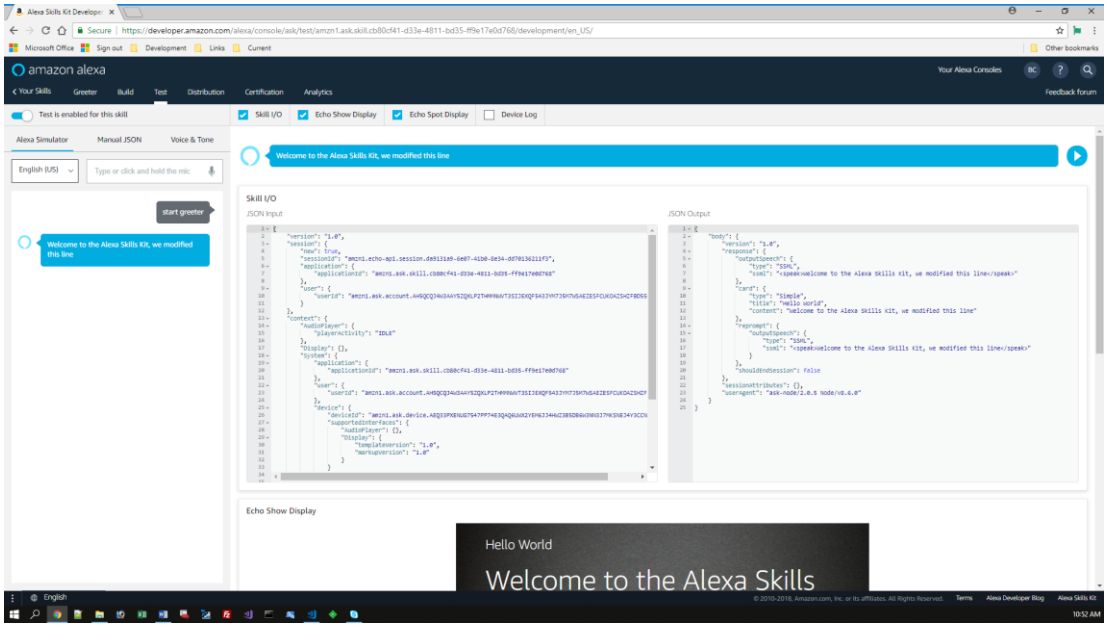


In the Alexa Skill Console, in the speech input box, enter ‘start greeter’ as shown below



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.



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.