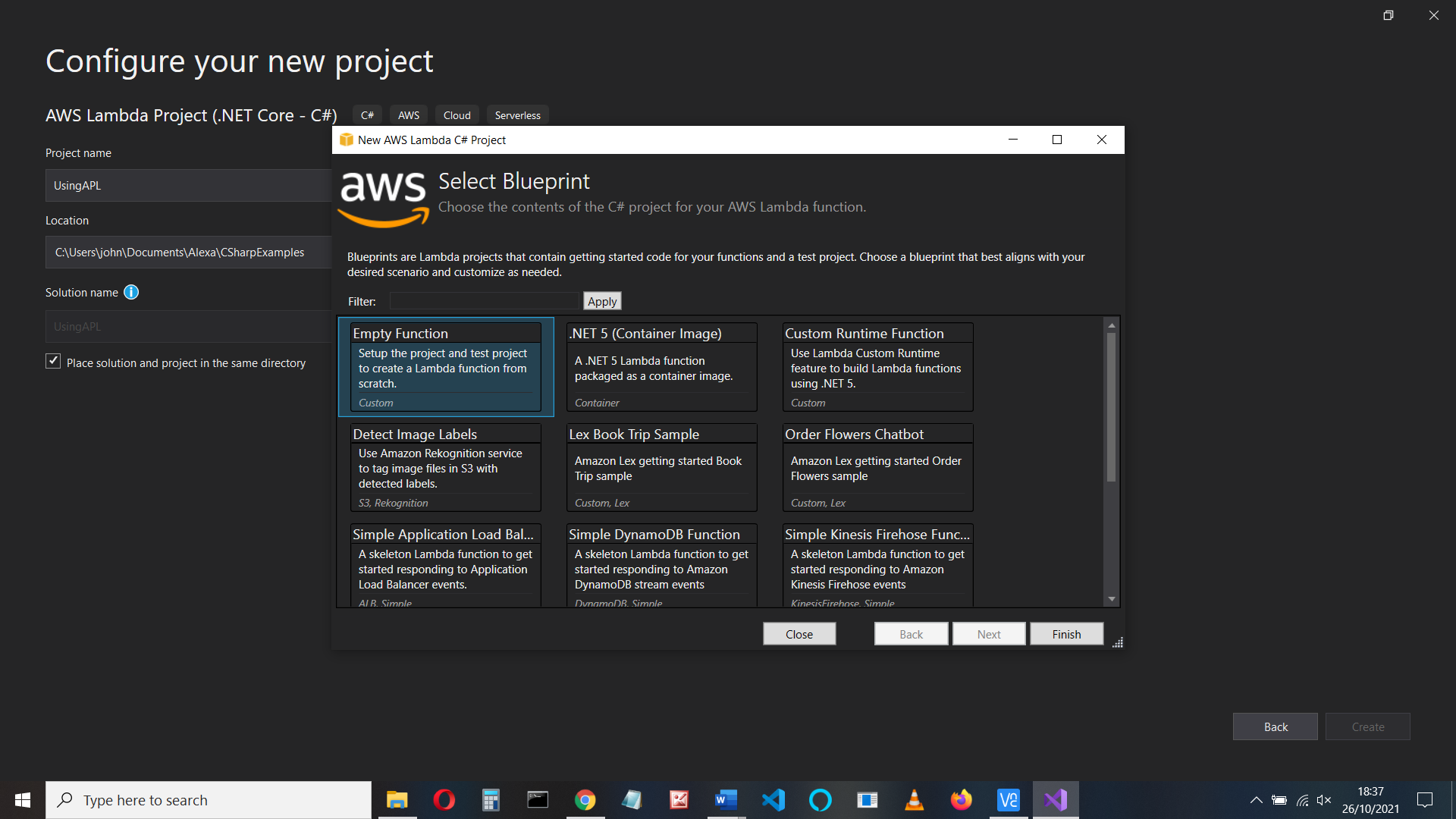
**Using APL and C# in VS code**

Part 1: <https://youtu.be/YBJXlpWhAJA>

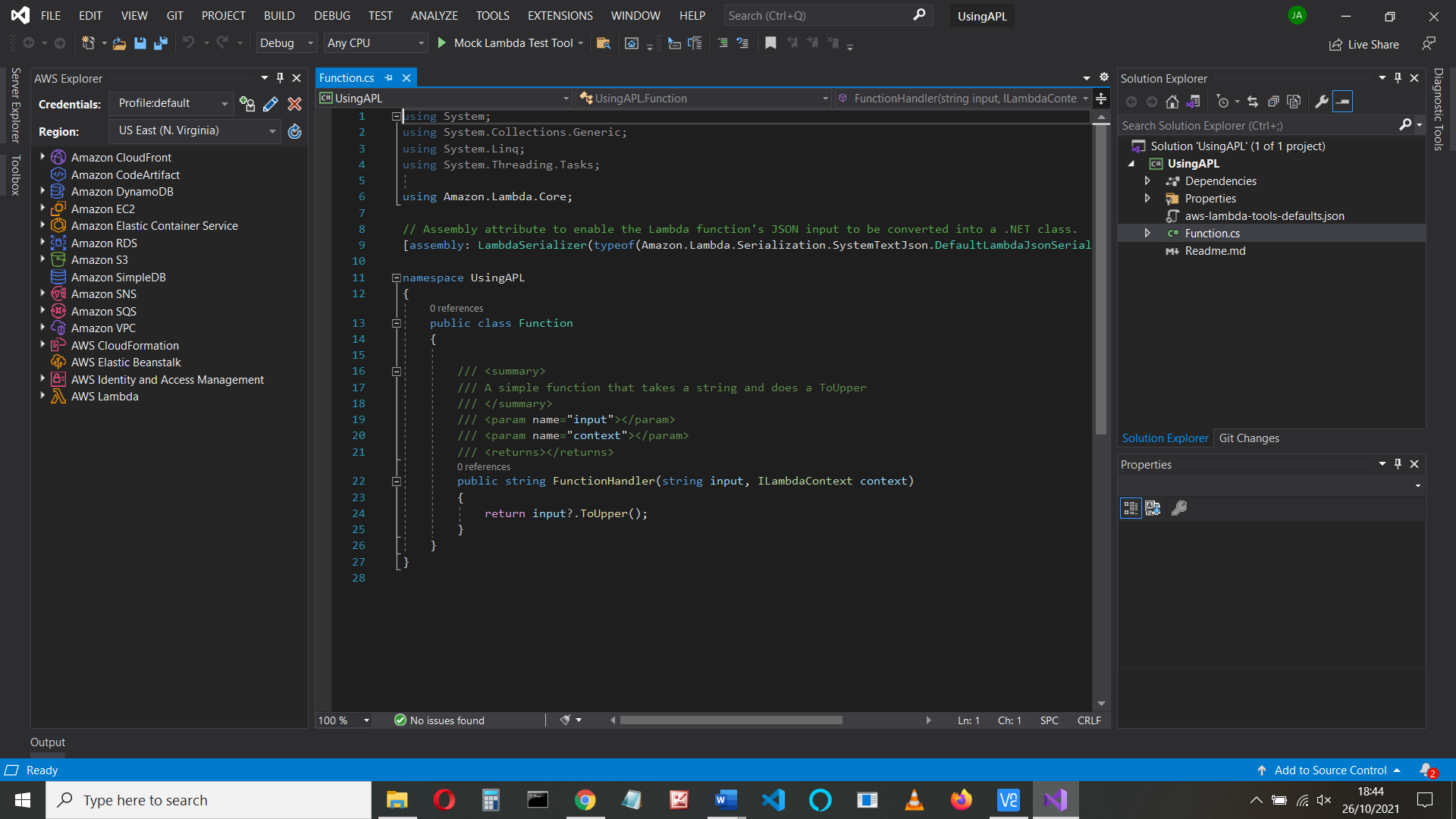
Part 2: <https://youtu.be/CPJGrVCekJE>

Open VS Code and start a new AWS Lambda Project (.NET core C#), complete the boxes (I called it UsingAPL) and choose empty function



Set up credentials if necessary. (see Alexa CSharp VS coding.docx for set up)

Select the function.cs code (this is found in the Solution Explorer) - it has a basic template:



Replace all the code you have with the following. **Change the namespace** if it’s not called AWSLambda1

using Amazon.Lambda.Core;

using Alexa.NET.Response;

using Alexa.NET.Request;

using Alexa.NET.Request.Type;

using Newtonsoft.Json;

**using Alexa.NET.APL;**

**using Alexa.NET.Response.APL;**

**using Alexa.NET.APL.Components;**

// See https://github.com/stoiveyp/Alexa.NET.APL

// Assembly attribute to enable the Lambda function's JSON input

// to be converted into a .NET class.

[assembly: LambdaSerializer(typeof(Amazon.Lambda.Serialization.Json.JsonSerializer))]

**namespace AWSLambda1**

{

public class Function

{

public SkillResponse FunctionHandler(SkillRequest input, ILambdaContext context)

{

APLSupport.Add();

RenderDocumentDirective.AddSupport();

SkillResponse response = new SkillResponse();

response.Response = new ResponseBody();

response.Response.ShouldEndSession = false;

IOutputSpeech innerResponse = null;

var log = context.Logger;

log.LogLine($"Skill Request Object:");

log.LogLine(JsonConvert.SerializeObject(input));

if (input.GetRequestType() == typeof(LaunchRequest))

{

log.LogLine($"Default LaunchRequest made");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Launch Request";

}

else if (input.GetRequestType() == typeof(IntentRequest))

{

var intentRequest = (IntentRequest)input.Request;

switch (intentRequest.Intent.Name)

{

case "AMAZON.CancelIntent":

log.LogLine($"AMAZON.CancelIntent: send StopMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Cancelled";

response.Response.ShouldEndSession = true;

break;

case "AMAZON.StopIntent":

log.LogLine($"AMAZON.StopIntent: send StopMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Stopping";

response.Response.ShouldEndSession = true;

break;

case "AMAZON.HelpIntent":

log.LogLine($"AMAZON.HelpIntent: send HelpMessage");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Help me!";

break;

case "HelloIntent":

log.LogLine($"Say Hello");

var directive = new RenderDocumentDirective

{

Token = "randomToken",

Document = new APLDocument // in Alexa.NET.Response.APL

{

MainTemplate = new Layout(new[]

{

new Container(new APLComponent[]{

new Text("APL in C#"){FontSize = "24dp",TextAlign= "Center"},

new Image("https://johnallworksbucket.s3.amazonaws.com/perkin.jpg"){Width = 600,Height=480}

})

})

}

};

response.Response.Directives.Add(directive);

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Hello Intent";

break;

default:

log.LogLine($"Unknown intent: " + intentRequest.Intent.Name);

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Unknown Intent";

break;

}

}

response.Response.OutputSpeech = innerResponse;

response.Version = "1.0";

log.LogLine($"Skill Response Object...");

log.LogLine(JsonConvert.SerializeObject(response));

return response;

}

}

}

Did you change the namespace?

The code creates an APL document (in the HelloIntent code) and adds the code to the response using a RenderDocumentDirective with a token and APLDocument that has a template with a layout containing a Container with a Text and Image components

Document = new APLDocument // in Alexa.NET.Response.APL

{

MainTemplate = new Layout(new[]

{

new Container(new APLComponent[]{

new Text("APL in C#"){FontSize = "24dp",TextAlign= "Center"},

new Image(".jpg"){Width = 600,Height=480}

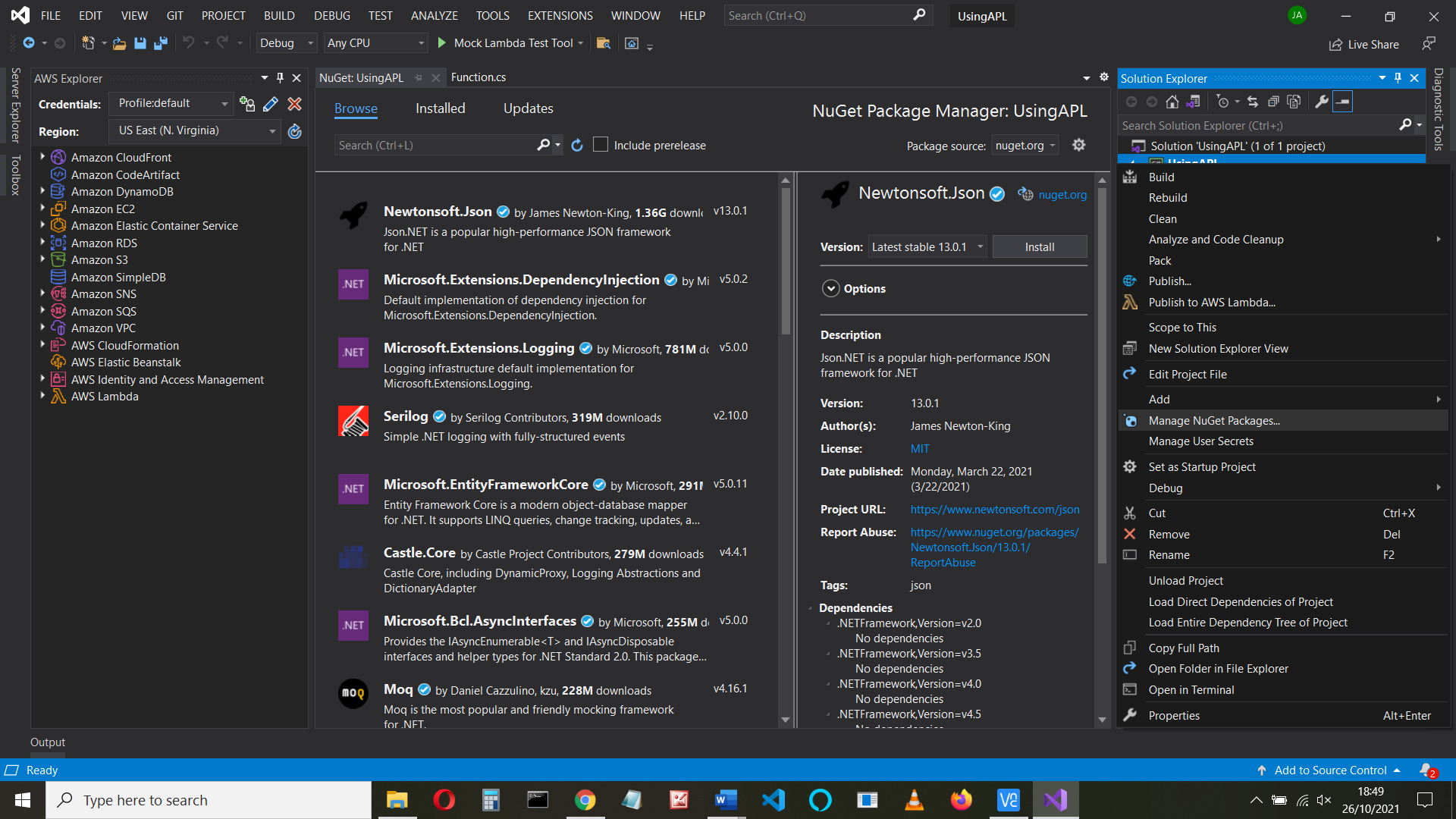
For more information on the RenderDocumentDIrective see

<https://github.com/stoiveyp/Alexa.NET.APL#sending-a-renderdocument-directive>

We need to add the libraries to our project (they are added in the code with the using clause)

**Add the libraries**

Click the Project name (“UsingAPL”), right-click packages and select Manage Nuget packages



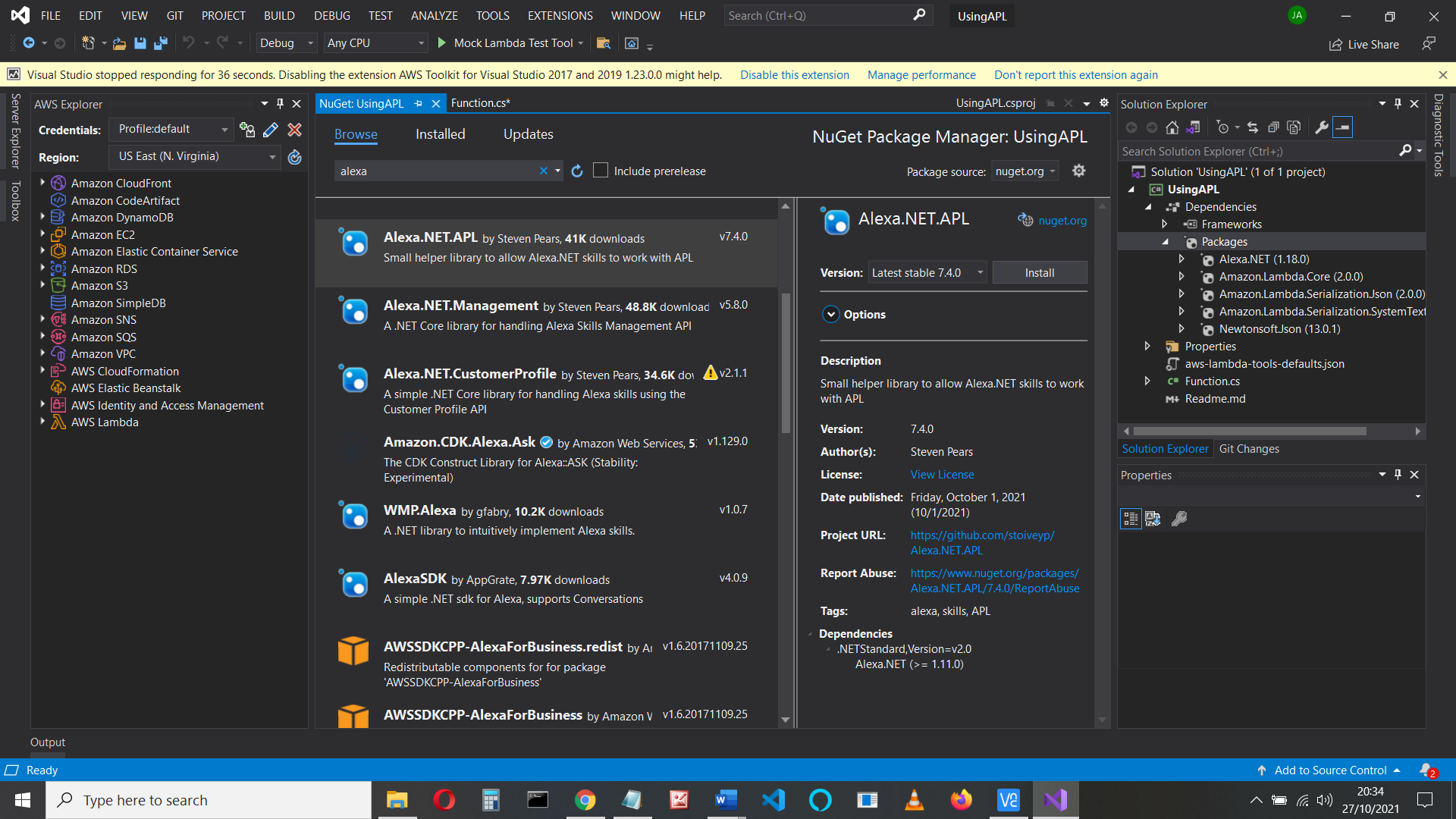
Click Browse, type json.net into the search box and install the **NewtonSoft.Json**

package.

And repeat for **Alexa.NET** by Tim Heuer

And for **Amazon.Lambda.Serialization.Json**

And for **Alexa.NET.APL**



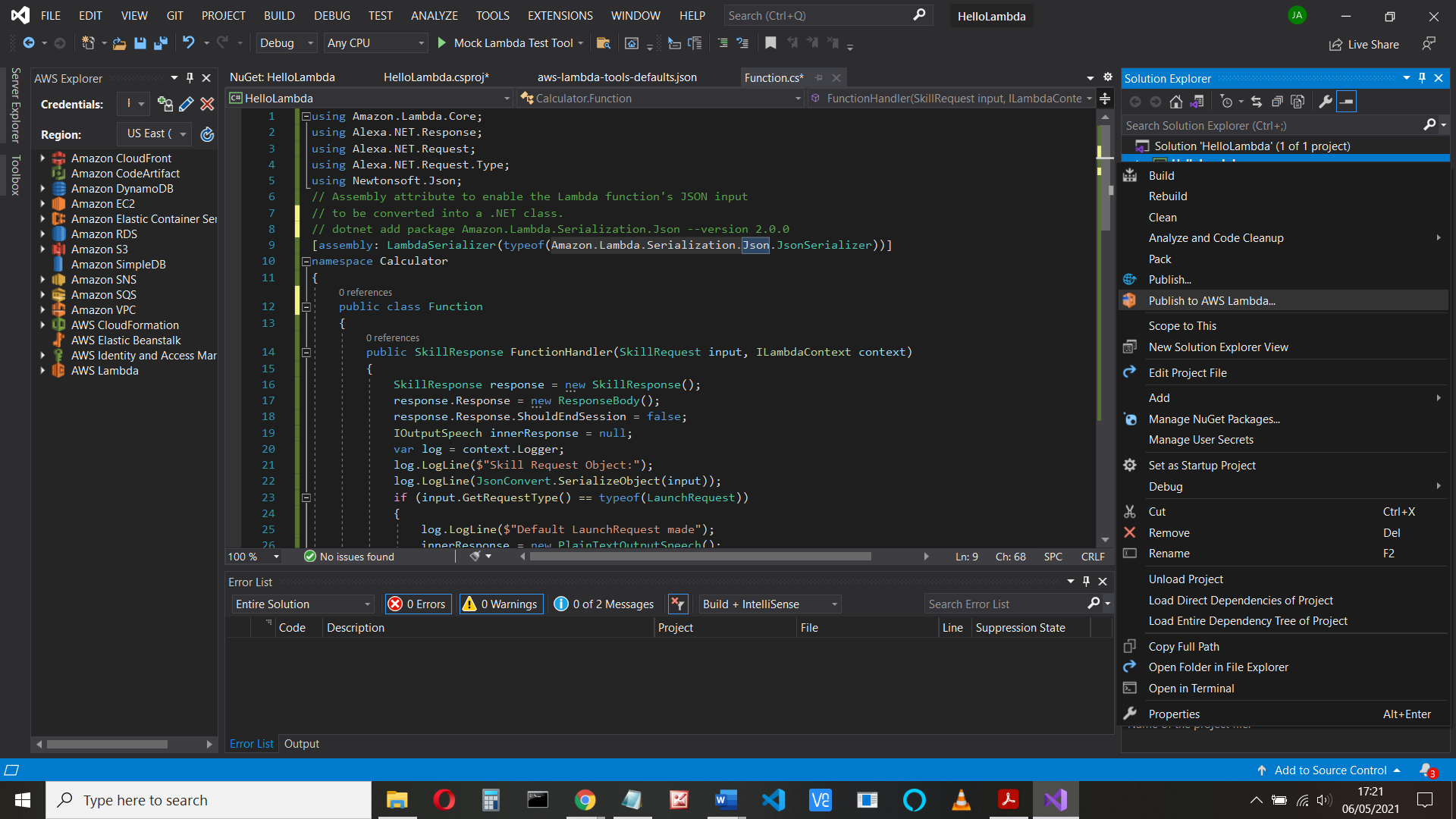
**SAVE ALL**

**Upload**

Upload the code to Amazon’s Lambda service. It won’t work yet.

.

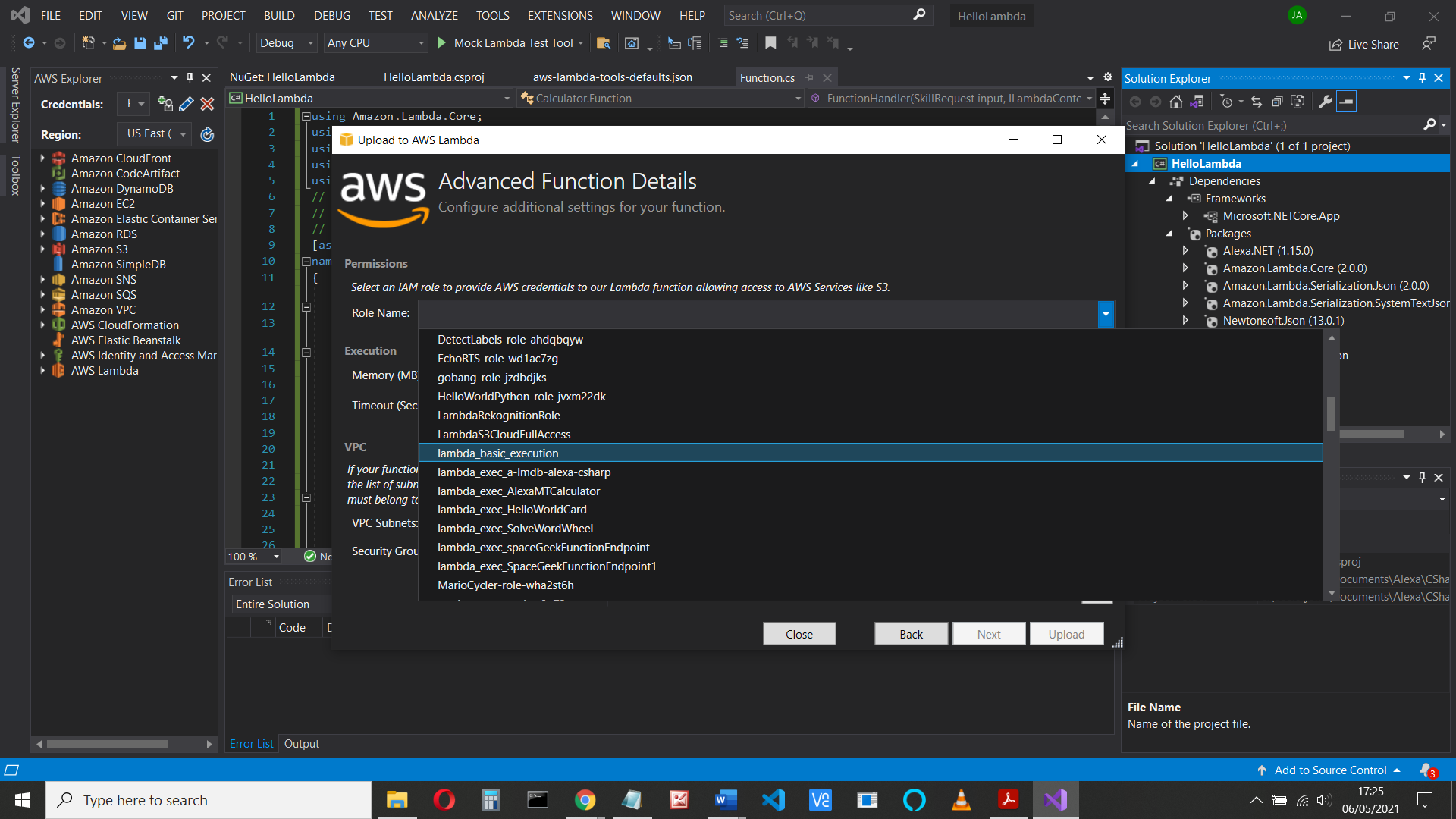
Right-click on your project and click Publish to AWS Lambda



Everything is completed except for the name, so type in whatever name you used/want **but it’s all one word**

In the next screen, for the Role Name select lambda\_basic\_execution from the drop down

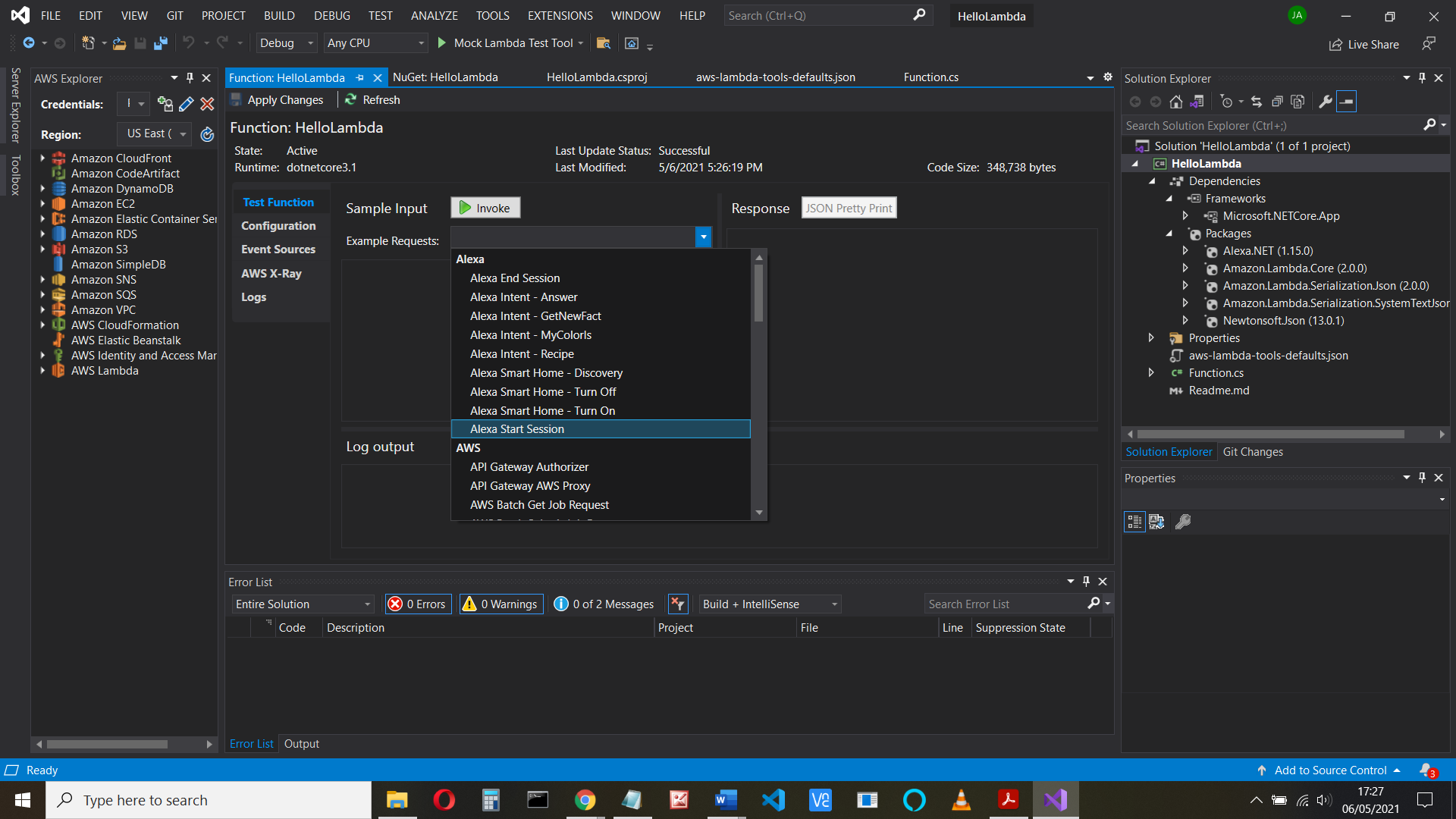
box, and then **Upload**.



Correct any errors you may have (did you save the file? Was the name all one word? Do you already have one called that?)

**Test**

You can test that it will run before we proceed. Choose Alexa Start Session from the Example Requests drop down box,



and click Invoke:



If you get the error

{

"errorType": "LambdaException",

"errorMessage": "Unable to load type 'HelloLambda.Function' from assembly 'HelloLambda'."

}

Check namespace and solution names are the same

**Skill Information and Interaction Model**

Now let’s see if our function is on the AWS site.

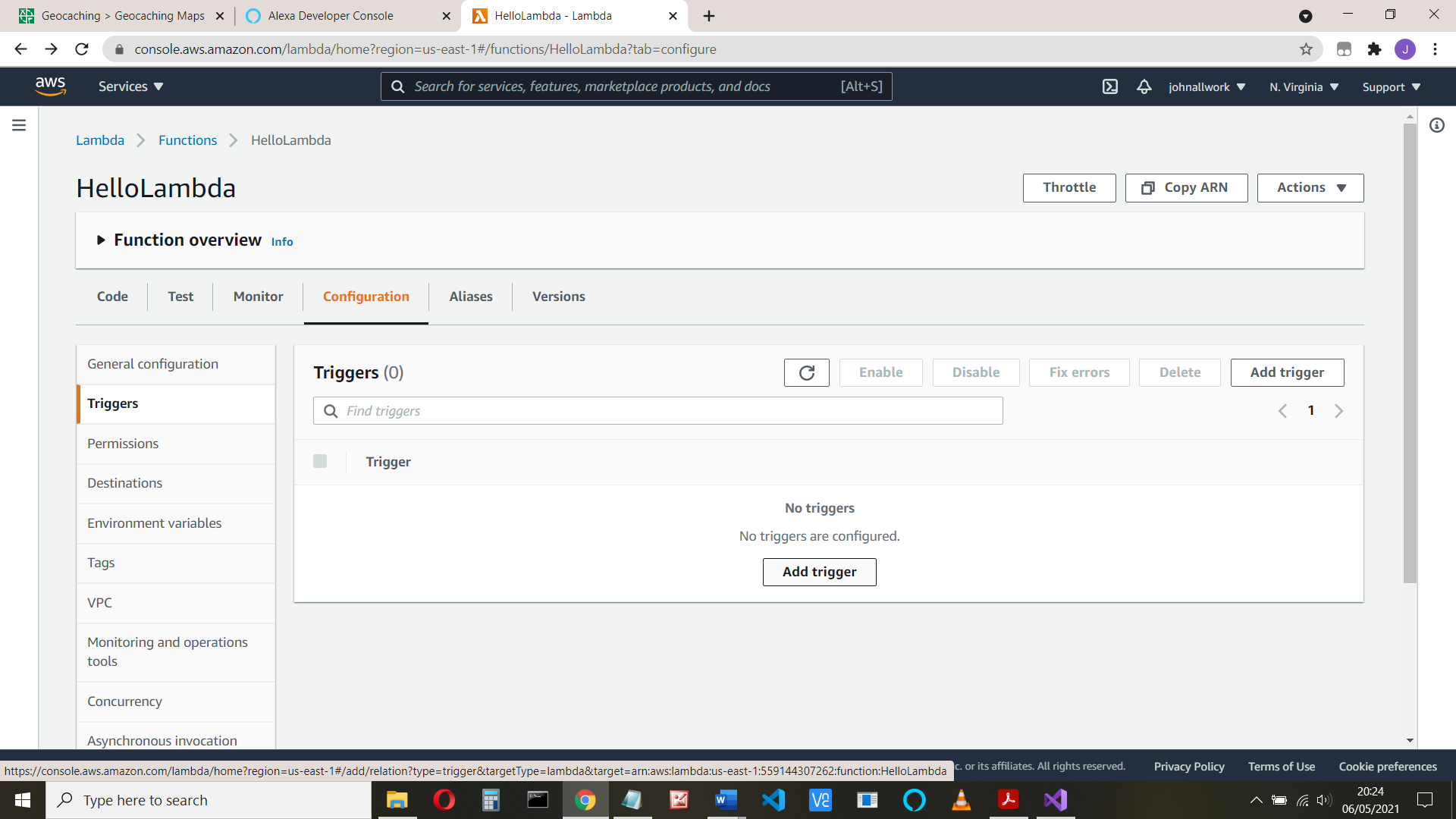
Note it’s not in the developer console (yet)

Link the lambda to our skill

Go to AWS console (https://console.aws.amazon.com/ ) and find Lambda > Functions

Find your skill and make a copy of the ARN

We need to add a trigger to the lambda function. Go to Configuration > Triggers



Do this before trying to link from your (as yet uncreated) skill.

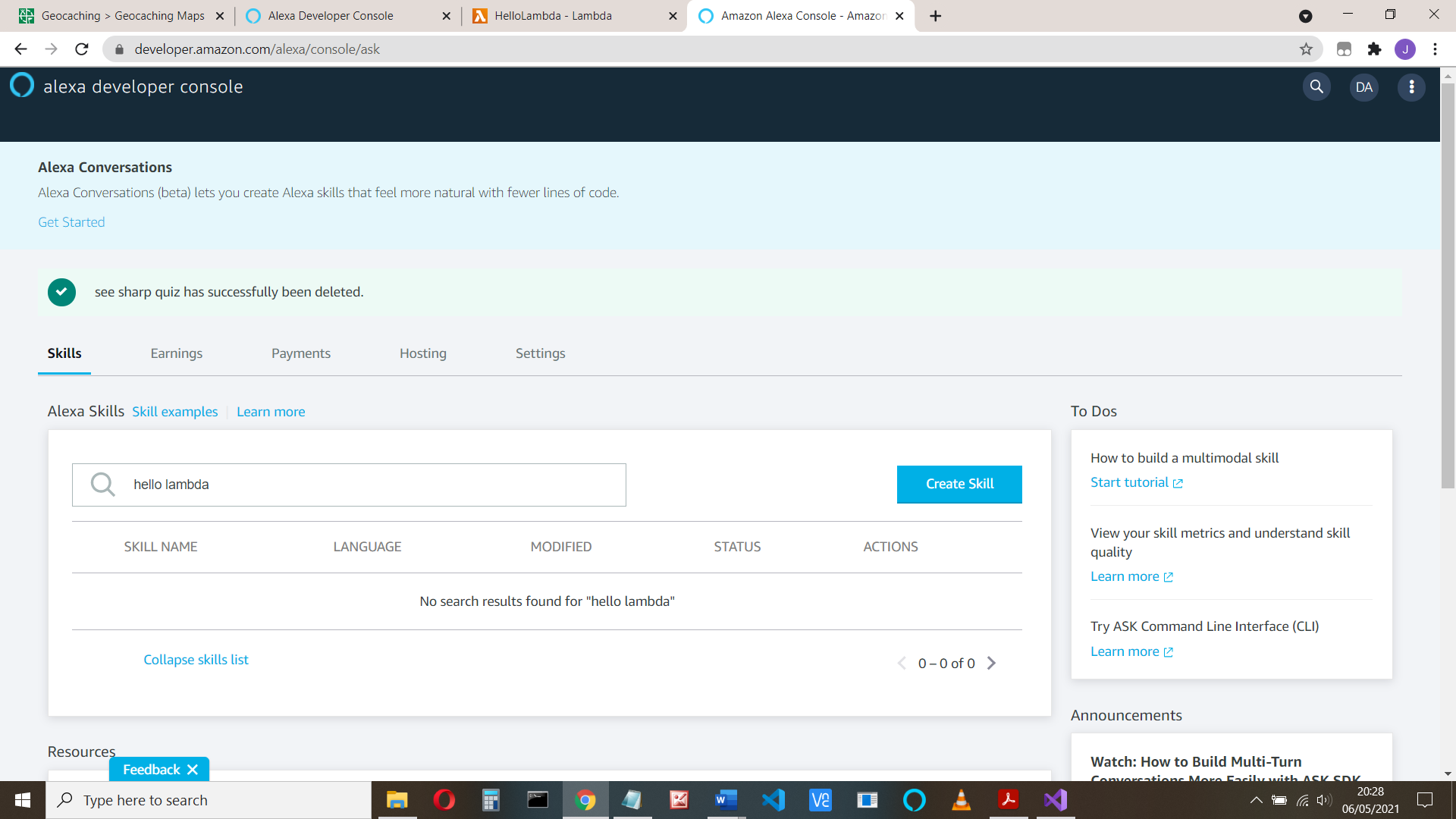
Choose Alexa Skills kit. But before we can complete this, we need the skill ID that comes from the Developer site. That in turn needs the Amazon Resource Number (ARN) – it’s in the top right. If you didn’t do so, make a copy of that now. It looks something like this:

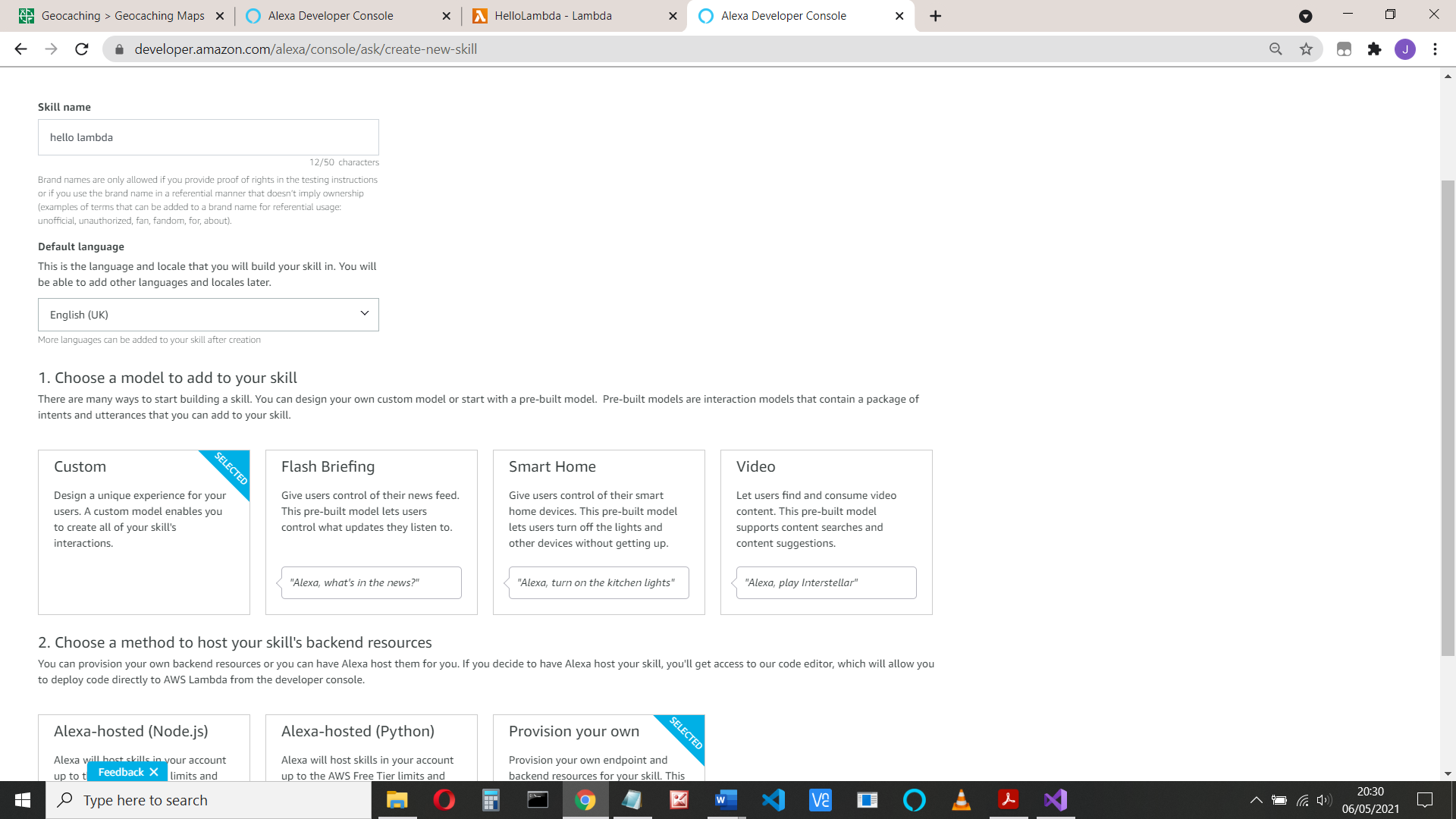
arn:aws:lambda:us-east-1:559144301234:function:usingAPL

We’ll return here in a minute

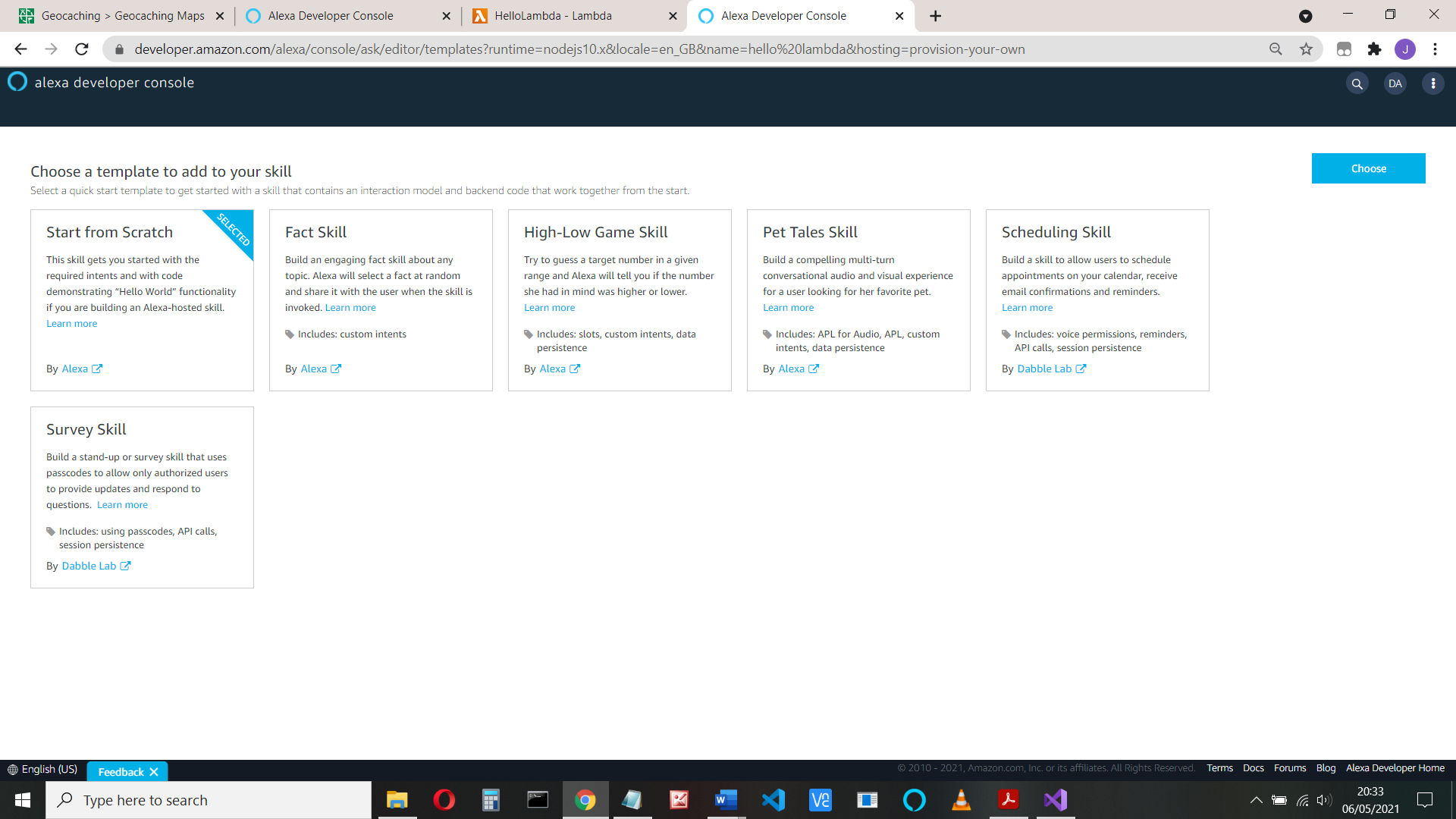
Open the developer dashboard (developer.amazon.com)

In the developer console, choose Alexa Skills Kit and Create skill.



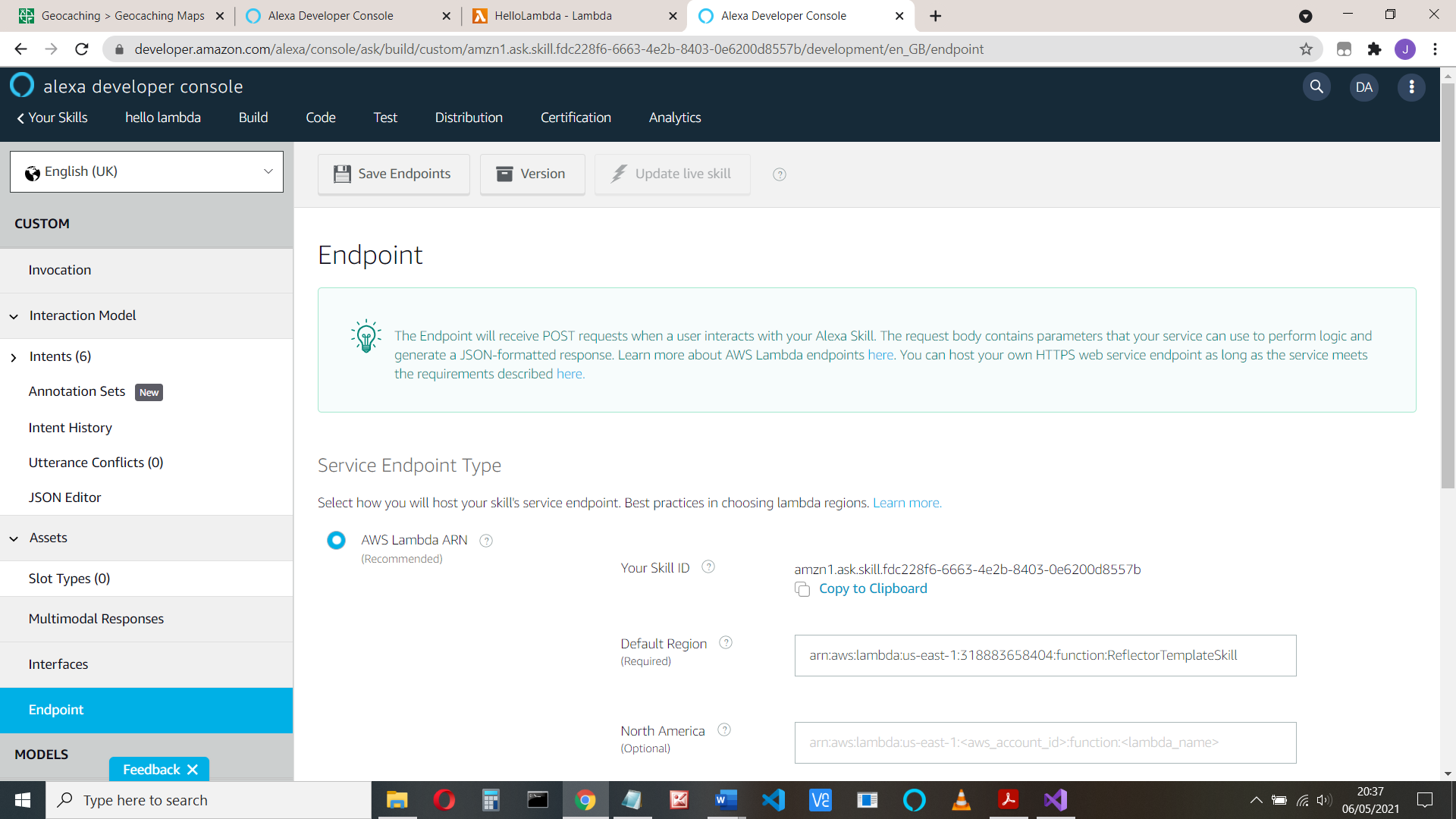


I’ve called it **see apple** (clever play on words there). Choose Start from Scratch for the template

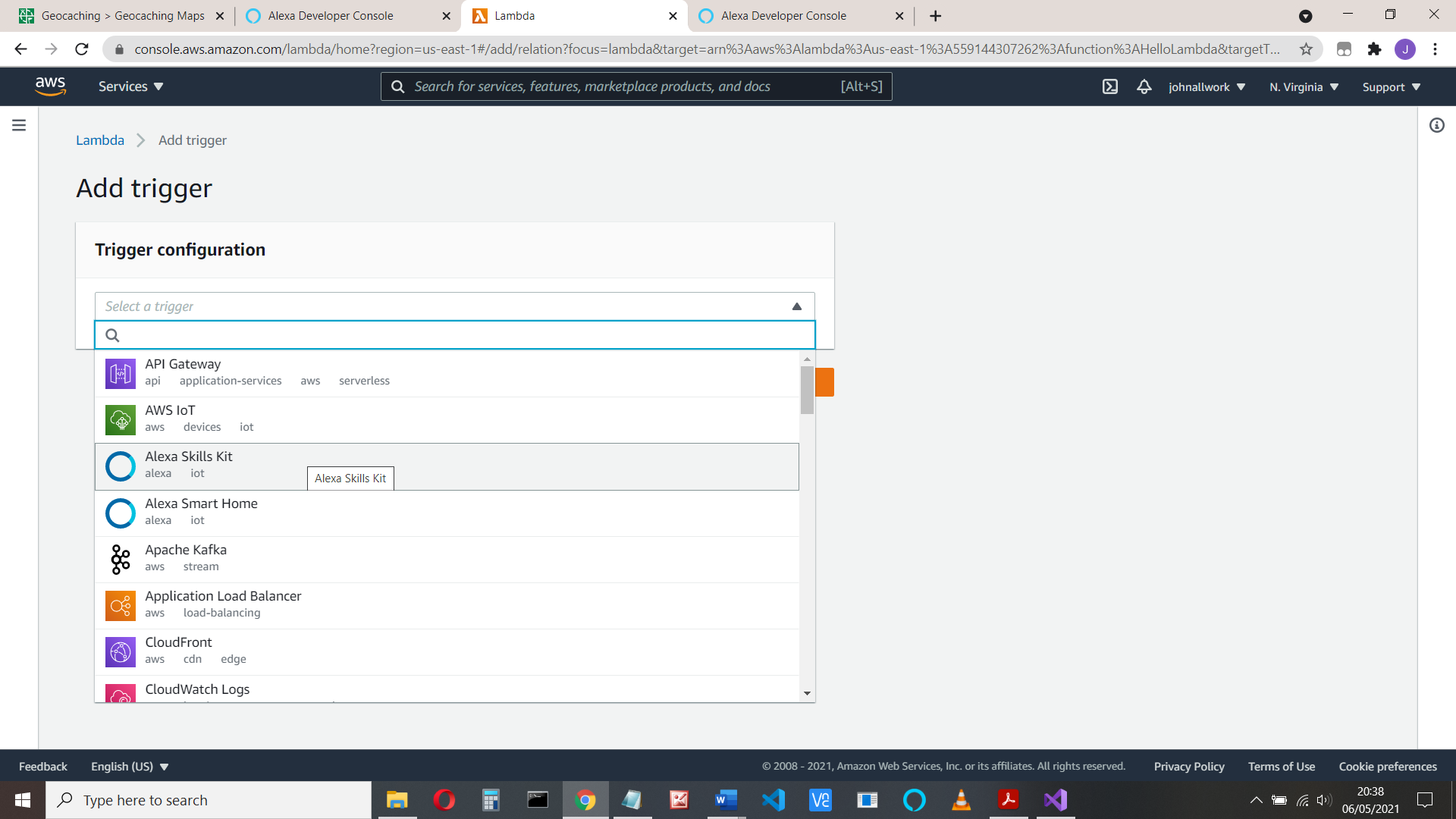


Now select Endpoint and copy the skill ID It begins with amzn1.ask.skill

amzn1.ask.skill.396ed377-1877-4e60-8ca2-e68606fdb003

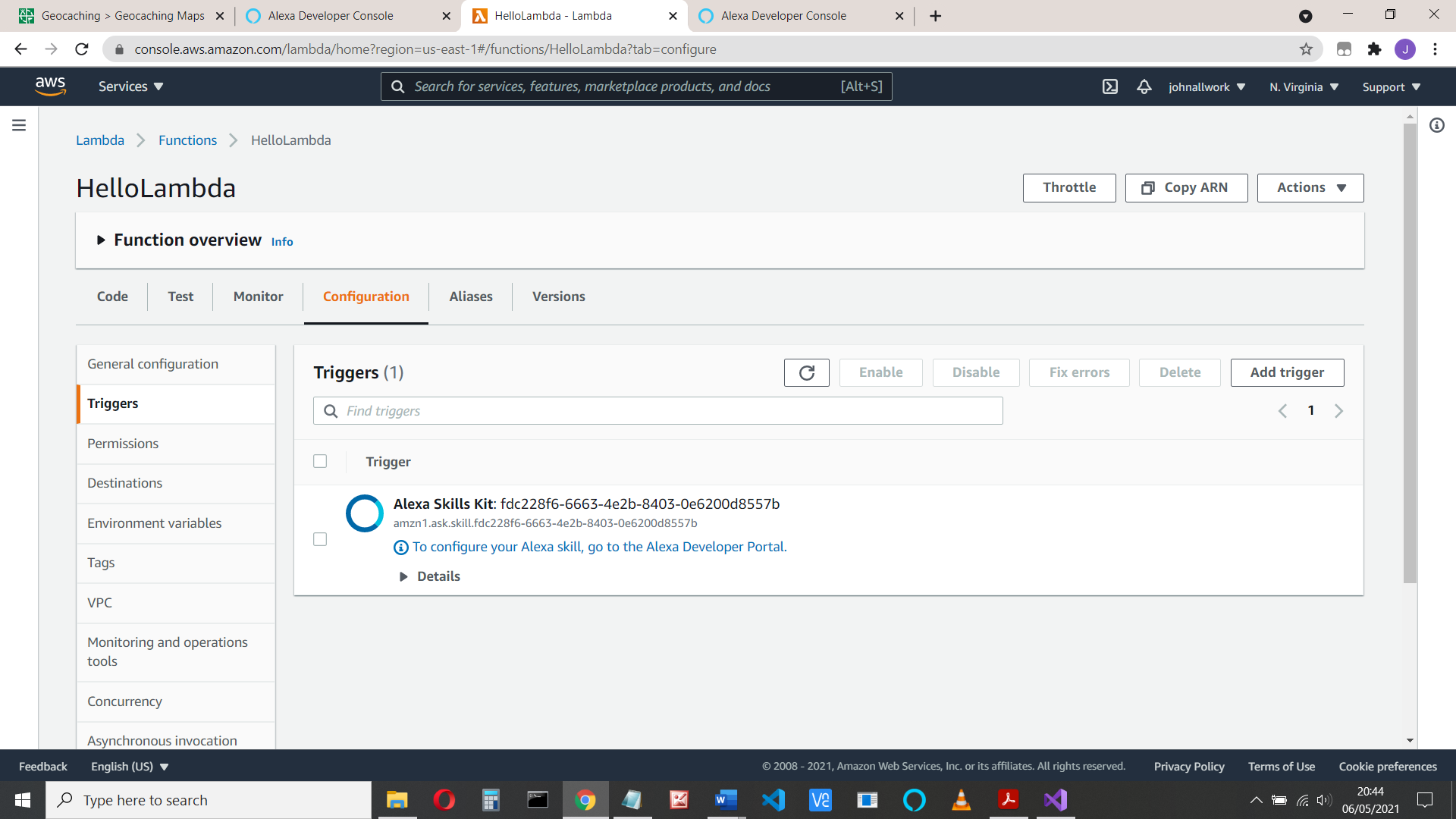


Return to the AWS screen, add the trigger, select Alexa Skill Kit and paste your skill ID



The reason for all that is so when a user uses your skill (says “open see apple”) the skill will trigger your C# lambda code, and it will respond.

(You could have other triggers, such as run some code when a photo is uploaded)

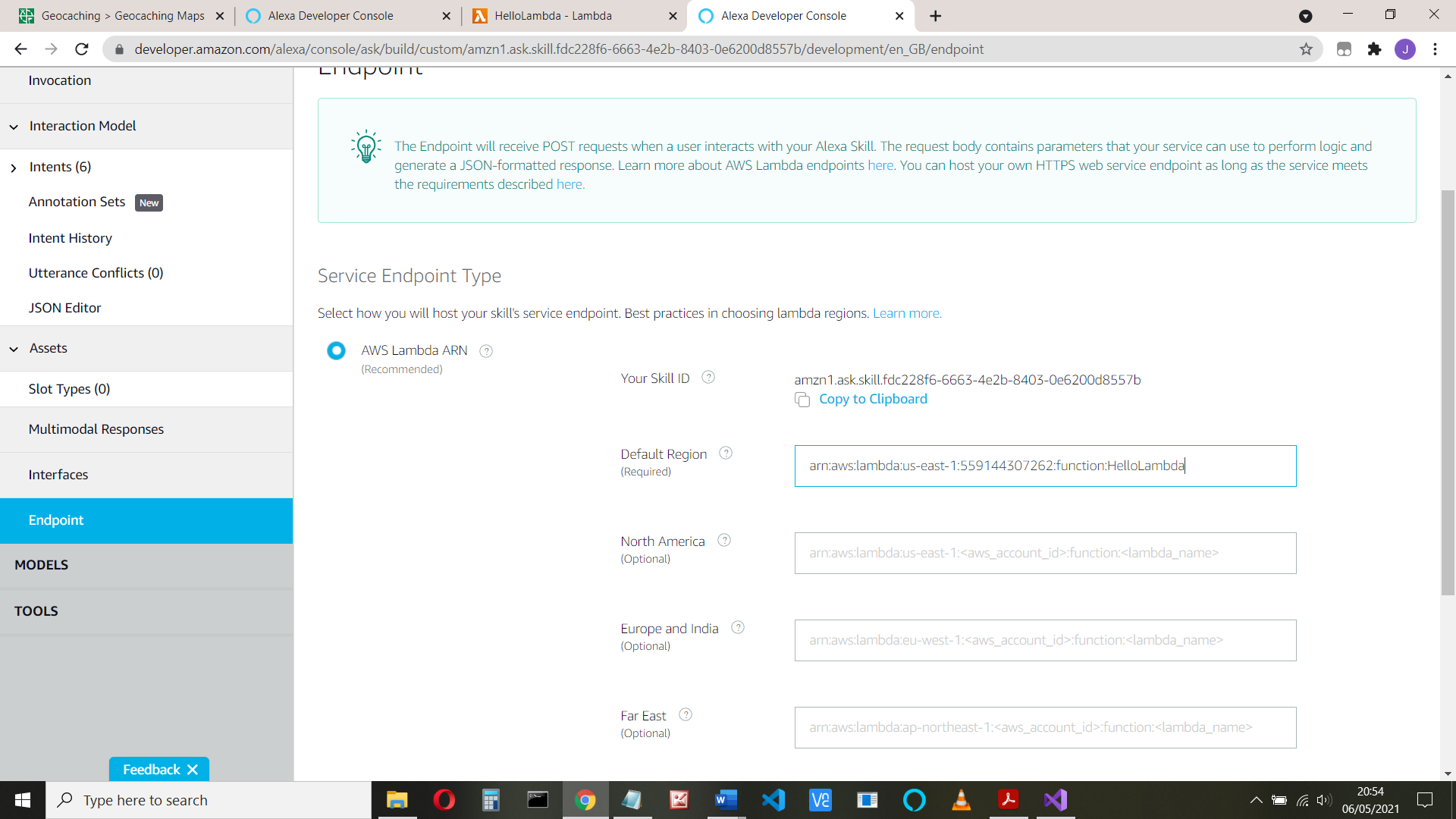


Go to the Developer Portal (developer.amazon.com) to configure the Utterances and intents

**Configure endpoint**

Your lambda code now knows about your Alexa skill, but not the other way, so add the ARN you copied (remember that? ) to the Default region in your endpoints:

(It’s something like arn:aws:lambda:us-east-1:55914121234:function:UsingAPL)



And save the endpoints, ignore the warning

**Add your intent**

Select Intents from the left-hand side.

Our code has the AMAZON intents: Cancel, Stop and Help and our Hellointent:

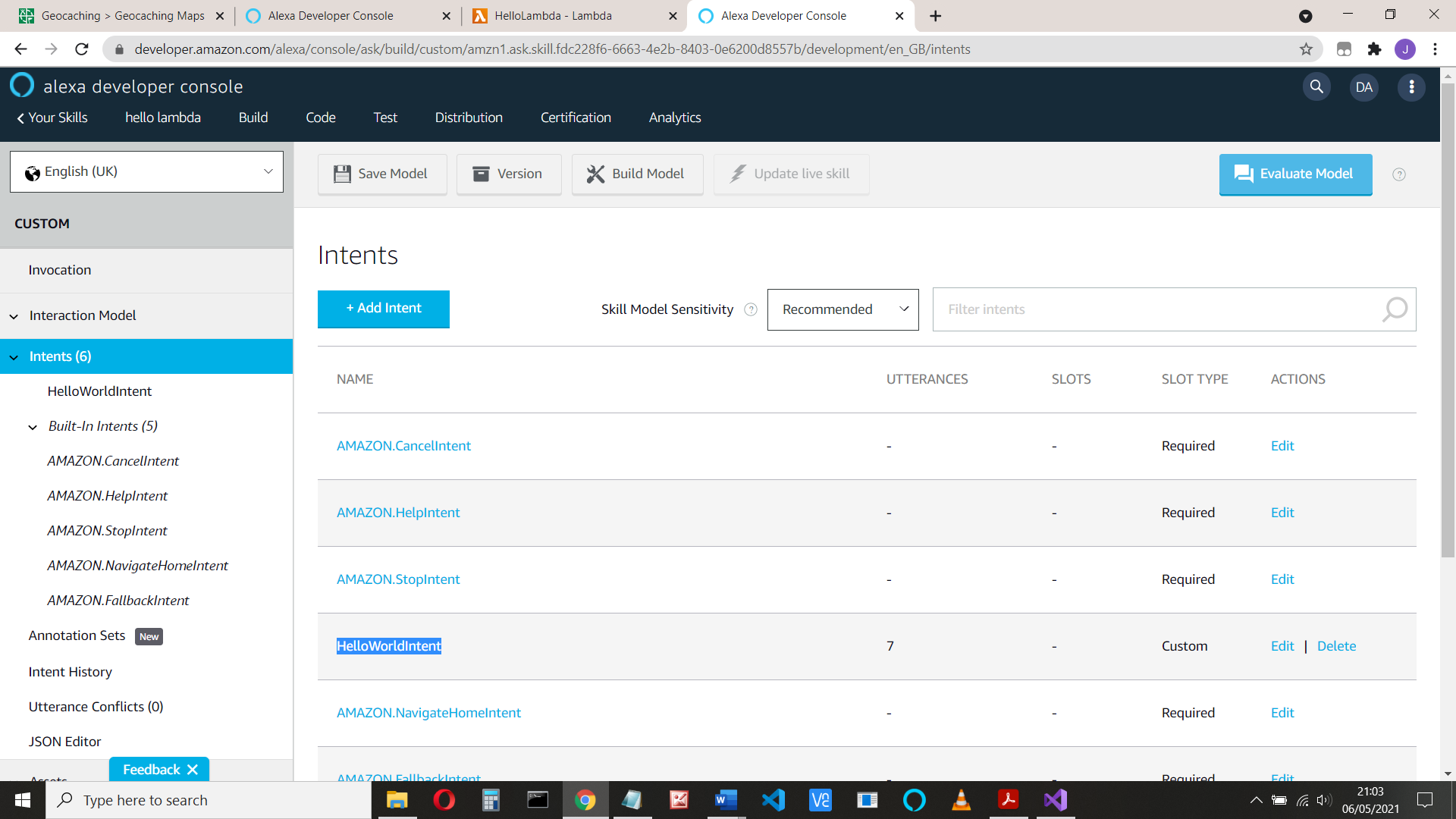
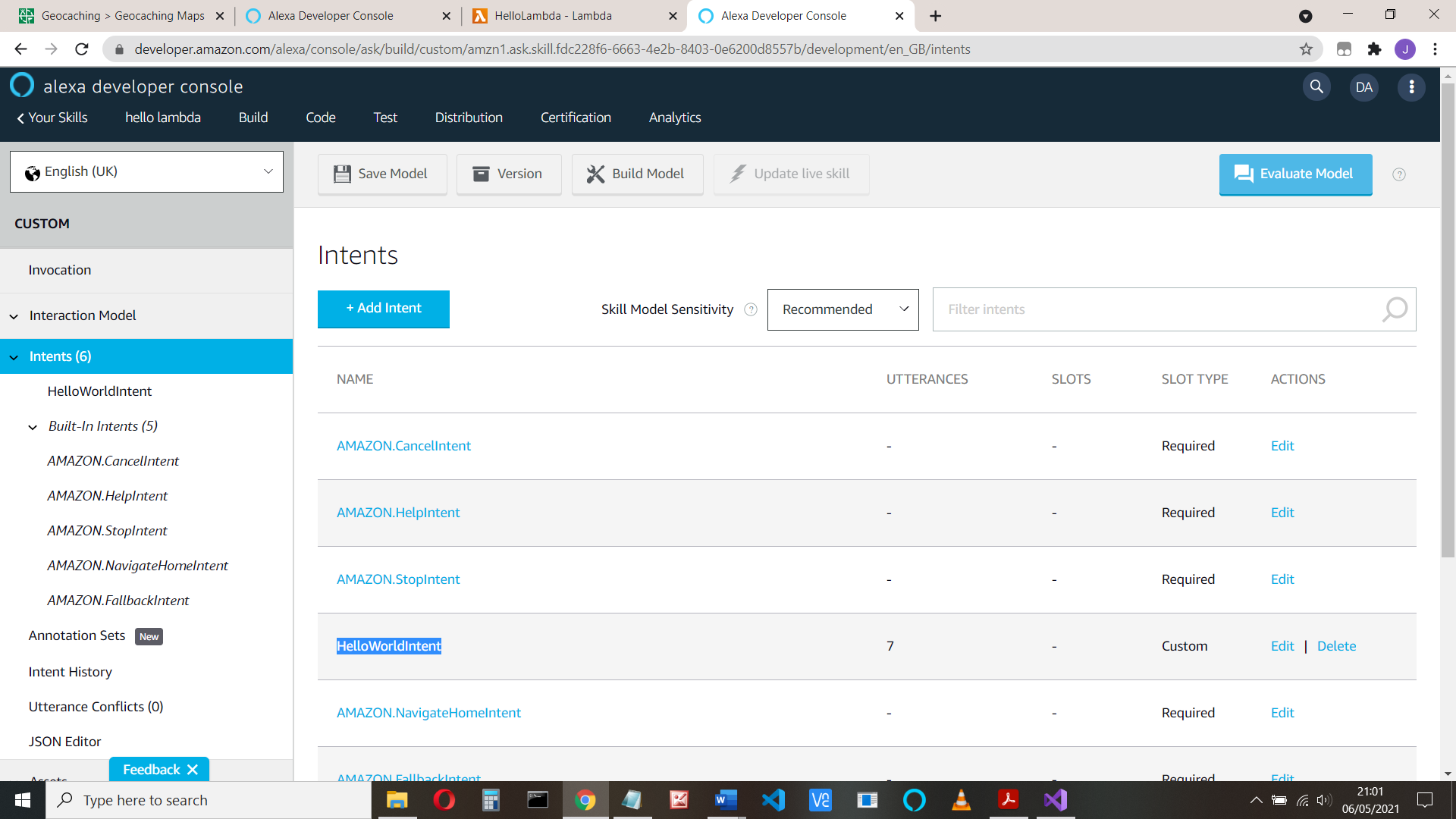
case "HelloIntent":

log.LogLine($"Say Hello");

innerResponse = new PlainTextOutputSpeech();

(innerResponse as PlainTextOutputSpeech).Text = "Hello from Johns Calculator";

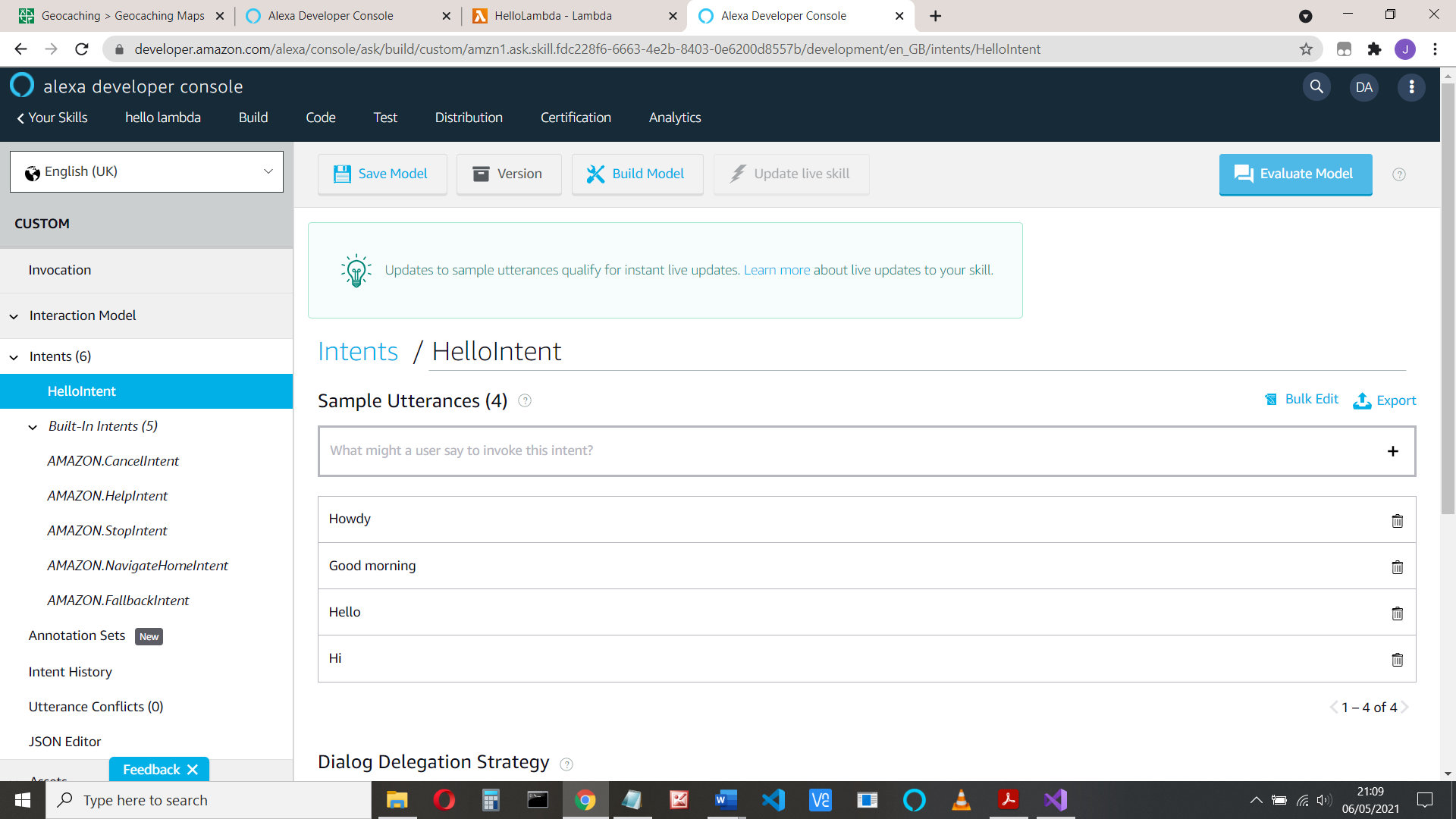
There’s already a **HelloWorldIntent**. Delete that.



Now click Add Intent and add an intent called HelloIntent (this must match your code)

Click Create Custom Intent and add some utterances:

Hi, Hello, good morning, etc:



And save and build the model (at the top of the page)

**Now let’s add the APL**

Open the skill in the developer console if you haven’t done so, select Interfaces, and enable APL



Click ‘Save Interfaces’ and ‘Build model’ to be on the safe side.

If you want to create your own APL code, see the Addendum below or my videos:

Simple Alexa APL text to speech using GUI design and transformers:

<https://youtu.be/x6425OyqD8E>

or

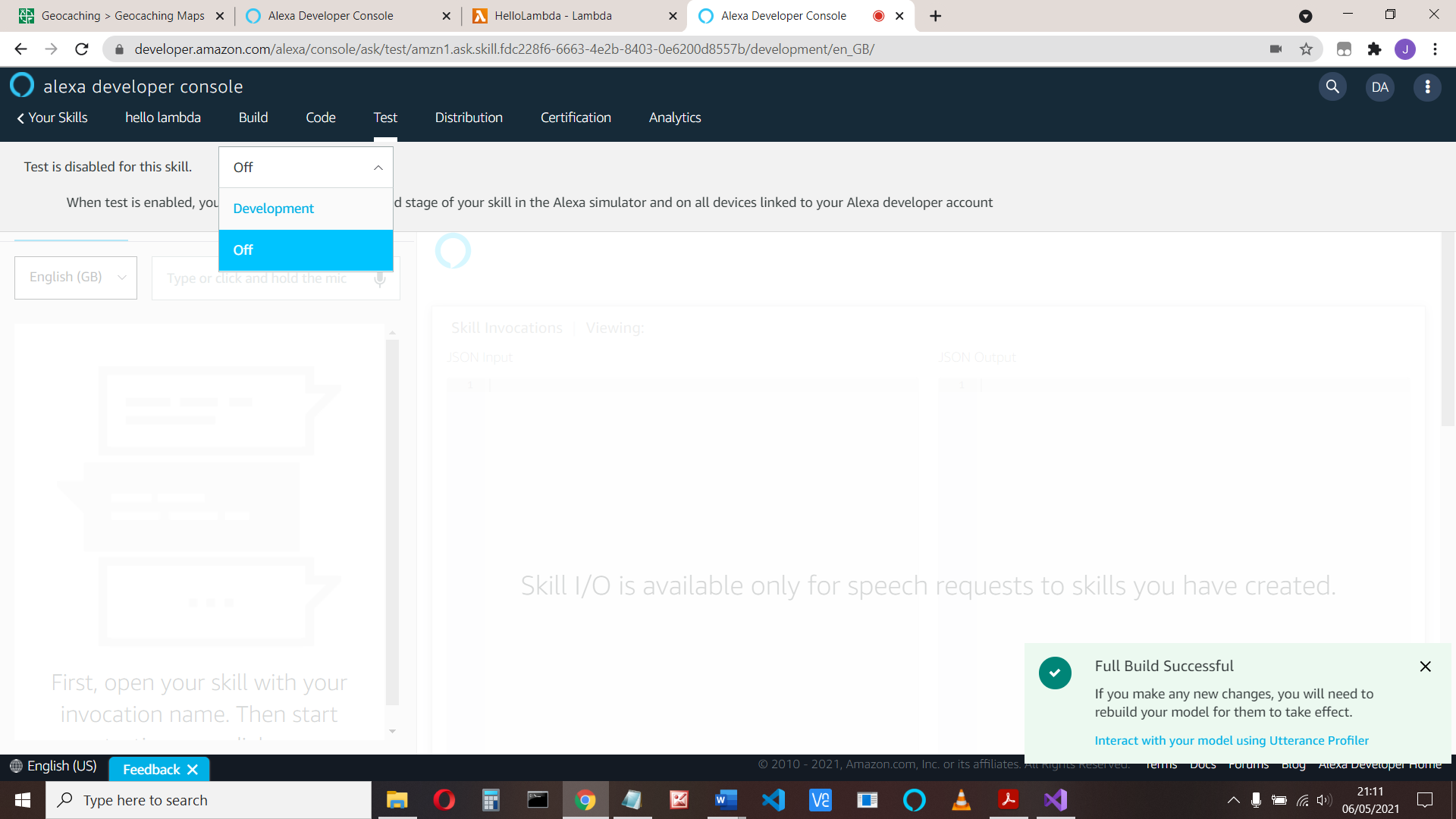
Alexa skills Python APL example

<https://youtu.be/84d8c8_LJM0>

**Let’s test it**

When the build id done, click the Test tab at the top and

Enable the skill for Development testing:



Type in or say “open see apple” (or open whatever your name is)

You should get a spoken response: ‘Launch request”

And if you type “hello”, the response: “Hello intent”

If that works, well done!

Ok time to try it on your device!

**Addendum**

Creating an APL document

Choose Developer console > Multimodal responses > Visual > Create >Blank document

Copy the following code into the APL box (see picture below if you don’t know where that is):

{

"type": "APL",

"version": "1.6",

"license": "Copyright 2021",

"settings": {},

"theme": "dark",

"import": [],

"resources": [],

"styles": {},

"onMount": [],

"graphics": {},

"commands": {},

"layouts": {},

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Container",

"items": [

{

"source": "${payload.data.properties.source}",

"type": "Image",

"id": "perkinImage",

"width": "100vw",

"height": "90vh"

},

{

"text": "${payload.data.properties.text}",

"type": "Text",

"width": "100vw",

"height": "10vh",

"id": "perkinText"

}

]

}

]

}

}

And copy this code into the Data box

{

"data": {

"properties": {

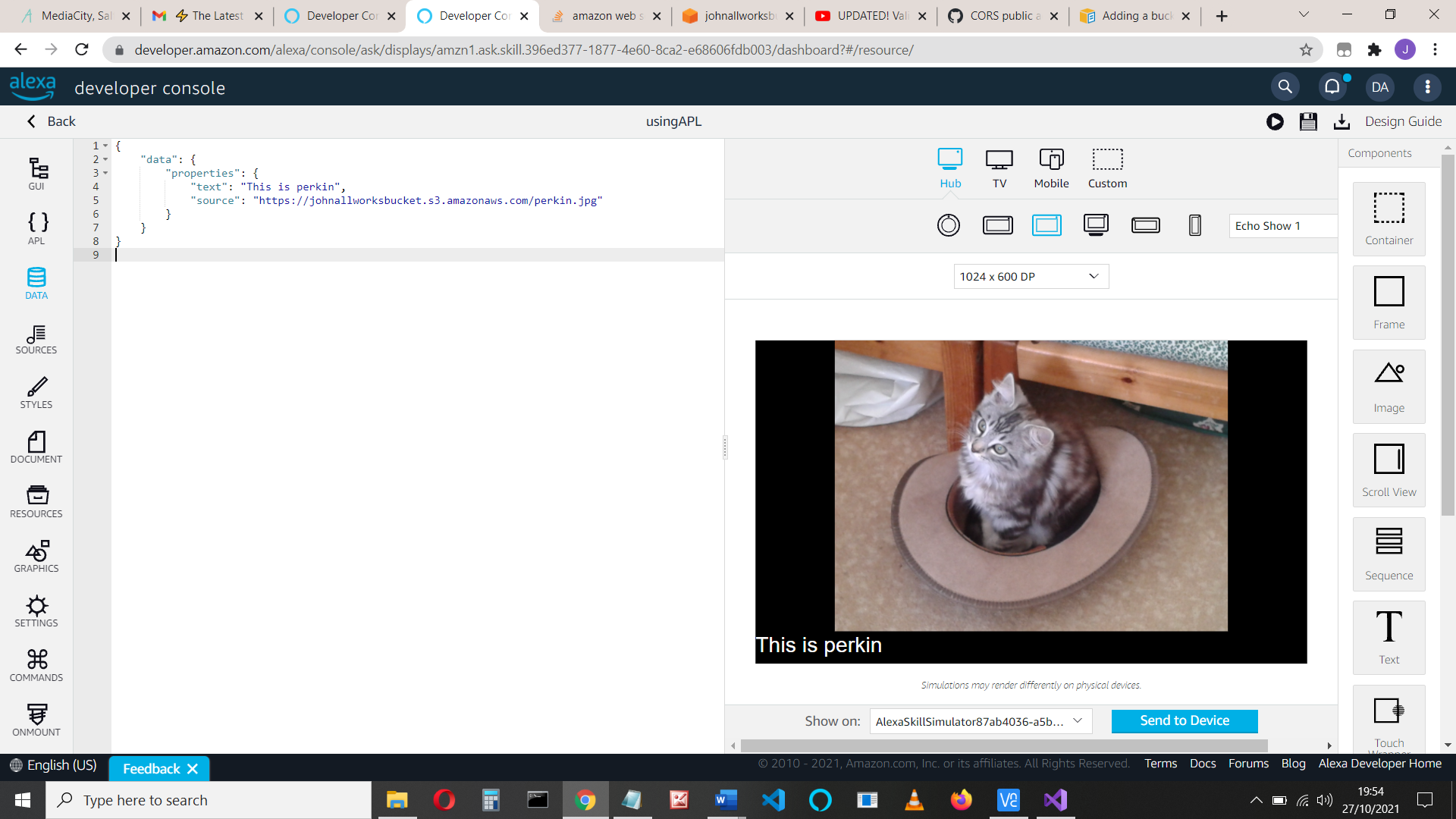
"text": "This is perkin",

"source": "https://johnallworksbucket.s3.amazonaws.com/perkin.jpg"

}

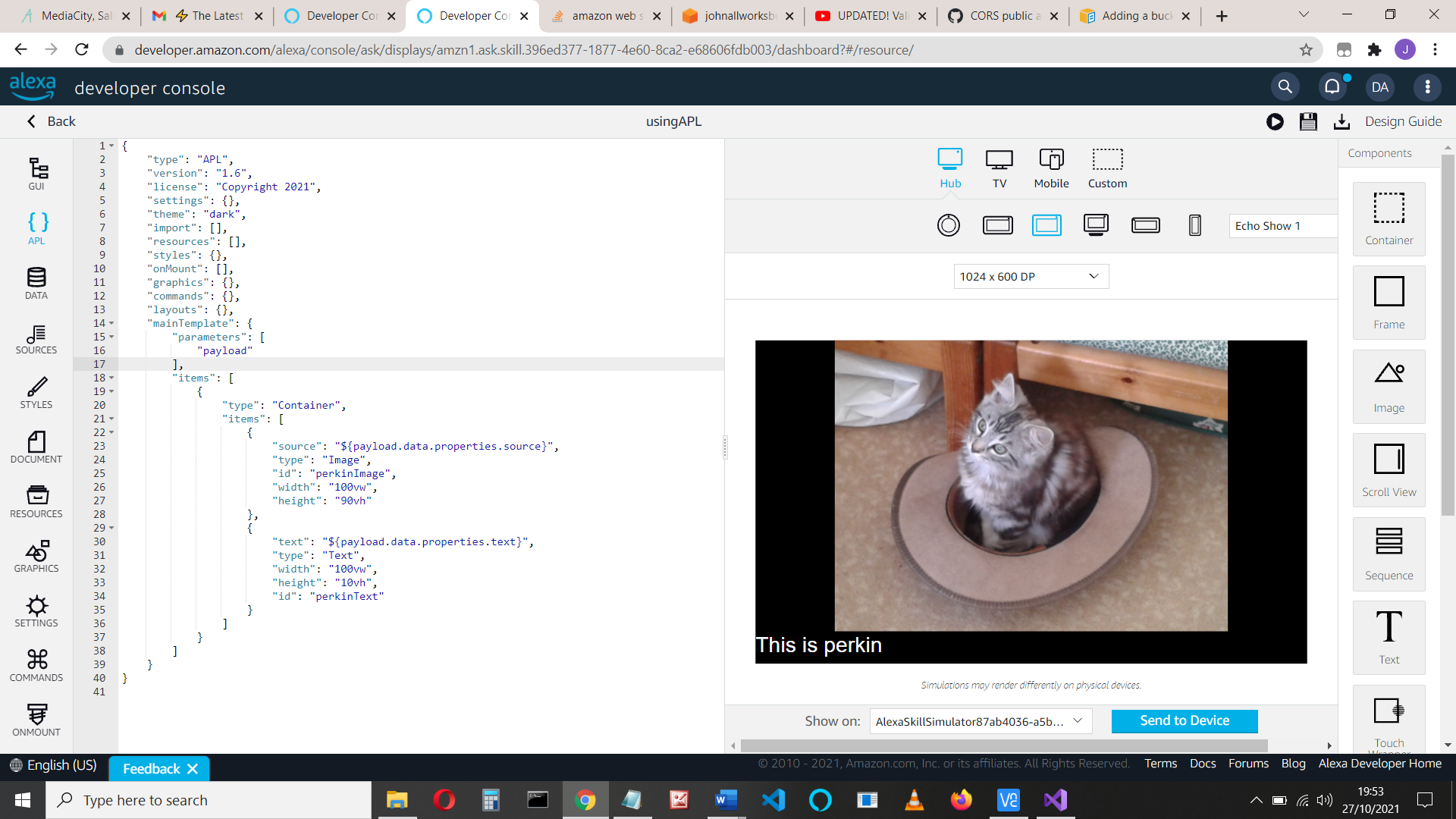
}

}



Note if you want the image above the text, use the GUI and drag the text item above the image.

Click Preview mode (That’s the Play button in the top right-hand corner) and check that it works.



if you want more information on how APL works, watch the Alexa video at:

https://www.youtube.com/watch?v=IGsvRT\_Ak\_M

For more on renderdocumentdirective, look at <https://johnallwork.weebly.com/files.html> and choose the pythonaplinstructions.docx

The python code is like this:

if get\_supported\_interfaces(handler\_input).alexa\_presentation\_apl is not None:

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask("A reprompt to keep the session open")

.add\_directive(

RenderDocumentDirective(

token="pagerToken",

document=\_load\_apl\_document("hello.json"),

datasources={}

)

)

.response

)

I’ll do something for C# soon ..