**Using PIL/Pillow in Alexa APL code (and how to display GIF files)**

YouTube video at <https://youtu.be/LY4Sa5au9q0>

Instructions on how to display a GIF file on an Alexa device. The GIF is downloaded, converted to a PNG image and saved in an S3 location, which is used to display the image.

GIF images aren’t supported by Alexa devices.

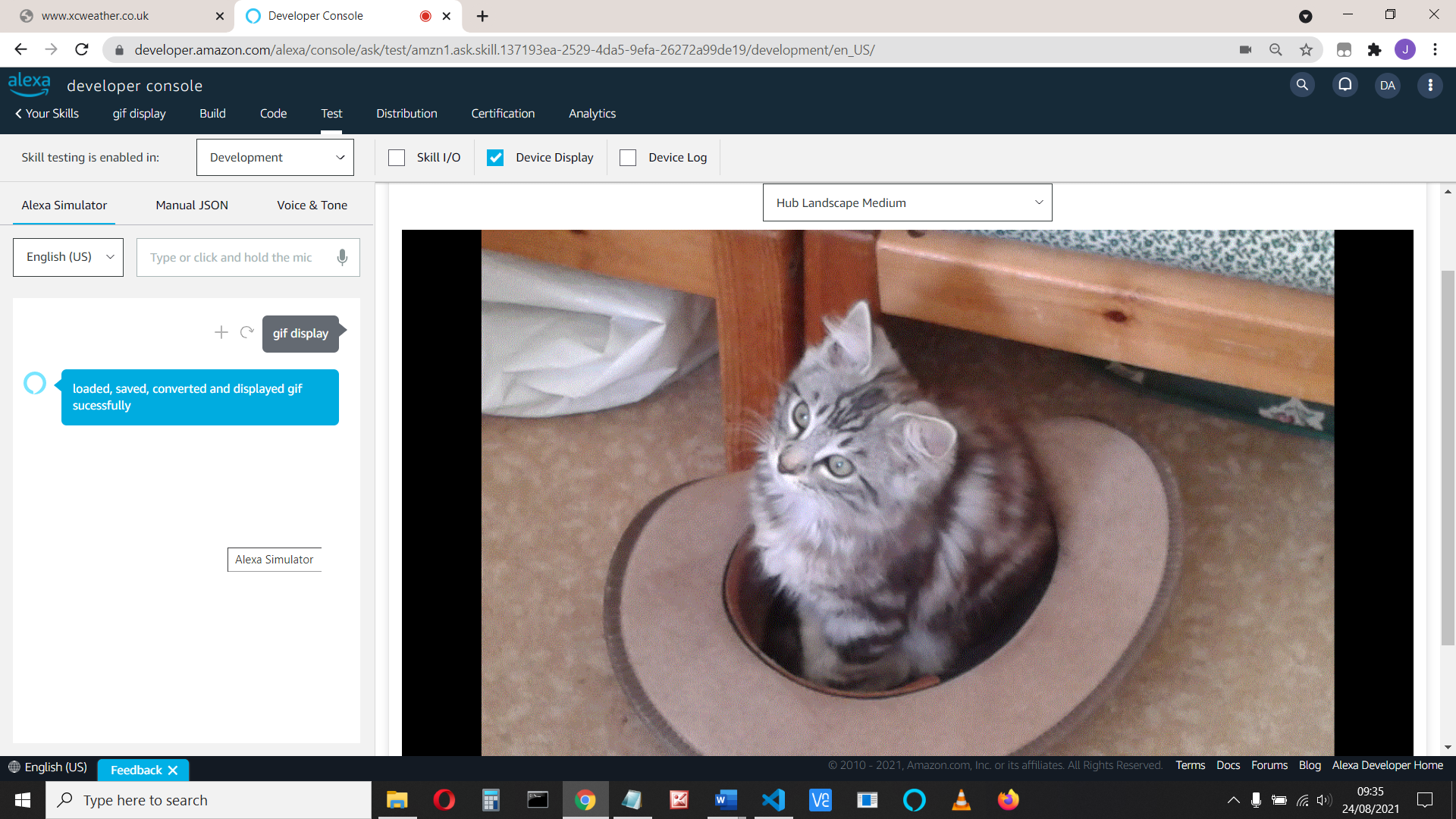
<https://developer.amazon.com/en-US/docs/alexa/alexa-design/presentation-imagery.html>

Here we will see how to convert a GIF image and display it using APL

There are two problems here:

1. To do this we will use the PIL library. If you use a lambda function, you have to download pre-compiled binaries.
2. To display the GIF file in an APL, we need to convert it to PNG or JPG and have an https location to read it from, and so need to save it in a S3 bucket (unless you have another https location)

This is what we are going to see:



1. We will use an Alexa Hosted skill to solve this problem (it loads the correct libraries), but if you want to use lambda, take a look at these references:

<https://aws.amazon.com/premiumsupport/knowledge-center/lambda-python-package-compatible/>

And: <https://www.py4u.net/discuss/173764> answer 4

2. Again an Alexa Hosted Skill helps as it provides S3 bucket access for https storage. We’ll see how to read a GIF file, save it locally and convert it to PNG then upload to a S3 bucket using a pre-signed url which we will then use in our JSON as the https source for our image to display using APL

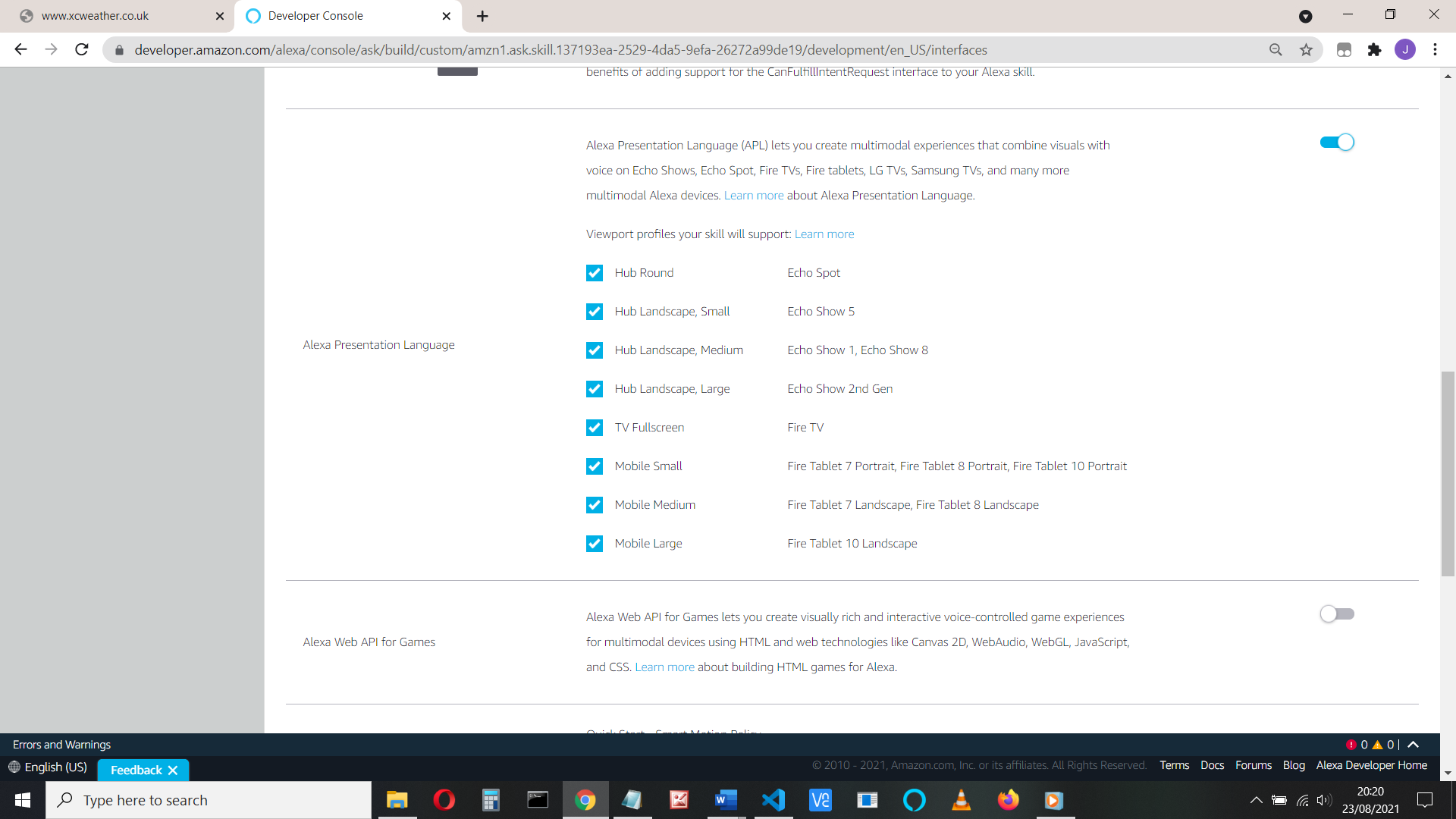
**Instructions**

Start by creating an Alexa Hosted skill, I’ve called it **GIF display**, use custom model, Alexa Hosted (Python) and your region

Select Start from Scratch and Create the skill

Check the invocation name

Click Interfaces and select the APL:



Save the model and build it.

Add the following imports and JSON code at the top of the code:

from ask\_sdk\_model.interfaces.alexa.presentation.apl import (

RenderDocumentDirective, ExecuteCommandsDirective, SpeakItemCommand, AutoPageCommand, HighlightMode)

import urllib.request

from utils import create\_presigned\_url

from PIL import Image

mainjson = {

"type": "APL",

"version": "1.7",

"license": "Copyright 2021 Amazon.com, Inc. or its affiliates blah blah",

"settings": {},

"theme": "dark",

"import": [],

"resources": [],

"styles": {},

"onMount": [],

"graphics": {},

"commands": {},

"layouts": {},

"mainTemplate": {

"parameters": [

"payload"

],

"items": [

{

"type": "Container",

"items": [

{

"source": "imageurl",

"type": "Image",

"width": "100vw",

"height": "100vh"

}

]

}

]

}

}

The JSON file is for the APL display and just displays a single image.

Modify the create\_presigned\_url code in utils.py to return the s3\_client and bucket\_name.Change the last line to: **return response, s3\_client, bucket\_name**

def create\_presigned\_url(object\_name):

"""Generate a presigned URL to share an S3 object with a capped expiration of 60 seconds

:param object\_name: string

:return: Presigned URL as string. If error, returns None.

"""

s3\_client = boto3.client('s3',

region\_name=os.environ.get('S3\_PERSISTENCE\_REGION'),

config=boto3.session.Config(signature\_version='s3v4',s3={'addressing\_style': 'path'}))

try:

bucket\_name = os.environ.get('S3\_PERSISTENCE\_BUCKET')

response = s3\_client.generate\_presigned\_url('get\_object',

Params={'Bucket': bucket\_name,

'Key': object\_name},

ExpiresIn=60\*1)

except ClientError as e:

logging.error(e)

return None

# The response contains the presigned URL

**return response, s3\_client, bucket\_name**

Save the file

Now change the LaunchRequestHandler to the following. It reads a GIF file from my github resource, but you can provide your own if you want. The code saves the GIF locally, then reads it and saves as a PNG (using PIL), It then gets a presigned url and uses that to upload the PNG file into the S3 bucket. The file is then accessed by the modified JSON to display the PNG image.

You can un-comment the code that lists the objects in the bucket if you want to.

class LaunchRequestHandler(AbstractRequestHandler):

"""Handler for Skill Launch."""

def can\_handle(self, handler\_input):

# type: (HandlerInput) -> bool

return ask\_utils.is\_request\_type("LaunchRequest")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Hello from image display!"

# retrieve GIF file and saving locally

urllib.request.urlretrieve('https://raw.githubusercontent.com/jallwork/savingTemporaryAlexaFiles/main/perkin.gif', "/tmp/perkin.gif")

imagegif= Image.open("/tmp/perkin.gif")

# now save image as png locally

imagegif.save(r'/tmp/perkin.png')

# now get presigned url and save

object\_name = 'S3perkin.png'

# I've modified create\_presigned\_url to return response, s3\_client & bucket\_name

image\_url, s3\_client, bucket\_name = create\_presigned\_url(object\_name)

s3\_client.upload\_file(Filename='/tmp/perkin.png', Bucket=bucket\_name, Key=object\_name)

"""

# lets' check - List objects present in the bucket

logger.info("listing object in bucket")

response = s3\_client.list\_objects(Bucket=bucket\_name)

logger.info("list object response: %s" % response)

"""

mainjson["mainTemplate"]["items"][0]["items"][0]["source"] = image\_url

speak\_output = 'loaded, saved, converted and displayed gif sucessfully'

return (

handler\_input.response\_builder

.speak(speak\_output)

.add\_directive(

RenderDocumentDirective(

token="pagerToken",

document=mainjson

)

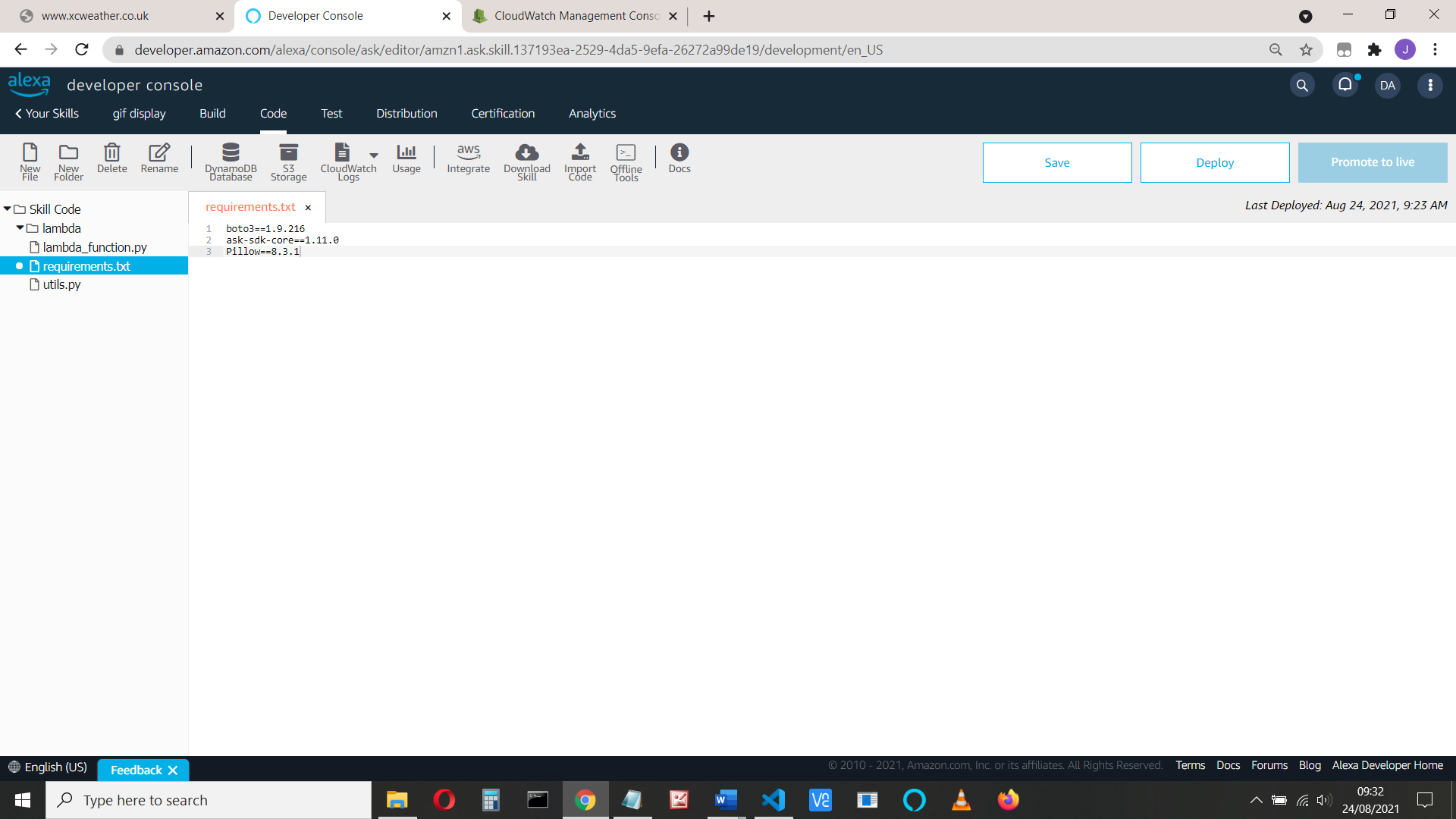
)

.response

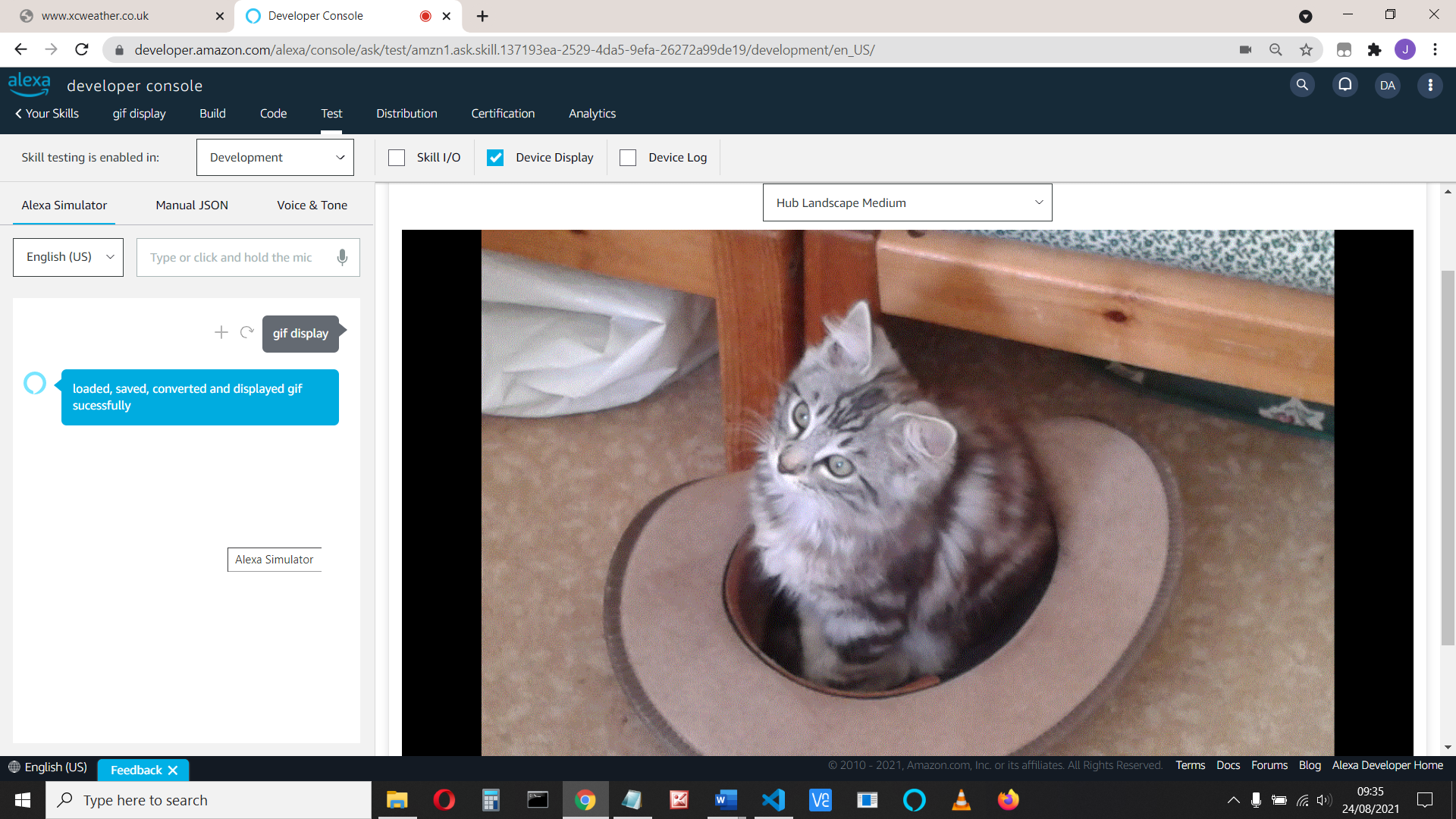
)

Finally, add the Pillow libraries to the requirements.txt file.

Add **Pillow==8.3.1**



Save, deploy and test your code: enable development and invoke your skill:



This is where we all go “Aaaahhhh”

If you get any errors, look at your cloudwatch logs (Code tab > Cloudwatch)

OK hope that helped