**Alexa ASK CLI V2 ISP**

<https://youtu.be/FD-CZ4i-GVU>

This follows on from my previous videos and github repos on In Skill Purchasing

Start here if you haven’t seen them: https://youtu.be/h0EIVhIfD98

I spent a long time looking at the documents but some still point to the old ASK CLI V1 instructions, e.g.:

<https://developer.amazon.com/en-US/docs/alexa/in-skill-purchase/use-the-cli-to-manage-in-skill-products.html>

And others were confusing. “ISP resources are now managed with the Skill Package resources.” – which implies a specific file structure.

I couldn’t find any instructions on how to do this with ASK CLI V2. So, here’s the easy way of using ASK CLI V2 and In Skill Purchasing

We’ll use ASK CLI to create a new AWS Lambda skill before using the development console to create a one-time purchase (entitlement) ISP and finally modify the (Python) code and then run the ISP skill. The skill tells a joke and the OTP give you access to more jokes

**Create the Lambda skill**

I use VS code

type **ask new**

Answer the questions:

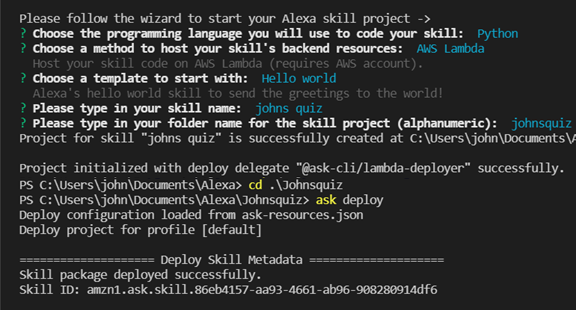
Language: Python

Host: AWS Lambda

Template: Hello world

skill name: johnsquiz

folder name: FOLDERNAME

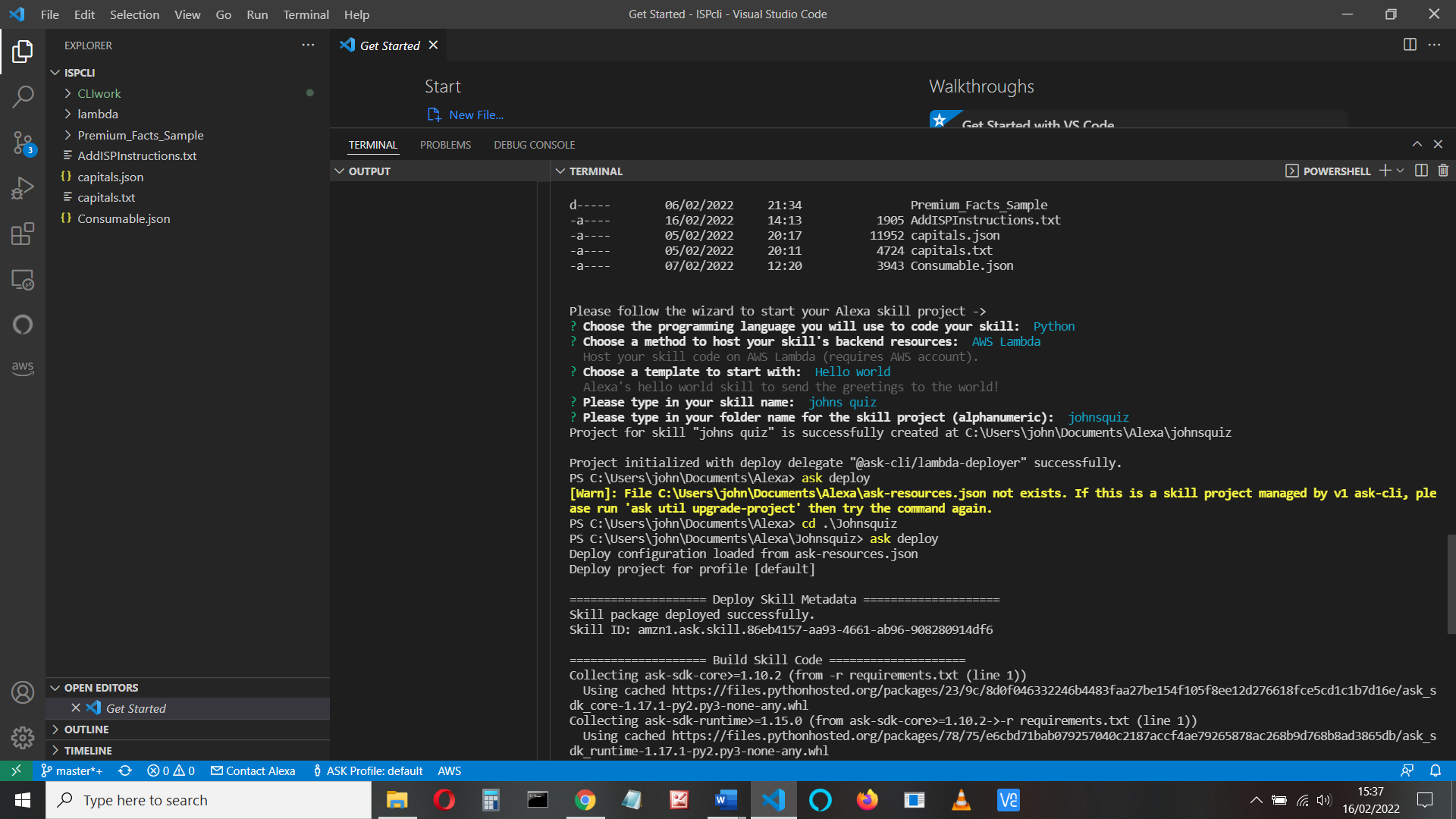


This creates the files we need. The important ones are our code in hello\_world.py; the intents and utterances are in interactionModels\custom\en-US.json and the overall skill description in ask-resources.json

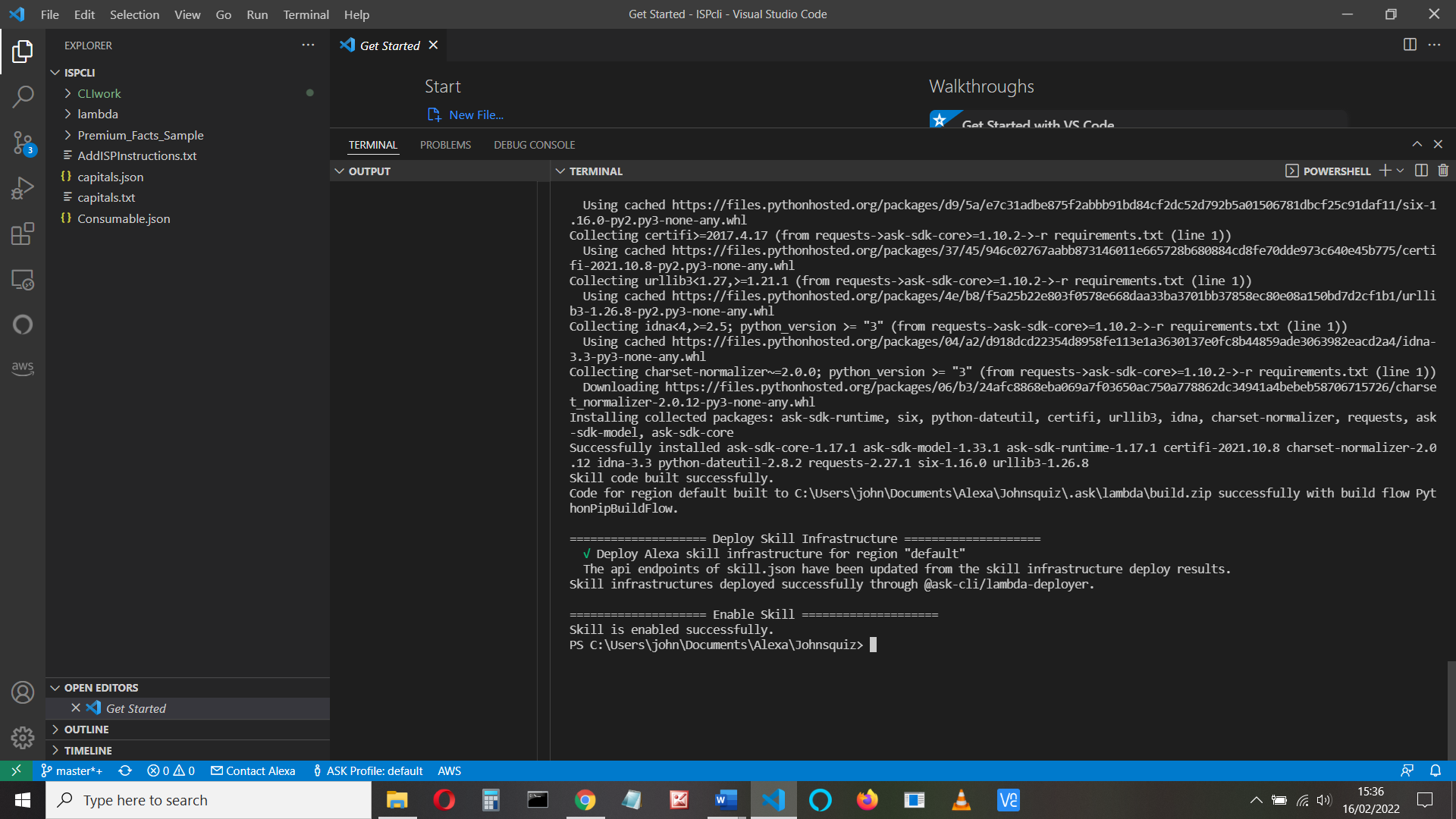
**Deploy your skill**

Deploying your skill uploads the code and creates the skill in the developer console.

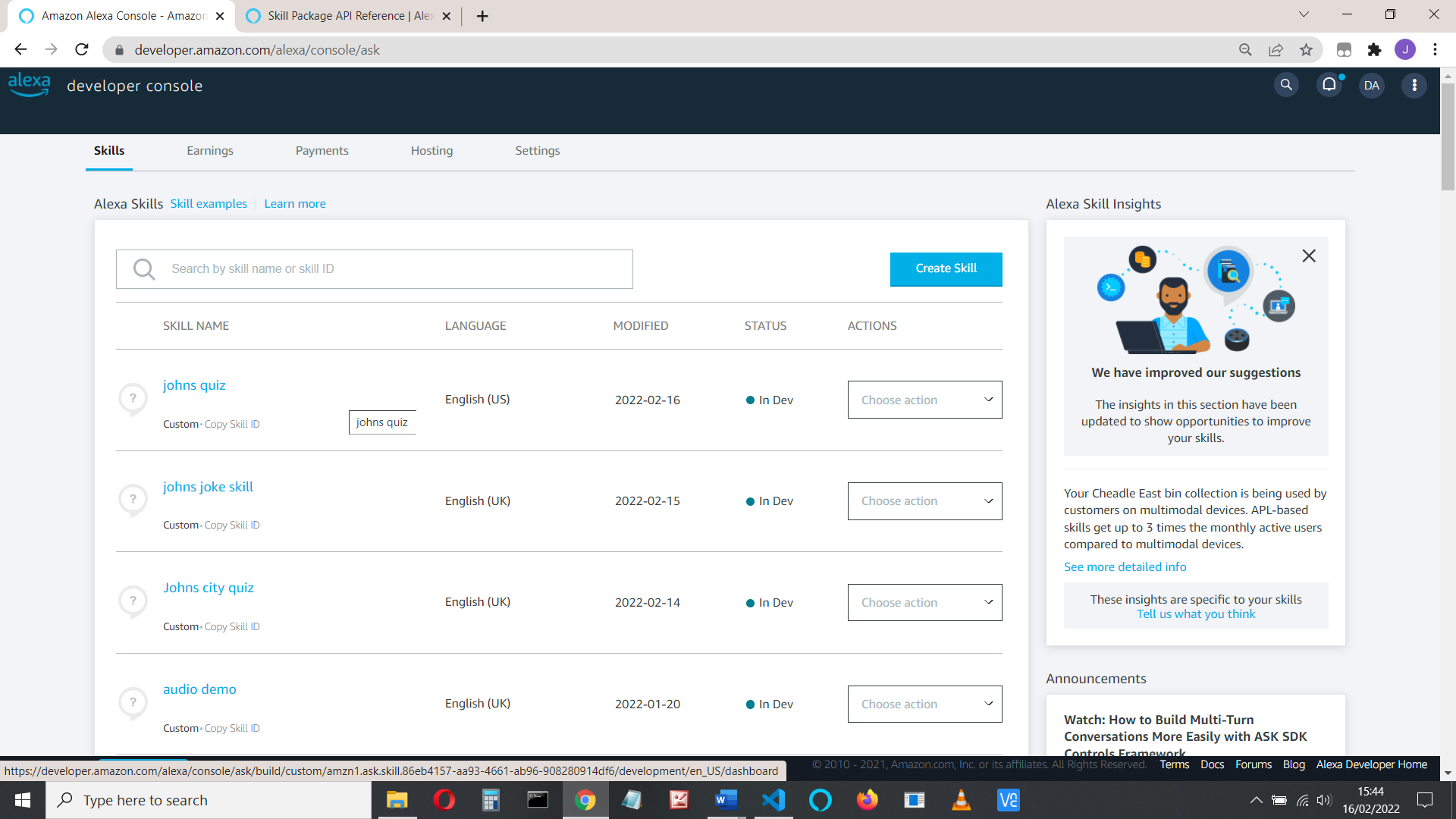
Change directory into your folder and type **ask deploy** (make sure you’ are in the correct folder!)



And a bit later..



Once created you can go to developer.amazon.com and find your skill there

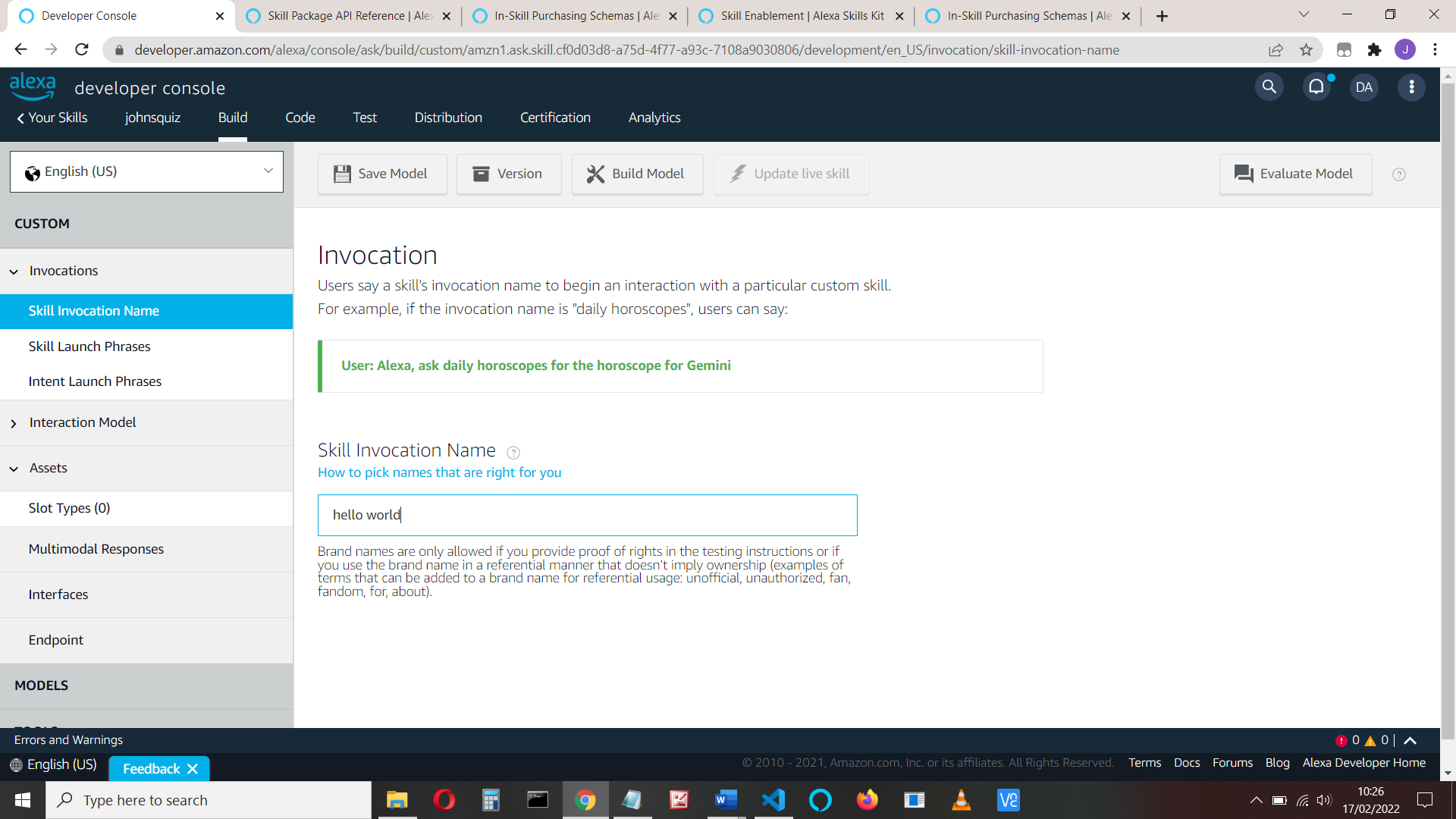


You may want the skill id, so save that, e.g.

amzn1.ask.skill.cf0d03d8-a75d-4f77-a93c-7108a9031234

**Test your skill**

You can test the skill in the developer console, but check the invocation name first of all. It might just be *hello world*:



Click on the Test tab, Enable development testing if necessary and type “open Hello world”

You can change the invocation name in skill-package/interaction models/custom/en-US.json if you want, but we will overwrite it later.

**Create the ISP**

We’ll do this in the Developer Console. This is covered in detail in my other videos, so briefly:

Choose Build > Models > Monetize your skill. Select Add In-skill product then Next

Then Create in-skill product.

Select OTP, Give your skill a reference name e.g. **morejokes**

Copy the product ID and click save, e.g.

**amzn1.adg.product.7fb47f73-ef55-4f7a-ac3e-96d152bb4321**

Add supported language (I’m using English(US), even though I’m in the UK) and answer the questions:

Display name –use **more jokes** – use a simple name

One sentence description – **This gives you access to more jokes**

Detailed description – **Buy this to gain access to more jokes.**

Small icon for this in-skill product – The image used for the product. \*

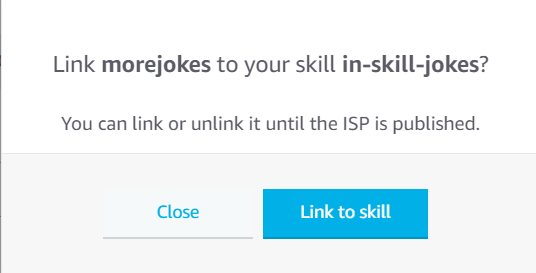
Large icon for this in-skill product – The image used for the product. \*

\* To create an icon, you can use <https://developer.amazon.com/docs/tools/icon-builder.html>

Purchase prompt description – What the customer hears when making a purchase – **Thank you, you've now got access to more jokes**

Purchase confirmation description – **You’ve just bought access to more jokes**.

And link to the skill



**Add the skill ISP code**

The code used here is the same as the previous videos, see

https://github.com/jallwork/Alexa\_ISP

Return to the VS code.

1. Change requirements.txt to:

boto3==1.9.216

ask-sdk-core==1.11.0

ask\_sdk

1. Add the intents we need - replace the en-US.json with the code below \*\*

Check that the invocation name is what you need

"invocationName": “johns in skill jokes ",

1. Add the joke code

Add a joke,py file (contents below) in the lambda folder.

1. Change the hello\_world.py code

Replace the hello\_world.py code with the code below.

1. Change the product\_id values in two lines (about lines 89 and 163) (Yes, I know they shouldn’t be hard-coded)

You can use smapi (Skill Management API) to see that the skill is linked:

ask smapi get-isp-list-for-skill-id --skill-id SKILL\_ID --stage development

e.g.

ask smapi get-isp-list-for-skill-id --skill-id amzn1.ask.skill.18e156d5-..-34 --stage development

The launch request calls .get\_monetization\_service which indicates if the product or skill is purchasable and/or already purchased by the current customer.

If the user has paid for extra jokes, this fact is stored in a session attribute and is mentioned in the response. Otherwise, the skill tells the user (s)he can buy more jokes.

If you want to know more, see:

<https://developer.amazon.com/en-US/docs/alexa/alexa-skills-kit-sdk-for-python/call-alexa-service-apis.html>

1. Finally **ask deploy and test the program**

* “open johns in skill jokes”
* “tell me a joke”
* “what can I buy”
* “upgrade” – “yes”
* “tell me a joke”

I got an error: “log() missing 1 required positional argument: 'msg'” It was because I’d copied the wrong product ID, so be careful of that.

The test didn’t work for 2 reasons: 1: I forgot to overwrite the en-US.json, 2: I then uploaded the hello\_world.py, which had the old product\_ID numbers, so failed. If it doesn’t work, check those.

Reference:

Entitlement ISP JSON example

{

"version": "1.0",

"type": "ENTITLEMENT",

"referenceName": "paidforjoke",

"publishingInformation": {

"locales": {

"en-US": {

"name": "paid joke",

"smallIconUri": "https://johnallworksbucket.s3.amazonaws.com/waveform108.png",

"largeIconUri": "https://johnallworksbucket.s3.amazonaws.com/waveform512.png",

"summary": "This gives you access to more jokes",

"description": "Buy this to gain access to more jokes.",

"examplePhrases": [

"let's buy it",

"upgrade",

"Can I buy it",

"purchase",

"I want to buy it"

],

"keywords": [

"Games"

],

"customProductPrompts": {

"purchasePromptDescription": "Thank you, you've now got access to more jokes",

"boughtCardDescription": "You've just bought access to more jokes."

}

}

},

"distributionCountries": [

"US"

],

"pricing": {

"amazon.com": {

"releaseDate": "2022-01-01",

"defaultPriceListing": {

"price": 0.99,

"currency": "USD"

}

}

},

"taxInformation": {

"category": "SOFTWARE"

}

},

"privacyAndCompliance": {

"locales": {

"en-US": {

"privacyPolicyUrl": "https://aws.amazon.com/privacy/"

}

}

},

"testingInstructions": "Start a game. Say 'upgrade' to purchase the product",

"purchasableState": "PURCHASABLE"

}

**ask-resources.json**

{

"askcliResourcesVersion": "2020-03-31",

"profiles": {

"default": {

"skillId": "",

"skillMetadata": {

"src": "./skill-package"

},

"code": {

"default": {

"src": "./lambda"

}

},

"skillInfrastructure": {

"userConfig": {

"runtime": "python3.6",

"handler": "hello\_world.handler",

"awsRegion": "us-east-1"

},

"type": "@ask-cli/lambda-deployer"

}

}

}

}

\*\* **en-US.json**

{

"interactionModel": {

"languageModel": {

"invocationName": "johns in skill jokes",

"intents": [

{

"name": "AMAZON.CancelIntent",

"samples": []

},

{

"name": "AMAZON.HelpIntent",

"samples": []

},

{

"name": "AMAZON.StopIntent",

"samples": []

},

{

"name": "AMAZON.NavigateHomeIntent",

"samples": []

},

{

"name": "HelloWorldIntent",

"slots": [],

"samples": [

"Tell me a joke",

"hello",

"how are you",

"say hi world",

"say hi",

"hi",

"say hello world",

"say hello"

]

},

{

"name": "BuyIntent",

"slots": [],

"samples": [

"let's buy it",

"upgrade",

"Can I buy it",

"purchase",

"I want to buy it"

]

},

{

"name": "TellMeMoreIntent",

"slots": [],

"samples": [

"What can I purchase",

"What can I get",

"Tell me more",

"What can I buy"

]

},

{

"name": "RefundPurchase",

"slots": [],

"samples": [

"i want a refund",

"return goods",

"refund purchase",

"refund"

]

},

{

"name": "AMAZON.FallbackIntent",

"samples": []

}

],

"types": []

}

}

}

**joke.py**

import random

def freejokes():

jokes = ["I was awake all night wondering where the sun went, but then it dawned on me",

"A burger walks into a bar. The bartender says 'Sorry, we don't serve food here'",

"Once you've seen one shopping center, you've seen the mall.",

"I didn't like my beard at first. Then it grew on me.",

"Time flies like an arrow. Fruit flies like a banana.",

"A backwards poet writes inverse.",

"To the guy who invented zero: Thanks for nothing!",

"A man walks into a bar. Ouch! It was an iron bar",

"What's brown and sticky? A stick."]

randNo = random.randint(0,len(jokes)-1)

return jokes[randNo]

def paidjokes():

jokes=["I'm reading a book about anti-gravity. I can't put it down.",

"Do you remember that joke I told you about my spine? It was about a weak back!",

"I just went to an emotional wedding. Even the cake was in tiers.",

"When's the best time to go to the dentist? Tooth-hurtie!",

"Why do seagulls fly over the sea? Because if they flew over the bay, they're bagels!",

"What do you call a farm that makes bad jokes? Corny!",

"Why do fish live in salt water? Because pepper makes them sneeze!",

"What kind of streets do ghosts haunt? Dead ends!",

"Why do you tell actors to break a leg? Because every play has a cast!",

"What kind of dogs love car racing? Lap dogs!",

"What did Winnie the Pooh say to his agent? 'Show me the honey!'",

"What do you call birds who stick together? Vel-crows.",

"Today I gave my dead batteries away. They were free of charge.",

"What do you call it when one cow spies on another? A steak out!"]

randNo = random.randint(0,len(jokes)-1)

return jokes[randNo]

**hello\_world.py**

# -\*- coding: utf-8 -\*-

# This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK for Python.

# Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,

# session persistence, api calls, and more.

# This sample is built using the handler classes approach in skill builder.

import logging

import ask\_sdk\_core.utils as ask\_utils

import joke

from ask\_sdk\_core.skill\_builder import SkillBuilder

from ask\_sdk\_core.dispatch\_components import AbstractRequestHandler

from ask\_sdk\_core.dispatch\_components import AbstractExceptionHandler

from ask\_sdk\_core.handler\_input import HandlerInput

from ask\_sdk\_model import Response

logger = logging.getLogger(\_\_name\_\_)

logger.setLevel(logging.INFO)

from ask\_sdk.standard import StandardSkillBuilder

from ask\_sdk\_model.services.monetization import (

EntitledState, PurchasableState, InSkillProductsResponse, Error,

InSkillProduct)

from ask\_sdk\_model.interfaces.monetization.v1 import PurchaseResult

from ask\_sdk\_model import Response, IntentRequest

from ask\_sdk\_model.interfaces.connections import SendRequestDirective

skill\_name = "johns joke skill"

def in\_skill\_product\_response(handler\_input):

"""Get the In-skill product response from monetization service."""

# type: (HandlerInput) -> Union[InSkillProductsResponse, Error]

locale = handler\_input.request\_envelope.request.locale

ms = handler\_input.service\_client\_factory.get\_monetization\_service()

return ms.get\_in\_skill\_products(locale)

def get\_all\_entitled\_products(in\_skill\_product\_list):

"""Get list of in-skill products in ENTITLED state."""

# type: (List[InSkillProduct]) -> List[InSkillProduct]

entitled\_product\_list = [

l for l in in\_skill\_product\_list if (

l.entitled == EntitledState.ENTITLED)]

return entitled\_product\_list

def get\_speakable\_list\_of\_products(entitled\_products\_list):

"""Return product list in speakable form."""

# type: (List[InSkillProduct]) -> str

product\_names = [item.name for item in entitled\_products\_list]

if len(product\_names) > 1:

# If more than one, add and 'and' in the end

speech = " and ".join(

[", ".join(product\_names[:-1]), product\_names[-1]])

else:

# If one or none, then return the list content in a string

speech = ", ".join(product\_names)

return speech

class TellMeMoreHandler(AbstractRequestHandler):

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("TellMeMoreIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In TellMeMoreHandler")

# Tell the user about the product(s)

speak\_output = 'You can upgrade to hear more jokes. Just say, "Upgrade" to purchase this product.'

reprompt = "I didn't catch that. What can I help you with?"

return handler\_input.response\_builder.speak(speak\_output).ask(

reprompt).response

class BuyHandler(AbstractRequestHandler):

"""Handler for letting users buy the product.

User says: Alexa, buy <category>.

"""

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("BuyIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyHandler")

product\_id = 'amzn1.adg.product.d0aed689-6761-483c-baf9-9d282ab7b788'

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Buy",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="buyToken")

).response

class BuyResponseHandler(AbstractRequestHandler):

"""This handles the Connections.Response event after a buy occurs."""

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

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Buy")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In BuyResponseHandler")

if handler\_input.request\_envelope.request.status.code == "200":

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = "You have bought the access to more jokes, enjoy"

reprompt = "You can ask for more jokes"

elif purchase\_result in (

PurchaseResult.DECLINED.value,

PurchaseResult.ERROR.value,

PurchaseResult.NOT\_ENTITLED.value):

speech = "Thanks for your interest. Ask for another free joke"

reprompt = "Ask for a joke"

elif purchase\_result == PurchaseResult.ALREADY\_PURCHASED.value:

logger.info("Already purchased product")

speech = " You've already purchased more jokes, just ask for another joke"

reprompt = "Ask for a joke, help or exit"

else:

# Invalid purchase result value

logger.info("Purchase result: {}".format(purchase\_result))

speech = "Sorry, there was an error, please try again"

reprompt = "Ask for a joke, help or exit"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Error: {}".format(

handler\_input.request\_envelope.request.status.message))

return handler\_input.response\_builder.speak(

"There was an error handling your purchase request. "

"Please try again or ask for help")

class RefundPurchaseHandler(AbstractRequestHandler):

#

#Deal with refund requests

#User says: Alexa, refund

#

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("RefundPurchase")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In RefundPurchaseHandler")

product\_id= 'amzn1.adg.product.d0aed689-6761-483c-baf9-9d282ab7b788'

return handler\_input.response\_builder.add\_directive(

SendRequestDirective(

name="Cancel",

payload={

"InSkillProduct": {

"productId": product\_id

}

},

token="correlationToken")

).response

class CancelResponseHandler(AbstractRequestHandler):

#This handles the Connections.Response event after a cancel occurs.

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

# type: (HandlerInput) -> bool

return (ask\_utils.is\_request\_type("Connections.Response")(handler\_input) and

handler\_input.request\_envelope.request.name == "Cancel")

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In CancelResponseHandler")

in\_skill\_response = in\_skill\_product\_response(handler\_input)

product\_id = handler\_input.request\_envelope.request.payload.get("productId")

if in\_skill\_response:

product = [l for l in in\_skill\_response.in\_skill\_products

if l.product\_id == product\_id]

if handler\_input.request\_envelope.request.status.code == "200":

speech = None

reprompt = None

purchase\_result = handler\_input.request\_envelope.request.payload.get(

"purchaseResult")

purchasable = product[0].purchasable

if purchase\_result == PurchaseResult.ACCEPTED.value:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = False

handler\_input.attributes\_manager.session\_attributes = session\_attr

speech = ("You have successfully cancelled your paid joke access.")

reprompt = "Ask for a free joke"

if purchase\_result == PurchaseResult.DECLINED.value:

if purchasable == PurchasableState.PURCHASABLE:

speech = ("You don't currently have paid joke access.")

else:

speech = "Ask for a free joke"

reprompt = "Ask for a free joke"

return handler\_input.response\_builder.speak(speech).ask(

reprompt).response

else:

logger.log("Connections.Response indicated failure. "

"Error: {}".format(

handler\_input.request\_envelope.request.status.message))

return handler\_input.response\_builder.speak(

"There was an error handling your cancellation "

"request. Please try again or contact us for "

"help").response

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 = "Welcome, you can ask for a joke or say hello"

in\_skill\_response = in\_skill\_product\_response(handler\_input)

if isinstance(in\_skill\_response, InSkillProductsResponse):

entitled\_prods = get\_all\_entitled\_products(in\_skill\_response.in\_skill\_products)

if entitled\_prods and len(entitled\_prods) > 0:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

#in\_skill\_response = in\_skill\_product\_response(handler\_input)

if isinstance(in\_skill\_response, InSkillProductsResponse):

entitled\_prods = get\_all\_entitled\_products(in\_skill\_response.in\_skill\_products)

if entitled\_prods and len(entitled\_prods) > 0:

session\_attr = handler\_input.attributes\_manager.session\_attributes

session\_attr["paid\_jokes"] = True

handler\_input.attributes\_manager.session\_attributes = session\_attr

if entitled\_prods:

speak\_output = (

"Welcome to {}. You currently own {} products. "

"To hear a paid joke, say, 'Tell me a joke.").format(

skill\_name,

get\_speakable\_list\_of\_products(entitled\_prods))

else:

logger.info("No entitled products")

speak\_output = (

"Welcome to {}. To hear a joke you can say "

"'Tell me a joke', or to hear about the paid for jokes "

"for purchase, say 'What can I buy', or ask for help"

).format(skill\_name)

reprompt = "I didn't catch that. What can I help you with?"

else:

logger.info("Error calling InSkillProducts API: {}".format(

in\_skill\_response.message))

speak\_output = "Something went wrong in loading your purchase history."

reprompt = speak\_output

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

class HelloWorldIntentHandler(AbstractRequestHandler):

"""Handler for Hello World Intent."""

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("HelloWorldIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

ask\_output = "Here's a free joke: " + joke.freejokes()

speak\_output = "Here's your free joke: " + joke.freejokes() + ". You can purchase more jokes, just say 'What can I buy'"

session\_attr = handler\_input.attributes\_manager.session\_attributes

if "paid\_jokes" in session\_attr:

if session\_attr["paid\_jokes"] == True:

speak\_output = "Here's your paid joke: " +joke.paidjokes()

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(ask\_output)

.response

)

class HelpIntentHandler(AbstractRequestHandler):

"""Handler for Help Intent."""

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("AMAZON.HelpIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "You can say hello to me! How can I help?"

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

class CancelOrStopIntentHandler(AbstractRequestHandler):

"""Single handler for Cancel and Stop Intent."""

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

# type: (HandlerInput) -> bool

return (ask\_utils.is\_intent\_name("AMAZON.CancelIntent")(handler\_input) or

ask\_utils.is\_intent\_name("AMAZON.StopIntent")(handler\_input))

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

speak\_output = "Goodbye!"

return (

handler\_input.response\_builder

.speak(speak\_output)

.response

)

class FallbackIntentHandler(AbstractRequestHandler):

"""Single handler for Fallback Intent."""

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

# type: (HandlerInput) -> bool

return ask\_utils.is\_intent\_name("AMAZON.FallbackIntent")(handler\_input)

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

logger.info("In FallbackIntentHandler")

speech = "Hmm, I'm not sure. You can say Hello or Help. What would you like to do?"

reprompt = "I didn't catch that. What can I help you with?"

return handler\_input.response\_builder.speak(speech).ask(reprompt).response

class SessionEndedRequestHandler(AbstractRequestHandler):

"""Handler for Session End."""

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

# type: (HandlerInput) -> bool

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

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

# Any cleanup logic goes here.

return handler\_input.response\_builder.response

class IntentReflectorHandler(AbstractRequestHandler):

"""The intent reflector is used for interaction model testing and debugging.

It will simply repeat the intent the user said. You can create custom handlers

for your intents by defining them above, then also adding them to the request

handler chain below.

"""

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

# type: (HandlerInput) -> bool

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

def handle(self, handler\_input):

# type: (HandlerInput) -> Response

intent\_name = ask\_utils.get\_intent\_name(handler\_input)

speak\_output = "You just triggered " + intent\_name + "."

return (

handler\_input.response\_builder

.speak(speak\_output)

# .ask("add a reprompt if you want to keep the session open for the user to respond")

.response

)

class CatchAllExceptionHandler(AbstractExceptionHandler):

"""Generic error handling to capture any syntax or routing errors. If you receive an error

stating the request handler chain is not found, you have not implemented a handler for

the intent being invoked or included it in the skill builder below.

"""

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

# type: (HandlerInput, Exception) -> bool

return True

def handle(self, handler\_input, exception):

# type: (HandlerInput, Exception) -> Response

logger.error(exception, exc\_info=True)

speak\_output = "Sorry, I had trouble doing what you asked. Please try again."

return (

handler\_input.response\_builder

.speak(speak\_output)

.ask(speak\_output)

.response

)

# The SkillBuilder object acts as the entry point for your skill, routing all request and response

# payloads to the handlers above. Make sure any new handlers or interceptors you've

# defined are included below. The order matters - they're processed top to bottom.

#sb = SkillBuilder()

sb = StandardSkillBuilder()

sb.add\_request\_handler(TellMeMoreHandler())

sb.add\_request\_handler(BuyHandler())

sb.add\_request\_handler(BuyResponseHandler())

sb.add\_request\_handler(RefundPurchaseHandler())

sb.add\_request\_handler(CancelResponseHandler())

sb.add\_request\_handler(LaunchRequestHandler())

sb.add\_request\_handler(HelloWorldIntentHandler())

sb.add\_request\_handler(HelpIntentHandler())

sb.add\_request\_handler(CancelOrStopIntentHandler())

sb.add\_request\_handler(FallbackIntentHandler())

sb.add\_request\_handler(SessionEndedRequestHandler())

sb.add\_request\_handler(IntentReflectorHandler()) # make sure IntentReflectorHandler is last so it doesn't override your custom intent handlers

sb.add\_exception\_handler(CatchAllExceptionHandler())

**handler = sb.lambda\_handler()**