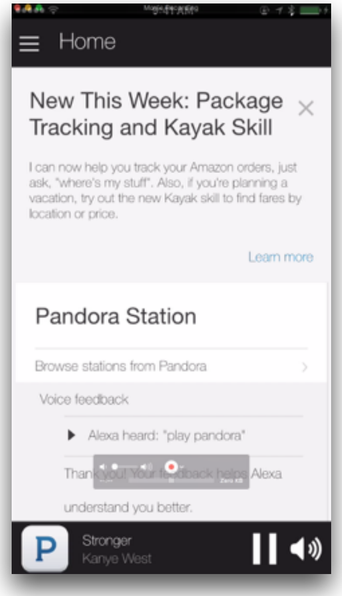
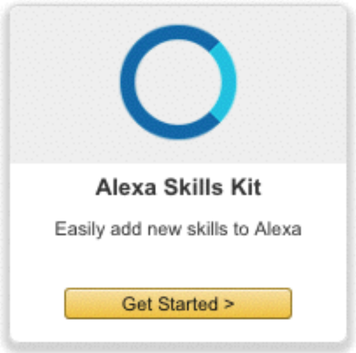
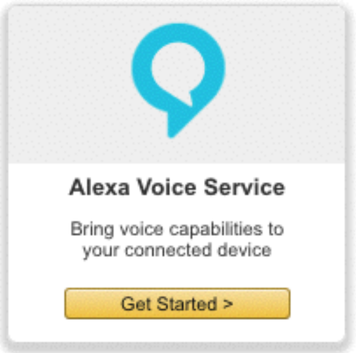
Famous Alexas of our time

Alexa and Python
mempy, May 16, 2016

Alexa and the Echo



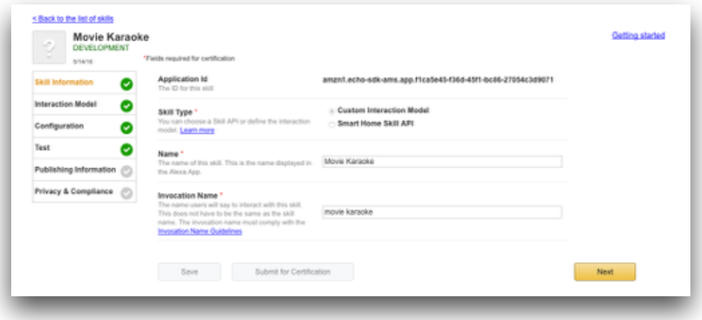




The 3 things

Alexa Skill

developer.amazon.com



A skill is your app
Skills start with an invocation name
Utterances are templated phrases
Utterances map to intents
Intents are your skill's base functions
Slots are the parameters passed to Intents
Intent events are passed to your lambda

Device

Alexa supported device
Echo, Tap, Dot, Triby
Raspberry Pi see Github

Lambda Instance

aws.amazon.com

A lambda is a micro PaaS
Returns a natural language response
Secured and tied to specific source
Event driven and scales
Java 8, Node 4, Python 2.7
Code in web editor or .zip
Can include libraries like requests

Skills

Invocation

Ask '**Movie Karaoke**'

Utterances

rateCityToCityIntent how much is it to ship from {**originationCity**} to {**destinationCity**}
rateCityToCityIntent how much does it cost to ship from {**originationCity**} to {**destinationCity**}

rateHereToCityIntent how much is it to ship from here to {**destinationCity**}
rateHereToCityIntent how much does it cost to ship from here to {**destinationCity**}
rateHereToCityIntent how much is it to ship to {**destinationCity**}
rateHereToCityIntent how much does it cost to ship to {**destinationCity**}

Intents

```
{
  "intents": [
    {
      "intent": "rateCityToCityIntent",
      "slots": [
        {
          "name": "originationCity",
          "type": "AMAZON.US_CITY"
        },
        {
          "name": "destinationCity",
          "type": "AMAZON.US_CITY"
        }
      ]
    }
}
```

Slots

Slot Types

AMAZON.DATE
AMAZON.DURATION
AMAZON.FOUR_DIGIT_NUMBER
AMAZON.NUMBER
AMAZON.TIME
AMAZON.US_CITY
AMAZON.US_FIRST_NAME
AMAZON.US_STATE
Custom - Named list of values

Lambda

aws:2hg435jhghjgsdf

“How much is it to ship from Collierville to Nashville?”

Request from Alexa

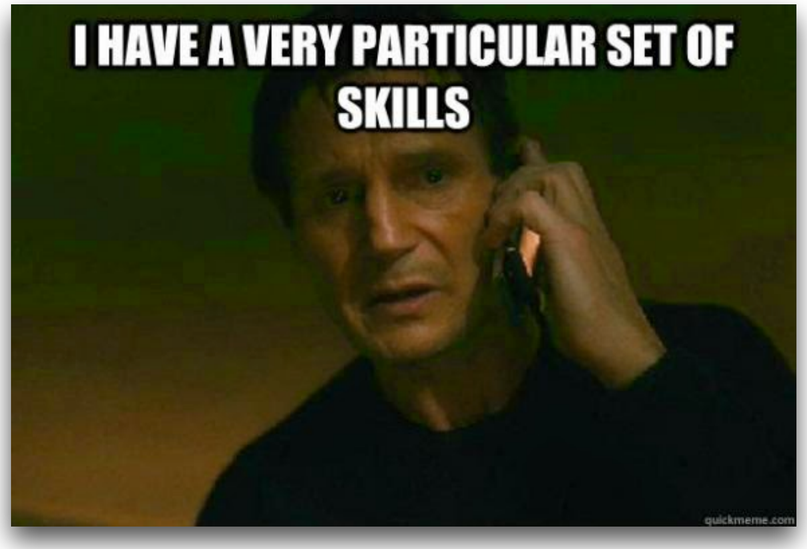

```
{
  "session": {
    "sessionId": "SessionId.3a1035",
    "application": {
      "applicationId": "amzn1.echo-sdk-e8"
    },
    "attributes": {},
    "user": {
      "userId": "amzn1.ask.account.AFP3ZWPOSY",
      "accessToken": "b0df04b2a76243689f0f6a"
    },
    "new": false
  },
  "request": {
    "type": "IntentRequest",
    "requestId": "EdwRequestId.32e51095-e79d",
    "timestamp": "2016-05-14T23:29:04Z",
    "intent": {
      "name": "rateCityToCityIntent",
      "slots": {
        "destinationCity": {
          "name": "destinationCity",
          "value": "Nashville"
        },
        "originationCity": {
          "name": "originationCity",
          "value": "Collierville"
        }
      }
    },
    "locale": "en-US"
  },
  "version": "1.0"
}
```


Amazon Lambda

Response from Lambda


```
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "type": "PlainText",
      "text": "FedEx found the following shipment rates from Collierville, Tennessee to Nashville, Tennessee. Standard Overnight costs 67 dollars and 98 cents. Fedex 2 Day costs 41 dollars and 72 cents. Fedex Ground costs 10 dollars and 23 cents. "
    },
    "card": {
      "content": "SessionSpeechlet - FedEx found the following shipment rates from Collierville, Tennessee to Nashville, Tennessee. Standard Overnight costs 67 dollars and 98 cents. Fedex 2 Day costs 41 dollars and 72 cents. Fedex Ground costs 10 dollars and 23 cents. ",
      "title": "SessionSpeechlet - Shipping Rate",
      "type": "Simple"
    },
    "reprompt": {
      "outputSpeech": {
        "type": "PlainText"
      }
    },
    "shouldEndSession": false
  },
  "sessionAttributes": {}
}
```

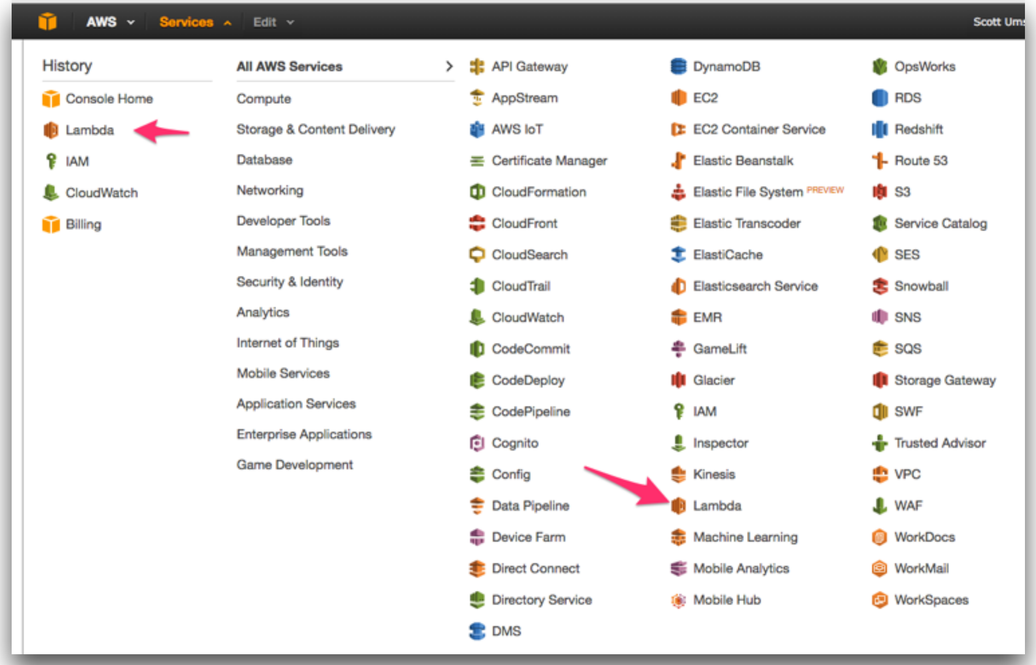
Create a skill

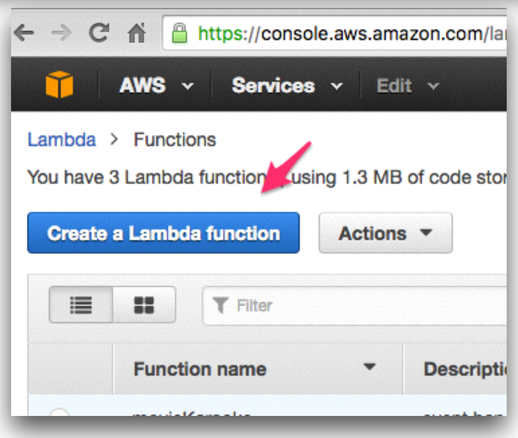


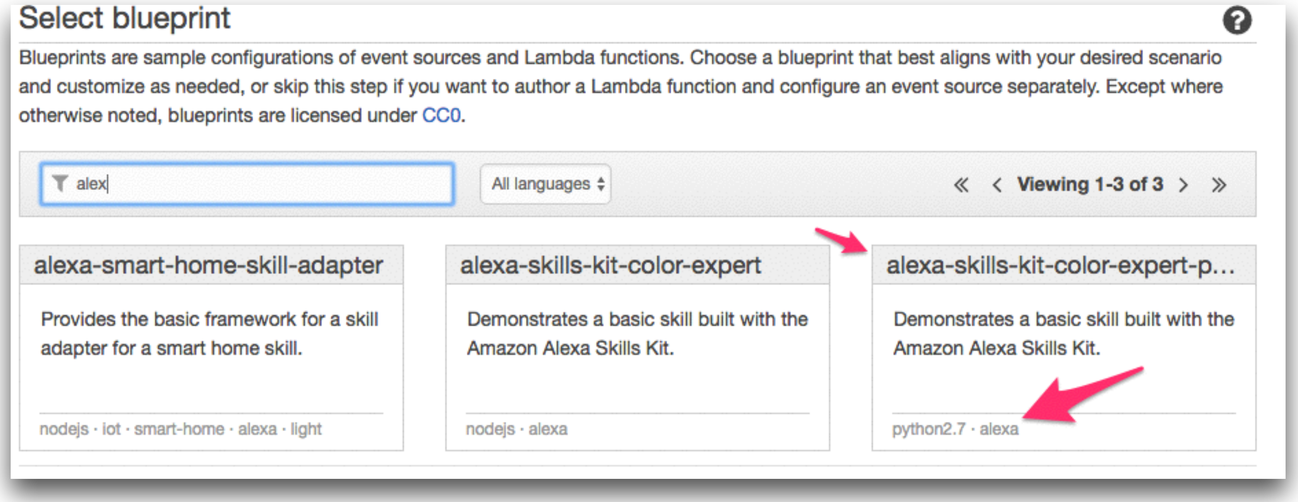
1. Create Lambda
2. Create Site
3. Move code to Lambda

Step 1 - Create Lambda

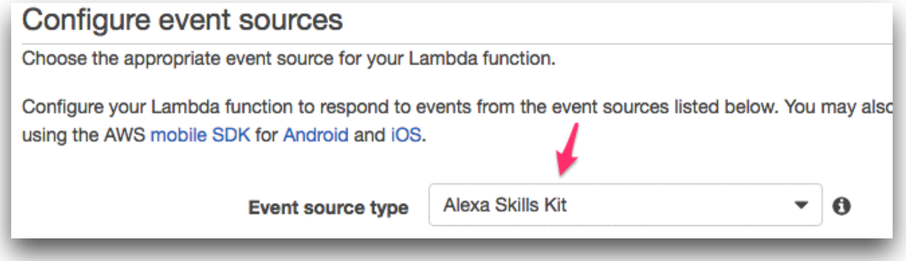
aws.amazon.com

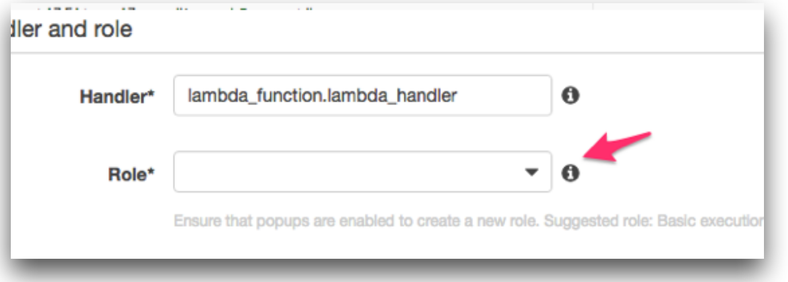




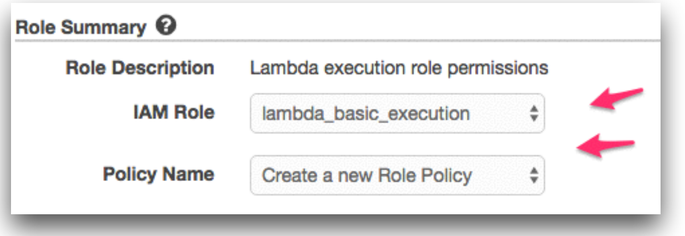


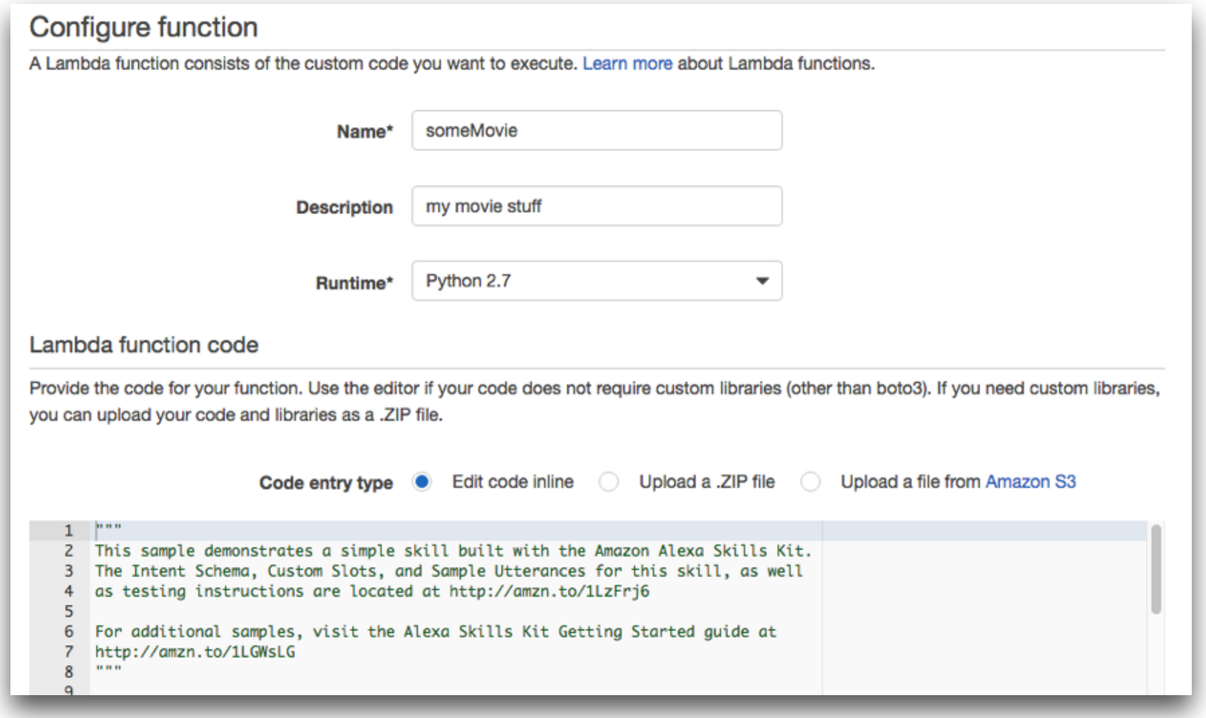
more lambda stuff

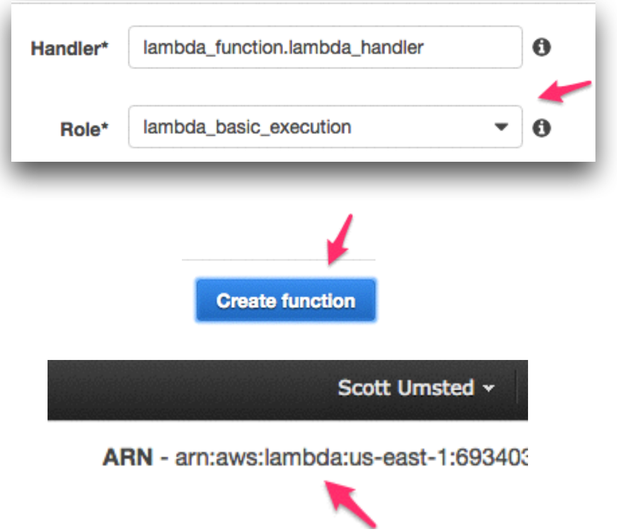




Weird Popup



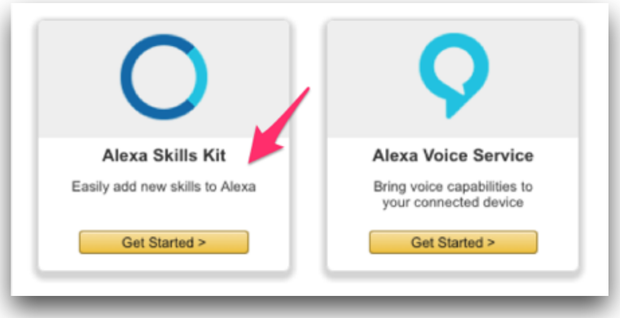


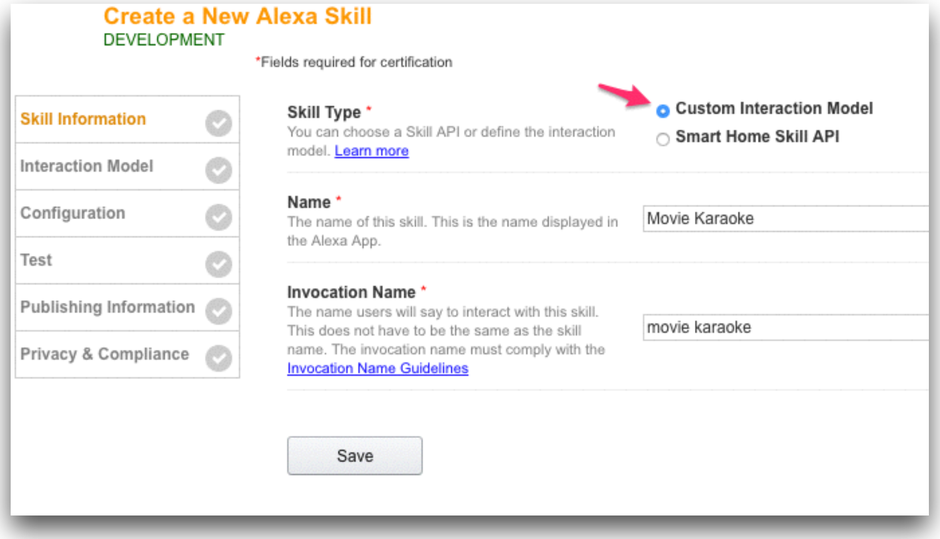


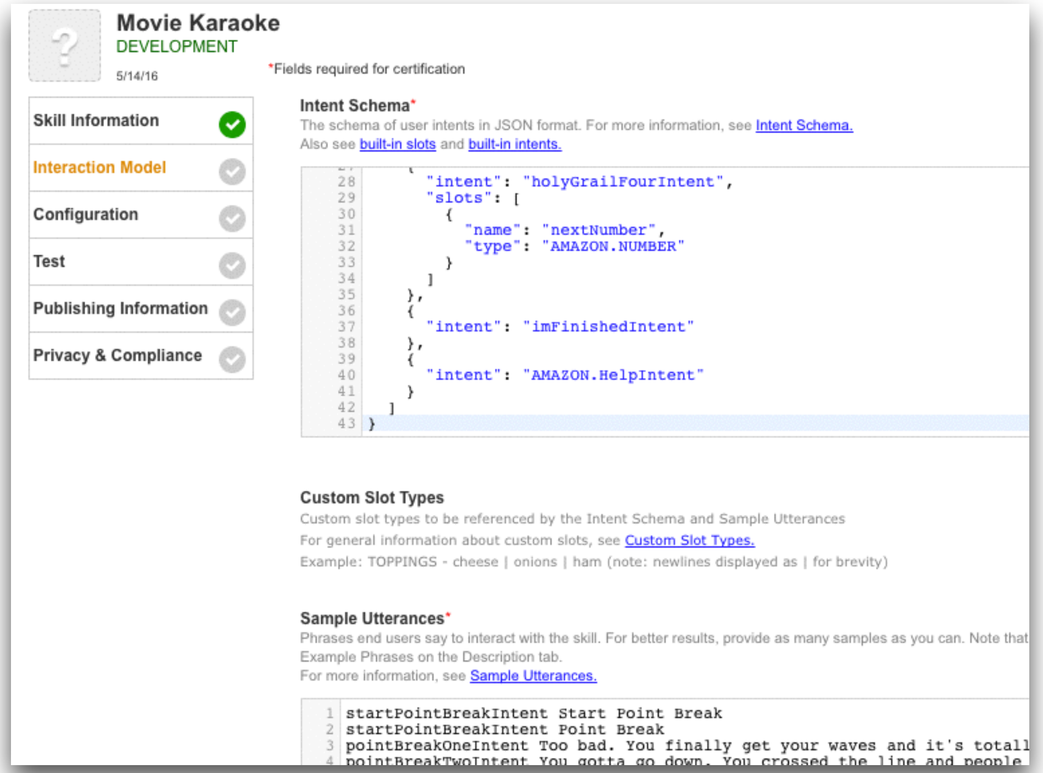
We'll add our code here later

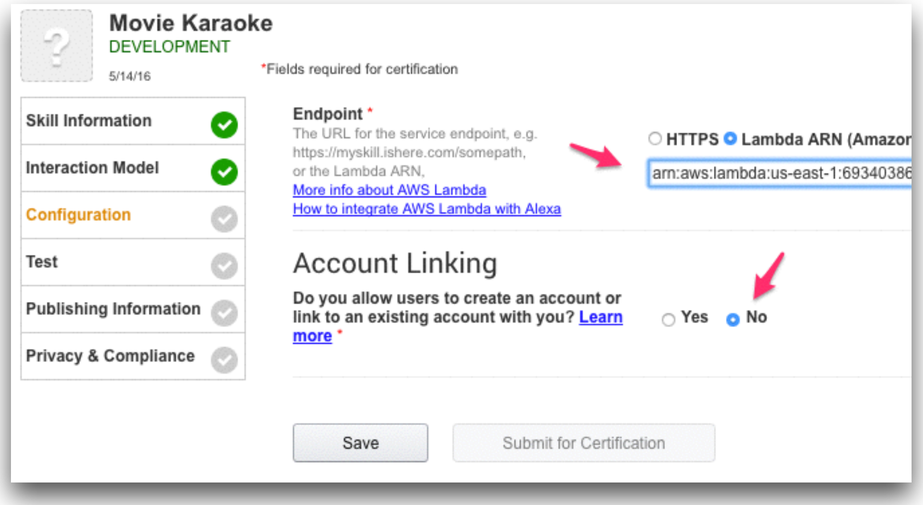
We'll need this for the skill

Step 2 - Create Skill

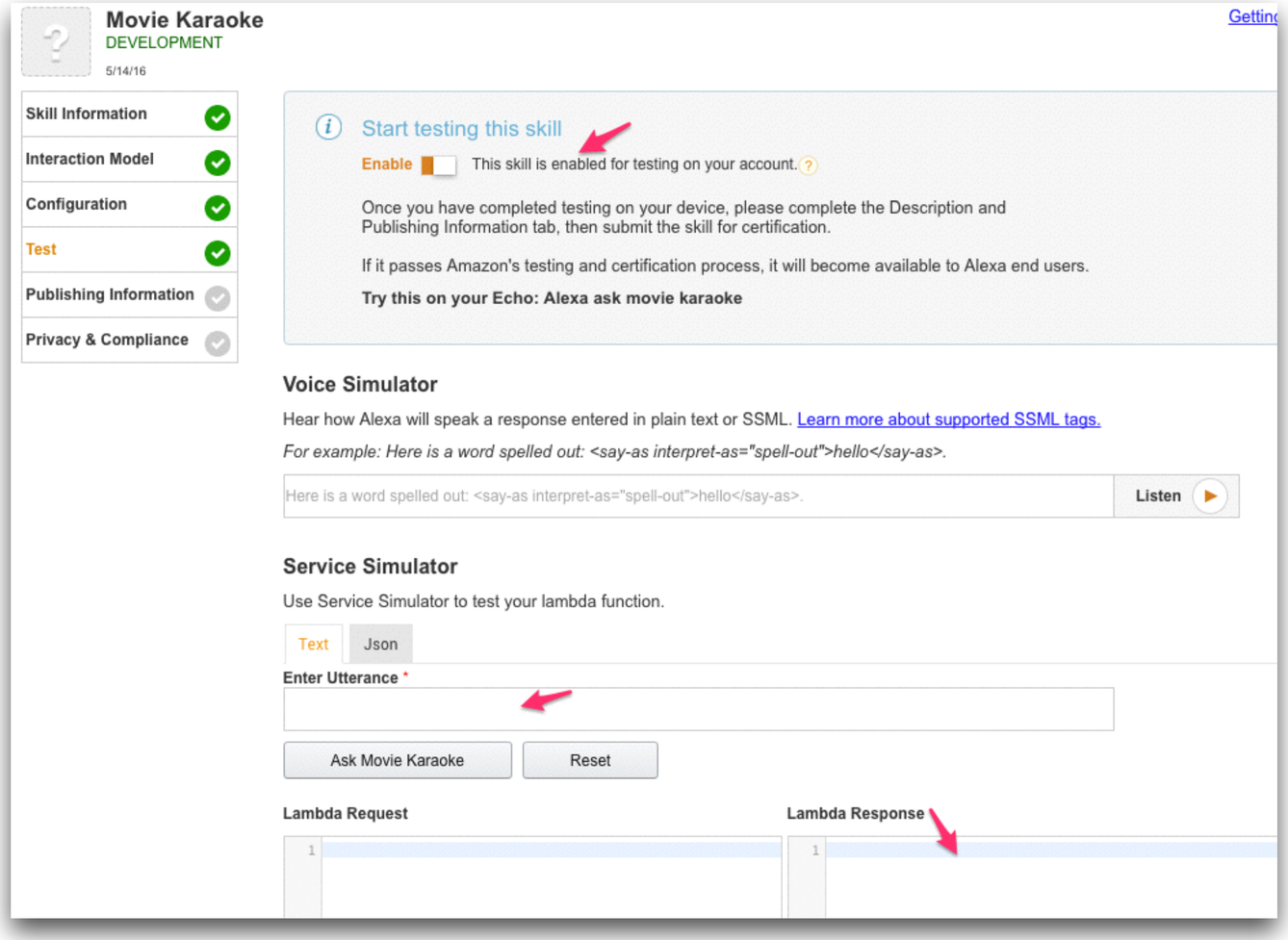








Where we test our skill once we finish writing our code

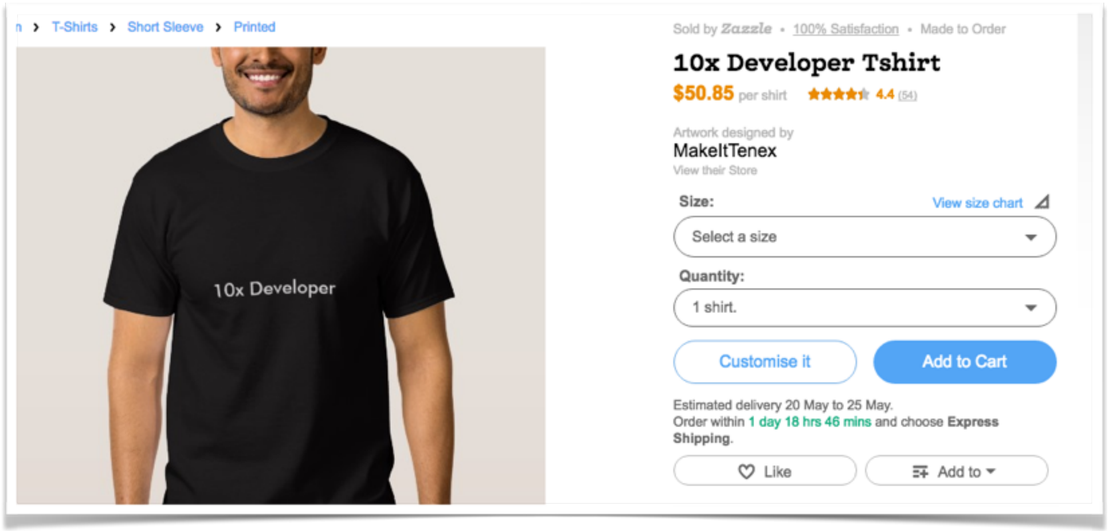


Step 3 Our Code

step 2 ~~steal~~ clone Amazon's code

```
def lambda_handler(event, context):
    """ Route the incoming request based on type (LaunchRequest, IntentRequest,
    etc.) The JSON body of the request is provided in the event parameter.
    """
    print("event.session.application.applicationId=" +
          event['session']['application']['applicationId'])

    """
    Uncomment this if statement and populate with your skill's application ID to
    prevent someone else from configuring a skill that sends requests to this
    function.
    """
    # if (event['session']['application']['applicationId'] !=
    #         "amzn1.echo-sdk-ams.app.[unique-value-here]"):
    #     raise ValueError("Invalid Application ID")

    if event['session']['new']:
        on_session_started({'requestId': event['request']['requestId']},
                           event['session'])

    if event['request']['type'] == "LaunchRequest":
        return on_launch(event['request'], event['session'])
    elif event['request']['type'] == "IntentRequest":
        return on_intent(event['request'], event['session'])
    elif event['request']['type'] == "SessionEndedRequest":
        return on_session_ended(event['request'], event['session'])
```

step 1 put on favorite coding t-shirt




```
def on_intent(intent_request, session):
    """ Called when the user specifies an intent for this skill """

    print("on_intent requestId=" + intent_request['requestId'] +
          ", sessionId=" + session['sessionId'])

    intent = intent_request['intent']
    intent_name = intent_request['intent']['name']

    # Dispatch to your skill's intent handlers
    if intent_name in ["startPointBreakIntent"]:
        return start_point_break(intent, session)
    elif intent_name in ["pointBreakOneIntent"]:
        return holy_grail_4(intent, session)
    elif intent_name in ["imFinishedIntent"]:
        return i_am_finished(intent, session)
    elif intent_name == "AMAZON.HelpIntent":
        return get_welcome_response()
    elif intent_name == "AMAZON.CancelIntent" or intent_name == "AMAZON.StopIntent":
        return handle_session_end_request()
    else:
        raise ValueError("Invalid intent")
```

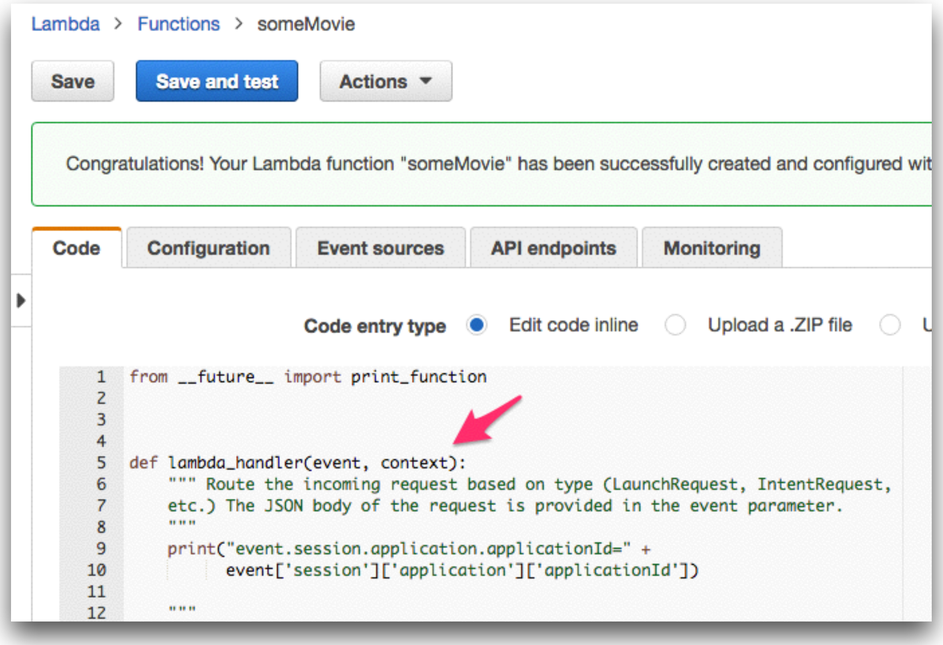
More with “our” lambda code

```
def point_break_1(intent, session):
    title = "Point Break"
    should_end_session = False
    session_attributes = {}
    reprompt_text = None

    speech_output = "Just waiting for my set."

    return build_response(session_attributes, build_speechlet_response(
        title, speech_output, reprompt_text, should_end_session))


def point_break_2(intent, session):
    title = "Point Break"
    should_end_session = False
    session_attributes = {}
    reprompt_text = None

    speech_output = "Yeah, it went bad, went real bad."

    return build_response(session_attributes, build_speechlet_response(
        title, speech_output, reprompt_text, should_end_session))

def holy_grail_4(intent, session):
    title = "Holy Grail"
    should_end_session = False
    session_attributes = {}
    reprompt_text = None

    if get_slot(intent, 'nextNumber', '3') == '3':
        speech_output = "Yes, Three! Boom!"
    else:
        speech_output = "Three, sir!"

    return build_response(session_attributes, build_speechlet_response(
        title, speech_output, reprompt_text, should_end_session))
```

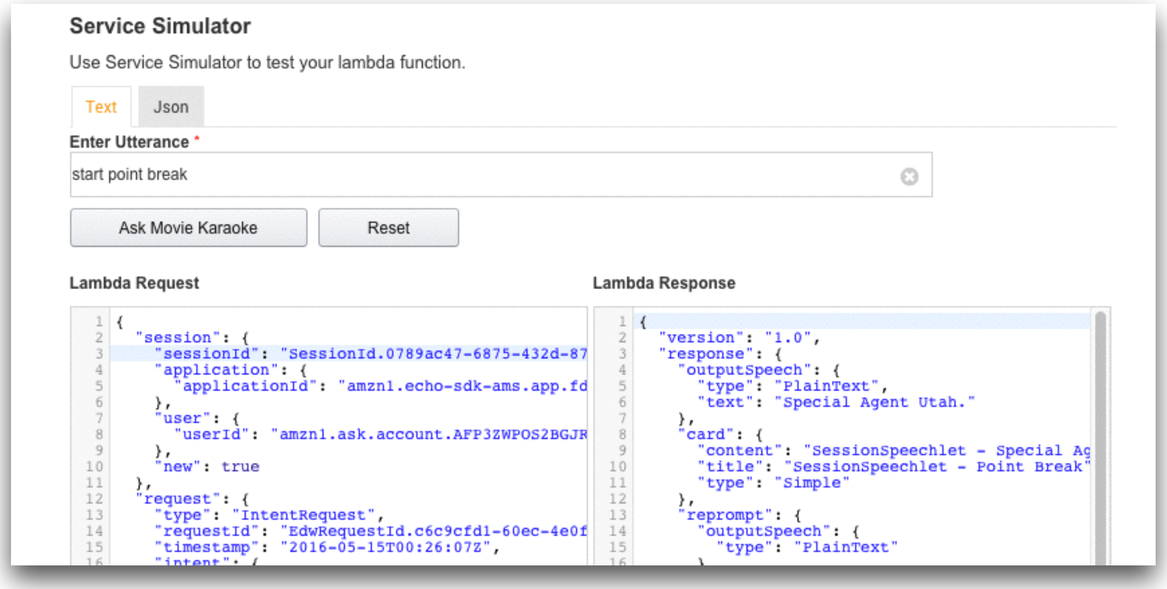
```
def get_slot(intent, key, default=None):
    try:
        return intent['slots'][key]['value']
    except:
        return default


def build_speechlet_response(title, output, reprompt_text, should_end_session):
    return {
        'outputSpeech': {
            'type': 'PlainText',
            'text': output
        },
        'card': {
            'type': 'Simple',
            'title': 'SessionSpeechlet - ' + title,
            'content': 'SessionSpeechlet - ' + output
        },
        'reprompt': {
            'outputSpeech': {
                'type': 'PlainText',
                'text': reprompt_text
            }
        },
        'shouldEndSession': should_end_session
    }


def build_response(session_attributes, speechlet_response):
    return {
        'version': '1.0',
        'sessionAttributes': session_attributes,
        'response': speechlet_response
    }
```

Paste into lambda and test

Lambda on aws.amazon.com



Test on developer.amazon.com



make something useful

Various Alexa Devices
Assistant + BT Speaker
Approximately 3M sold
Goal of 2 second response

amazon alexa

Enable skills
Identify users
Translate utterances
Invoke intents
Pass slots

amazon web services

Lightweight PaaS
Event based service
Coupled to Alexa
Intent based routes
Easily scales

heroku

PaaS for web and db
Services and web
Authorization provider
User profile, packages list
Translate external services

VPS + mailgun

Splash page
DNS routing of MX records
Route track request emails
Email based linking
Email activation process

some shipping company

Favorite Track
and Rate APIs

Enabling users while providing unique and individual experiences

Your 10x Code

enable → store stuff and present login form

type username password → establish account authenticate user generate access token

enabled ← redirect to Amazon state, token, response type

Request from Alexa


```
{
 "session": {
  "sessionId": "SessionId.3a1035",
  "application": {
   "applicationId": "amzn1.echo-sdk-e8"
  },
  "attributes": {},
  "user": {
   "userId": "amzn1.ask.account.AFP3ZWPOSY",
   "accessToken": "b0df04b2a76243689f0f6a"
  },
  "new": false
 },
 "request": {
  "type": "IntentRequest",
  "requestId": "EdwRequestId.32e51095-e79d",
  "timestamp": "2016-05-14T23:29:04Z",
  "intent": {
   "name": "rateCityToCityIntent",
   "slots": {
    "destinationCity": {
     "name": "destinationCity",
     "value": "Nashville"
    },
    "originationCity": {
     "name": "originationCity",
     "value": "Collierville"
    }
   }
  },
  "locale": "en-US"
 },
 "version": "1.0"
}
```