Installing Aplexaruss

HOW TO GET THIS DAMN APP WORKING…HOPEFULLY

1/17/2017

TWITTER: @RUSS152

Disclaimer

- This took me 2 weeks to get going.
- Granted, I didn't know how-to-Node prior to starting, and I had to update the code to support changes to the plex api, niether which you have to do now, but still...
- Expect this to take a few hours to get working
- The biggest hurdle was a lack of documentation, thus this ppt, sooooo if you can't get it to work with all of this... This might not be a project for you to do solo. Call a buddy, someone who is REALLY good at troubleshooting
- Good luck!

Capabilities & Limitations

- Capabilities
  - Enable Alexa to play KNOWN movies and tv episodes with commands like
    - Play Episode 1 Season 2 of The Office
    - Play the movie Godzilla 2000
    - Play a good episode of The Walking Dead
- Limitations
  - Launches only one default client, in our case a Roku
  - Can't dynamically see new content, new series/movies must be listed in the Alexa Skill's list of valid show/movie names
  - Only one server and does not play content of shared servers

Required Services

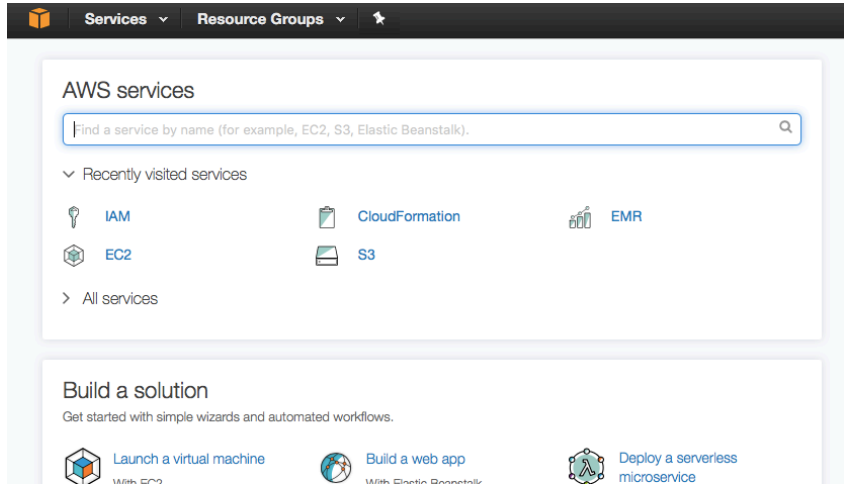
- This application requires 3 separate services, all of which reside in the AWS ecosystem. For personal/low use like this, it shouldn't cost you anything
  - Alexa Skills Kit
    - Used to create a skill that you install on your Amazon Echo (dot)
  - Lambda
    - This will run our Node.js app, and will be the end point for our Alexa Skill
  - DynamoDB
    - This is used by our Node.js to store information about our server and connections

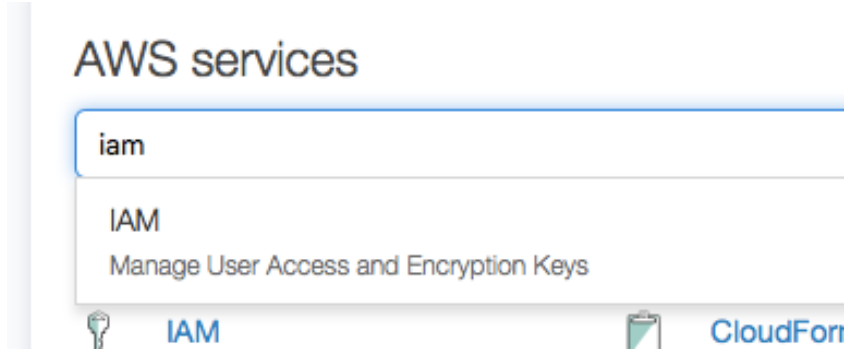
Plan of attack

- Setup our AWS Services
- Enable our workstation to deploy our services and app
- Configure our app
- Deploy our app
- Configure our network to allow Lambda to talk to our client

Setting Stuff Up

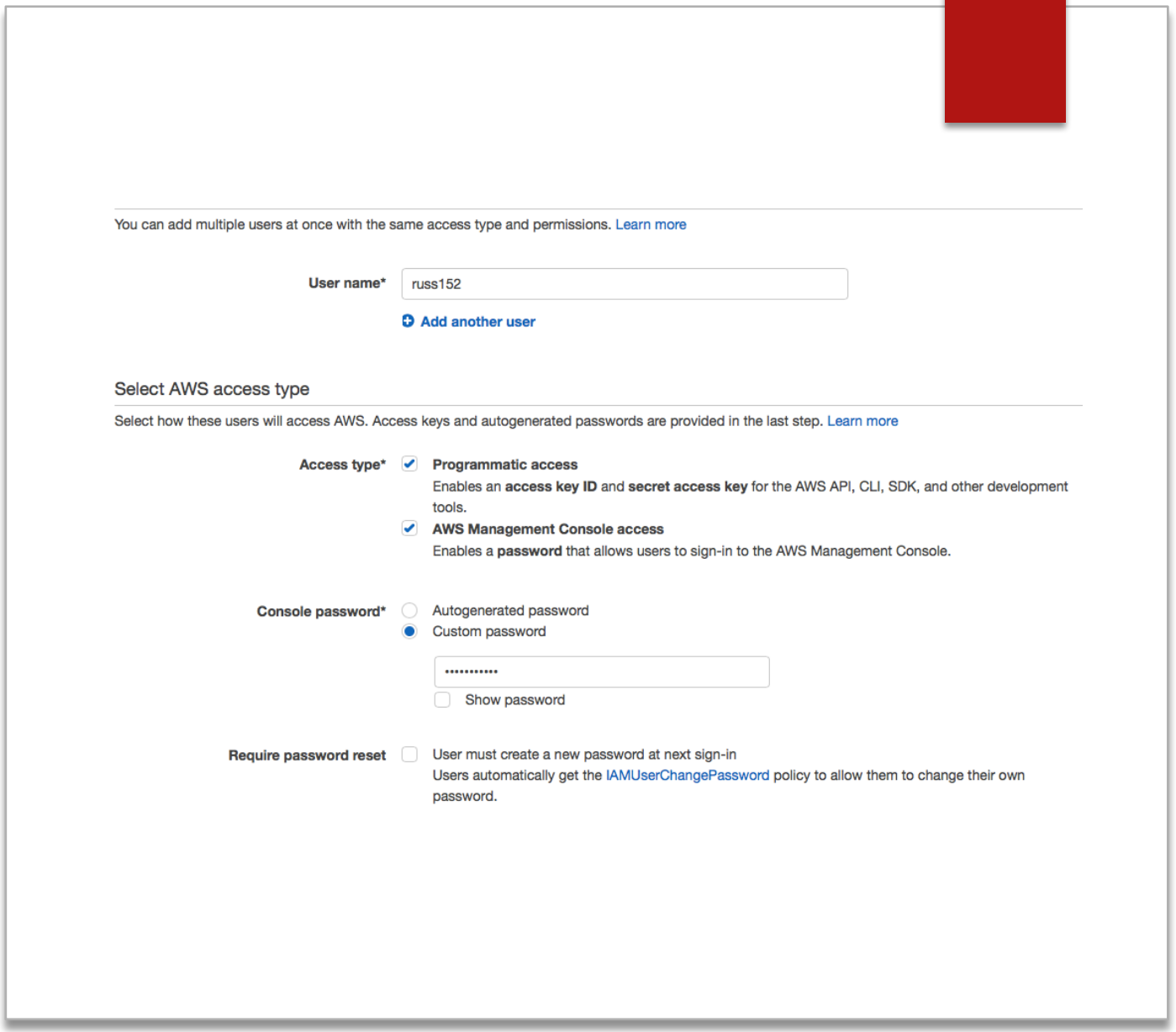
- Create an AWS account
- Afterwards you'll have access to the console, pictured right
- Type 'IAM' in the services box
- We need to our user keys for the next step





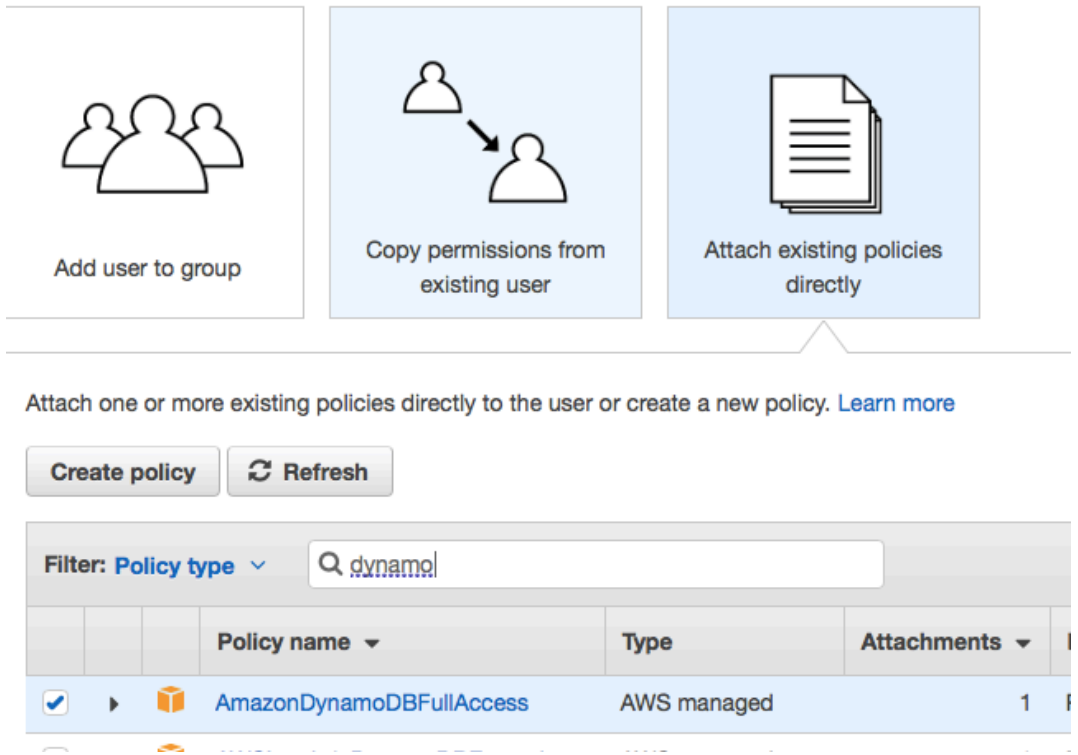
Create a User

- On the left, click create user
- Then At the top, Add user
- Fill in the prompts, make sure programmatic access is checked
- Add a password, uncheck reset requirement
- Click next



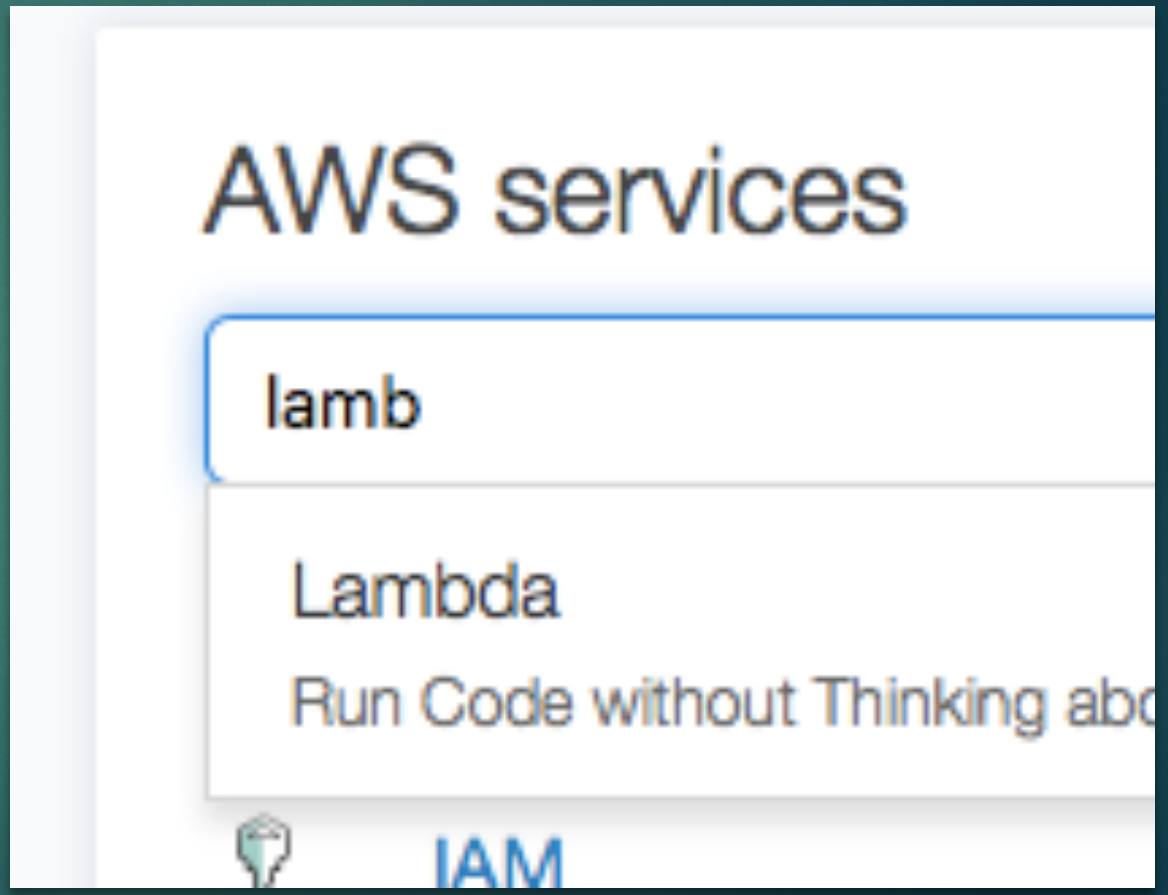
Permissions

- Next you can create a group, or you can just give the user the following permissions by clicking 'Attach existing policies directly'
  - AWSLambdaFullAccess
  - AmazonDynamoDBFullAccess
- Click next, then create user
- On the next screen click download csv
- We're now ready to setup the CLI
- But first, let's setup the other services



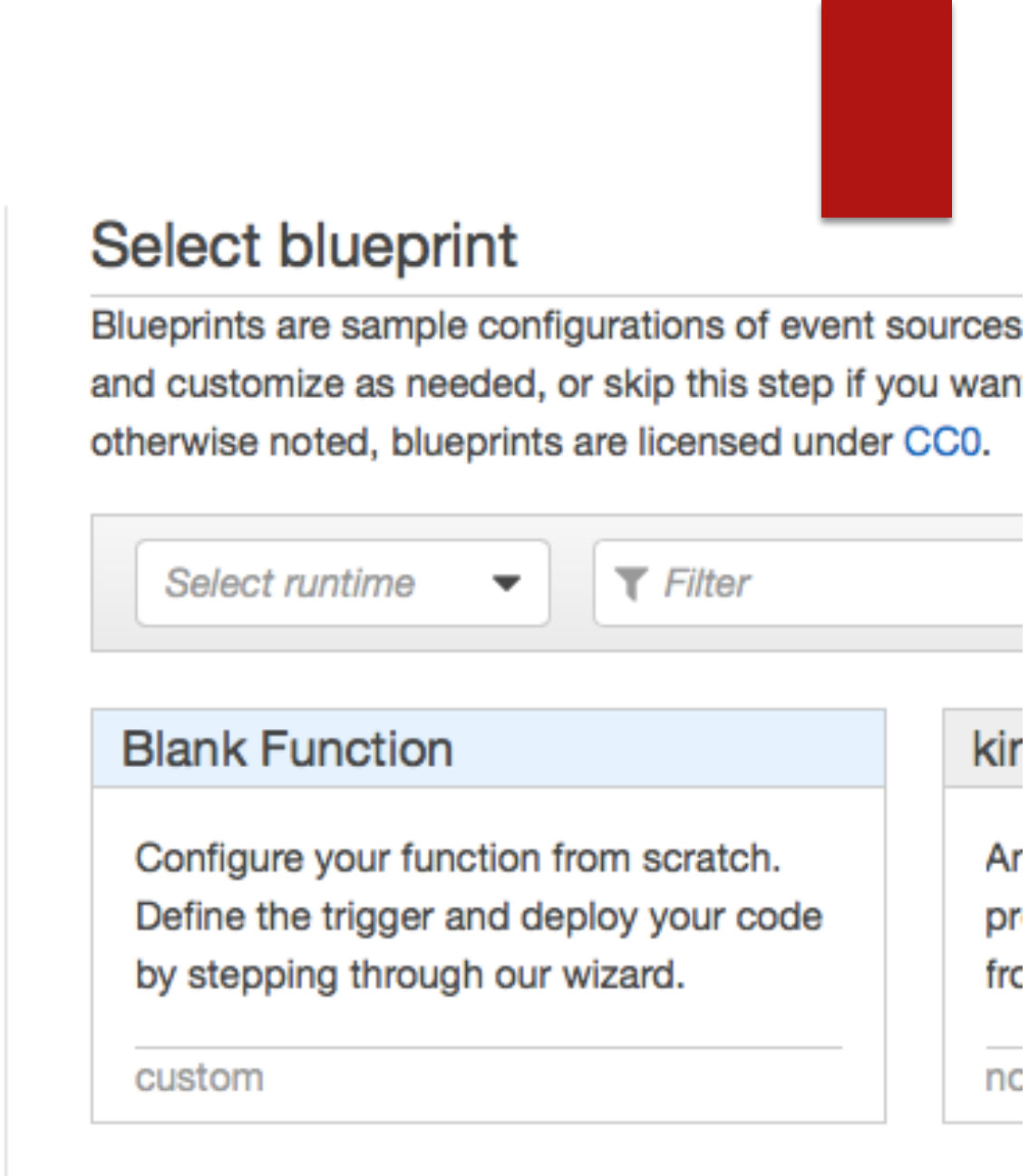
AWS Lambda

- AWS Lambda lets us run code, without running a server. When we deploy our app, it will look for an existing AWS function to attach to, so we need to create a blank one
- Go back to the console and type lambda
- It might say ‘Get Started’ or Create Function, regardless, click that



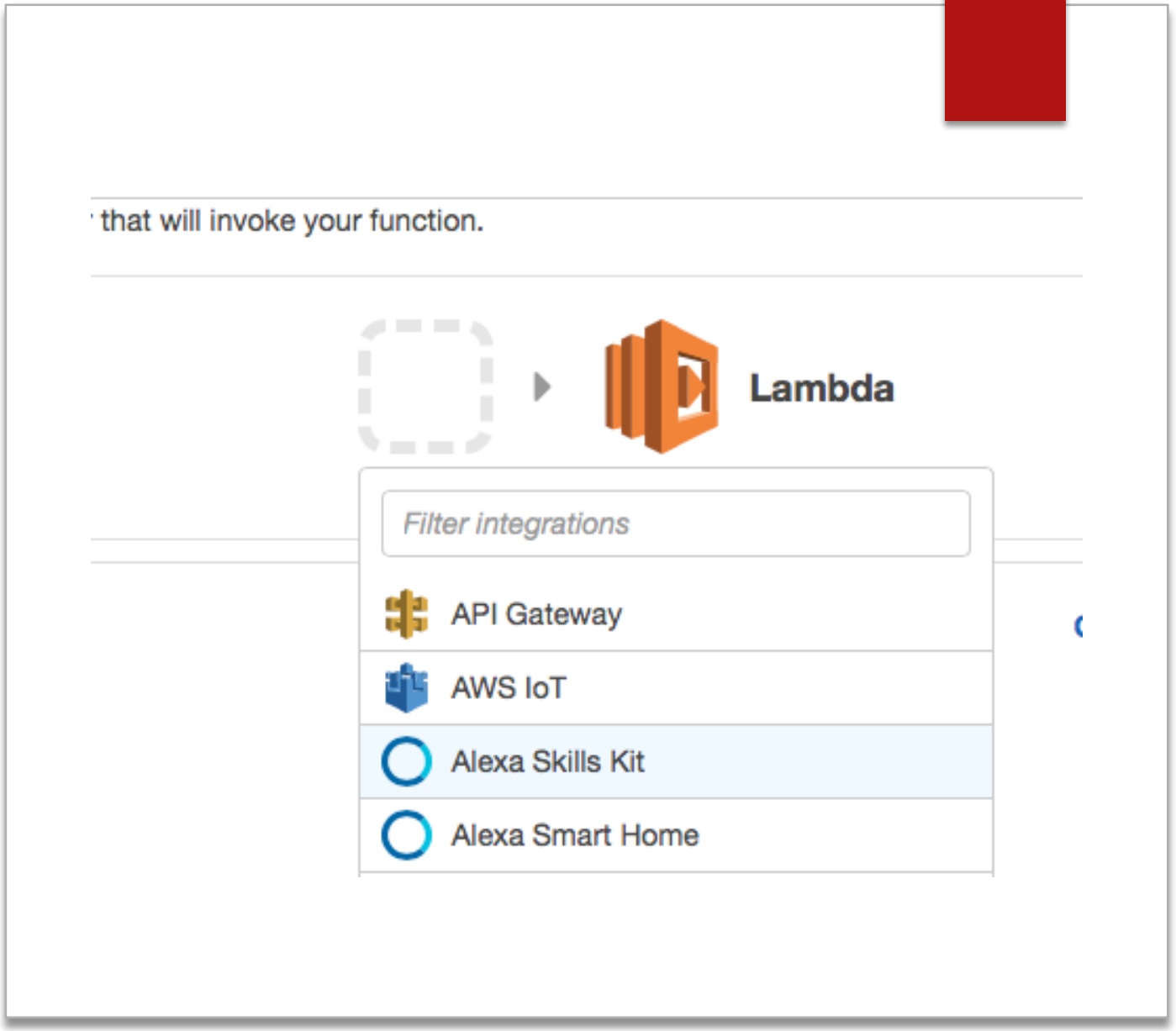
Create a Blank function

- Use the blank blueprint



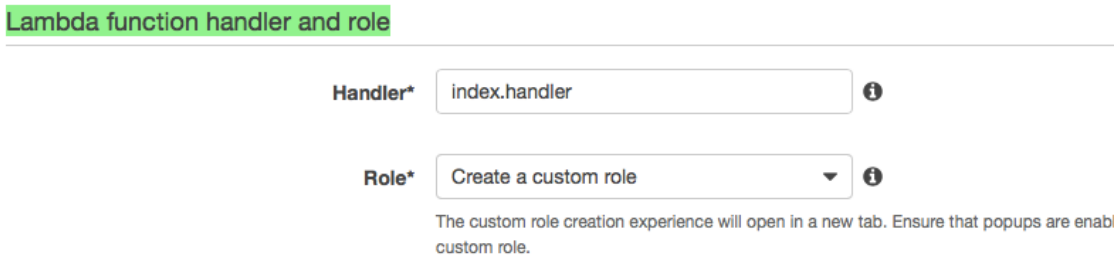
Set Triggers

- Click on the dotted-line box and select Alexa Skills Kit
- (Not Smart Home)
- Click next



Name and Role

- The Lambda function needs access to our DynamoDB (not created yet), so to do that we need to create a role.
- On the next screen, give the EXACT name:
  - **alexa-plex**
- Then scroll down to Lambda function handler and role section and select ‘Create a Custom Role’

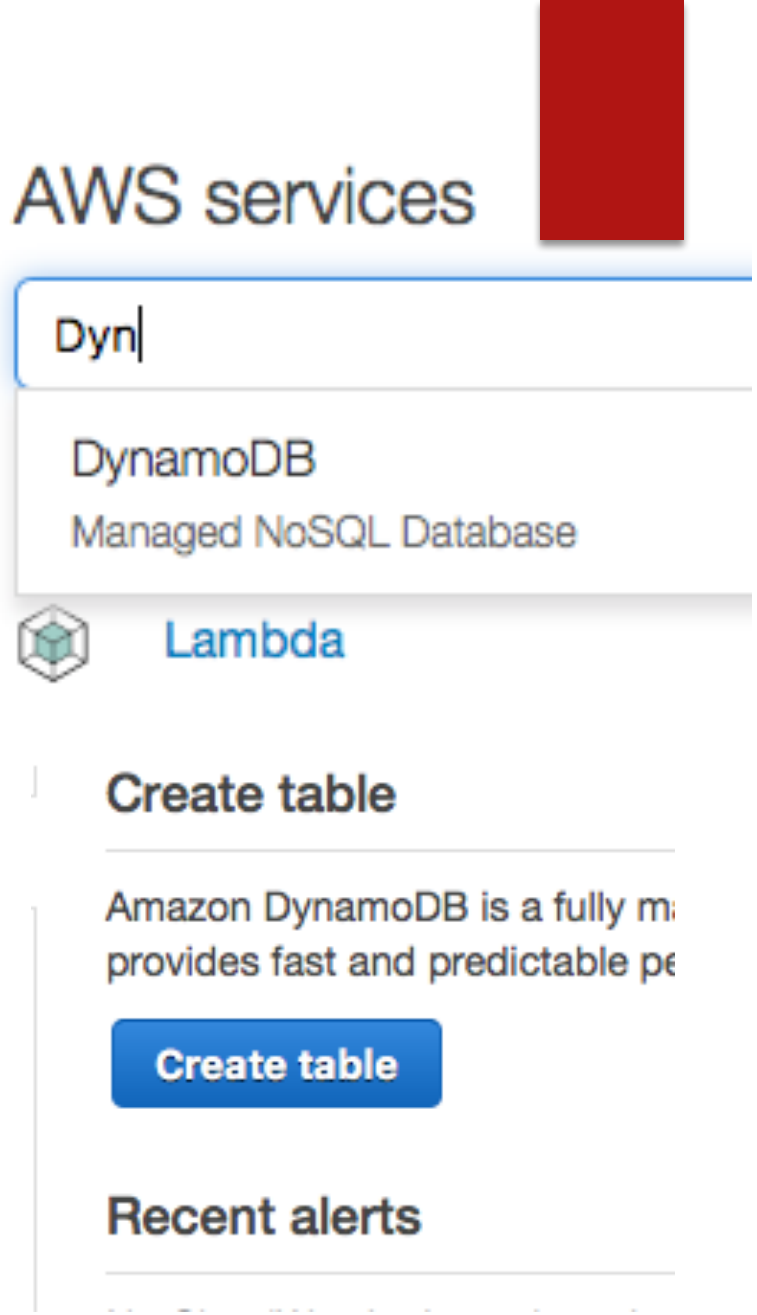


Creating A Role

- This will open another tab, give the role a name
- Click edit and paste the contents of docs/dynamoDB_role.txt inside the box
- Click Apply Role and you'll be taken back to the prior tab
- Click done, resolve any errors that come up
- So long as the name is alexa-plex, the role has DynamoDB access, and runtime is set to node.js, nothing else should really matter what the other settings are, especially the code, that will get overwritten completely

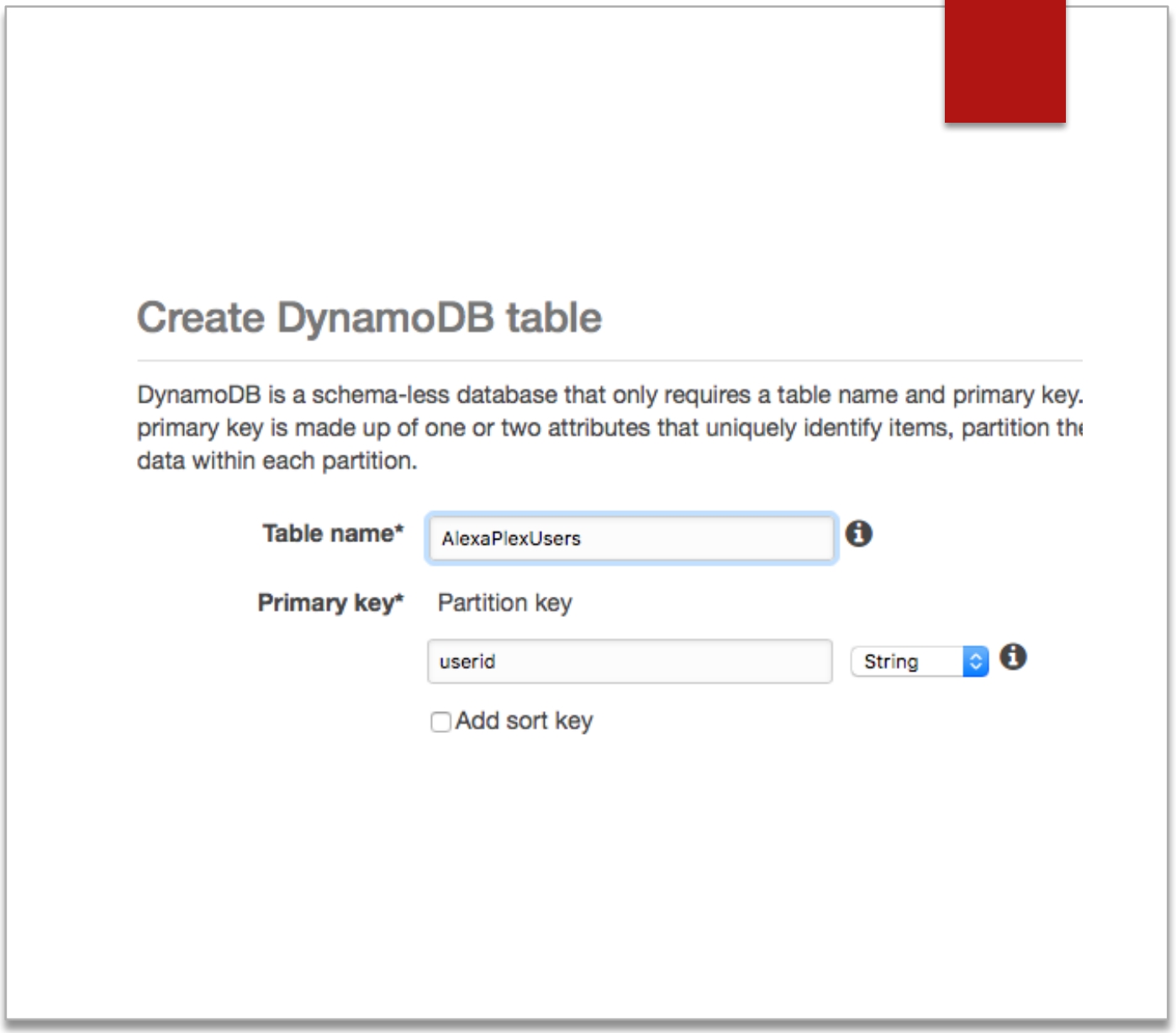
Setup DynamoDB

- Now we need to setup DynamoDB
- Type in Dynamo into the service box like before and select it
- Click either ‘Get Started’ or ‘Create Tables’



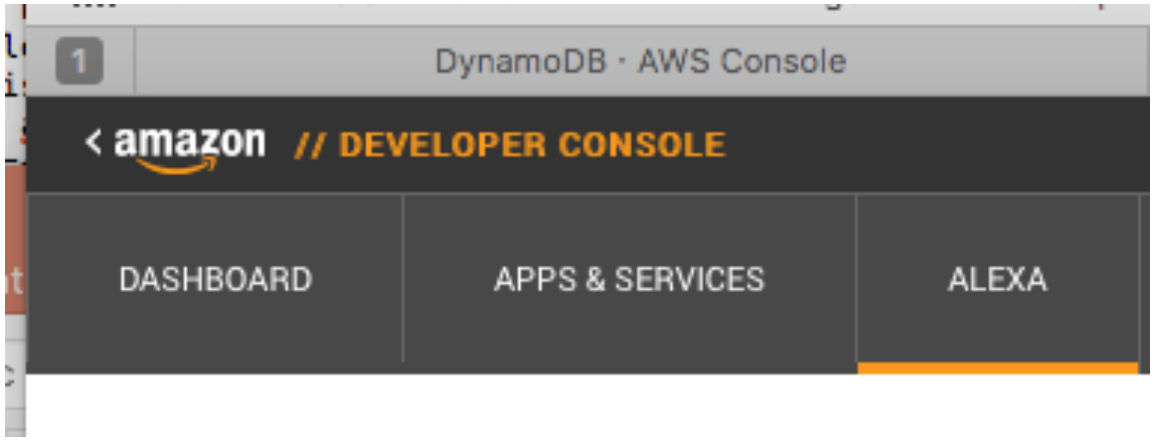
Create Table

- Fill in the prompts with
  - Table Name: AlexaPlexUsers
  - Primary Key: userid
- These have to be exact, the app will look for them by name
- Click create



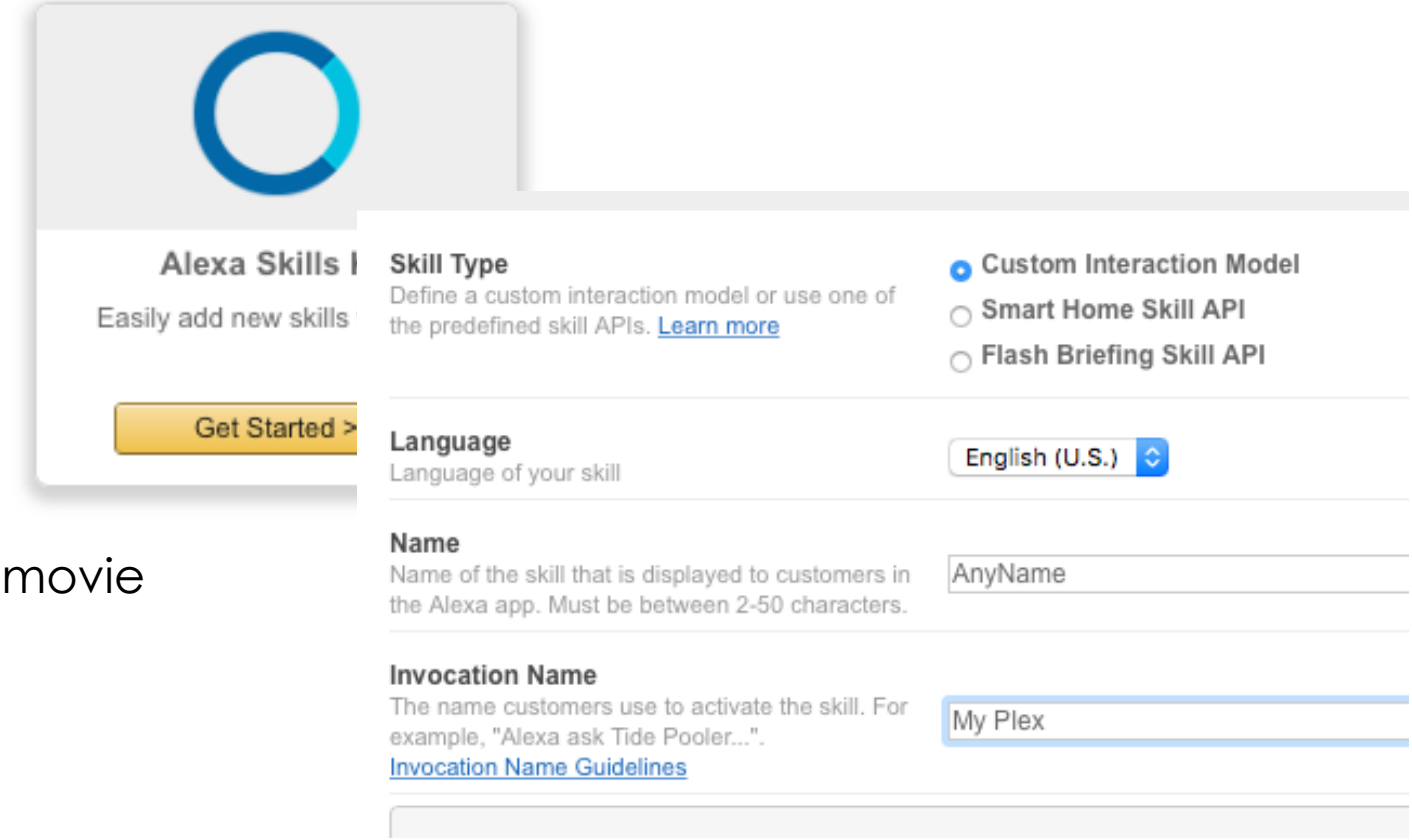
Alexa Skill

- It's time to create our Alexa Skill, this is what we will install on our Echo.
- Sign into the devloper console and click 'Alexa'



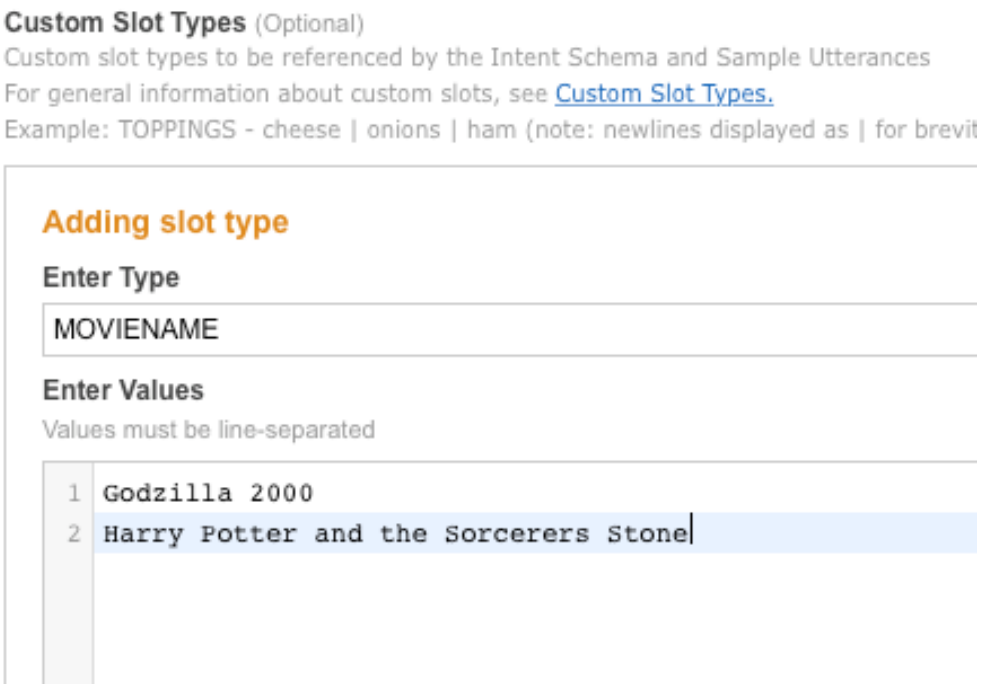
Alexa Skill Kit

- Click Alexa Skills Kit
- Then Add New skill
- Give your skill any name
- And for Invocation Name, use something that's easy to say
  - '**My Plex**' here will mean my commands are:
  - "Alexa, ask **My Plex** to play the movie Gone with the Wind"
- Click next



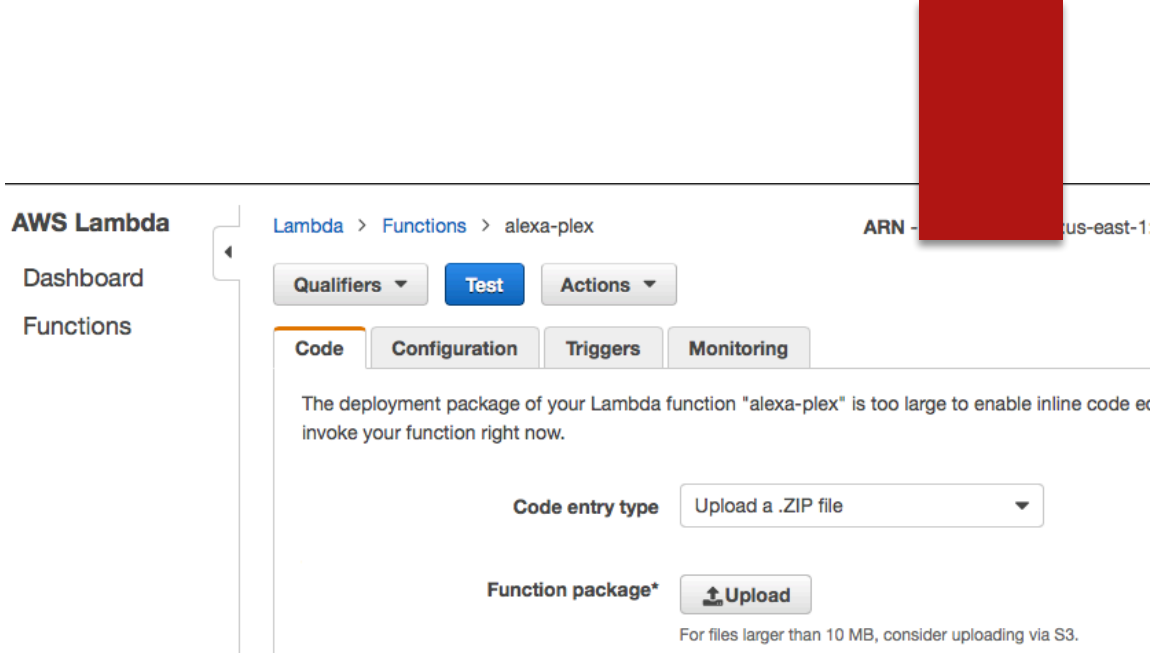
Configure Skill

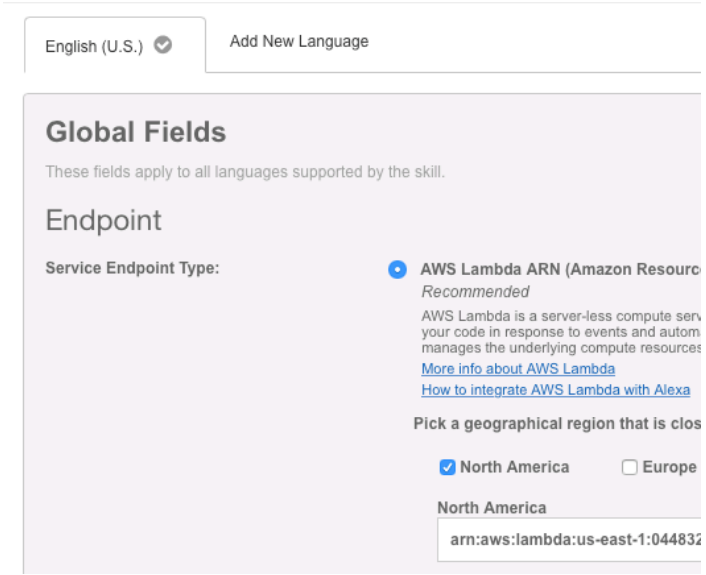
- For each prompt in the next screen, there is a corresponding file in the /ask_configuration folder
- Paste the intent-schema.json file into the first prompt
- Click the 'create slot' and enter slot name SHOWNAME, then paste in the list of shows you want alexa to be able to launch. SHOWNAME.txt is an example (note, there are no commas, each show is on a separate line)
- Repeat this process for MOVIENAME
  - I used plex2csv plex plugin to export my shows and movies, I'd suggest just typing in one show and movie getting it working with that one first
- Paste in sample-utterances.txt into the last prompt



Link to Lambda ARN

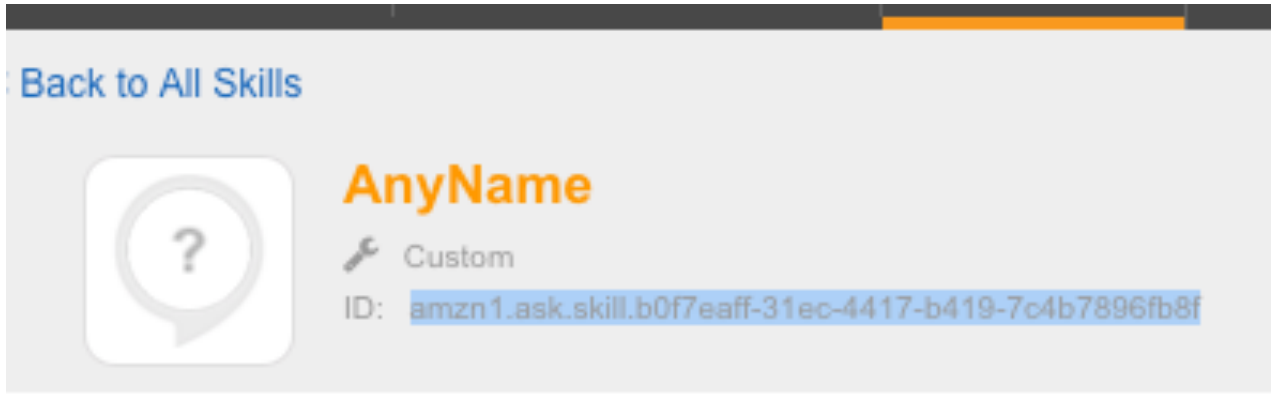
- In the next screen we need to paste the ARN of our lambda function
- Go back to the lambda panel and click on our alexa-plex function
- In the top right corner you'll see the ARN, copy the whole thing from arn:… to function:alexa-plex
- Paste this in to the Endpoint prompt





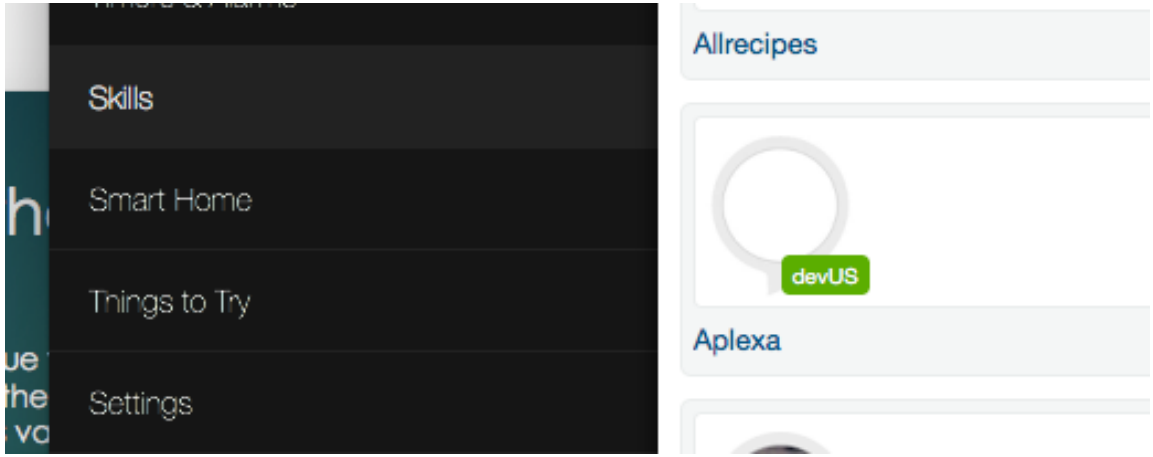
Get the Skill ID

- At the top of this page, you should see your Alexa skill id
- Copy this out, we'll need it for the next step, configuring the .env file



Finish the Skill setup

- Continue through the prompts, giving the minimum required to pass validations, at publishing prompt, click ‘SAVE’ not ‘SUBMIT for Certification’
- On the last page just click Save
- Your skill should now show up in your Skill list of your Alexa app
- We can’t test yet though, because our lambda function doesn't’t do anything yet



Install 7zip and configure .env

- You need to have 7zip installed and added to your path (check the box during installation) for the included deploy scripts to work
- Open the sample.env and edit the last 4 variables listed
  - APP_IDENTIFIER: This is just a random key you create so that plex doesn't spawn off a new app ever time you make a request
  - ALEXA_APP_ID: the id of the alexa skills app we created in the last step
  - AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY: downloaded in slide 8
- Save the edited file as just '.env' with no prefix name (take off 'sample' if your on mac, this might make the file hidden in finder

Setup your network

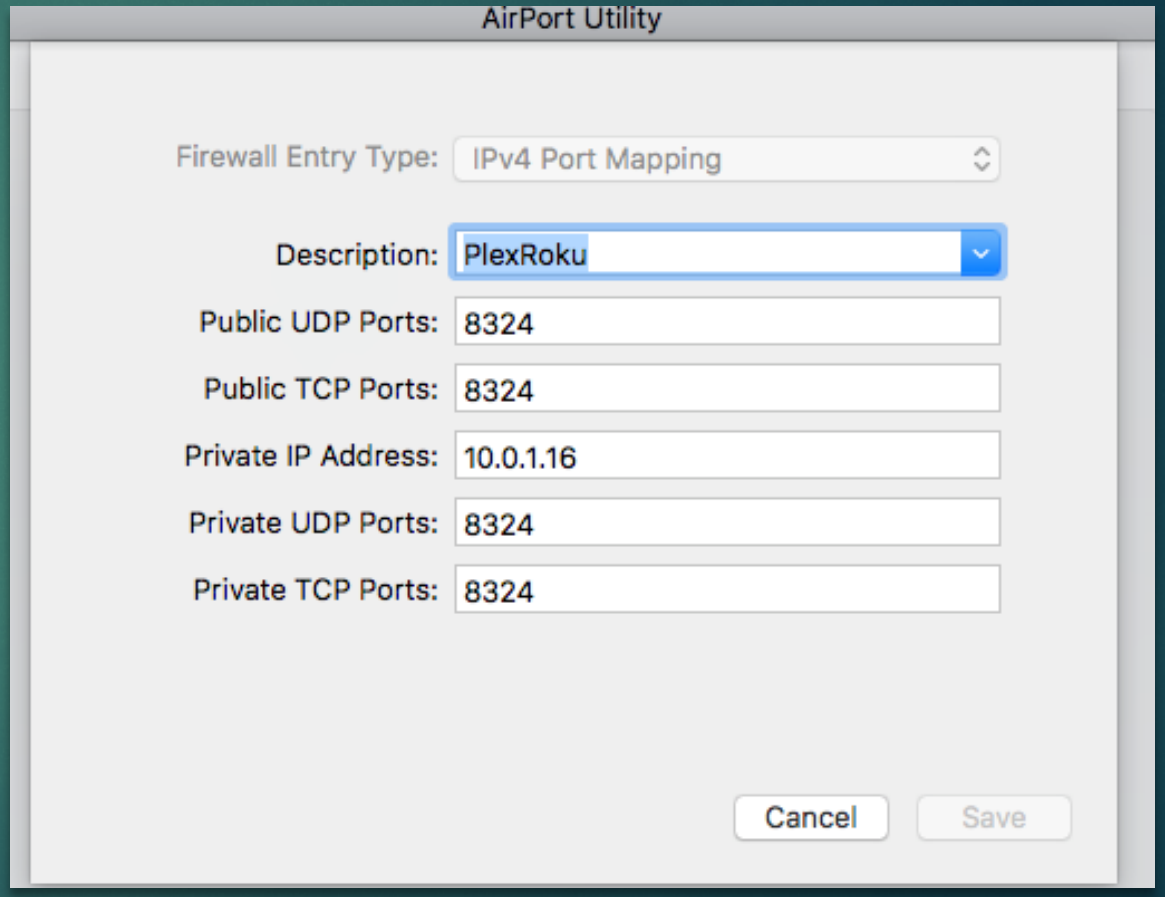
- Ok, before we can have our node function work, we need to have an endpoint for it to talk to
- The alexa-plex used to be able to initiate communication with the server and take over from there, but based on my research, plex changed the way you issue commands to clients, and now you have to talk to the client directly, so this is the work around I chose.
  - Alternative: run the node app on your own home computer/server and point the Alexa skill to it, then your node app can use local ip address

Getting your client information

- Sign into plex.tv
- Navigate to https://plex.tv/devices.xml
  - This will show you all the devices you have configured, your public ip, your clients ip, ports, etc
  - All we need is the publicAddress and connection.uri of the client we want
  - Notice it probably has a local address (10.0.1.16 in this example)
  - What we want is the port (8324) in this case

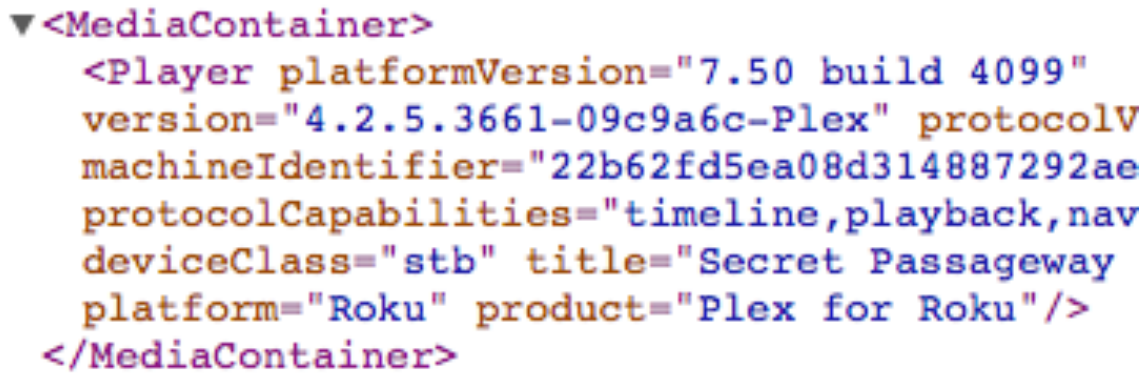
Setup your router

- What we want to be able to is take our publicAddress, add our port, and see the client information
- However by default, your router doesn't know we want this
- Google 'how to port forward <router name>' and follow the instructions
- In the example here, I would port forward my Airport Extreme's public port 8324 to private address 10.0.1.16 and private port 8234



Save and Test

- Verify your forwarding is working by navigating to
  - http://<your_public_ip>:<your_clients_port>/resources.xml
- If you see xml like the following, your good to go, but keep your IP and client port handy, we need to tell our app those values



Tell the app where our client is

- Get your IP address from google
  - Alternatively, create an A record for a subdomain on a website and point it to your ip address, like 'myhouse.mywebsite.com'
- Update the file /lib/plexutils.js and set the two variables at the top
  - var clientIP = '123.12.12.123';
  - var forwardedPort = '8234';
- For clientIP, this is your home ip
- For portfoward, pop in the port we just tested

Now we can setup our CLI

- Setup the CLI for windows or mac following these instructions:
  - http://docs.aws.amazon.com/cli/latest/userguide/installing.html#choosing-an-installation-method
- Once done, run in terminal/cmd
  - aws configure
- Paste in the Access Key and Secret found in the downloaded csv from slide 8
- For default region, use the closest to you in the list here. Someone in Florida would use us-east-1
- Leave the output format empty, just hit enter

Region Name	Region
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1

Install node and npm

- In order to deploy our app, we need Node.js on our workstation and NPM (the node package manager)
- On mac, you can just use homebrew for both
  - Brew install node
  - Brew install npm
- Everything else, even mac, can use the following link
  - https://nodejs.org/en/download/

Ready to deploy

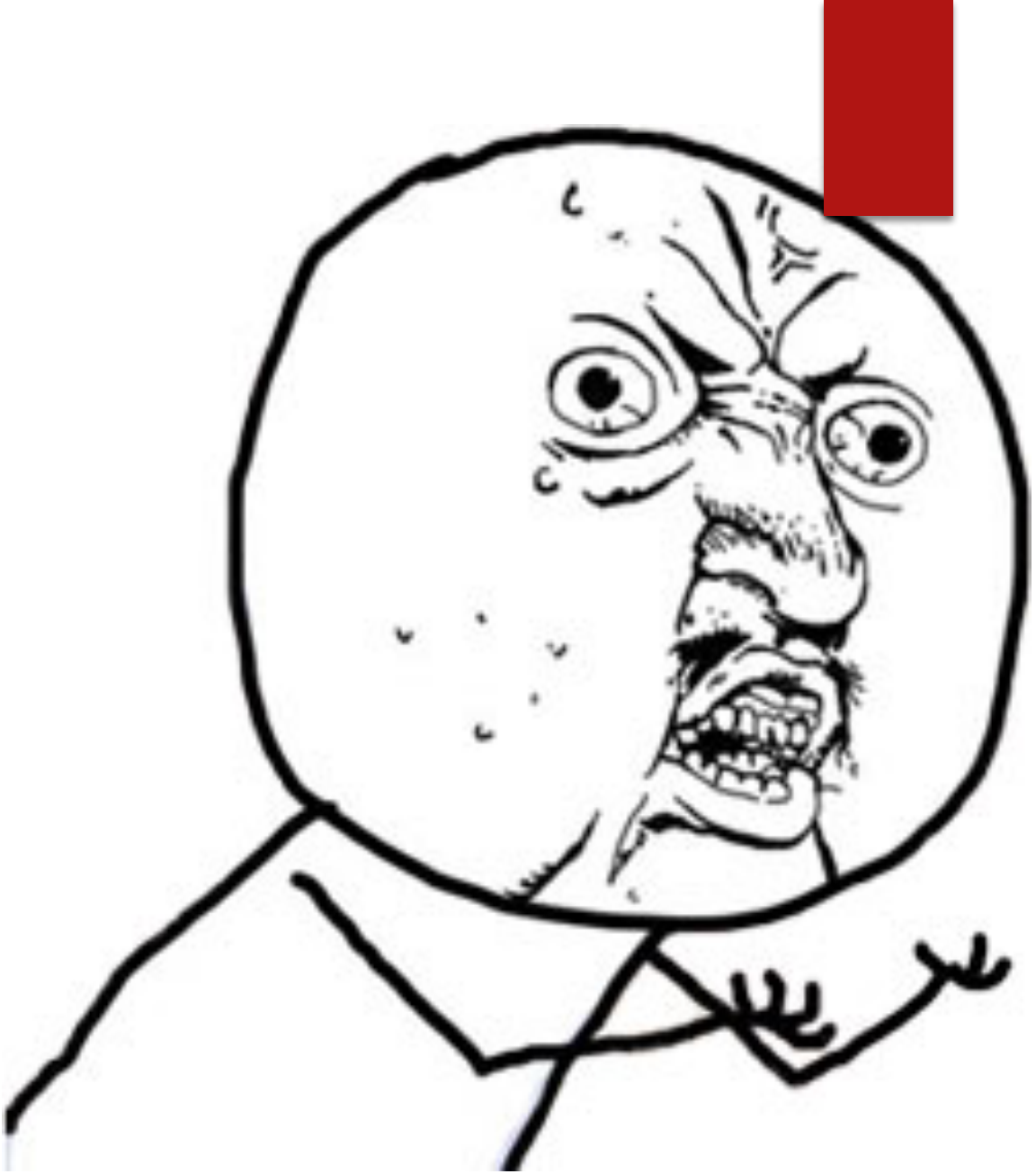
- 29 slides later, 1+ hour(s), and 3 headaches later, and we should now be able to deploy our alexa plex application
- The include bash scripts has been tested to work on my mac and, but you might have to tweak it in some way, and I haven't extensively tested the windows batch file
- Ultimately, the files will build us a node package (hence why we have to install node/npm) and create a zip file, then the script will auto upload the file to our lambda function.
- Alternatively, we can just upload the dist.zip file (after you've built it via npm) in the web UI, but where's the fun in that :P
- Run deploy_mac.sh if your on mac, or deploy.bat if you're on windows

Errors

- If you get errors here, read them. Either AWS sdk isn't working, or node can't compile the script, or the zip file isn't where it should be, or god just hates you. Scripts suck. Get the code compiled and move it to lambda, this doc assumes you can figure that out and we move onto the next step
- You might have to run it with sudo
- Try running the commands one at a time and see where it goes wrong
- These scripts didn't work for me out of the box, I've saved the changes necessary to get the running for Mr. Mee-agi-only

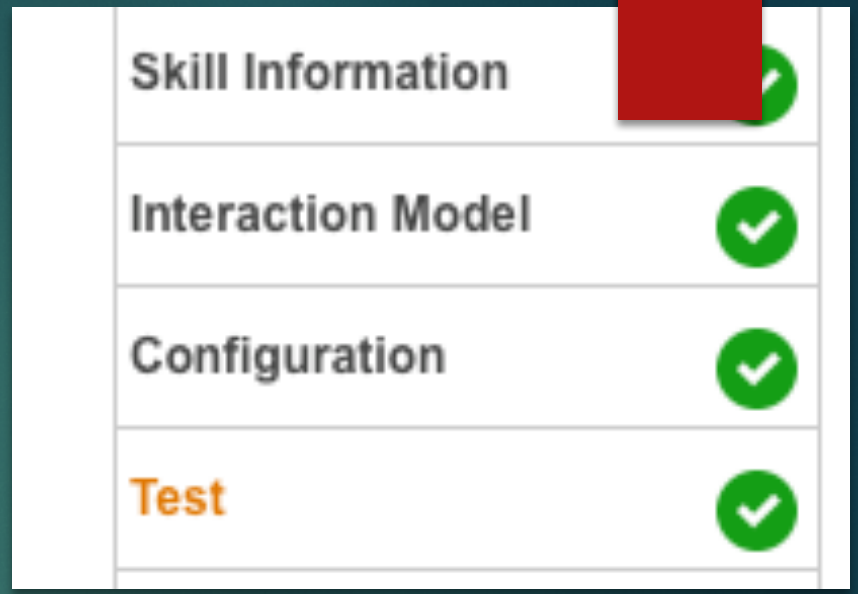
Sorry, I offer no support

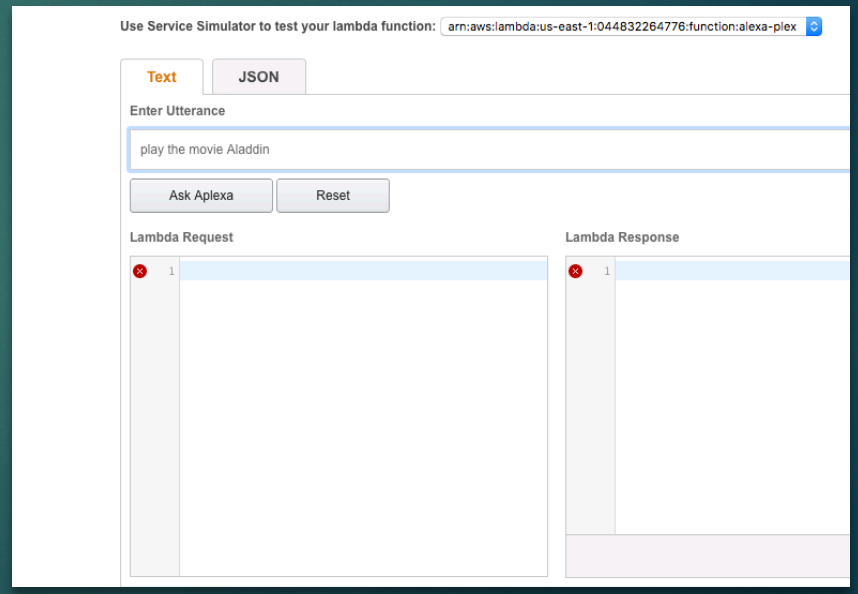
- Seriously, from here on out all sorts of things might go wrong, we've setup several services, and this requires all of them to be setup correctly for it to work at all
- I do have some testing options for you though



Alexa Skills Test Screen

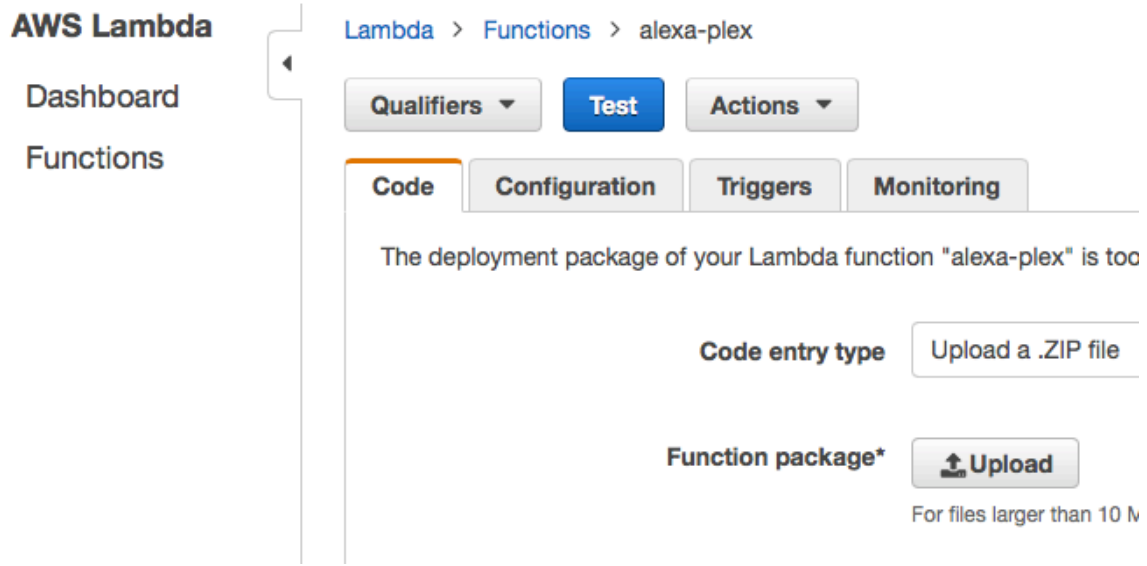
- In the alexa skill, there is a Test screen
- Scroll down and you can type in the alexa commands, follow the speech formats in the utterances file
- Press ‘ask alexa’ and this will build out the request alexa sends, and it will retun the lambda result, which will likely be ‘something went wrong’
- Useless, right?





But that's not all!

- Copy the json from the left side, and go to the Lambda test screen!
- Click the lambda function and you'll see the dashboard
- Click 'Actions' and 'Configure new test event
- Paste your json text
- Press save and test



Log Output

- Your node function will output some info near the middle of the page
- This is the same response sent back to alexa
- If you look in the bottom right, you'll see the actual log file
- This is still only a subset, click the 'Click here' link above it for even more, stored in cloudwatch!


```
      "content": "Playing Movie The Accountant"
    },
    "outputSpeech": {
      "type": "PlainText",
      "text": "",
      "ssml": "<speak>I'm sorry, but there was an error in the Plex Skill. The error has b
    }
  }
}
```

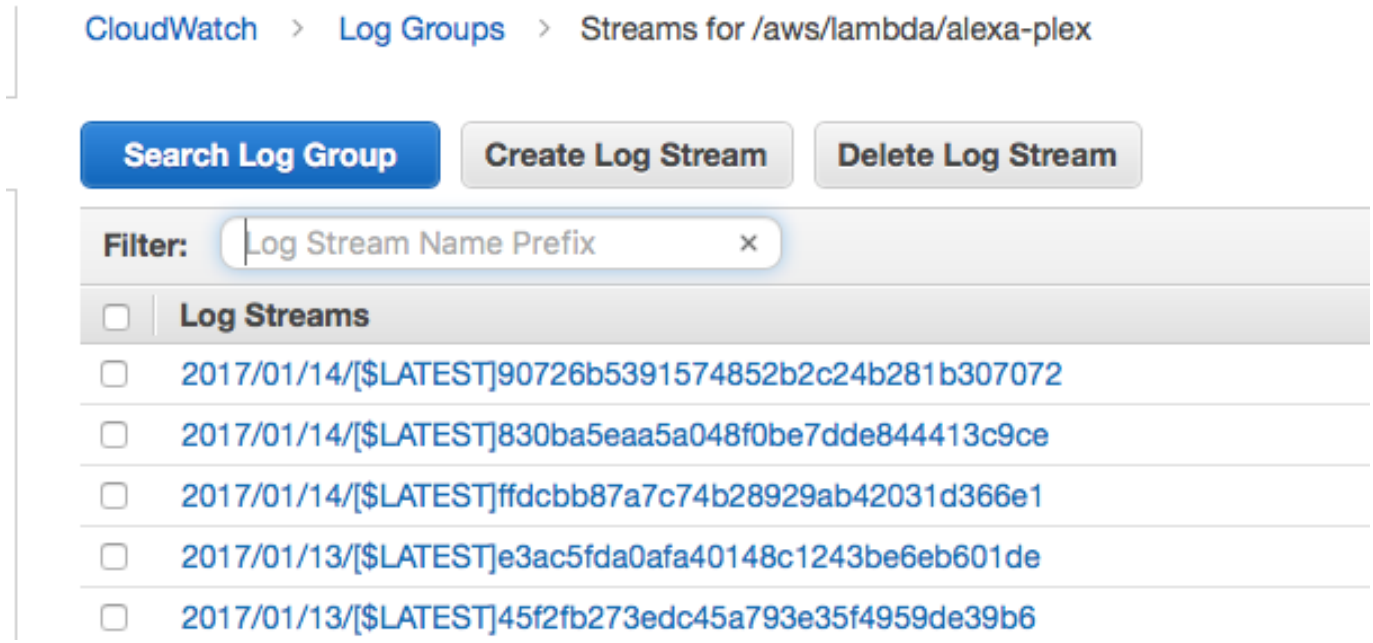

Log output

The area below shows the logging calls in your code. These corresp
log group corresponding to this Lambda function. Click here to view


```
        duration: 5478360,
        originallyAvailableAt: '1985-09-10',
        addedAt: 1482890507,
        updatedAt: 1482937520,
        ratingImage: 'imdb://image.rating',
        Media: [Object],
        Genre: [Object],
        DirEND RequestId: 74e4af18-daa2-11e6-ae30-4b93b53b3
REPORT RequestId: 74e4af18-daa2-11e6-ae30-4b93b53b3819  Du
```

Cloud Watch

- The next screen shows the log files generated
- Click the top one, it's the most recent
- Now you can see each line, scroll to the bottom for the error and see what you need to fix
- I did this for 12 days before it worked so good luck!



That’s it!

- Easy as pie…