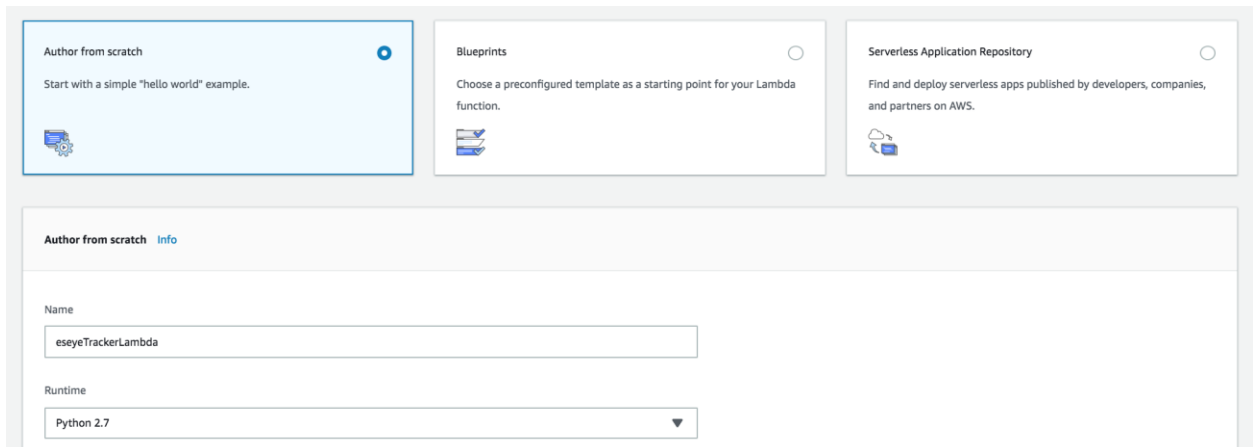


APN Blog

Coarse Location Tracking with AnyNet Secure SIM and AWS IoT

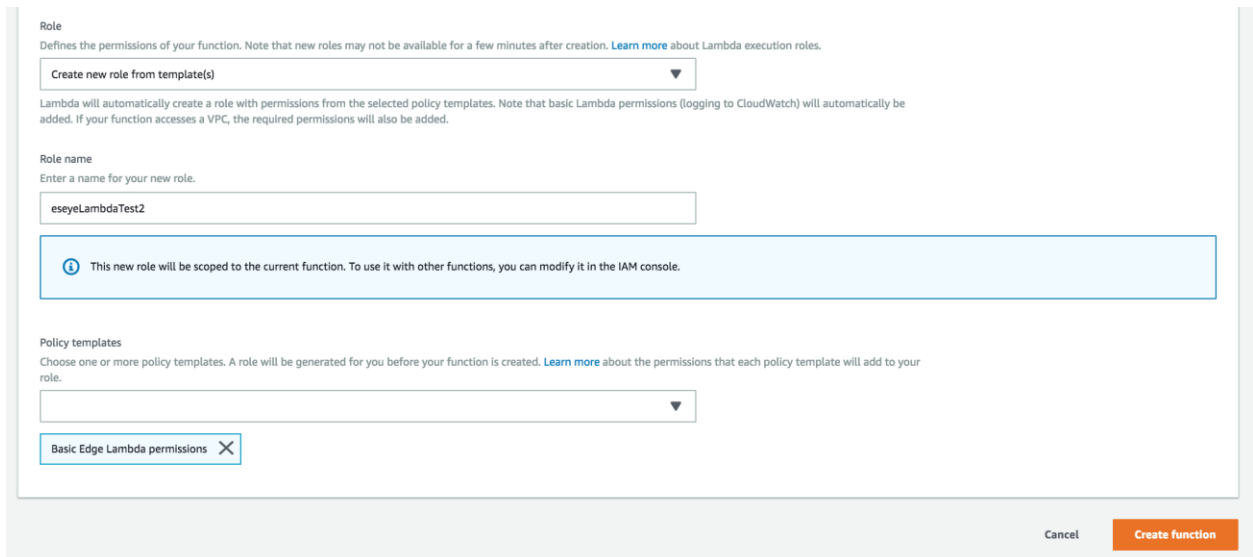
Appendix II – AWS Lambda Setup Instructions and Code

1. Create a new Lambda function from scratch; make sure to use Python 2.7



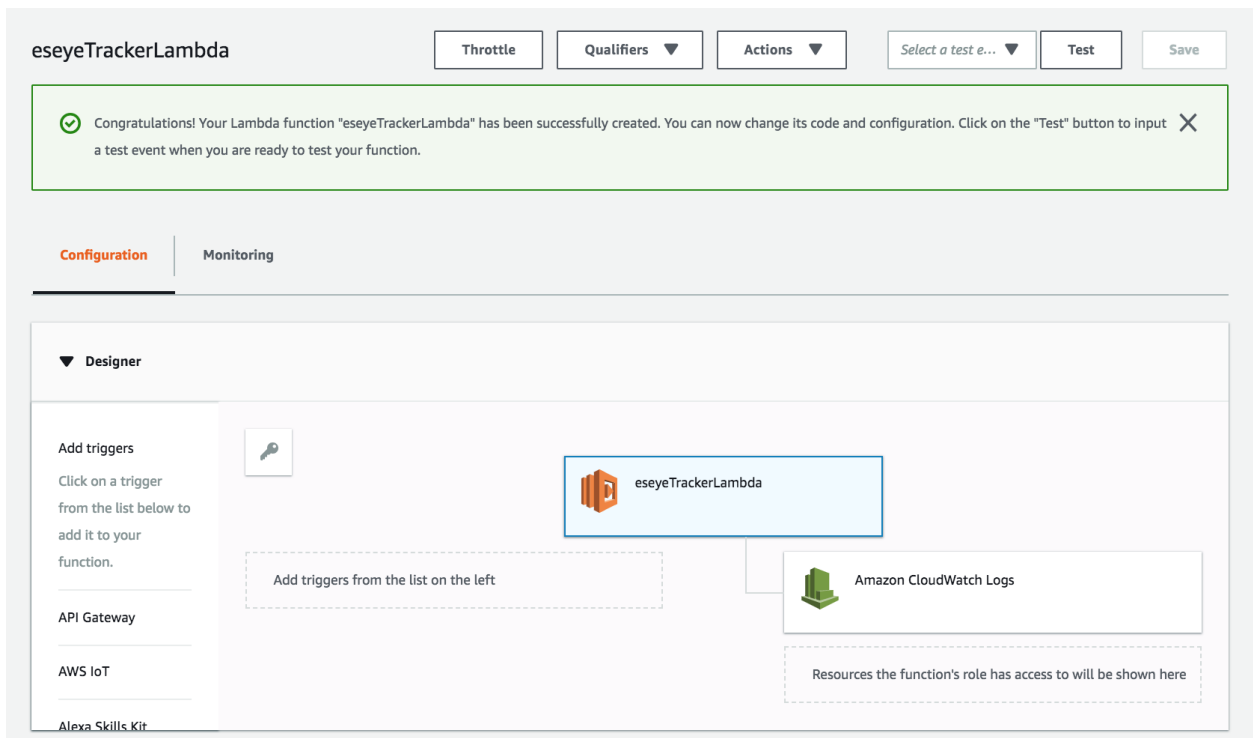
The screenshot shows the 'Author from scratch' step in the AWS Lambda console. It features three tabs: 'Author from scratch' (selected), 'Blueprints', and 'Serverless Application Repository'. The 'Author from scratch' tab contains a text input field for 'Name' with the value 'eseyeTrackerLambda' and a dropdown menu for 'Runtime' set to 'Python 2.7'.

2. For the Role, select "Create new role from template(s)" and then add Basic Edge Lambda permissions and click Create Function.



The screenshot shows the 'Role' step in the AWS Lambda console. It includes a dropdown menu for 'Role' set to 'Create new role from template(s)', a text input field for 'Role name' with the value 'eseyeLambdaTest2', and a dropdown menu for 'Policy templates' with 'Basic Edge Lambda permissions' selected. A blue information box states: 'This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.' At the bottom right, there are 'Cancel' and 'Create function' buttons.

- You can see that Lambda doesn't have the permission to use AWS IoT, so we'll have to go to the IAM console and modify the role.



The screenshot shows the AWS Lambda console for the function 'eseyeTrackerLambda'. At the top, there are buttons for 'Throttle', 'Qualifiers', 'Actions', 'Select a test e...', 'Test', and 'Save'. A green notification box states: 'Congratulations! Your Lambda function "eseyeTrackerLambda" has been successfully created. You can now change its code and configuration. Click on the "Test" button to input a test event when you are ready to test your function.' Below this, there are tabs for 'Configuration' and 'Monitoring'. The 'Designer' section is active, showing a visual flow diagram. On the left, under 'Add triggers', there are options for 'API Gateway', 'AWS IoT', and 'Alexa Skills Kit'. The main diagram shows a box for 'eseyeTrackerLambda' connected to 'Amazon CloudWatch Logs'. A dashed box below the logs says 'Resources the function's role has access to will be shown here'.

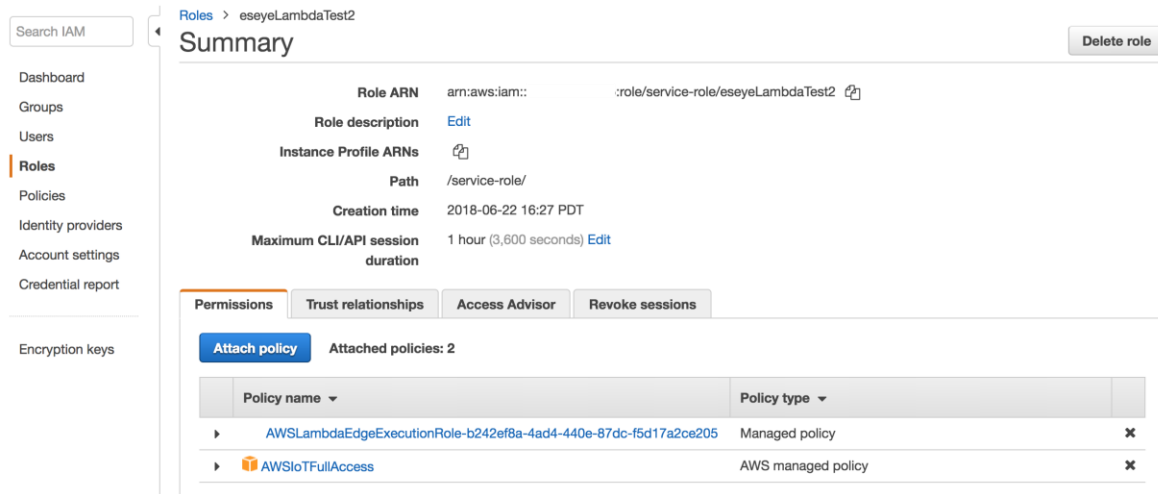
- Search for the newly created role; sometimes it takes time for it to show up.



The screenshot shows the AWS IAM console. On the left is a navigation menu with 'Roles' selected. The main area has a search bar with 'eseyeLambdaTest2' entered. Below the search bar, a table shows one result:

Role name	Description	Trusted entities
<input type="checkbox"/> eseyeLambdaTest2		AWS service: lambda and 1 more

5. Attach the policy and add 'AWSIoTFullAccess' for the demo.



Search IAM

Roles > eseyeLambdaTest2

Summary Delete role

Role ARN: arn:aws:iam:::role/service-role/eseyeLambdaTest2

Role description: [Edit](#)

Instance Profile ARNs: [Edit](#)

Path: /service-role/

Creation time: 2018-06-22 16:27 PDT

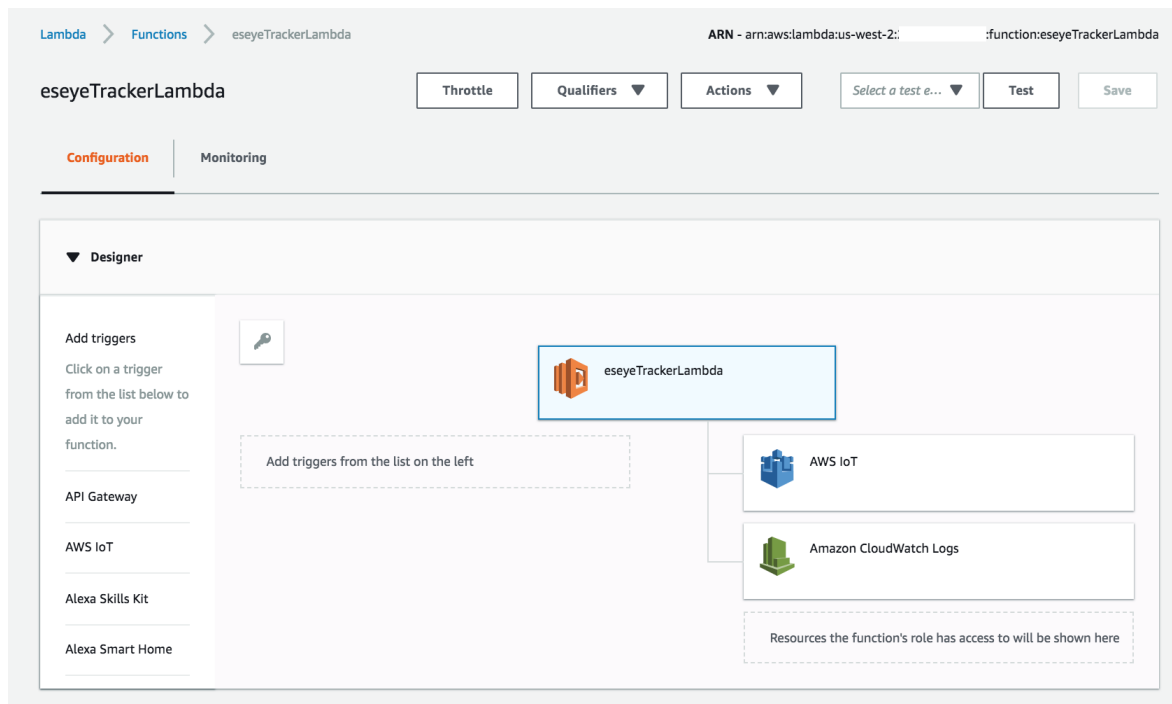
Maximum CLI/API session duration: 1 hour (3,600 seconds) [Edit](#)

Permissions | Trust relationships | Access Advisor | Revoke sessions

[Attach policy](#) Attached policies: 2

Policy name	Policy type	
AWSLambdaEdgeExecutionRole-b242ef8a-4ad4-440e-87dc-f5d17a2ce205	Managed policy	✕
AWSIoTFullAccess	AWS managed policy	✕

6. Your Lambda will have the right permissions now.



Lambda > Functions > eseyeTrackerLambda

ARN - arn:aws:lambda:us-west-2:::function:eseyeTrackerLambda

eseyeTrackerLambda Throttle Qualifiers Actions Select a test e... Test Save

Configuration | Monitoring

▼ Designer

Add triggers

Click on a trigger from the list below to add it to your function.

API Gateway

AWS IoT

Alexa Skills Kit

Alexa Smart Home

eseyeTrackerLambda

Add triggers from the list on the left

AWS IoT

Amazon CloudWatch Logs

Resources the function's role has access to will be shown here

7. Add the code below, and make sure to change the <Your SIM ID> to the ID of your Eseye SIM.

8. Python is indentation-specific, so make sure indentation doesn't change.

9. Change the Lambda handler to “lambda_function.setup_eseye_shadow” (format is fileName.functionName), and finally change the timeout to 5 minutes.

```

import boto3
import json
import datetime
import time
from botocore.exceptions import ClientError

clientIoT = boto3.client('iot')
clientIoTData = boto3.client('iot-data')

shadowPayload = "{\"state\" : {}}"

while True:
    #shadowReady variable is used to detect if eseye had updated the
    shadow or not
    # this variable stores the value of "status" element in the JSON
    shadowReady = "Pending"
    #print("step 1 - update attribute")

    # 1 - update attribute and set the ActionRequest Variable value as
    "obtainmetadata"
    responseThingUpdate = clientIoT.update_thing(
        thingName='EseyeDemoThing',
        thingTypeName='AnyNetThingType',
        attributePayload=
            {'attributes':
                {'ActionRequest': 'obtainmetadata', 'SimId' : '<Your
SIM ID>' }
            ,
            'merge' : True})
    #print("!!!!!!step 1 - attribute updated!!!!")
    time.sleep(20)

    # 2 - stay in this while loop till the status element in the shadow
    has value "provsioned"
    while (shadowReady != "Provisioned"):
        #print "step 2 - read shadow"

        # 2.1 - try reading the shadow value
        try:
            responseShadow = clientIoTData.get_thing_shadow(
                thingName='EseyeDemoThing'
            )

            streamingBody = responseShadow["payload"]
            jsonState = json.loads(streamingBody.read())
        except ClientError as e:
            #2.1.1 - if shadow document doesnt exist then create a new one
            before proceeding
            print("no shadow Doc hence creating new one")
            responseCreateNewDoc =
clientIoTData.update_thing_shadow(
                thingName = 'EseyeDemoThing',
                payload = shadowPayload

```

```

    )
    time.sleep(20)
#2.1.2 - once created read the shadow

    responseShadow = clientIoTData.get_thing_shadow(
        thingName='EseyeDemoThing'
    )

    streamingBody = responseShadow["payload"]
    jsonState = json.loads(streamingBody.read())
#print(jsonState)

    try:
        #print("reading JSON to check for status")
        # 3 - from the shadow JSON extract the value for status element
        shadowReady =
jsonState["state"]["reported"]["anynet"]["status"]
    except KeyError, e:
        print("cant find the key will try again in 20 sec")

        #print(shadowReady)

        # 4 - if the status element value is "provisioned" then extract
lat,lon and print the values
        if (shadowReady == "Provisioned") :
            lat =
jsonState["state"]["reported"]["anynet"]["metadata"]["location"]["lat"]
            lon =
jsonState["state"]["reported"]["anynet"]["metadata"]["location"]["lon"]
            print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M") +
"-- " + lat + "," + lon)
            #print("!!!!step 2 - shadow read!!!!")

            time.sleep(20)

        # 5 - reset the shadow status value to pending for next time
shadowReady = "Pending"
        #print("step 3 - delete shadow")

        # 6 - delete the shadow document
response_delShadow = clientIoTData.delete_thing_shadow(
    thingName='EseyeDemoThing'
)

        #print("!!!step 3 - shadow delete complete!!!")

        time.sleep(20)
        #print("step 4 - updated shadow")

        # 7 - create a new shadow document
responseCreateNewDoc = clientIoTData.update_thing_shadow(
    thingName = 'EseyeDemoThing',
    payload = shadowPayload
)
        #print("!!!step 4 - shadow update completed!!!!")
        time.sleep(1)

```

10. Given that Lambda can run for a maximum of 5 minutes, it's not possible to run an infinite loop. Instead, we'll use an Amazon CloudWatch trigger to kick-off a Lambda function every 3 minutes.

Rule
Pick an existing rule, or create a new one.

Create a new rule ▼

Select or create a new rule

Rule name*
Enter a name to uniquely identify your rule.

eseyeCloudwatchEventTemp2

Rule description
Provide an optional description for your rule.

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.

Event pattern

Schedule expression

Schedule expression*
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

rate(3 minutes)

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon CloudWatch Events to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Enable trigger