AWS IoT QuickStart

Kirk S. Kalvar

5/12/2017

# Create AWS IoT Device

1. Create <myiotdevice> device in AWS IoT/Registry/Things
2. Update Android application CUSTOMER\_SPECIFIC\_ENDPOINT with HTTPS endpoint from Registry/Things/<myiotdevice>/Interact

# Create AWS Cognito Federated Identity

1. Cognito/ Manage Federated Identities
2. Identity pool name: <myiotdevice\_identity>
3. Unauthenticated identities: Enable access to unauthenticated identities
4. Click on “Create Pool”
5. Click on “Allow”
6. Update Android application COGNITO\_POOL\_ID with highlighted Identity Pool ID in “Get AWS Credentials” section

# Update AWS IAM Cognito Roles

1. Update IAM/Roles/Cognito\_<myiotdevice\_identity>Unauth\_Role
2. Attach Policy/AWSIoTFullAccess
3. Click on “Attach Policy”

# Test <myiotdevice> Connectivity

1. Check activity from AWS IoT/Registry/Things/<myiotdevice>/Activity
2. Send a few packets through to AWS IoT via Mobile Application

# Create AWS IoT Device Rule

1. Create <myiotdevice\_rule> rule in AWS IoT/Rules
2. Attribute: \*
3. Topic filter: $aws/things/<myiotdevice>/shadow/update/accepted

# Create AWS IoT Device Rule S3 Bucket

1. Click on “Add Action”
2. Select “Store messages in an Amazon S3 bucket”
3. Click on “Configure action”
4. Click on “Create a new resource” # this will open a new browser tab
5. Clock on “Create bucket”
6. Bucket Name: <myiotdevice-s3-bucket> # must be lowercase, only allows underscores
7. Next
8. Next
9. Next
10. Click on “Create bucket”
11. Back on AWS IoT “Configure action” browser tab click “refresh” icon between S3 bucket and “Create a new resource”
12. Using S3 bucket pull down, Select <myiotdevice\_s3\_bucket>
13. Key: ${topic()}/${timestamp()}.txt
14. Click on “Create a new role”
15. IAM Role name: <myiotdevice\_role\_s3>
16. Click on “Create a new role”
17. Using IAM role name pull down, Select <myiotdevice\_role\_s3>
18. Click on “Update role”
19. Click on “Add action”
20. Click on “Create rule”

# Update AWS IAM S3 Roles

1. Update IAM/Roles/<myiotdevice\_role\_s3>
2. Attach Policy/AWSIoTFullAccess

# Test <myiotdevice\_rule> S3 bucket

# Create AWS IoT Device Rule Lambda

1. AWS IoT/Rules/<myiotdevice>
2. Click on “Configure action”
3. Select “Invoke a Lambda function passing the message data”
4. Click on “Create a new resource” # will take you to new browser tab
5. Select “Blank Function”
6. Click on dotted box and Select “AWS IoT”
7. IoT Type Pull down: Custom IoT Rule
8. Rule name: <myiotdevice\_rule> # enter complete rule name before fields will auto populate
9. Enable trigger: click on checkbox to enable
10. Next
11. Name: <myiotdevice\_lambda>
12. Runtime: Node.js 6.10
13. Code entry type: Edit code inline
14. Cut and Paste lambda.js from project into code pane.
15. Update lambda.js variable endpointAddress: CUSTOMER\_SPECIFIC\_ENDPOINT
16. Update lambda.js variable thingName: <myiotdevice>
17. Lambda function handler and role: Role: Create a custom role # will take you to a new browser tab
18. Role name: <myiotdevice\_role\_lambda>
19. Allow
20. Back on AWS IoT “Configure function” browser tab
21. Advanced Settings: Timeout: 10 secs
22. Next
23. Click on “Create function”
24. Back on AWS IoT “Configure action” browser tab click “refresh” icon between “Function name” and “Create a new resource”
25. Using “Function name” pull down, Select < myiotdevice\_role\_lambda >
26. Click on “Add action”

# Update AWS IAM Lambda Roles

1. Update IAM/Roles/< myiotdevice\_role\_lambda >
2. Attach Policy/AWSIoTFullAccess

# Test <myiotdevice\_rule> Lambda function

1. Using Android Application Open Window and then set Temperature > 72 Degrees

# Create AWS IoT Device Rule Elasticsearch

1. AWS IoT/Rules/<myiotdevice\_rule>
2. Click on “Add action”
3. Select “Send messages to the Amazon Elasticsearch Service”
4. Click on “Configure action”
5. Click on “Create new domain”
6. Elasticsearch domain name: <myiotdevice-es> # must be lowercase, no underscores allowed
7. Next
8. Instance count: 1
9. Instance type: t2.small.elasticsearch
10. Next
11. Set up access policy: Select a template: allow open access to the domain
12. Confirm “Policy Risk Statement”
13. Next
14. Click on “Confirm”
15. Domain status: loading # allow completion before proceeding; note it takes a few minutes!
16. Back on AWS IoT “Configure action” browser tab click “refresh” between Domain name and “Create a new resource”
17. Using Domain name pull down, Select <myiotdevice-es>
18. Be sure Endpoint is pre-populated before continuing
19. Id: ${newuuid()}
20. Index: <myiotdevices>
21. Type: < myiotdevices >
22. Click on “Create a new role”
23. Enter <myiotdevice\_role\_es>
24. Enter Create
25. Using IAM role name pull down, Select <myiotdevice\_role\_es>
26. Click on “Update role”
27. Click on “Add action”

# Update AWS IAM ES Roles

1. Update IAM/Roles/<myiotdevice-es>
2. Attach Policy/AWSIoTFullAccess

# Access Elasticsearch dashboard

1. AWS Elasticsearch/Dashboards
2. Click on <myiotdevice-es>
3. Click on Kibana endpoint

# Test <myiotdevice-es> Elasticsearch

1. Send a few packets through to AWS IoT via the Mobile Application so Elasticsearch has some records to work with.

# Configure Kabana Index Pattern

1. Index name or pattern: \*
2. Uncheck “Index contains time-based events “
3. Click on “Create”
4. Set filter: state
5. Click on Star graphic to set as default index

# Configure Kabana Dashboard

Kabana is a little more involved to configure so I’ll let you figure out the basics. AWS has online documentation as well for Kabana. I’ve included snapshots of the configuration values to visualize an Open Window bar chart.



