

Smart Meeting Room

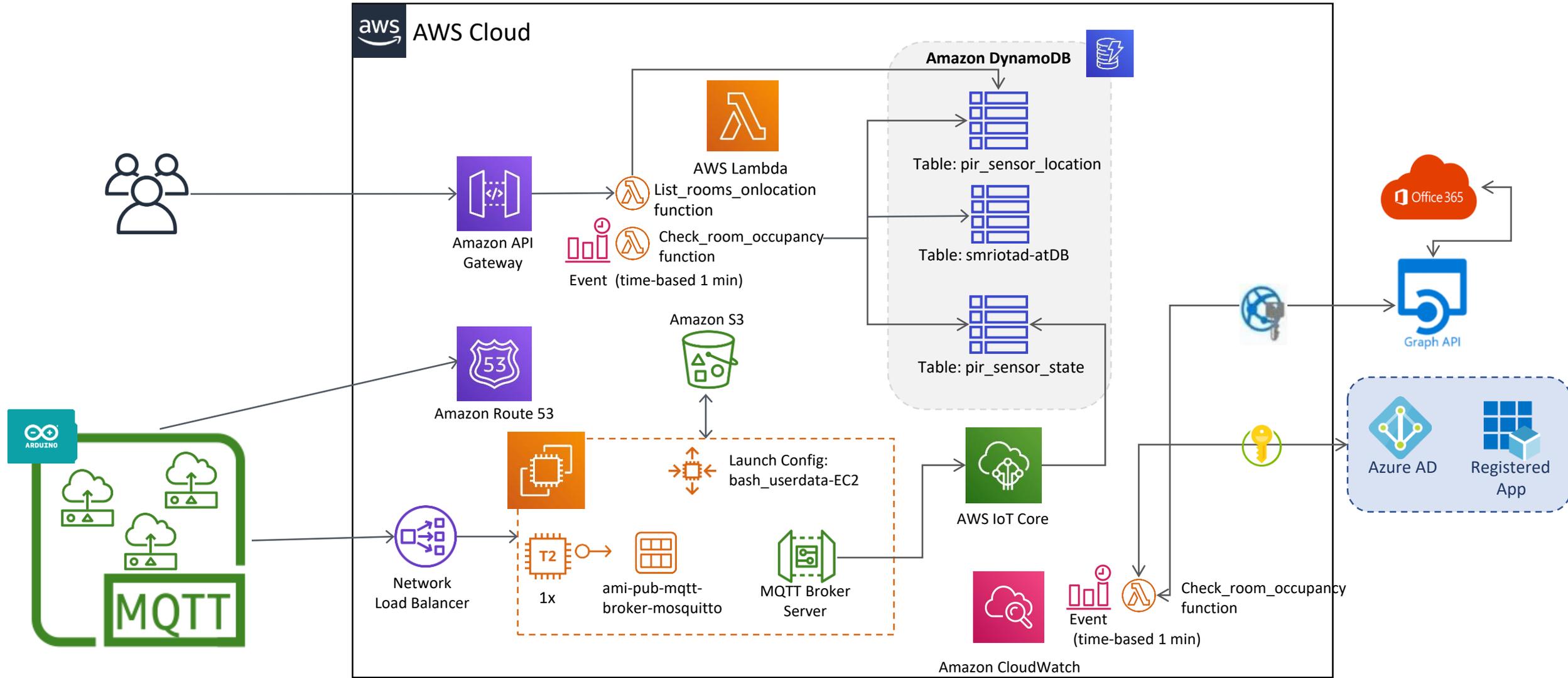
Bringing efficiency to space utilization at most end-user location
convenience

Background and Goal

- Meeting rooms within companies and offices premises are often under utilization at an expensive building cost;
- Majority of companies utilizes MS Azure AD as company Active Directory for corporate user identities handlings, MS Outlook as email and calendar platforms, and where meeting rooms are seen as resources type of users, owning mailboxes and been handled/booked via MS Outlook (administrated via o365 admin console);
- To improve meeting rooms utilization and provide enhanced location based meeting rooms availability, this project integrates motion sensors into meeting rooms, where motion detection (PIR) and location (Google Wi-Fi Geolocation API) telemetric info is provided to backend non-SQL database (AWS DynamoDB) and applications (AWS Lambda), where this info is transformed into meaningful data for automatic meeting rooms reservation released due no activity/usage and provide to end-users closest available meeting rooms for utilization, based on their location.
- Sensors telemetry data is sent using MQTT (Message Queuing Telemetry Transport) protocol towards a MQTT broker (Mosquitto Server deployed in a AWS EC2 and S3), which in turns is integrated with AWS IoT (using TLS authentication) and backend non-SQL DB as mentioned above.
- For prototyping purposes, client app is developed using Python, making use of Google Wi-Fi based Geolocation and AWS API Gateway, also integrated with backend AWS Lambda applications for meeting room lookup purposes.

This is a work in progress with final goal to have a full programmatically “one-click” server-side architecture using AWS Cloud Formation for MQTT Broker / IoT Core / Cloud Watch and back-end Lambda functions. I will update this project continuously. Please check the last page for Project Roadmap.

E2E Architecture



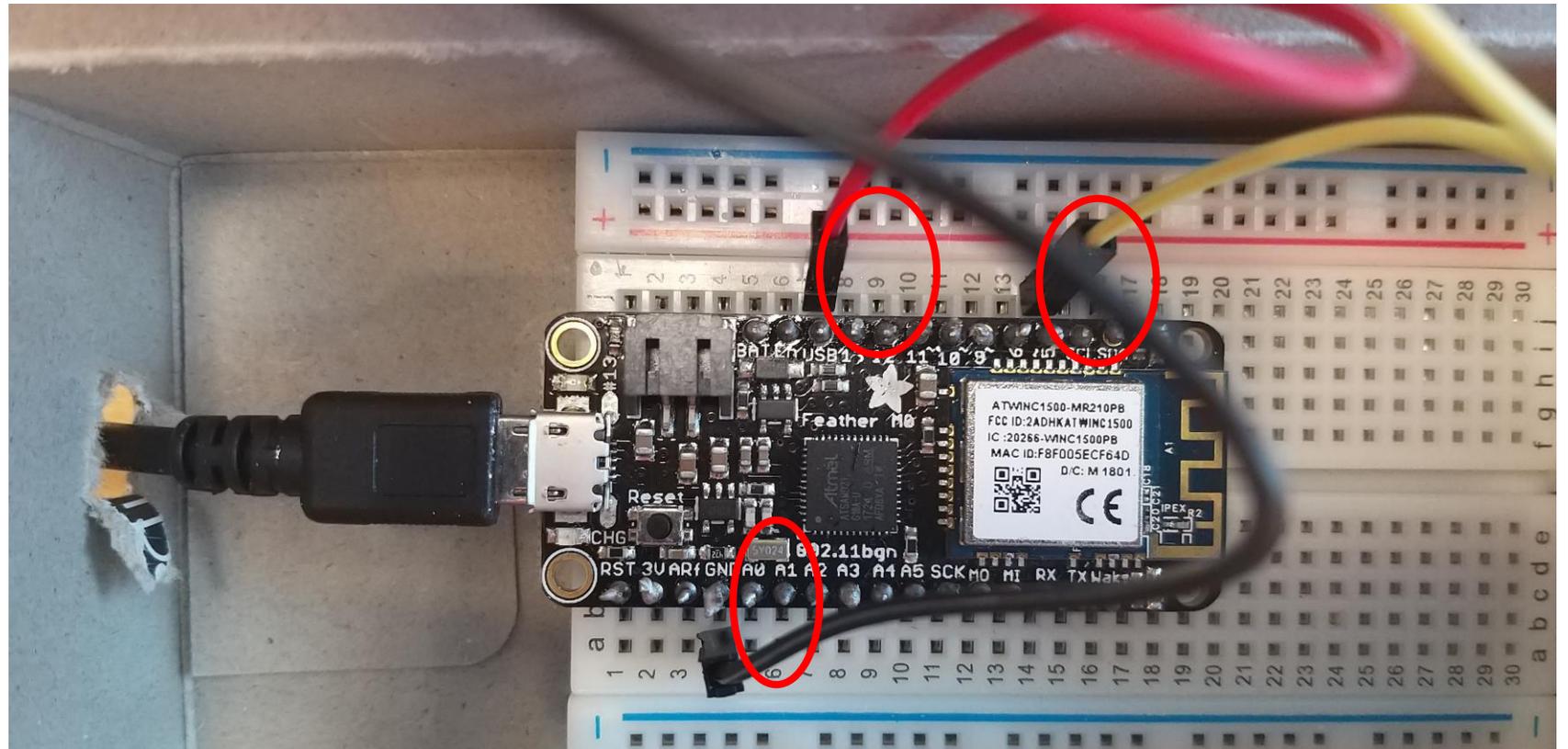
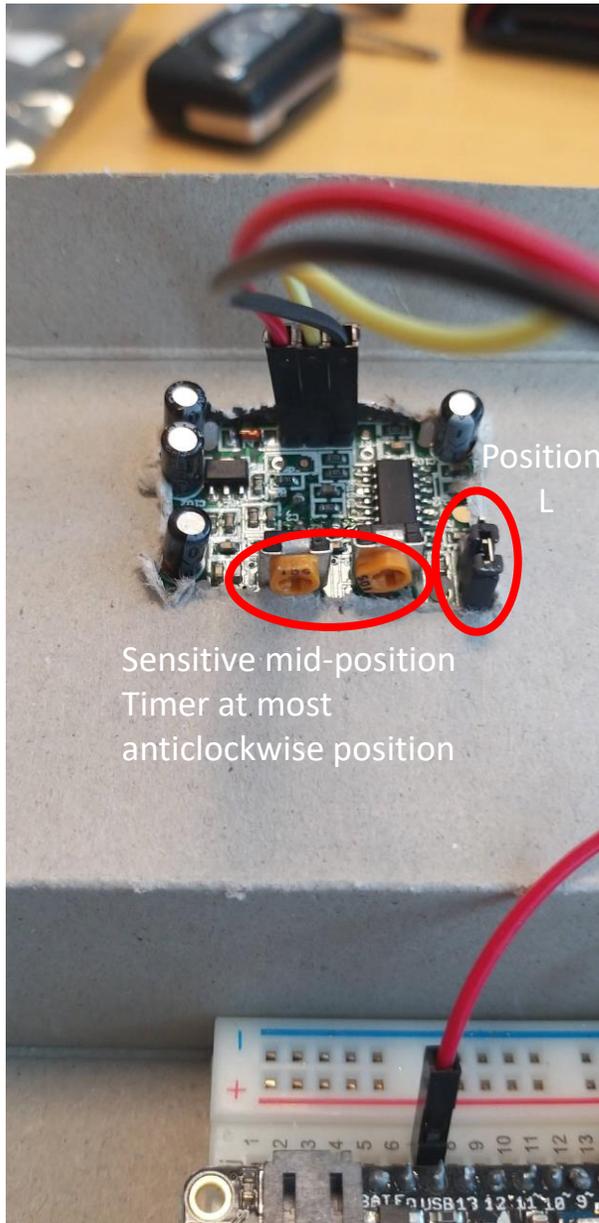
Adafruit Feather M0 Wi-Fi (atwinc1500) and PIR Pinout

PIR motion has to be set up using jumper to L position where the sensor will 'toggle' (change state) every time motion is detected - providing the on-board timer has timed out. This is unlikely to be of much use in some applications, but not this one. At this IoT project we need to have a timely manner to track last presence state in a certain meeting room in order to determine occupancy from last state change and current time. Also for future purposes, the ambition is to optimize the solution in order to roughly determine the amount of people with a meeting room. The sensor (or sensors) shall be placed in a position (i.e: center of the room/table) where motion can be determined frequently based on peoples natural behavior (i.e.: body movements).

Cabling / Pin sets: Red cable at USB/5V pin, black at GND (ground) and yellow at pin 5 (GPIO). Feather M0 has to be well stacked at the bread board using socket headers and **properly soldered**, to avoid motion detection misbehavior.



Adobe Acrobat
Document



IoT Sensor configuration using IDE Arduino

Microcontroller has to be uploaded with the attached Arduino code and be provided with the following data:

```
const char* googleApiKey = "xxxxxxxxxxxxxxxxxxxxxx";  
char ssid[] = "xxxxxx";  
char pass[] = "xxxxxx";
```



Adobe Acrobat
Document

googleApiKey has to be acquired from <https://developers.google.com/maps/documentation/geolocation/intro>, where billing details has to be provided. In case do you exceed the free tier credit, this will not incur in any charge. Note that the code request Geolocation of the microcontroller once at power-on, so it is unlikely that you exceed the free tier credit, even you decide to use for dozens of rooms.

SSID from closest access point is required, along with password. pass is optional and is required when WPA/WPA2 auth method is used instead of open access.

MQTT Broker details:

```
#define host    "<MQTT Broker IP address or FQDN if DNS is used: i.e.: AWS Route53>"  
#define clientid "<prefix>-meetingroom1"  
//Common prefix is used to implement some level of access control at MQTT Broker Server  
//clientid also must match user-part of room mailbox address, configured via o365 admin  
#define username "sensor1"  
#define password "sensor123"  
//username and password is used for ACL purposes too at MQTT Broker side. More details will be provided MQTT Server side.
```

Mosquitto MQTT Broker Server Setup

Mosquitto is the choice of MQTT Broker, placed between MQTT clients (publisher sensors) and AWS IoT Core (IoT Connectivity Manager). The broker is responsible for receiving all messages, filtering the messages (on #topic basis), determining who is subscribed to each message, and sending the message to these subscribed clients. The broker also holds the sessions of all persisted clients, including subscriptions and missed messages. Another responsibility of the broker is the authorization of clients done on MQTT Broker and initiate TLS authentication procedure towards the AWS IoT Core.

MQTT Broker AMI based on Mosquitto 1.4.5 is available at UE-EAST-1 region (N. Virginia), as found below. Just select Community AMIs at left frame and search by “mqtt” string.

Launch Configuration for ASG can be created using indicated AMI and attached “user-data” in order to automate a couple tasks, such as “MQTT broker thing” creation at IoT Core, configuration of Mosquitto broker, etc. Zip file contains Broker configuration files, http index file (for http server used for Network Load Balancing monitoring) and customized init.d/mosquitto service file (stored in a S3 bucket and downloaded via VPC endpoint to avoid public access). In order to run under AWS free tier, ASG is made of 1 desired/max EC2 VM.

aws Services Resource Groups Fabio N. Virginia Support

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

Architecture	AMI Name	AMI ID	Root device type	Virtualization type	ENA Enabled	Action
<input type="checkbox"/> 32-bit (x86) <input type="checkbox"/> 64-bit (x86) <input type="checkbox"/> 64-bit (Arm)	HiveMQ 4.0.1 Enterprise MQTT Broker AMI (single node)		ebs	hvm	Yes	64-bit (x86)
<input type="checkbox"/> EBS	Mosquitto MQTT Broker - Public AMI	ami-0a460cf5d798d36c9	ebs	hvm	Yes	Select 64-bit (x86)



s3-mqtt-broker.zip

AWS EC2/NLB/ASG-LaunchConfig/S3/SG/VPC-EP screenshots configuration of MQTT Broker

Create Auto Scaling group Actions

Filter: Filter Auto Scaling groups

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grace Period
ASG-mqtt-Broker	MQTT Broker Server v5	1	1	0	1	us-east-1d, us-east-1e	300	300

Auto Scaling Group: ASG-mqtt-Broker

Details | Activity History | Scaling Policies | Instances | Monitoring | Notifications | Tags | Scheduled Actions | Lifecycle Hooks

Launch Configuration: MQTT Broker Server v5

Availability Zone(s): us-east-1d, us-east-1e

Subnet(s): subnet-38a9007, subnet-b56749b

Classic Load Balancers: none

Target Groups: mqtt-broker-targets

Health Check Type: ELB

Health Check Grace Period: 300

Instance Protection: none

Termination Policies: Default

Suspended Processes: none

Placement Groups: none

Default Cooldown: 300

Create Security Group Actions

Filter by tags and attributes or search by keyword

Name	Group ID	Group Name	VPC ID	Owner
Mosquito-M...	sg-65c8892d	Mosquito-MQTT-SG	vpc-b96b3c2	617734212808

Security Group: sg-65c8892d

Description | Inbound | Outbound | Tags

Edit

Type	Protocol	Port Range	Source
HTTP	TCP	80	10.10.0.0/16
SSH	TCP	22	0.0.0.0/0
Custom TCP Rule	TCP	8883	0.0.0.0/0
Custom TCP Rule	TCP	1883	0.0.0.0/0

Create Load Balancer Actions

Filter by tags and attributes or search by keyword

Name	DNS name	State	VPC ID	Availability Zones	Type
NLB-MQTT-Server	NLB-MQTT-Server-2efa256...	active	vpc-b96b3c2	us-east-1e, us-east-1d	network

Load balancer: NLB-MQTT-Server

Description | Listeners | Monitoring | Integrated services | Tags

A listener checks for connection requests using its configured protocol and port, and the load balancer uses the listener rules to route requests to targets. You can add, remove, or edit listeners and listener rules.

Add listener | Edit | Delete

Listener ID	Security policy	SSL Certificate	Default action
TCP : 1883	arn...fe41a44cf06a0c1e*	N/A	Forward to mqtt-broker-targets

Amazon S3 > mqtt-broker

Overview | Properties | Permissions

Type a prefix and press Enter to search. Press ESC to clear.

Upload | Create folder | Download | Actions | Versions | Hide | Show

Name	Last modified
conf	--
init.d	--
www	--

Create target group Actions

Filter by tags and attributes or search by keyword

Name	Port	Protocol	Target type	Load Balancer	VPC ID	Monitoring
mqtt-broker-targets	1883	TCP	instance	NLB-MQTT...	vpc-b96b3c2	
MyTarget-EC2-DMZ	80	HTTP	instance	vpc-a7aa55df		

target group: mqtt-broker-targets

Description | Targets | Health checks | Monitoring | Tags

Basic Configuration

Name: mqtt-broker-targets

ARN: arn:aws:elasticloadbalancing:us-east-1:617734212808:targetgroup/mqtt-broker-targets/e584d16b8ae9625f

Protocol: TCP

Port: 1883

Target type: instance

VPC: vpc-b96b3c2

Load balancer: NLB-MQTT-Server

Attributes

Create Endpoint Actions

Filter by tags and attributes or search by keyword

Name	Endpoint ID	VPC ID	Service name	Endpoint type	Status	Creation time
	vpc-e67fec07	vpc-b96b3c2	com.amazonaws.us-east-1-s3	Gateway	available	May 29, 2018 at 4:59:31 PM UTC+3

Endpoint: vpc-e67fec07

Details | Route Tables | Policy | Tags

Endpoint ID	VPC ID	Status	Creation Time	Service name	Endpoint type
vpc-e67fec07	vpc-b96b3c2	available	May 29, 2018 at 4:59:31 PM UTC+3	com.amazonaws.us-east-1-s3	Gateway

Launch Configuration: MQTT Broker Server v5

Details

AMI ID	ami-d72846a8	Instance Type	t2.micro
IAM Instance Profile	IoT_Mosquito_Broker	Kernel ID	
Key Name		Monitoring	false
EBS Optimized	false	Security Groups	sg-65c8892d
Spot Price		Creation Time	Wed Jan 16 01:06:02 GMT+300 2019
RAM Disk ID		Block Devices	idev/xvda
User data		IP Address Type	Assign a public IP address to every instance.

User data

```
#!/bin/bash
yum -y update

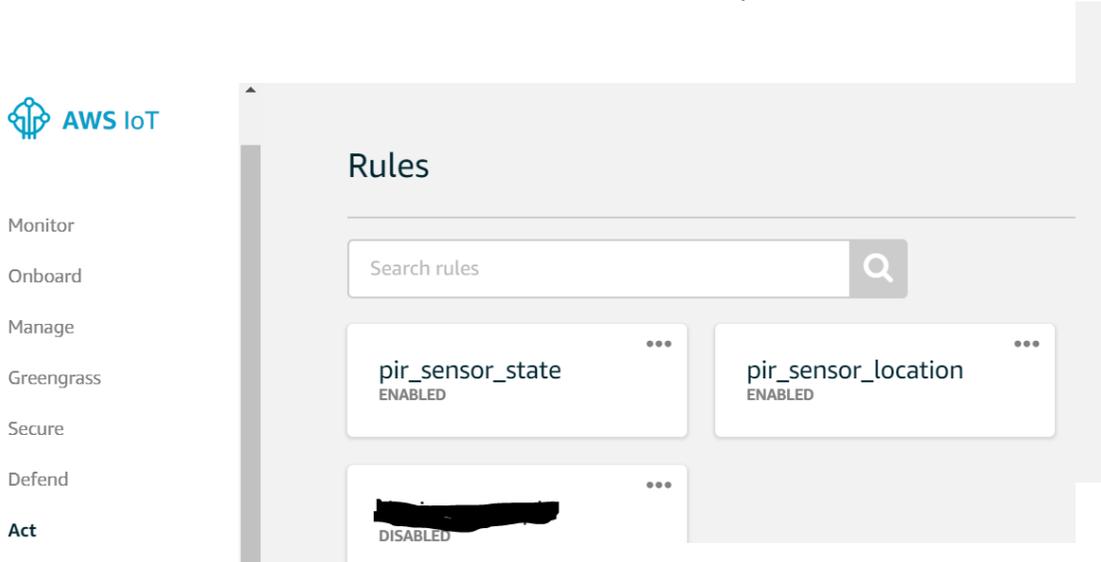
#Configure the CLI with your region, leave access/private keys blank
aws configure set default.region us-east-1
```

bash_userdata-EC2 - pub.bash

AWS IoT Core Configuration

NOTE that Mosquitto Broker once instantiated, will trigger broker thing creation, IAM policy for the bridge creation, certificates and keys creation and download, attach the policy to the certificate and finally attach the policy to your thing automatically, via EC2 user-data bash scripting. Please check slide 7 for details.

Rules creation are still not automated, not thus shared here.



Insert a message into a DynamoDB table pir_sensor_state

Table name pir_sensor_state

Partition key topic

Partition key value \${topic()}

Sort key timestamp

Sort key value \${timestamp()}

Write message data to this column state

Operation INSERT

IAM Role service-role/TestThingRole

Insert a message into a DynamoDB table pir_sensor_location

Table name pir_sensor_location

Partition key topic

Partition key value \${topic()}

Sort key

Sort key value

Write message data to this column location

Operation UPDATE

IAM Role service-role/TestThingRole

Rule query statement Edit

The source of the messages you want to process with this rule.

```
SELECT * FROM 'pir_sensor/state/#'
```

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Insert a message into a DynamoDB table

pir_sensor_state

Remove

Edit

Rule query statement

The source of the messages you want to process with this rule.

```
SELECT * FROM 'pir_sensor/location/#'
```

Using SQL version 2016-03-23

Actions

Actions are what happens when a rule is triggered. [Learn more](#)



Insert a message into a DynamoDB table

pir_sensor_location

Remove

Check_room_occupancy Lambda Function, environment variables used and execution timer (1/2)

Check_room_occupancy Throttle Qualifiers ▼ Actions ▼

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code.

DB_SMRIOTAD_AT	smriotad-atDB	Remove
DB_TABLE_LOCATION	pir_sensor_location	Remove
DB_TABLE_STATE	pir_sensor_state	Remove
authorityHostUrl	https://login.microsoftonline.com	Remove
clientId	[Azure AD Application ID]	Remove
clientSecret	[Azure AD Application Secret]	Remove
resource	https://graph.microsoft.com	Remove
tenant	[tenant name].onmicrosoft.com	Remove
topic_location	pir_sensor/location	Remove
topic_state	pir_sensor/state	Remove
aad_domain	[custom domain]	Remove
Key	Value	

Execution timer of 10 secs is suggested for 3 meeting rooms made available at same time, due MS Graph API callbacks execution to check and release rooms. This timer shall be increase if more rooms are added. Cost Analysis shall be carried between lambda services and dedicated EC2 instances if intended for commercial purposes.



Check_room_occupancy.zip

Deployment package (Python 3.7) found above. Please note that it is greater than 3MB and and it is not possible to edit the code at lambda GUI directly. Azure AD related variables values (**where to locate?**), are found at slide 15.

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role ▼

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/Smart_room_IoT ▼

[View the Smart_room_IoT role](#) on the IAM console.

Basic settings

Description

Check Smart Room occupancy. Default non-occupancy

Memory (MB) [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

0 min 10 sec



Event (time-based 1 min)

Rules > job_check_room_occupancy

Actions ▾

Summary

ARN ⓘ `arn:aws:events:us-east-1:██████████:rule/job_check_room_occupancy`

Schedule Fixed rate of 1 minutes

Status Disabled

Description This event job execute Check_room_occupancy Lambda function every minute and provides fixed JSON constant data to the function

Monitoring [Show metrics for the rule](#)

Targets

Filter:

« < Viewing 1 to 1 of 1 Targets > »

Type	Name	Input	Role	Additional parameters
Lambda function	Check_room_occupancy	Constant: {"occupancy_timer": 600}		

An event-driven Lambda function (Check_room_occupancy) is triggered every 1 minute, in order check reported presence/motion by IoT sensors installed at each meeting room, against current timestamp and configured "occupancy_timer", this last to used determine the pre-booked meeting rooms in idle state and entitled for auto-cancellation.

Check_room_occupancy Lambda Function, environment variables used and execution timer (2/2)

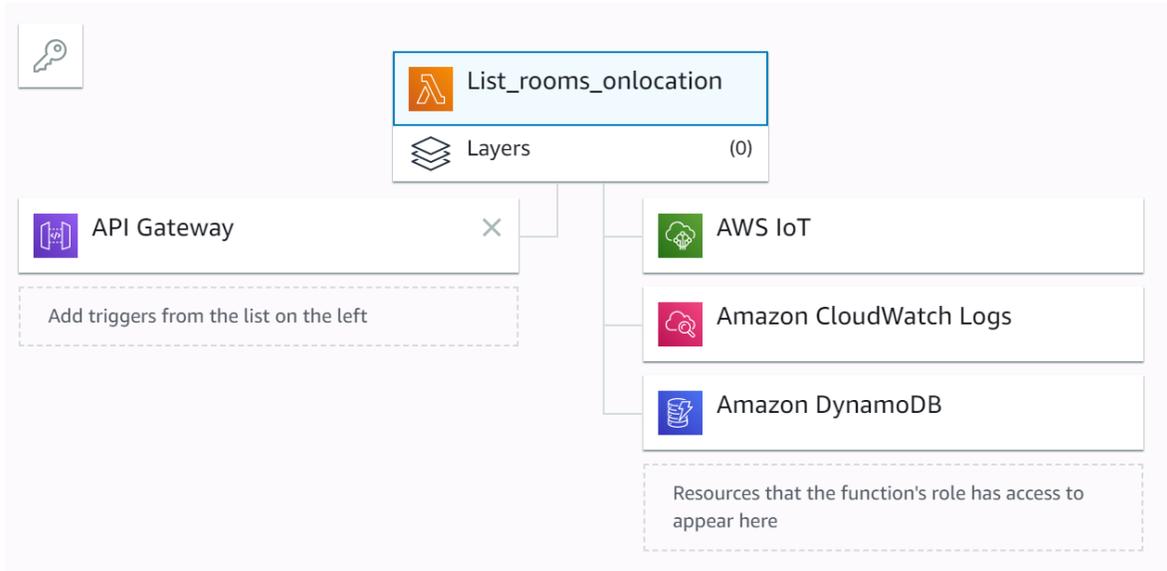
```
PS C:\Users\[REDACTED]\AWS\MQTT Broker & IoT\Check_room_occupancy> dir

Directory: C:\Users\[REDACTED]\AWS\MQTT Broker & IoT\Check_room_occupancy

Mode                LastWriteTime         Length Name
----                -
d-----          5/18/2019  11:18 PM             adal
d-----          5/18/2019  11:18 PM      adal-1.2.1.dist-info
d-----          5/18/2019  11:18 PM      asn1crypto
d-----          5/18/2019  11:18 PM      asn1crypto-0.24.0.dist-info
d-----          5/18/2019  11:18 PM             bin
d-----          5/18/2019  11:18 PM      certifi
d-----          5/18/2019  11:18 PM      certifi-2019.3.9.dist-info
d-----          5/18/2019  11:18 PM      cffi
d-----          5/18/2019  11:18 PM      cffi-1.12.3.dist-info
d-----          5/18/2019  11:18 PM      chardet
d-----          5/18/2019  11:18 PM      chardet-3.0.4.dist-info
d-----          5/18/2019  11:18 PM      cryptography
d-----          5/18/2019  11:18 PM      cryptography-2.6.1.dist-info
d-----          5/18/2019  11:18 PM      dateutil
d-----          5/18/2019  11:18 PM      idna
d-----          5/18/2019  11:18 PM      idna-2.8.dist-info
d-----          5/18/2019  11:18 PM      jwt
d-----          5/18/2019  11:18 PM      pycparser
d-----          5/18/2019  11:18 PM      pycparser-2.19-py3.7.egg-info
d-----          5/18/2019  11:18 PM      PyJWT-1.7.1.dist-info
d-----          5/18/2019  11:18 PM      python_dateutil-2.8.0.dist-info
d-----          5/18/2019  11:18 PM      requests
d-----          5/18/2019  11:18 PM      requests-2.22.0.dist-info
d-----          5/18/2019  11:18 PM      six-1.12.0.dist-info
d-----          5/18/2019  11:18 PM      urllib3
d-----          5/18/2019  11:18 PM      urllib3-1.25.2.dist-info
d-----          5/18/2019  11:18 PM      __pycache__
-a-----          5/29/2019  12:47 PM          4482 lambda_function.py
-a-----          5/18/2019  11:18 PM        32452 six.py
-a-----          5/18/2019  11:18 PM       174592 _cffi_backend.cp37-win_amd64.pyd
```

```
aws lambda update-function-code --function-name
Check_room_occupancy --zip-file fileb://Check_room_
occupancy.zip
```

List_rooms_onlocation Lambda Function and environment variables



Deployment package written in Python 3.7 and environment variables used below.

API Gateway configuration found in the next page.

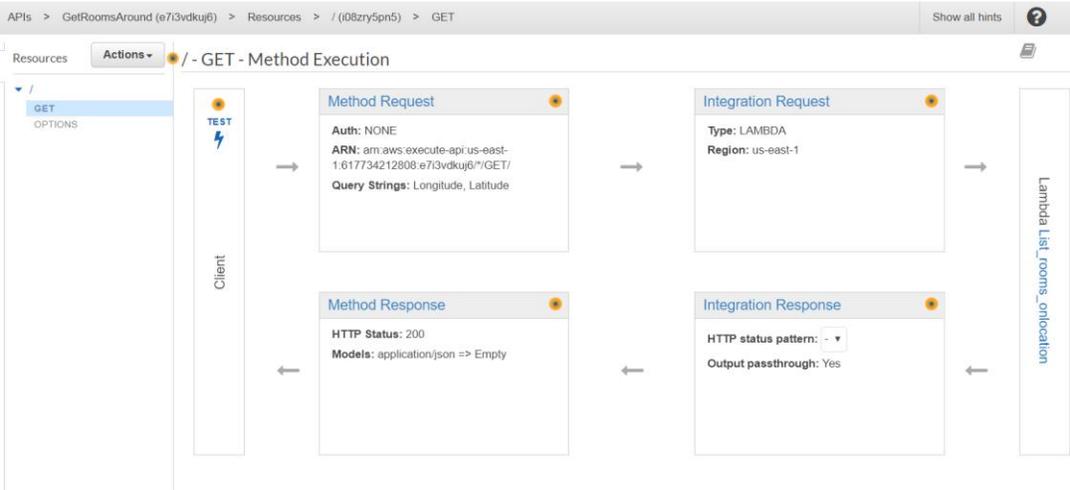
Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code.

[Learn more](#)

DB_TABLE_LOCATION	pir_sensor_location	Remove
DB_TABLE_STATE	pir_sensor_state	Remove
topic_location	pir_sensor/location	Remove
<i>Key</i>	<i>Value</i>	Remove

AWS API Gateway for HTTP/REST requests configuration and backend lambda function integration



Method Execution / - GET - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Settings

- Authorization: NONE
- Request Validator: NONE
- API Key Required: false

URL Query String Parameters

Name	Required	Caching	
Latitude	<input type="checkbox"/>	<input type="checkbox"/>	
Longitude	<input type="checkbox"/>	<input type="checkbox"/>	

[Add query string](#)

HTTP Request Headers

Method Execution / - GET - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type: Lambda Function

- HTTP
- Mock
- AWS Service
- VPC Link

Use Lambda Proxy integration:

Lambda Region: us-east-1

Lambda Function: List_rooms_onlocation

Execution role:

Invoke with caller credentials:

Credentials cache: Do not add caller credentials to cache key

Use Default Timeout:

URL Path Parameters

Mapping Templates

Request body passthrough

- When no template matches the request Content-Type header
- When there are no templates defined (recommended)
- Never

Content-Type

application/json

[Add mapping template](#)

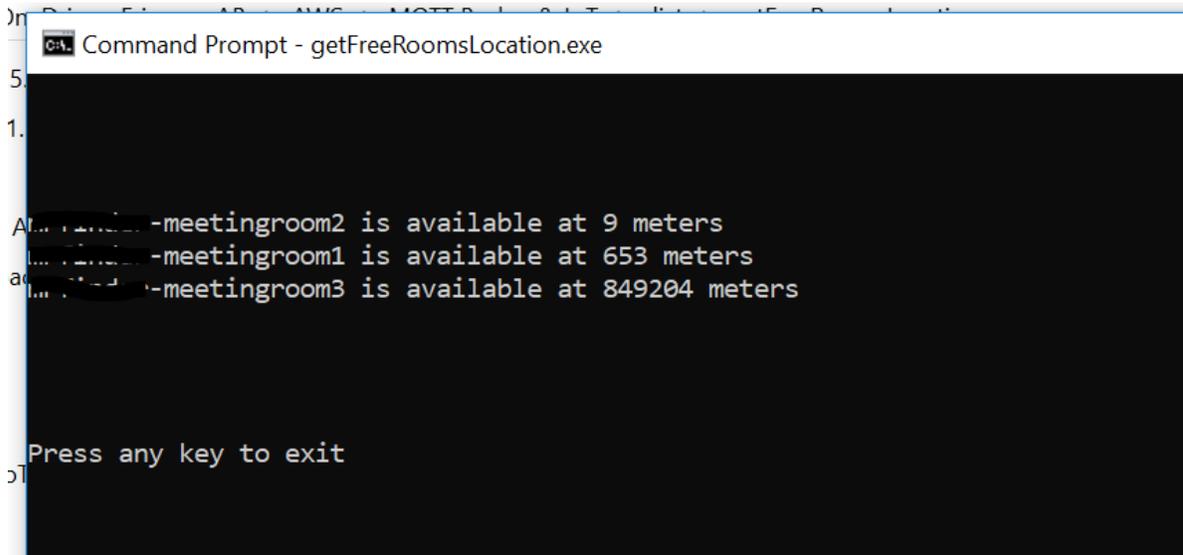
application/json

Generate template:

```
1 - {
2   "Latitude" : "$input.params('Latitude')",
3   "Longitude" : "$input.params('Longitude')",
4 }
```

Client Desktop App

Client desktop App is coded in python 3.7 for simplicity and prototyping purposes.



```
Command Prompt - getFreeRoomsLocation.exe

-meetingroom2 is available at 9 meters
-meetingroom1 is available at 653 meters
-meetingroom3 is available at 849204 meters

Press any key to exit
```



Run it using `c:\python .\getFreeRoomsLocation.py`

Or use pyinstaller for executable creation and easy distribution. Suggest to pack it one directly instead of one file due performance purposes:

What to configure/customize?

Google Geolocation api-endpoint + Key

URL = <https://www.googleapis.com/geolocation/v1/geolocate?key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

api-gateway endpoint, found at AWS API Gateway GUI, left frame menu "Dashboard" of concerned API

URL_APIG = <https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>

MS Azure AD App registration, Client secrets and API permissions

Add Enterprise application o365 Exchange Online for test environment purposes

smr-iot-app

Search (Ctrl+/) Delete Endpoints

Overview Quickstart

Manage Branding Authentication Certificates & secrets API permissions Expose an API

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? [Learn more](#)

Display name: smr-iot-app

Application (client) ID: [REDACTED]

Directory (tenant) ID: [REDACTED]

Object ID: [REDACTED]

Supported account types: My organization only

Redirect URLs: [Add a Redirect URI](#)

Managed application in local directory: smr-iot-app

Home > SMR-IOTAD - App registrations > smr-iot-app - API permissions

smr-iot-app - API permissions

Search (Ctrl+/)

Overview Quickstart

Manage Branding Authentication Certificates & secrets API permissions Expose an API Owners Manifest

API permissions

Applications are authorized to use APIs by requesting permissions. These permissions show up during the consent process where users are given the opportunity to grant/deny access.

[+ Add a permission](#)

API / PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ Microsoft Graph (4)			
Calendars.Read	Application	Read calendars in all mailboxes	Yes ✔ Granted for SMR-IOT...
Calendars.ReadWrite	Application	Read and write calendars in all mailboxes	Yes ✔ Granted for SMR-IOT...
User.Read.All	Application	Read all users' full profiles	Yes ✔ Granted for SMR-IOT...
User.ReadWrite.All	Application	Read and write all users' full profiles	Yes ✔ Granted for SMR-IOT...

These are the permissions that this application requests statically. You may also request user consent-

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as applicati

[+ New client secret](#)

DESCRIPTION	EXPIRES	VALUE
app secret	12/31/2299	Hidden

Enterprise applications - All applications

SMR-IOTAD - Azure Active Directory

[+ New application](#) Columns

Overview Overview

Manage All applications Application proxy User settings

NAME	HOMEPAGE URL	OBJECT ID	APPLICATION ID
Graph explorer	https://developer.microsoft.com/en-us/grap...	ef8d97d9-dd42-4d68-bebd-4b9f7...	de8bc8b5-d9f9-48b1-a8a...
Microsoft Teams		5b00d9c4-c68e-4338-a06b-f6e48f...	cc15fd57-2c6c-4117-a88c...
Office 365 Exchange Online	http://office.microsoft.com/outlook/	d21fe7c6-fa2f-41db-9c5c-f9c67321...	00000002-0000-0ff1-ce0...
Office 365 Management API		0077fa5f-7b64-425b-ab88-f007e7a...	c5393580-f805-4401-95e...

Microsoft o365 Resource Rooms configuration

Access to admin console is required. Create an user with admin privileges for the respective tenant in the Azure AD, and associate it to the Enterprise Application o365 Exchange Online

SMR-IOTAD

[Preview on](#)

Rooms & equipment

<input type="checkbox"/>	Name	Email	Type
<input type="checkbox"/>	[redacted] Meeting Room 1	[redacted]-meetingroom1@smriotad.onmicr...	Room
<input type="checkbox"/>	[redacted] - Meeting Room 2	[redacted]meetingroom2@smriotad.onmicr...	Room
<input type="checkbox"/>	[redacted] - Meeting Room 3	[redacted]meetingroom3@smriotad.onmicr...	Room

clientid used in the sensor side to identify the room must match user-part of room mailbox address, configured via o365 admin. @domain.com is configured inside the Lambda functions.

AWS DynamoDB tables

Choose a table group
Actions
Viewing 3 of 3 Tables

Name	Status	Partition key	Sort key	Indexes	Total read capacity	Total write capacity
pir_sensor_location	Active	topic (String)	-	0	5	5
pir_sensor_state	Active	topic (String)	timestamp (Number)	0	5	5
smriotad-atDB	Active	record_type (String)	-	0	5	5

Table details

Table name	pir_sensor_location
Primary partition key	topic (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	March 10, 2019 at 1:25:36 PM UTC+3
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5(Auto Scaling Disabled)
Provisioned write capacity units	5(Auto Scaling Disabled)
Last decrease time	-
Last increase time	-

Table details

Table name	pir_sensor_state
Primary partition key	topic (String)
Primary sort key	timestamp (Number)
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	March 6, 2019 at 3:53:58 PM UTC+3
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5(Auto Scaling Disabled)
Provisioned write capacity units	5(Auto Scaling Disabled)

Table details

Table name	smriotad-atDB
Primary partition key	record_type (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT Manage Encryption
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	May 26, 2019 at 2:01:07 PM UTC+3
Read/write capacity mode	Provisioned
Last change to on-demand mode	-
Provisioned read capacity units	5(Auto Scaling Disabled)
Provisioned write capacity units	5(Auto Scaling Disabled)
Last decrease time	-