Edge Computing (Framework: EdgeX)

Nelli Aghajanyan, Fargina Mahmud, Ruchit Dineshbhai Dobariya, Bhargav Anghan

Frankfurt University of Applied Sciences
Faculty of Computer Science and Engineering
Cloud Computing, Prof. Dr. Christian Baun
SoSe 2022

Overview

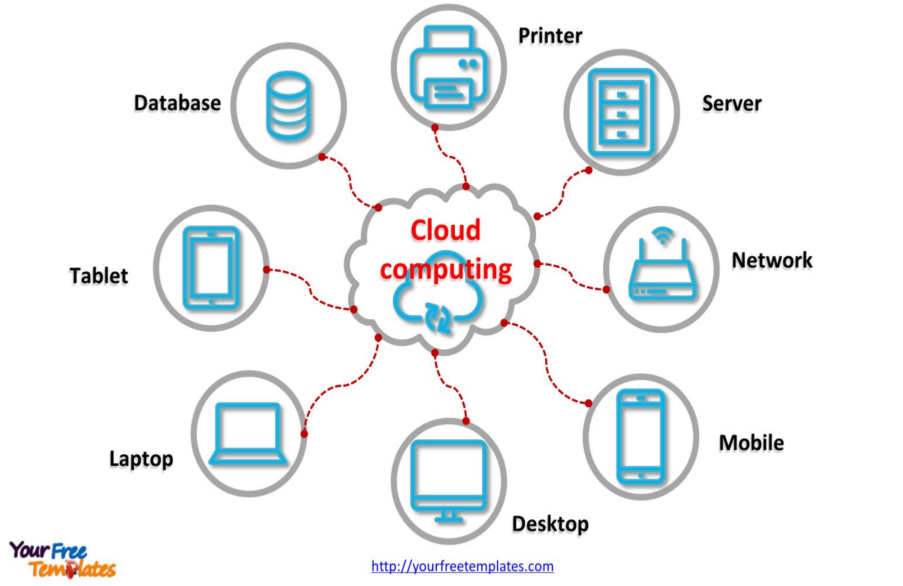
- What is Cloud Computing?
- What is Edge Computing?
- Docker
- EdgeX Foundry
- EdgeX Foundry installation on Ubuntu
- Raspberry Pi OS
- DHT22 Sensor
- Kuiper Engine
- AWS DynamoDB & IAM
- AWS Lambda & API Gateway
- Live Demo

Cloud Computing

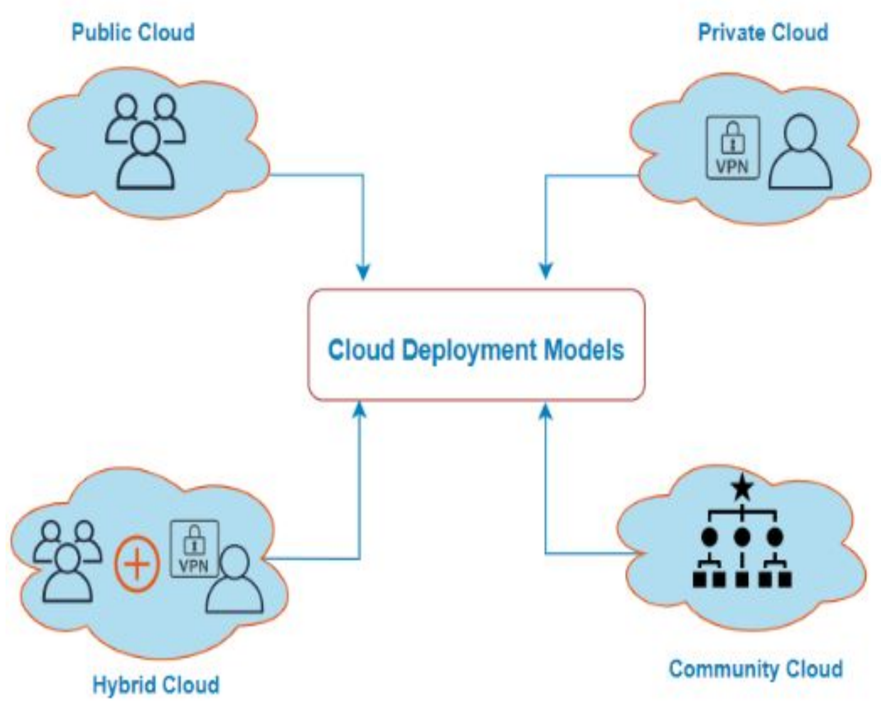
❖ Distributed computing on internet Or delivery of computing service over the internet.

❖ Instead of running a program on your computer, you log in to a Web account remotely. The software and storage for your account doesn't exist on your computer -- it's on the service's computer cloud.

Cloud computing metaphor



Cloud Deployment Model



Type of Cloud Model

- Public cloud
- Private cloud
- Community cloud
- Hybrid Cloud

Edge Computing : the concept

Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers.

"For edge devices to be smart, they need to process the data they collect, share timely insights and if applicable, take appropriate action. Edge computing is the science of having the edge devices do this without the need for the data to be transported to another server environment. Put another way, edge computing brings the data and the compute closest to the point of interaction."

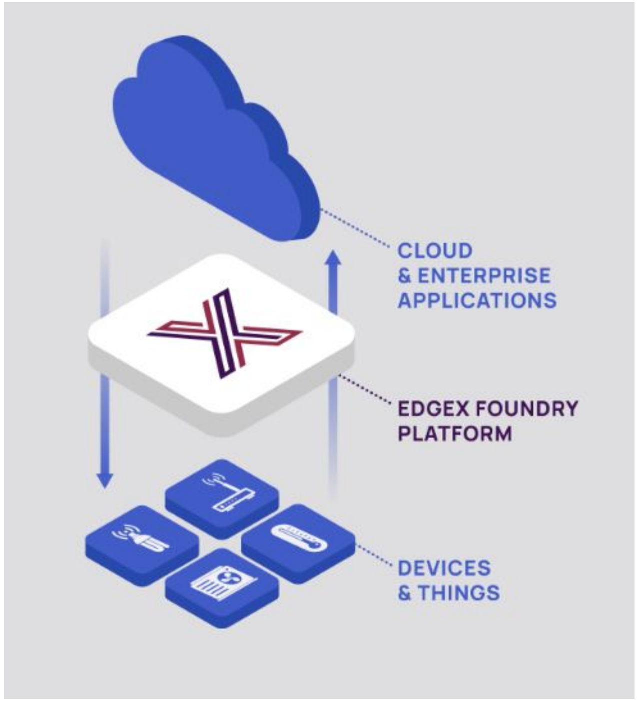
says Red Hat chief technology strategist E.G. Nadhan

DOCKER

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

Docker is an open source platform for building, deploying, and managing containerized applications.

EdgeX Foundry



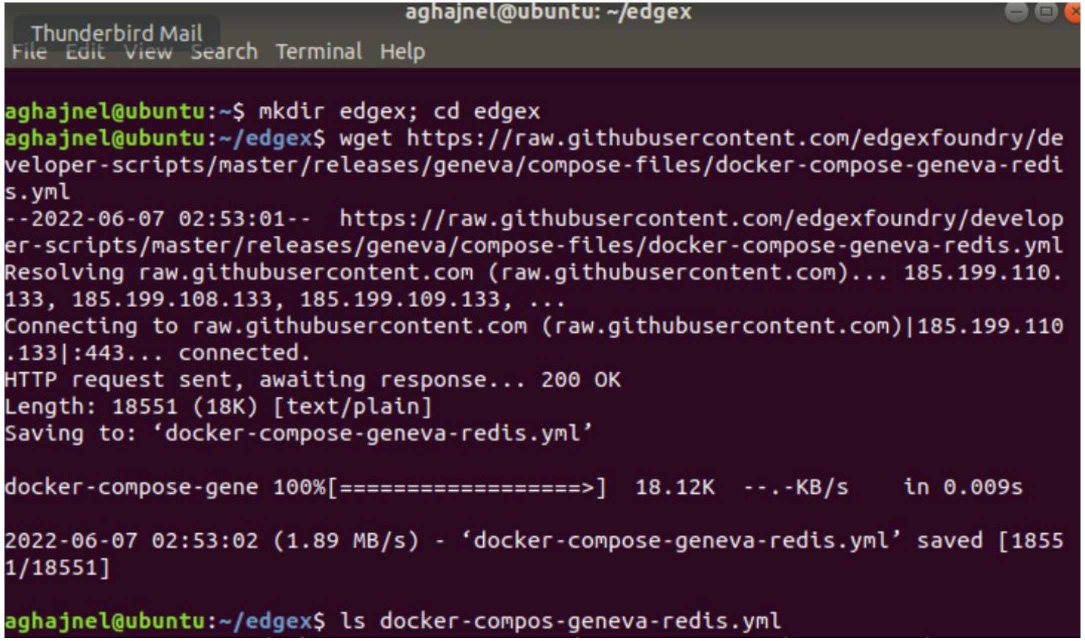
EdgeX Foundry is an open source, vendor neutral, Edge IoT middleware platform.

EdgeX enables autonomous operations and intelligence at the Edge.

EdgeX is a key enables or digital transformation and AI across IoT use cases and businesses in all vertical markets.

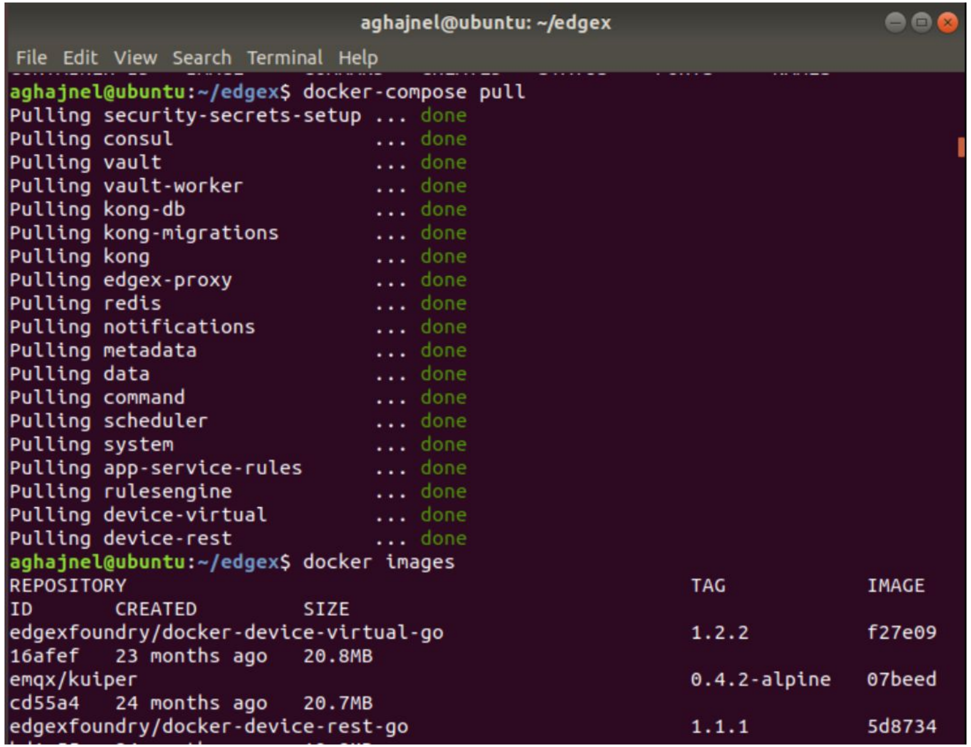
EdgeX Foundry Installation on Ubuntu

- **Step 1: Installing Docker and docker-compose**

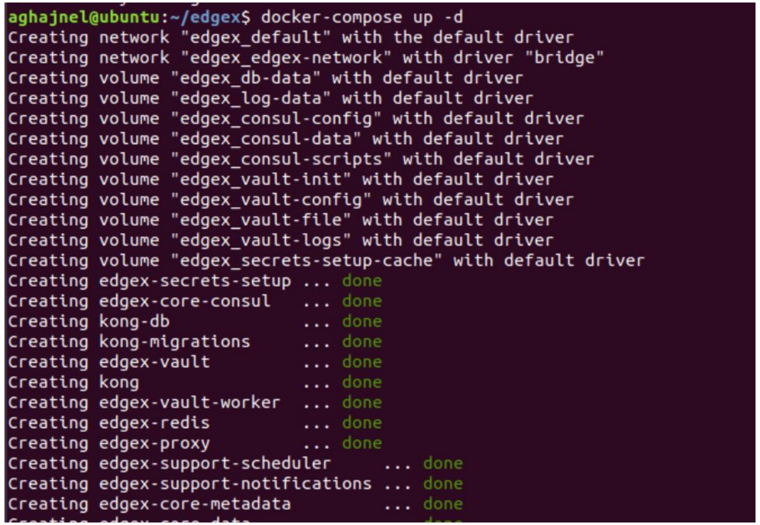


```
aghajnel@ubuntu:~$ mkdir edgex; cd edgex
aghajnel@ubuntu:~/edgex$ wget https://raw.githubusercontent.com/edgexfoundry/de
veloper-scripts/master/releases/geneva/compose-files/docker-compose-geneva-redi
s.yml
--2022-06-07 02:53:01--  https://raw.githubusercontent.com/edgexfoundry/develop
er-scripts/master/releases/geneva/compose-files/docker-compose-geneva-redis.yml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.
133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110
.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18551 (18K) [text/plain]
Saving to: ‘docker-compose-geneva-redis.yml’

docker-compose-gene 100%[===================>]  18.12K  --.-KB/s    in 0.009s

2022-06-07 02:53:02 (1.89 MB/s) - ‘docker-compose-geneva-redis.yml’ saved [1855
1/18551]

aghajnel@ubuntu:~/edgex$ ls docker-compos-geneva-redis.yml
```

- **Step 2: Running EdgeX Foundry**

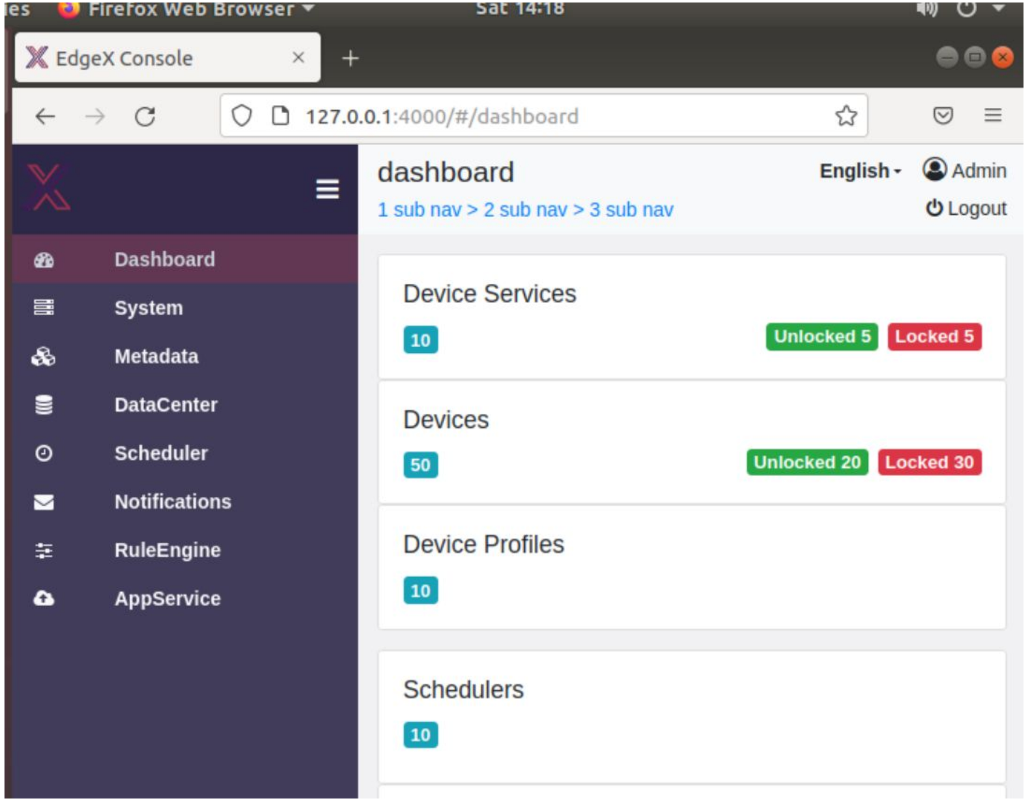


```
aghajnel@ubuntu:~/edgex$ docker-compose pull
Pulling security-secrets-setup ... done
Pulling consul                 ... done
Pulling vault                  ... done
Pulling vault-worker           ... done
Pulling kong-db                ... done
Pulling kong-migrations        ... done
Pulling kong                   ... done
Pulling edgex-proxy            ... done
Pulling redis                  ... done
Pulling notifications          ... done
Pulling metadata               ... done
Pulling data                   ... done
Pulling command                ... done
Pulling scheduler              ... done
Pulling system                 ... done
Pulling app-service-rules      ... done
Pulling rulesengine            ... done
Pulling device-virtual         ... done
Pulling device-rest            ... done
aghajnel@ubuntu:~/edgex$ docker images
REPOSITORY                                                  TAG             IMAGE
ID         CREATED          SIZE
edgexfoundry/docker-device-virtual-go                       1.2.2           f27e09
16afef     23 months ago    20.8MB
emqx/kuiper                                                 0.4.2-alpine    07beed
cd55a4     24 months ago    20.7MB
edgexfoundry/docker-device-rest-go                          1.1.1           5d8734
```



```
aghajnel@ubuntu:~/edgex$ docker-compose up -d
Creating network "edgex_default" with the default driver
Creating network "edgex_edgex-network" with driver "bridge"
Creating volume "edgex_db-data" with default driver
Creating volume "edgex_log-data" with default driver
Creating volume "edgex_consul-config" with default driver
Creating volume "edgex_consul-data" with default driver
Creating volume "edgex_consul-scripts" with default driver
Creating volume "edgex_vault-init" with default driver
Creating volume "edgex_vault-config" with default driver
Creating volume "edgex_vault-file" with default driver
Creating volume "edgex_vault-logs" with default driver
Creating volume "edgex_secrets-setup-cache" with default driver
Creating edgex-secrets-setup ... done
Creating edgex-core-consul   ... done
Creating kong-db             ... done
Creating kong-migrations     ... done
Creating edgex-vault         ... done
Creating kong                ... done
Creating edgex-vault-worker  ... done
Creating edgex-redis         ... done
Creating edgex-proxy         ... done
Creating edgex-support-scheduler     ... done
Creating edgex-support-notifications ... done
Creating edgex-core-metadata         ... done
```

- **Step 3: Accessing EdgeX Foundry**

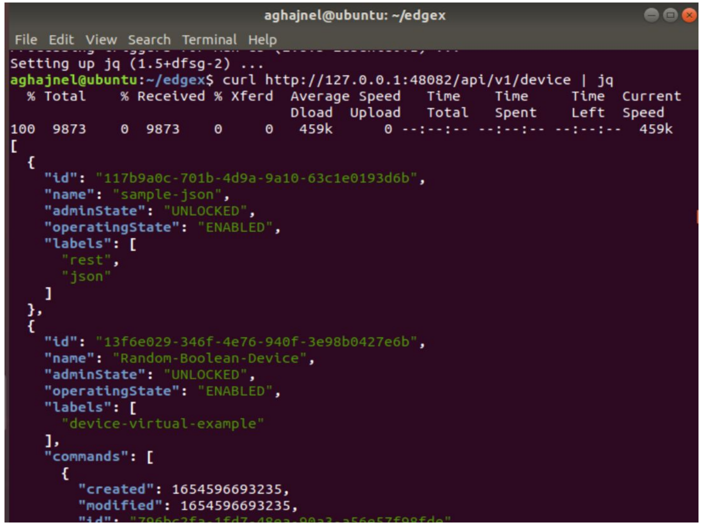


Device Services
10 Unlocked 5 Locked 5

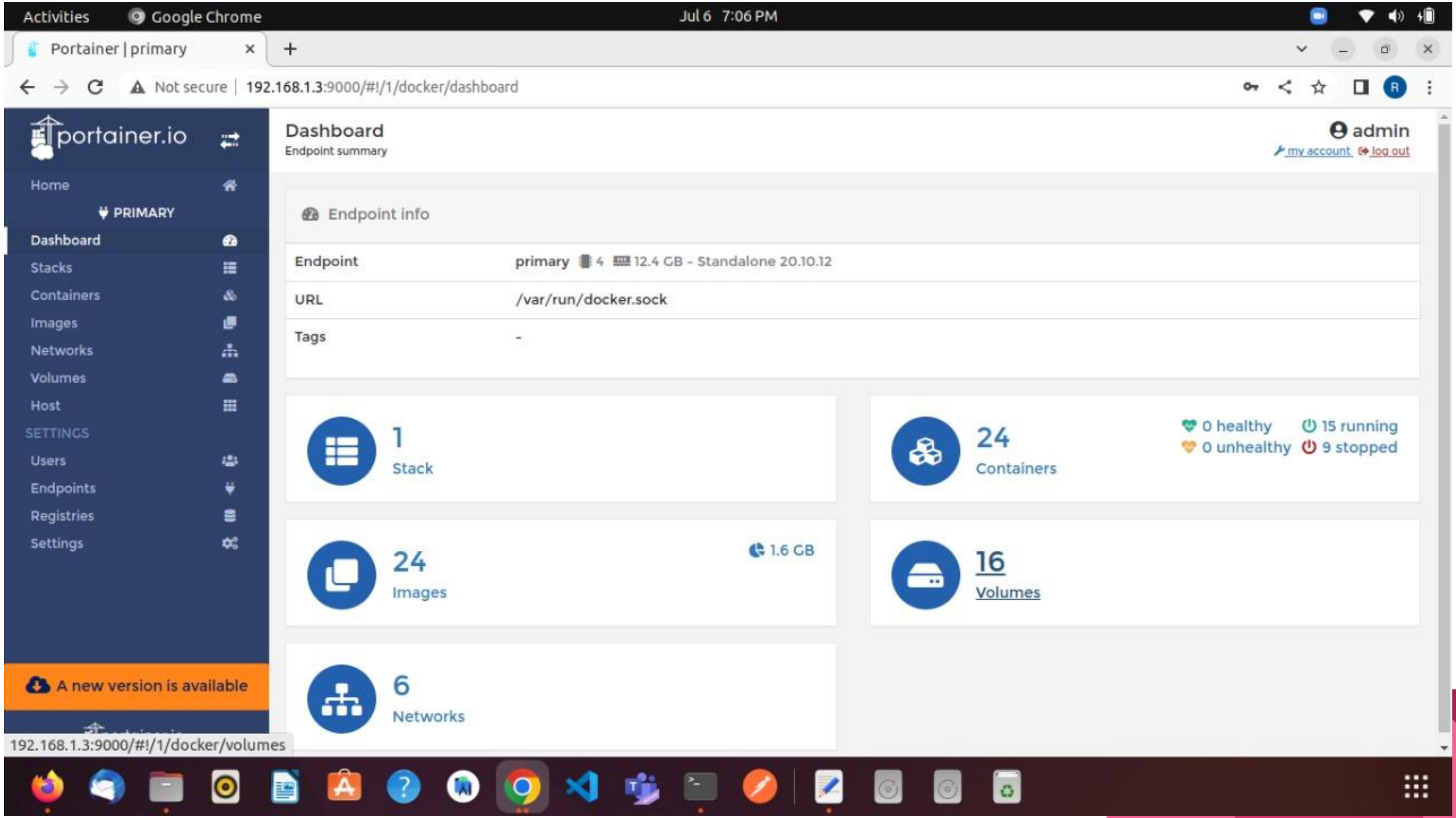
Devices
50 Unlocked 20 Locked 30

Device Profiles
10

Schedulers
10



```
Setting up jq (1.5+dfsg-2) ...
aghajnel@ubuntu:~/edgex$ curl http://127.0.0.1:48082/api/v1/device | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9873    0  9873    0     0   459k      0 --:--:-- --:--:-- --:--:--  459k
[
  {
    "id": "117b9a0c-701b-4d9a-9a10-63c1e0193d6b",
    "name": "sample-json",
    "adminState": "UNLOCKED",
    "operatingState": "ENABLED",
    "labels": [
      "rest",
      "json"
    ]
  },
  {
    "id": "13f6e029-346f-4e76-940f-3e98b0427e6b",
    "name": "Random-Boolean-Device",
    "adminState": "UNLOCKED",
    "operatingState": "ENABLED",
    "labels": [
      "device-virtual-example"
    ],
    "commands": [
      {
        "created": 1654596693235,
        "modified": 1654596693235,
```



Dashboard
Endpoint summary

Endpoint info

Endpoint	primary 4 12.4 GB - Standalone 20.10.12
URL	/var/run/docker.sock
Tags	-

1 Stack

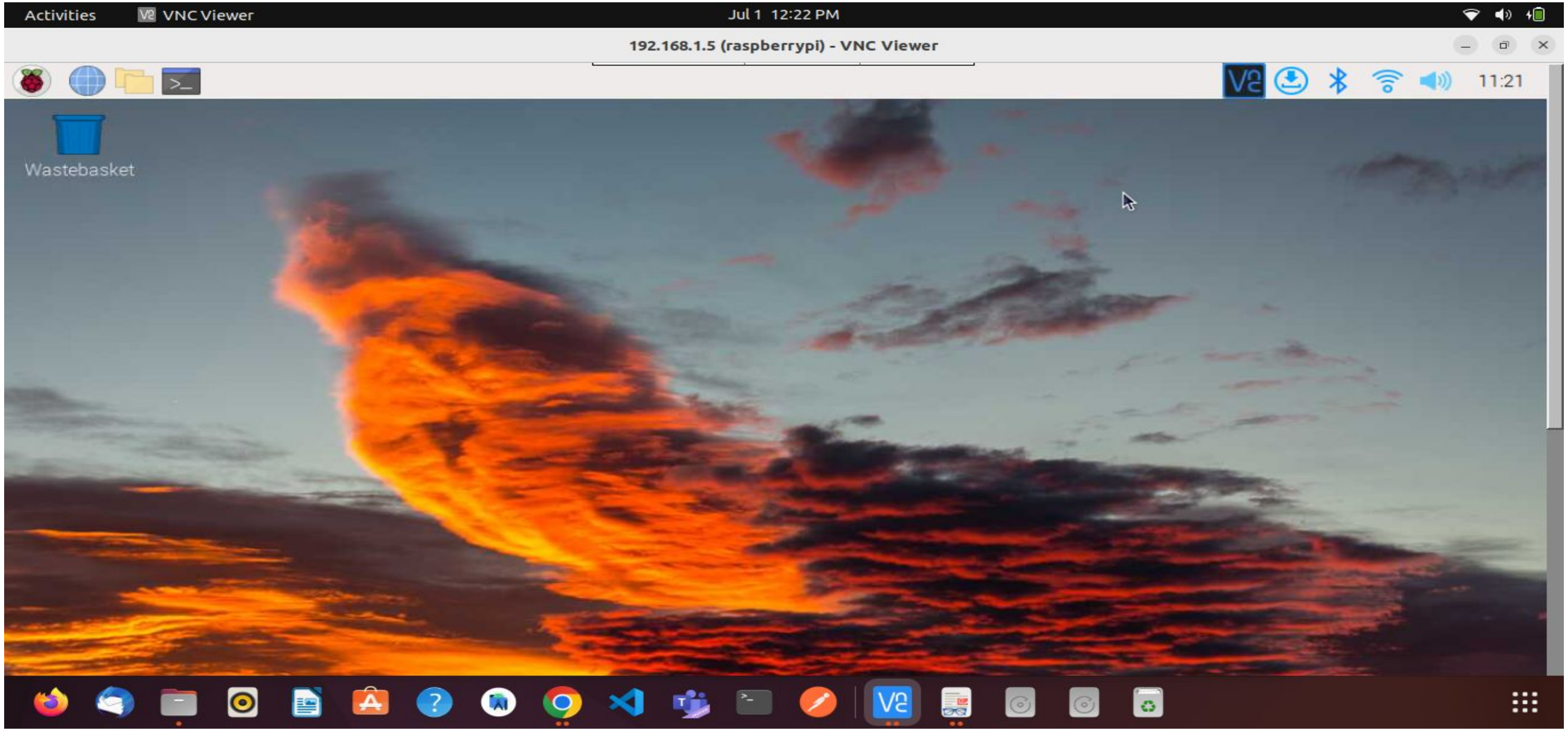
24 Containers — 0 healthy, 0 unhealthy, 15 running, 9 stopped

24 Images — 1.6 GB

16 Volumes

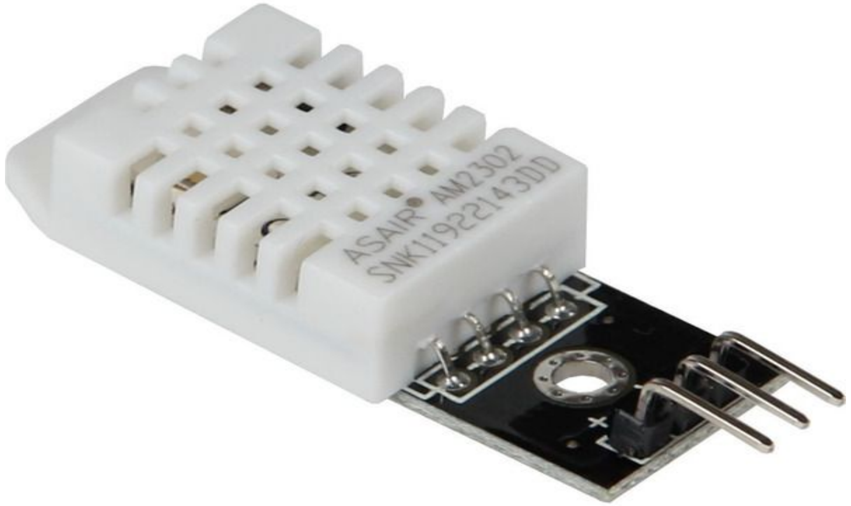
6 Networks

Raspberry Pi OS



DHT22 Sensor

- ❖ Accuracy:±2
- ❖ Humidity Range:0 100
- ❖ Temperature:-40°C 80°C
- ❖ Output Type:Digital
- ❖ Voltage - Supply:3.3V 6V



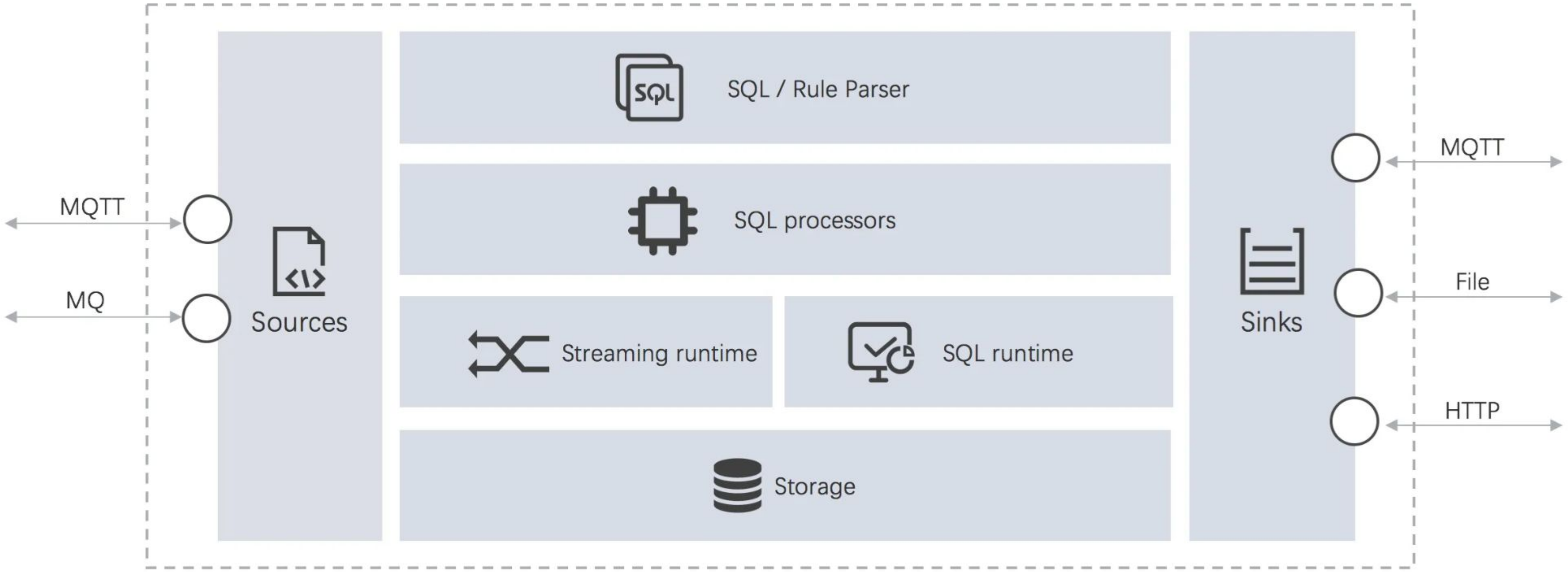
Kuiper Rules Engine

- Lightweight
- Cross-platform
- Data analysis support
- Management:
  - streams and rules management through CLI & REST API

Kuiper Rule engine components

- **Source**: Source of stream data, such as data from an MQTT server. For EdgeX, the data source is an EdgeX message bus, which can be implemented by ZeroMQ or MQTT.
- **SQL**: SQL is where the specified business logic is processed. Kuiper provides SQL statements to extract, filter, and transform data.
- **Sink**: Used to send the analysis result to a specific target, such as sending the analysis results to EdgeX's Command service, or an MQTT broker in the cloud.

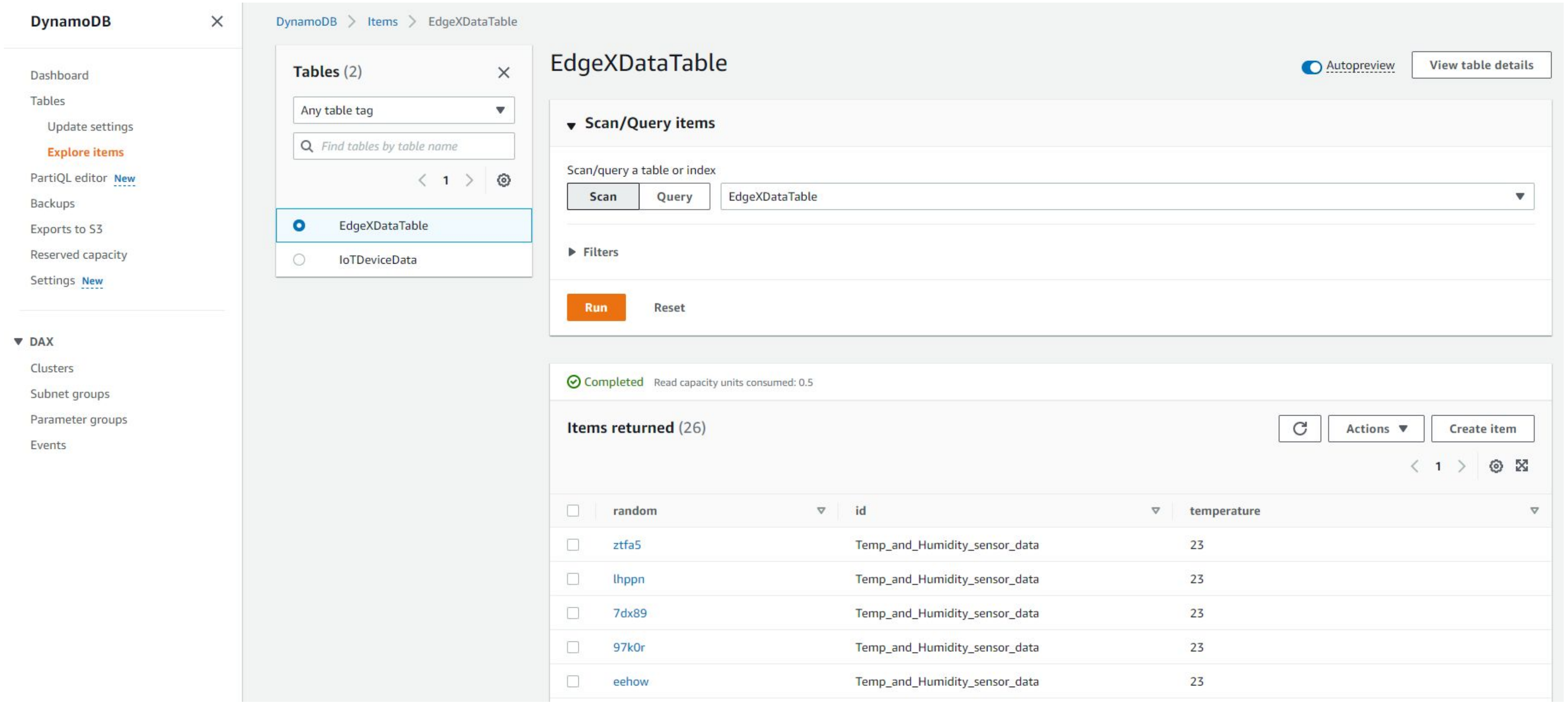
Kuiper Rules Engine



AWS DynamoDB

- ❖ A fast and flexible NoSql database service.
- ❖ Supports key-value and document databases.
- ❖ In DynamoDB we have created a table called “EdgeXDataTable” with primary key “random” as string.

AWS DynamoDB



DynamoDB

- Dashboard
- Tables
  - Update settings
  - Explore items
- PartiQL editor New
- Backups
- Exports to S3
- Reserved capacity
- Settings New

DAX

- Clusters
- Subnet groups
- Parameter groups
- Events

DynamoDB > Items > EdgeXDataTable

Tables (2)

Any table tag

Find tables by table name

● EdgeXDataTable

○ IoTDeviceData

EdgeXDataTable

Autopreview | View table details

Scan/Query items

Scan/query a table or index

Scan | Query | EdgeXDataTable

Filters

Run | Reset

Completed Read capacity units consumed: 0.5

Items returned (26)

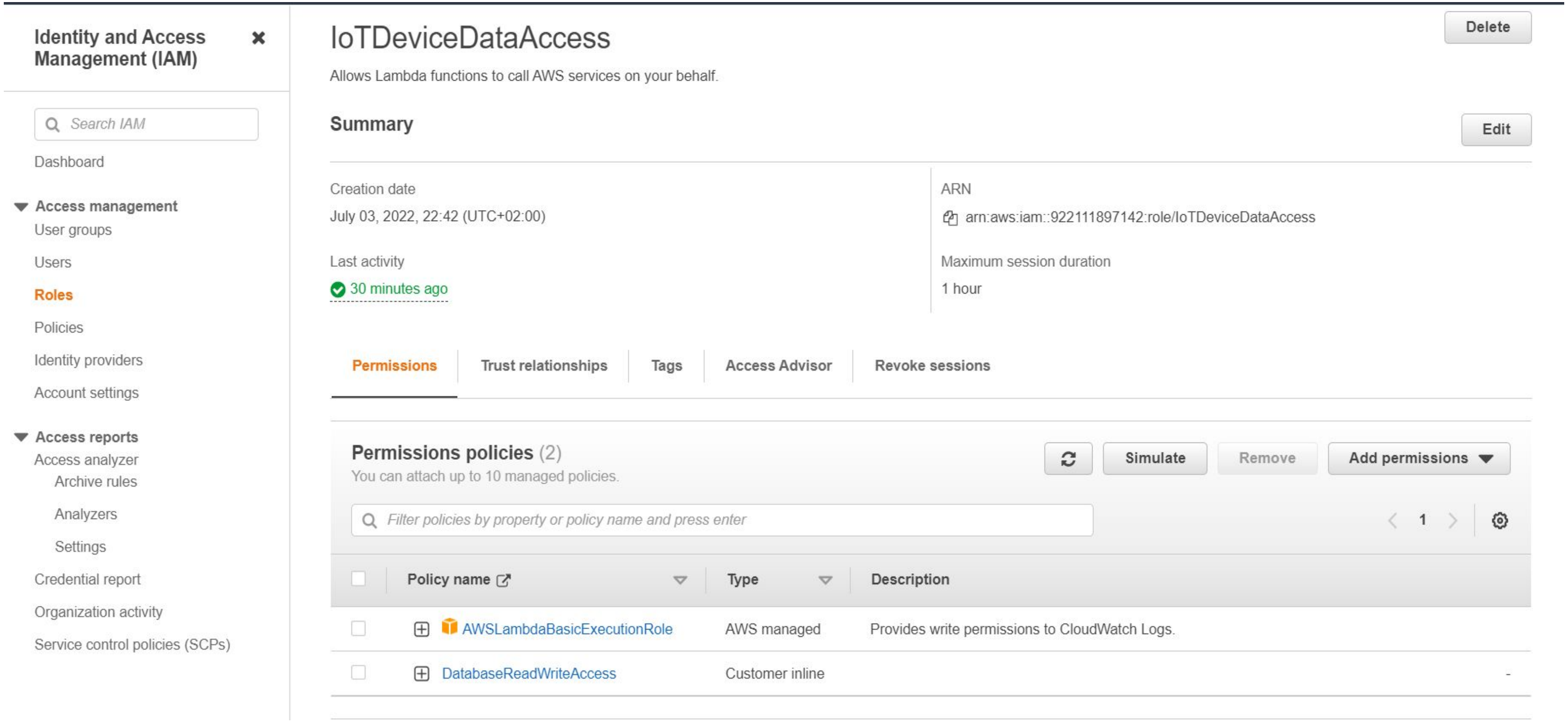
Actions | Create item

random	id	temperature
ztfa5	Temp_and_Humidity_sensor_data	23
lhppn	Temp_and_Humidity_sensor_data	23
7dx89	Temp_and_Humidity_sensor_data	23
97k0r	Temp_and_Humidity_sensor_data	23
eehow	Temp_and_Humidity_sensor_data	23

IAM

- ❖ To control access to service resources.
- ❖ Used to implement lambda rule.
- ❖ We have created “IoTDeviceDataAccess” role using IAM service.

IAM



Identity and Access Management (IAM)

Search IAM

Dashboard

Access management
- User groups
- Users
- **Roles**
- Policies
- Identity providers
- Account settings

Access reports
- Access analyzer
  - Archive rules
  - Analyzers
  - Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

IoTDeviceDataAccess

Delete

Allows Lambda functions to call AWS services on your behalf.

Summary

Edit

Creation date	ARN
July 03, 2022, 22:42 (UTC+02:00)	arn:aws:iam::922111897142:role/IoTDeviceDataAccess
Last activity	Maximum session duration
30 minutes ago	1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (2)

You can attach up to 10 managed policies.

Simulate | Remove | Add permissions

Filter policies by property or policy name and press enter

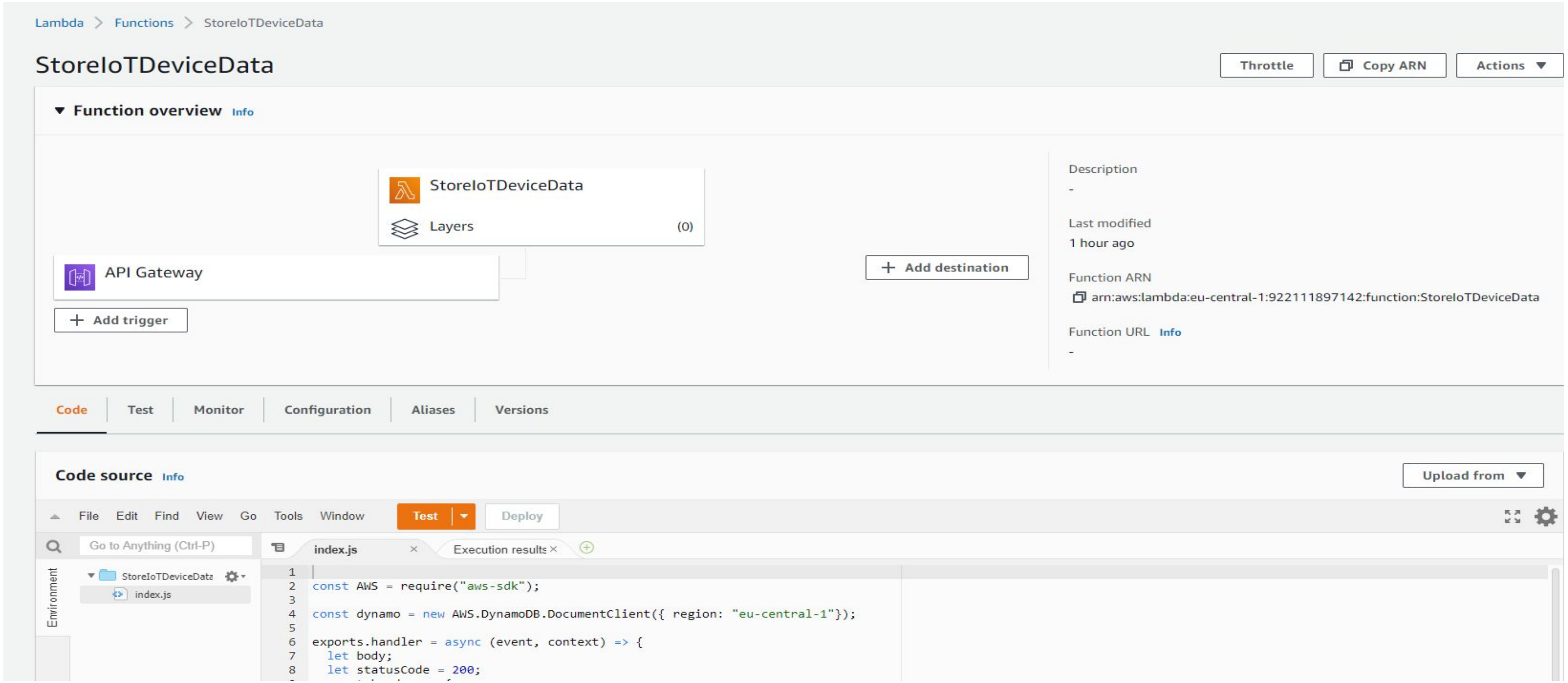
1

Policy name	Type	Description
AWSLambdaBasicExecutionRole	AWS managed	Provides write permissions to CloudWatch Logs.
DatabaseReadWriteAccess	Customer inline	-

AWS Lambda

- ❖ AWS Lambda is a serverless function.
- ❖ For accessing the created DynamoDB table lambda function is needed.
- ❖ A Lambda function is created and named as “StoreIoTDeviceData”
- ❖ A javascript file with Node.js backend to parse API Gateway data and store it DynamoDB

AWS Lambda



Description
-

Last modified
1 hour ago

Function ARN
arn:aws:lambda:eu-central-1:922111897142:function:StoreIoTDeviceData

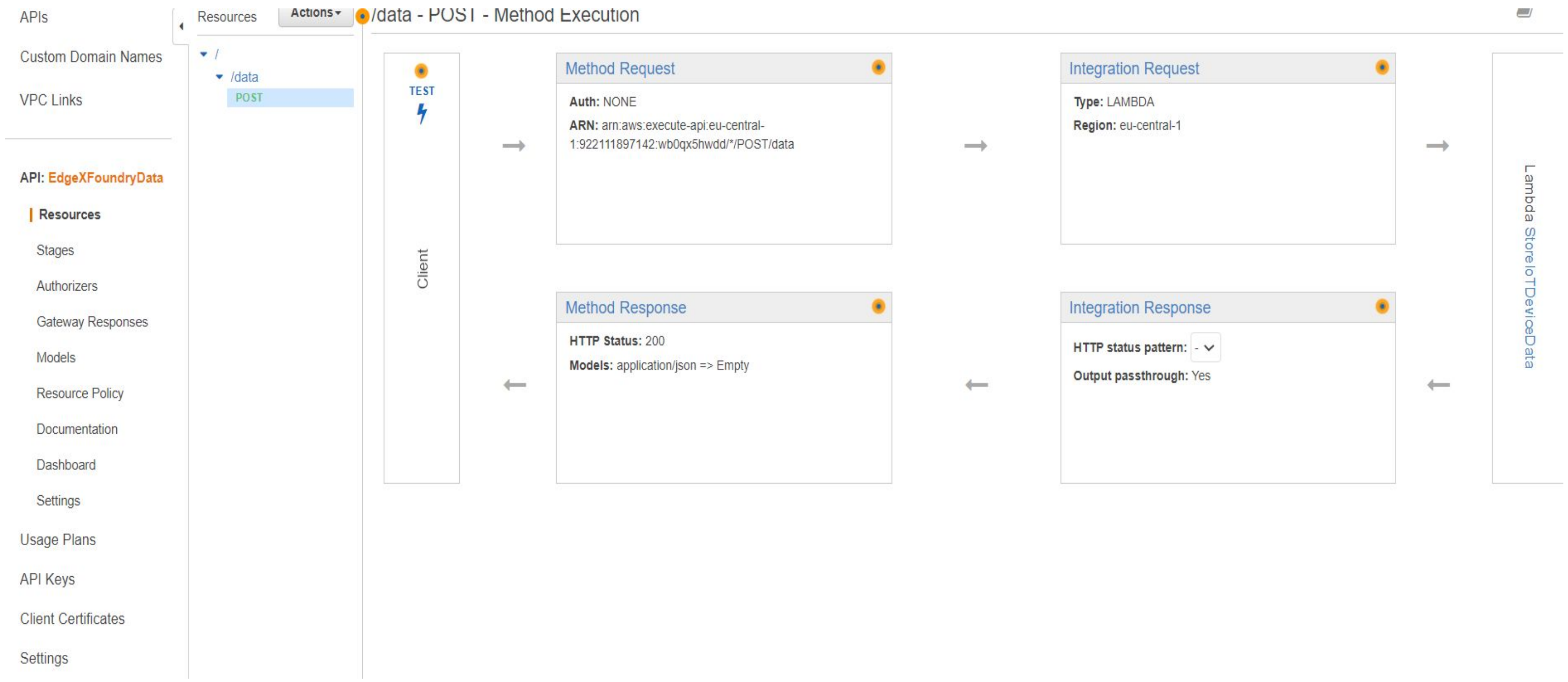
Function URL Info
-

```

const AWS = require("aws-sdk");

const dynamo = new AWS.DynamoDB.DocumentClient({ region: "eu-central-1"});

exports.handler = async (event, context) => {
  let body;
  let statusCode = 200;
```

API Gateway

- ❖ To implement a gateway we have used AWS API gateway service so that EdgeX can communicate with AWS services
- ❖ We created API gateway “EdgexFoundaryData”.
- ❖ The Gateway API service supports RESTful-APIs .
- ❖ POST HTTP method is used to send data to AWS Lambda.

API Gateway



APIs
Custom Domain Names
VPC Links

API: EdgeXFoundryData
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
API Keys
Client Certificates
Settings

Resources Actions

/
/data
POST

/data - POST - Method Execution

TEST
Client

Method Request

Auth: NONE

ARN: arn:aws:execute-api:eu-central-1:922111897142:wb0qx5hwdd/*/POST/data

Integration Request

Type: LAMBDA

Region: eu-central-1

Method Response

HTTP Status: 200

Models: application/json => Empty

Integration Response

HTTP status pattern: -

Output passthrough: Yes

Lambda StoreIoTDeviceData

Live Demo