Easy as a Pi: EdgeX on Raspberry Pi for Automotive IoT

Alex Courouble - VMware

August 2019

acourouble@vmware.com
Agenda

Edge-based Architecture

A Python Sample Automotive IoT Application

Intro to EdgeX Foundry

A ‘Full Stack’ App with EdgeX on a Raspberry pi

Future work
Edge Base Architecture

Source: https://wiki.edgexfoundry.org/display/FA/Executive+Summary
Open Glossary of Edge Computing (v0.9.0-Beta)

- Version: v0.9.0-beta
- Date: June 12, 2018 1:08 PM Pacific Time
- License: Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

Overview

The Open Glossary of Edge Computing is a wiki-like, open source lexicon of terms related to edge computing. It has been built using a collaborative process and is designed for easy adoption by the entire edge computing ecosystem, including by open source projects, vendors, standards groups, analysts, journalists, and practitioners.

Glossary

3G, 4G, 5G
3G, 4th, and 5th generation cellular technologies, respectively. In simple terms, 3G represents the introduction of the smartphone along with their mobile web browsers; 4G, the current generation cellular technology, delivers true broadband internet access to mobile devices; the coming 5G cellular technologies will deliver massive bandwidth and reduced latency to cellular systems, supporting a range of devices from smartphones to autonomous vehicles and large-scale IoT. Edge computing at the infrastructure edge is considered a key building block for 5G.

See also: Infrastructure Edge

Access Edge Layer

The sub-layer of infrastructure edge closest to the end user or device, zero or one hops from the last mile network. For example, an edge data center deployed at a cellular network site. The Access Edge Layer functions as the front line of the infrastructure edge and may connect to an aggregation edge layer higher in the hierarchy.

See also: Aggregation Edge Layer

Access Network

A network that connects subscribers and devices to their local service provider. It is contrasted with the core network, which connects service providers to one another. The access network connects directly to the infrastructure edge.

See also: Infrastructure Edge

Aggregation Edge Layer


Sample Automotive IoT App

Architecture

Devices

- OBD
- GPS

Edge

- Raspberry Pi

Cloud Endpoints

- Events

Sample Available at https://github.com/vmwaresamples/automotive-iot-samples
Intro to EdgeXFoundry

Hardware / OS agnostic
Plug and Play Components
Microservices
Containers, Snap, or Go binaries
LF Edge
Apache 2.0
A Full Stack App with EdgeXFoundry

Overview

Cloud End Points
“Northbound”

Gateways
“Edge”

Device Layer
“Southbound”
A Full Stack App with EdgeXFoundry

Southbound: Build a Device Service with Device SDK

Device SDK:

- Create Device Service to Read/Write to device
- Define a Device Profile
  - Commands
  - Resources (reading type)

https://github.com/edgexfoundry/device-sdk-go
Southbound
GPS device service
https://github.com/edgexfoundry-holding/device-gps

- Implements Device-SDK-go
- AddDevice()
  - Starts a \textit{go} routine to read and convert GPS coordinate dump
- HandleReadCommand()
  - Returns the latest coordinates found by the \textit{go} routine
- Reading includes:
  - Latitude
  - Longitude
  - Speed
  - Time
- Includes a mock data file to get started with the device service without buying a GPS device
A Full Stack App with EdgeXFoundry
Southbound: Build a Device Service with Device SDK

App Function SDK

- Create a Function Pipeline to filter, transform, and export your data to the Cloud (or elsewhere!)
- Use built-in function or Custom function
- Function can be called on every event added to core data

https://github.com/edgexfoundry/device-sdk-go
Northbound
Exporting filtered events to the cloud with app-function-sdk-go
https://github.com/edgexfoundry/app-functions-sdk-go

Easy to build function pipeline to filter, transform and export your data and events

Series of built-in functions to perform common actions on events

MQTT export to popular cloud providers coming soon
Edge Gateway
Raspberry Pi 3 B+

64-bit ARM
1GB RAM
Ethernet, WiFi, BLE
SD card reader
4 USB ports
Edge Gateway
Installing EdgeXFoundry with docker-compose

OS: Ubuntu Server 18 ARM64

Install Go 1.12

Install Docker

Install Docker-compose
  • Use pip3 and install `libssl-dev libffi-dev` before docker-compose

Download ARM64 docker-compose file

Create 2GB of swap

Download and build Device-GPS

Create and build your Function Pipeline to export your data

Ready to go!

Complete tutorial: https://github.com/vmware-samples/automotive-iot-samples/tree/master/edgex_sample
Southbound
Cloud Endpoints

User-facing app

EdgeX pushes events to the cloud endpoint in HTTP

Custom built endpoint in Python + Flask

UI in JS + React

Outside of EdgeX's focus!

<table>
<thead>
<tr>
<th>Client ID</th>
<th>Client Name</th>
<th>Time</th>
<th>Distance</th>
<th>Fuel</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd1234566</td>
<td>someone</td>
<td>2018-11-02T17:58:3...</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>batmobile1</td>
<td>Batman</td>
<td>2018-11-02T18:00:...</td>
<td>250</td>
<td>24</td>
</tr>
</tbody>
</table>
Future work

Helping developers running EdgeX on Raspberry Pi

ARM 64 docker-compose file more readily available

Up-to-date tutorial on how to run on RPI

Automotive-iot-sample updated with EdgeX components on Github
Future work

Relevant features coming in the next release (Fuji)

Official release of App-function-SDK

Export functions to major cloud providers (AWS IoT, Azure IoT, Alibaba)
  • Help wanted if you want your cloud provider to be supported!

Device-GPS to be merged in official EdgeXFoundry github organization
Thank You