Edge Computing para IoT

María Dolores Medina Barroso
Máster de Ingeniería Informática

Félix Freitag

Enero 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Edge Computing para IoT
Nombre del autor:	María Dolores Medina Barroso
Nombre del consultor:	Félix Freitag
Fecha de entrega (mm/aaaa):	01/2019
Área del Trabajo Final:	Sistemas Distribuidos
Titulación:	*Máster de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

El tamaño y variedad de los datos que circulan por las redes actuales está aumentado exponencialmente e IoT contribuye significativamente a este aumento del volumen, esto implica un incremento sustancial del tráfico que llega a los centros de datos para su análisis. Este aumento de tráfico afecta de forma directa al tiempo de respuesta de los sistemas de IoT, lo que en entornos críticos puede resultar inaceptable.

Con este trabajo se pretende analizar un nuevo paradigma de computación llamado Edge Computing que acerca parte de las capacidades de análisis y procesamiento que actualmente ofrecen los sistemas en la nube al lugar en el que se generan los datos, disminuyendo de esta forma el volumen de información que debe enviarse a la red y mejorando el tiempo de respuesta de los sistemas.

Abstract (in English, 250 words or less):

The size and variety of the data circulating in the current networks is exponentially increased and the IoT contributes significantly to this increase in volume, this implies a substantial increase in the traffic that reaches the data centers for analysis. This increase in traffic directly affects the response time of IoT systems, which in critical environments can be unacceptable.

The aim of this paper is to analyze a new computing paradigm called Edge Computing that brings together some of the analytical and processing capabilities currently offered by cloud systems to the place where the data is generated, thus reducing the volume of information that must be sent to the network and improving the response time of the systems**.**

Palabras clave (entre 4 y 8):

IoT, Cloud Computing, Edge Computing, Fog Computing, EdgeX Foundry

Índice

Lista de Ilustraciones

1 Introducción

1.1 Contexto y justificación del trabajo

La llegada de Internet y los avances tecnológicos de los últimos años han permitido que cada vez estemos más conectados con el entorno que nos rodea. El uso de teléfonos móviles y otros dispositivos inteligentes han pasado a formar parte de nuestra vida diaria. Dando lugar no sólo a una conexión entre personas, sino a una interconexión que engloba a cualquier elemento presente en nuestro entorno.

El Internet de las Cosas [1], también conocido como Internet of Things o más comúnmente por sus siglas IoT surgió de esta posibilidad real de interconectar objetos cotidianos (electrodomésticos, bombillas, semáforos, vehículos, …) a través de internet. Hoy en día IoT es una realidad, y según un informe de Gartner [2] se estima que en 2020 habrá más de 20.000 millones de objetos conectados a la red. Lo que supone un desafío para las infraestructuras centralizadas actuales. Hay que tener en cuenta que el despliegue masivo de sensores como parte del IoT, el incremento de transmisiones de vídeo 4K, la realidad virtual aumentada y los avances en otras tecnologías están provocando un incremento en el tráfico procedente de Internet que llega a los centros de datos.

Si bien la nube nos brinda acceso a cómputo, almacenamiento y conectividad de manera fácil y rentable, un aumento de tráfico en la red puede provocar que estos recursos centralizados creen retrasos y problemas de rendimiento cuando la fuente que genera los datos está lejos de la infraestructura que los procesa. Los datos recogidos por sensores y dispositivos de IoT se envían a centros de datos en la nube donde se almacenan y se procesan para obtener información útil que permita tomar decisiones inteligentes. Altas latencias en las respuestas por parte de la nube pueden resultar inaceptable en entornos que necesitan tomar decisiones en tiempo real.

Edge Computing [3] es un paradigma que intentan dar solución al problema de la latencia, evitando el envío masivo de datos a la red. Tiene como objetivo realizar un pre-procesado de los datos lo más cerca posible de la fuente que los genera, se eliminan datos erróneos, se les da formato y se realiza un pequeño procesado; la idea es desplazar parte de la inteligencia que ofrecen los centros de datos en la nube a los dispositivos cercanos al usuario, reduciendo de esta forma la cantidad de información que se debe enviar a infraestructuras en la nube para su almacenamiento y procesamiento.

Compañías como Google, Amazon y Microsoft han visto el potencial que tiene este cambio de paradigma sobre entornos de IoT y han empezado a adaptar sus ecosistemas de aplicaciones para llevar la capacidad de cómputo y almacenamiento a los dispositivos cercanos al usuario. Por su parte, Linux Foundation ha creado el proyecto EdgeX Foundry [5] para construir un framework común y abierto para la computación en el borde de IoT.

1.2 Objetivos del trabajo

Identificar los problemas actuales a los que se enfrentan los sistemas de IoT y qué soluciones y ventajas ofrece el paradigma de Edge Computing aplicado a dichos sistemas, evaluando y analizando el framework desarrollado por EdgeX Foundry que tiene como objetivo contribuir a la estandarización de IoT creando un marco de referencia común para Edge Computing.

Los objetivos generales de este trabajo son:

- Identificar las ventajas que ofrece y los problemas a los que intenta dar solución Edge Computing en entornos de IoT.
- Analizar y evaluar el framework desarrollado por EdgeX Foundry desde un punto de vista teórico y práctico.
- Seleccionar las herramientas necesarias para desplegar una plataforma EdgeX Foundry y evaluar su funcionalidad.

Los objetivos específicos de este trabajo son:

- Contribuir a entender mejor el enfoque de la tecnología Edge Computing en entornos IoT, donde se pasa de una infraestructura centralizada en la nube a una infraestructura descentralizada en el borde de la red.
- Evaluar de forma práctica la funcionalidad de la plataforma EdgeX Foundry.

El trabajo se estructurará en dos partes, una primera parte teórica en la que se profundizará en este nuevo cambio de paradigma, analizando de forma teórica las ventajas de esta infraestructura descentralizada en el borde de la red, los problemas a los que intenta dar solución en entornos de IoT y cómo el framework de EdgeX Foundry contribuye a la estandarización de estos sistemas. Y una segunda parte práctica en la que se pretende evaluar la viabilidad y funcionalidad de la arquitectura propuesta por EdgeX Foundry.

1.3 Enfoque y método seguido

Para conseguir los objetivos indicados, el primer paso será realizar un análisis teórico de las tecnologías que se pretenden abordar en este trabajo, este análisis se centrará en la búsqueda y lectura de artículos relacionados con dichas tecnologías, los pasos a seguir serán:

- Análisis teórico de los sistemas de IoT.
- Análisis teórico del modelo de computación Edge Computing.
- Análisis teórico de la arquitectura propuesta por EdgeX Foundry.

Una vez completada la base teórica, se pasará a la fase de análisis y evaluación de un sistema existente, el marco de referencia propuesto por el proyecto EdgeX Foundry, utilizando para ello la documentación oficial que el proyecto pone a disposición de cualquier interesado.

1.4 Planificación del Trabajo

En la Ilustración 1 se muestra las fases y los hitos que reflejan la planificación inicial de este proyecto.

Nombre	Fecha de inicio	Fecha de fin
Fase 0 - Plan de trabajo	21/09/18	12/10/18
Elección y compresión área del proyecto	21/09/18	28/09/18
Revisión junto con el consultor	25/09/18	25/09/18
Revsión junto con el consultor	29/09/18	29/09/18
Contexto, justificación y objetivos TFM	1/10/18	5/10/18
Revisión junto con el consultor	5/10/18	5/10/18
Revisión contexto, justificación y objetivos TFM	5/10/18	10/10/18
Elaboración diagrama de Gantt	10/10/18	12/10/18
Fase 1 - Contexto teórico	15/10/18	8/11/18
Internet of Thing	15/10/18	20/10/18
Edge Computing	22/10/18	26/10/18
EdgeX Foundry	27/10/18	2/11/18
Elección herramientas prueba de concepto	3/11/18	8/11/18
Fase 2 - Prueba de concepto	10/11/18	22/12/18
Despliegue de componentes	10/11/18	3/12/18
Validación componentes	4/12/18	14/12/18
Validación funcionalidad	15/12/18	20/12/18
Análisis de resultados	20/12/18	22/12/18
Fase 3 - Conclusiones y Entrega Final	29/12/18	5/01/19
Memoria TFM	29/12/18	5/01/19

Ilustración 1. Diagrama de Gantt

Las diferentes fases en las que se ha desglosado el trabajo se definen a continuación:

- Fase 0: Se llevó acabo un análisis inicial de las áreas en las que se podría centrar este proyecto, tras barajar varias opciones se optó por

Edge Computing al tratarse de una tecnología en auge aplicable a muchos ámbitos de IoT.

- Fase 1: Esta fase se corresponde con el estado del arte, la parte en la que se analiza de forma teórica los sistemas de IoT, el modelo de computación Edge Computing y la arquitectura propuesta por EdgeX Foundry.
- Fase 2: En esta fase se evalúa de forma práctica la funcionalidad de un sistema EdgeX, a través de los componentes que el proyecto EdgeX Foundry pone a disposición de los usuarios que quieran desplegar un sistema con las capacidades mínimas de edge.
- Fase 3: Finalmente, se documentarán en esta memoria las conclusiones extraídas del análisis teórico y práctico realizado.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos al finalizar este trabajo son:

- Máquina virtual sobre la que se han desplegado las imágenes de contenedores Docker que el proyecto EdgeX Foundry pone a disposición de cualquier usuario. Una vez que la imágenes han sido desplegadas, se dispone de un sistema que proporciona las capacidades de una plataforma edge mínima.

1.6 Breve descripción de los otros capítulos de la memoria

La memoria se ha estructurado en los siguientes capítulos:

- Capítulo 2 (Fase 1): Se corresponde con el estado del arte, en el que se ha realizado un análisis teórico de las diferentes tecnologías propuestas en la memoria. Analizando la arquitectura de los sistemas de IoT, la relación que tienen con Cloud Computing, los problemas a los que se enfrentan y a los que Edge Computing y EdgeX Foundry intentan dar solución.
- Capítulo 3 (Fase 2): Se realiza el despligue de una plataforma de EdgeX Foundry que nos proporciona las capacidades mínimas de una arquitectura edge y que nos permitirá evaluar la funcionalidad del sistema.
- Capítulo 4 (Fase 3): Conclusiones finales tras el análisis teórico y práctico realizado.

2 Estado del arte

2.1 Internet of Things

2.1.1 Introducción

El término de Internet of Things fue propuesto en 1999 por Kevin Ashton [6] en el Instituto de Tecnología de Massachusetts (MIT), donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores. Pensaba que si todos los objetos de nuestro entorno eran equipados con este tipo de tecnologías, los ordenadores podrían observar, identificar y comprender el mundo [7].

Hoy en día, IoT engloba al conjunto de tecnologías que permiten que objetos cotidianos puedan comunicarse a través de la red con el objetivo de recopilar información que nos permita supervisar el estado y comportamiento de dichos objetos. Un dispositivo IoT se caracteriza por ser un sistema electrónico de tamaño reducido equipado con un procesador, sensores que le permiten medir el entorno, actuadores que le permiten realizar determinadas acciones en respuesta a los datos recibidos y módulos de comunicación que utilizan protocolos de red. Un ejemplo de uso sencillo lo encontramos en los hogares inteligentes, que mediante la instalación de sensores en diferentes zonas de la vivienda, conectados a un sistema central, permiten, entre otras cosas, optimizar el gasto de luz, agua, y consumo energético.

El eje central de IoT son los datos recogidos por sensores y dispositivos, estos datos se envían a servidores remotos o a la nube para su tratamiento, una vez que se ha extraído de ellos la información que se considera importante, los dispositivos de IoT pueden recibir del servidor o de la nube una serie de instrucciones para realizar una determinada acción. En términos empresariales, el valor para las organizaciones está en la información que se puede extraer de dichos datos porque permite automatizar procesos, optimizar recursos y tomar mejores decisiones, lo que conlleva una mayor eficiencia operativa [8]. IoT tiene efectos notables en muchos ámbitos de la vida diaria (domótica, salud y bienestar, vehículos conectados, ...) y en sectores empresariales e industriales (logística, automatización y control de procesos de producción, seguridad, …). Ofrece grandes oportunidades de mercado y tiene un alto impacto económico.

El tamaño y variedad de los datos que circulan por las redes actuales está aumentado exponencialmente e IoT contribuye significativamente a este aumento del volumen Ilustración 2. Cisco estima que a finales del 2019 el IoT generará más de 500 zettabytes de datos al año, lo que implica un incremento

sustancial del tráfico que circula por las redes y llega a centros de datos en la nube para su tratamiento [9].

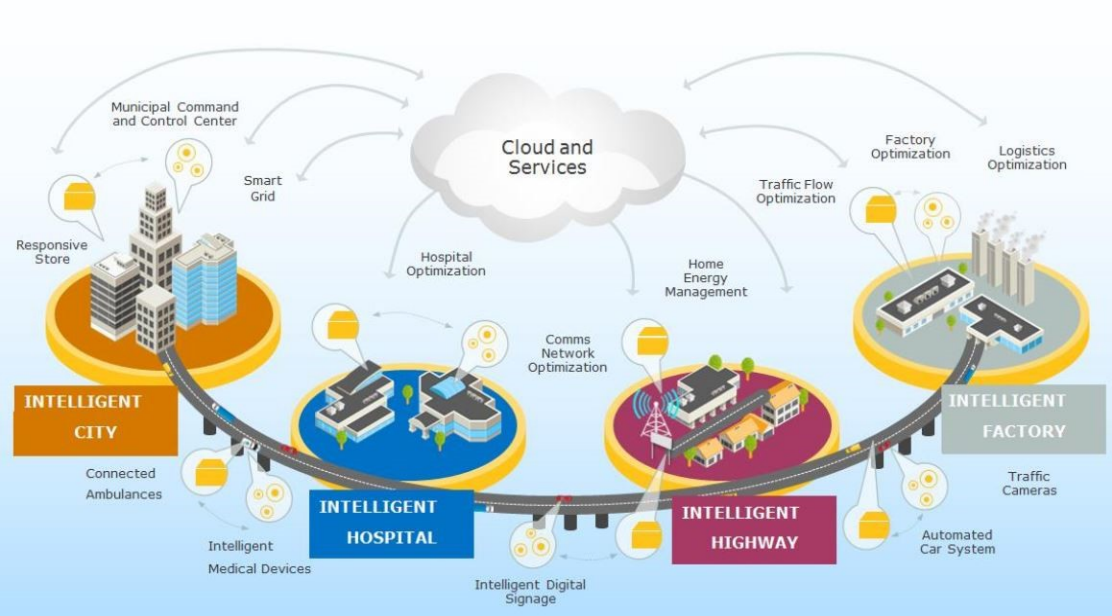

Ilustración 2. Escenarios de aplicación para IoT

Cloud computing [11] es una tecnología que ha impulsado el uso de sistemas de IoT ofreciendo las capacidades de almacenamiento y de cómputo necesarias para tratar y extraer información de los datos generados por dichos sistemas. Hablamos de arquitecturas en las que los datos se transfieren a infraestructuras centralizadas donde son procesados y luego devueltos al solicitante. Sin embargo, el tiempo de respuesta a la hora de almacenar, procesar y recuperar los datos de la nube puede verse afectado por el incremento de tráfico que se está produciendo en las redes, lo que puede resultar inaceptable en entornos que necesitan dar respuesta en tiempo real. Un vehículo autónomo no puede permitirse un retraso entre detectar una posible colisión y actuar en consecuencia (reducir la velocidad, detenerse, …), los sistemas de seguridad industriales, como alarmas contra incendios y detectores de humo tampoco pueden permitirse retrasos en la transmisión de datos. La latencia, definida como la cantidad de tiempo que tarda un servicio basado en la nube en responder a la solicitud de un usuario, es un factor crítico en estas infraestructuras centralizadas que dan soporte a sistemas de IoT. Por ello, las organizaciones necesitan soluciones que permitan procesar los flujos de datos en el menor tiempo posible para obtener respuestas inmediatas.

Reducir la latencia implica acercar el procesamiento a los dispositivos finales, los que generan y consumen datos. De hecho, acercar los servicios al

perímetro de la red ha sido un reto desde que Internet se convirtió en un servicio universal y la demanda y consumo de contenidos se generalizó. Prueba de ello son las redes de distribución de contenido, Content Delivery Network o CDN [10], cuyo objetivo es superar las limitaciones inherentes de Internet en términos de calidad de servicio percibida por el usuario, proporcionado servicios que mejoren el rendimiento de la red al maximizar el ancho de banda y mejorar la accesibilidad. Todo ello a través de una agrupación colaborativa de nodos en la red ubicados en las cercanías de los clientes.

2.1.2 Modelo de referencia

En la Ilustración 3. se muestra el modelo de referencia de IoT propuesto por Cisco en el IoT World Forum de 2014 [12].

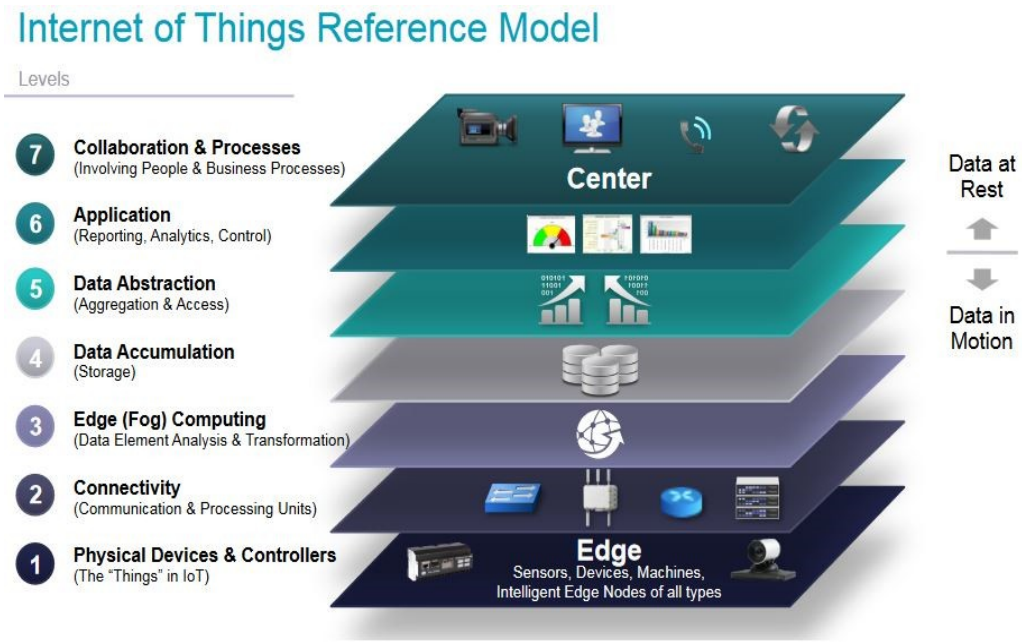

Ilustración 3. Modelo de referencia propuesto por Cisco para IoT

El modelo, definido como un compromiso entre siete capas o niveles, describe como debe operar cada uno de ellos para mantener la simplicidad, permitir una alta escalabilidad y garantizar la compatibilidad de todos los componentes de un sistema de IoT, en el que los datos fluyen en ambas direcciones, desde el nivel 1 al nivel 7 y viceversa.

- Nivel 1: Encontramos en este nivel los sensores, actuadores y dispositivos físicos que pueden generar, enviar y recibir datos a través de un protocolo de red.

- Nivel 2: Se corresponde con los equipos de red que permiten que los dispositivos del nivel 1 se puedan comunicar entre ellos y con niveles superiores. Generalmente, el nivel 1 y el 2 se encuentra en las instalaciones del cliente.
- Nivel 3: Tiene como objetivo realizar un análisis inicial del flujo de datos, puede verse como una capa intermedia entre el hardware y las infraestructuras remotas en la nube, se debe ubicar lo más cerca posible del origen de los datos, en el perímetro de la red, y se le da el nombre de Edge o Fog Computing. En este nivel se realiza una evaluación inicial de los datos para determinar si deben ser procesados por un nivel superior o por el nivel en el que se encuentran, se les da formato, se elimina datos erróneos, se comprimen, … se evita de esta forma sobrecargar la red con información poco útil, optimizando el uso de recursos de red y ancho de banda disponible.
- Nivel 4 al nivel 7: Se encuentran las infraestructuras remotas, las aplicaciones y los servicios que ofrece Cloud Computing, se convierten los datos en información y conocimiento, es dónde encontramos el mayor potencial de IoT.

Es importante señalar que el valor de IoT proviene de una combinación de Edge y Cloud Computing, no de lo uno o lo otro. Procesar los flujos de datos en el menor tiempo posible implica disminuir el alto grado de centralización que los servicios en la nube tienen en la actualidad, para ello hay que propiciar el análisis y la generación de conocimiento cerca del lugar en el que estos se generan. En este aspecto, la capa de Edge (Fog) Computing tiene un papel fundamental.

2.2 Edge Computing

2.2.1 Introducción

Edge Computing es un modelo de computación que acerca las capacidades de análisis y procesamiento al lugar en el que se generan los datos. Al no tener que enviarlos a infraestructuras remotas para su evaluación y análisis se reducen latencias, se mejora el tiempo de respuesta de las aplicaciones y disminuye el volumen de datos enviado a la red. Tiene un gran impacto en entornos de IoT que requieren respuesta en tiempo real o en aquellos cuya conectividad con la nube es limitada o esporádica. La virtualización y la computación distribuida son factores claves dentro de este modelo de arquitectura descentralizada que debe ser capaz de escalar horizontalmente.

El término Edge device*,* en este contexto*, se* refiere a elementos con capacidades limitadas que tienen su propio conjunto de recursos: CPU, memoria, almacenamiento y red. Pueden ser smartphones, smartglasses, smartwatches, tablets, routers, vehículos autónomos*,…,* o cualquier dispositivo de IoT con capacidad de proceso. Así, Edge Computing se refiere a cómo parte de los procesos que se realizan ahora en centros de datos en la nube se trasladan y se ejecutan en dispositivos Edge ó nodos Edge que en ocasiones pueden represantar pequeños centros de datos ubicados en las cercanías del cliente.

Una arquitectura de Edge Computing puede resultar compleja y se enfrenta a grandes desafíos derivados de la heterogeneidad de los dispositivos que la forman. La seguridad se presenta como un problema ya que todo dispositivo conectado es susceptible de ser atacado. La comunicación puede verse limitada debido a la movilidad de sus componentes, se deben gestionar servicios que pueden residir en parte en dispositivos locales y en parte en la nube, además se deben establecer mecanismos para asegurar la disponibilidad de la infraestructura local, formada por dispositivos que pueden aparecer y desaparecer aleatoriamente, y resulta necesario una capa intermedia de red que interactúe con los dispositivos edge y con la nube, con la flexibilidad necesaria para adaptarse a diferentes tipos de tráfico, seguridad y niveles de servicio requeridos. En este aspecto, las redes SDN (Software Defined Networking) se presentan como una solución. Aunque no hay una arquitectura de referencia definida para Edge Computing en la Ilustración 4 se muestra un ejemplo de los niveles o capas intermedias que deberían contemplar este tipo de arquitecturas.

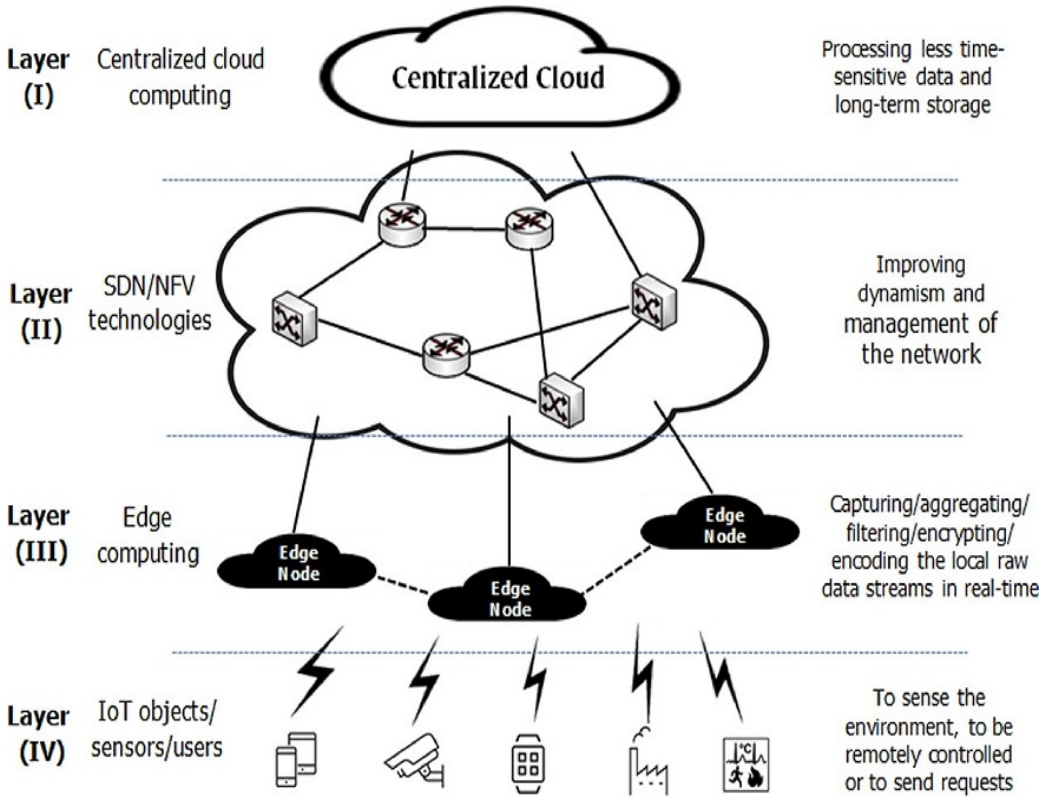

Ilustración 4. Arquitectura de ejemplo para Edge Computing [19]

Acercar el procesamiento a la fuente de datos a través de un proceso colaborativo de nodos no es un concepto nuevo, históricamente se encuentran estudios e investigaciones que datan de la década de los 90 [14]. Sin embargo, no es hasta 2016 que surgen iniciativas como Open Edge Computing [15] y Edge Computing Consortium [16], con el objetivo de promover el uso de Edge Computing y establecer un marco de referencia en materia de buenas prácticas, seguridad y eficiencia energética, impulsado, en gran medida, por el impacto económico que se espera que tenga en los próximos años. Así, Open Edge Computing define una arquitectura de Edge Computing como un conjunto de pequeños centros de datos [17] o Edge Nodes Ilustración 4 situados en las cercanías de los clientes que ofrecen capacidades de cómputo y almacenamiento.

Si revisamos la literatura disponible, un término asociado a Edge Computing es Fog Computing [13], acuñado por Cisco en 2012 para promover la escalabilidad en infraestructuras de IoT. La idea detrás de ambos conceptos es la misma, acercar el procesamiento a la fuente de datos. En este caso se habla de *Fog* o "niebla" y se ve como una forma de distribuir servicios de computación, comunicación, control y almacenamiento hacia los dispositivos finales.

Ambos términos se confunden con frecuencia, en algunas de las fuentes consultadas [18] se define Fog Computing como una arquitectura horizontal que abarca a Edge Computing y a la red requerida para llevar los datos preprocesados a su destino final. Con frecuencia, se sitúa la diferencia entre ambas arquitecturas en el lugar en el que se ubica la capacidad de procesamiento. En Fog Computing se establece en las cercanías de la red local donde el flujo de datos es enviado hacía nodos de niebla, *gateway de IoT* o pequeños servidores que procesan y enrutan el tráfico hacia donde se necesite Ilustración 5.

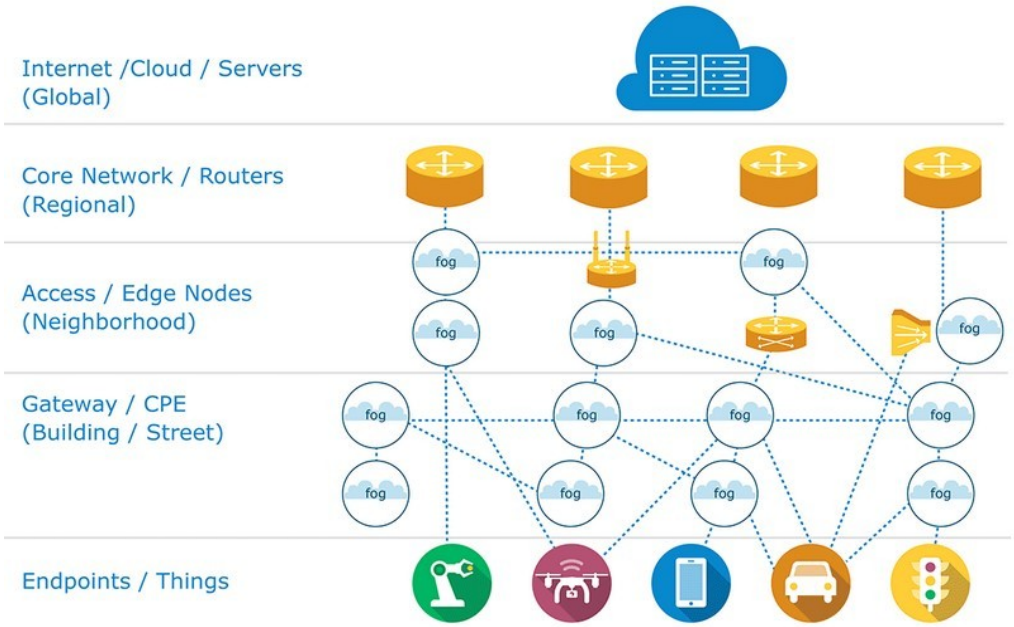

Ilustración 5. Visión general de Fog Computing

Sin embargo, en Edge Computing esa capacidad de procesamiento se sitúa en la fuente de datos o lo más cerca posible a esta, en sensores conectados a controladores programables, en la maquinaria de una fábrica, en vehículos autónomos, …, así un nodo de computación Edge se define como un hardware con capacidad de cómputo, situado físicamente cerca de los dispositivos o equipos que hacen uso de sus recursos.

Por lo tanto, Edge Computing permite habilitar el pre-procesado de los datos en la propia fuente sin que sea necesario disponer de conectividad con la red, esto favorece a aquellos sistemas que tienen una conectividad intermitente o reducida, como puede ser el caso de una plataforma petrolífera.

2.2.2 Ventajas

Muchos sectores (fabricación, salud, telecomunicaciones, finanzas, agricultura, ...) pueden verse beneficiados de las ventajas que ofrece acercar el procesamiento y el análisis a la fuente que genera los datos:

- Baja latencia: Parte de los datos se analizan cerca de la fuente, eliminando el viaje de ida y vuelta a la nube por lo que la latencia se reduce drásticamente. Permite a las organizaciones tratar datos importantes casi en tiempo real y actuar en consecuencia, mejorando el rendimiento y la eficiencia de servicios y aplicaciones. Muchos sistemas se verán beneficiados: vehículos autónomos que deben reaccionar en milisegundos para operar de forma segura, fábricas que necesitan reaccionar a eventos a medida que suceden, instituciones financieras que usan datos en tiempo real para informar de decisiones comerciales o controlar el fraude, etc.
- Mayor seguridad: Cuantos menos datos almacene un sistema en un entorno en la nube, menos vulnerable será si ese entorno se ve comprometido, se produce una descentralización de los datos. En este aspecto, una arquitectura de Edge Computing debe establecer los medios necesarios para que la seguridad sea la adecuada porque hay que tener en cuenta que los dispositivos e*dge* son susceptibles de ser atacados. Por ejemplo, con este enfoque de computación las organizaciones pueden filtrar información confidencial de identificación personal y procesarla localmente, enviando a la nube la información no confidencial para su posterior análisis.
- Menores costos: Procesar los datos cerca de la fuente implica: eliminar lecturas erróneas, comprimirlos, darles formato, …, reduciendo de esta forma el volumen de datos que se debe enviar a la red. Así la organizaciones podrán reducir el consumo de ancho de banda y los requisitos de almacenamiento y cómputo en infraestructuras en la nube. La escalabilidad que ofrece también es menos costosa porque permite a las empresas ampliar las capacidades de computación a través de dispositivos de IoT y pequeños centros de datos perimetrales.

De esta forma, Edge Computing se presenta como una solución para muchos servicios de IoT emergentes que demandan procesamiento en tiempo real y mejorar la eficiencia en la recopilación y análisis de grandes volúmenes de datos.

2.3 EdgeX Foundry

2.3.1 Introducción

Uno de los grandes retos a los que se enfrentan las soluciones de IoT es la interoperabilidad, la capacidad que tienen los sistemas o sus componentes de comunicarse entre sí. Solo hay que pensar en la diversidad de dispositivos que forman este tipo de soluciones donde entran en juego diferentes requisitos y protocolos de comunicación. En este aspecto, es necesario una colaboración entre fabricantes que permita alcanzar un conjunto de estándares comunes para que la comunicación entre dispositivos no se vea afectada por limitaciones técnicas o comerciales. De la misma forma, las plataformas de desarrollo también tienen que propiciar esa interoperabilidad entre dispositivos para impulsar y facilitar el despliegue de estas soluciones.

EdgeX Foundry [5] es un proyecto de código abierto dentro de Linux Foundation [20] que surgió en abril del 2017 con el apoyo de 50 organizaciones miembros Ilustración 6, y que actualmente cuenta con más de 70. Para ponerlo en marcha Dell aportó mas de una docena de microservicios y 125.000 líneas de código en Apache 2.0.

Ilustración 6. Miembros fundadores del proyecto EdgeX Foundry

El proyecto tiene como objetivo contribuir a la estandarización de IoT creando un marco de referencia común para Edge Computing que permita la interoperabilidad entre las aplicaciones y los estándares de conectividad existentes. Pone el foco en los sistemas de IoT industriales donde la complejidad, la fragmentación del mercado y la falta de un framework común

está dificultando su adopción. Esta iniciativa se centra en componentes de capacidades limitadas distribuidos geográficamente, cualquier elemento capaz de recopilar y procesar datos podrá hacer de enlace entre los dispositivos de IoT y los sistemas en la nube. No llega al nivel del sensor, pero el proyecto asegura que los dispositivos (routers, switches, servidores, ...) que los conectan hablen todos el mismo idioma Ilustración 7.

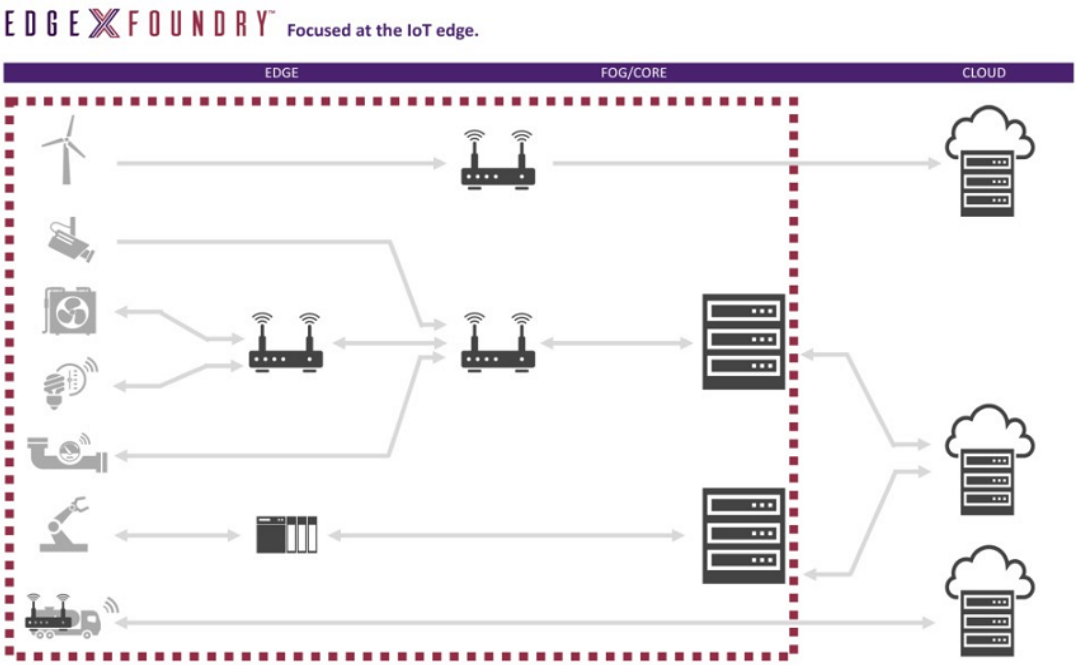

Ilustración 7. EdgeX Foundry centrado en dispositivos IoT Edge

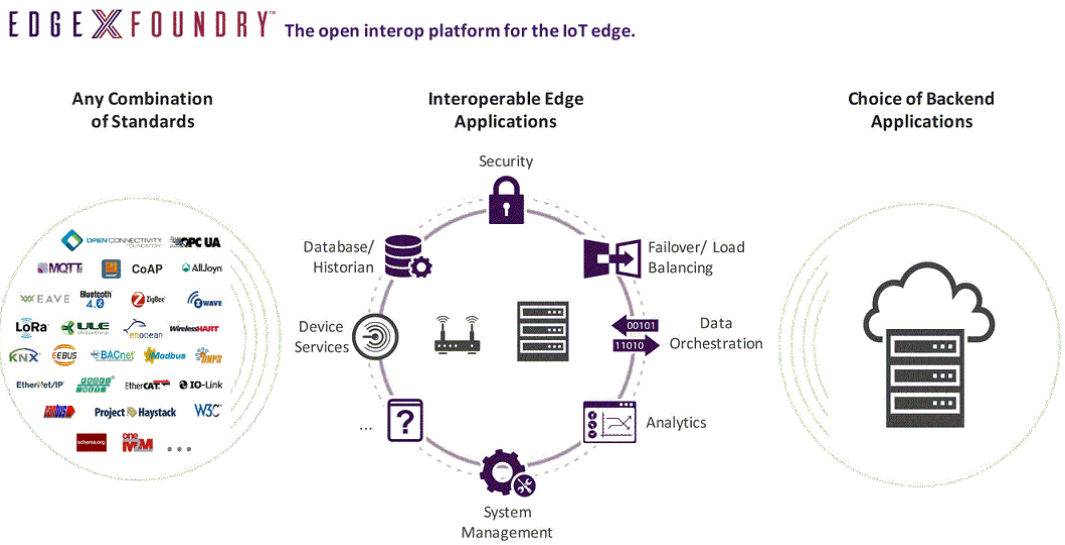

Ilustración 8: EdgeX Foundry como plataforma interoperable

De esta forma, mediante un ecosistema de componentes interoperables definidos por software, independientes del hardware y del sistema operativo, las organizaciones podrán integrar con facilidad los estándares seleccionados para los dispositivos de IoT con sus soluciones de Backend (locales o en la nube) Ilustración 8.

2.3.2 Objetivos

Los objetivos del proyecto EdgeX Foundry pasan por:

- Fomentar el crecimiento de las soluciones de IoT industriales a través de una comunidad que crea y mantiene servicios y APIs comunes.
- Promocionar EdgeX como una arquitectura abierta y común para el uso de Edge Computing.
- Alentar a la comunidad de proveedores de soluciones de IoT industriales a colaborar con la creación de un ecosistema de componentes *plug-and-play* interoperables.
- Certificar los componente de EdgeX para garantizar su interoperabilidad y compatibilidad.
- Proporcionar herramientas para crear de forma rápida soluciones e*dge* para el mercado de IoT industrial basadas en componentes de EdgeX que se puedan adaptar fácilmente a las necesidades cambiantes de los negocios.
- Colaborar con proyectos de código abierto, grupos de estándares, y crear alianzas con la industria para garantizar la coherencia y la interoperabilidad en todos los niveles de IoT Ilustración 3.

2.3.3 Organización

Se distinguen tres categorías de miembros dentro de la comunidad del proyecto: *Platinum, Silver* y *Associate* [21]. Sin embargo, como proyecto de código abierto se da la bienvenida a cualquiera que contribuya y se involucre en la comunidad de desarrollo [22]. Solo los miembros *Platinum* y *Silver* podrán formar parte del Consejo Directivo del proyecto. Los miembros *Associate* se limitan a organizaciones sin ánimo de lucro, proyectos de código abierto y entidades gubernamentales, se requerirá la aprobación del Consejo Directivo.

Los tres tipos de miembros tienen derecho a participar en reuniones generales del proyecto, iniciativas y cualquier otra actividad, de la misma forma podrán identificarse como miembros o participantes del proyecto EdgeX Foundry.

El trabajo técnico del proyecto se define a través del Comité de Dirección Técnica [23], responsable de la supervisión del código, de los procesos de publicación y de la comunidad técnica. El presidente de este comité es miembro del Consejo Directivo.

Tal como se muestra en la Ilustración 9 el Comité de Dirección Técnica (*Technical Steering Committee - TSC*) está compuesto por grupos de trabajo, cada uno de estos grupos es una unidad organizativa centrada en un dominio técnico específico (Seguridad, DevOps, SDK, QA y Test …). A su vez, cada grupo de trabajo se encarga de uno o más proyectos, y cada proyecto entregará un elemento de trabajo, como puede ser un nuevo desarrollo o la modificación de un desarrollo existente. Todos los proyectos tiene una página en EdgeX Foundry Wiki [24]. Compañías miembros como Dell, tienen equipos de desarrollo trabajando exclusivamente para el proyecto.

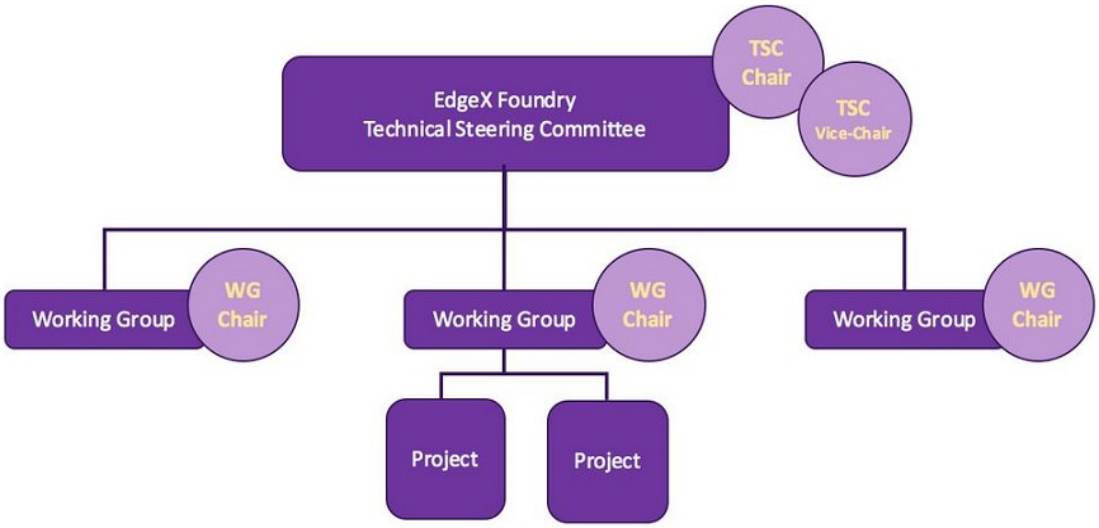

Ilustración 9. Organización del proyecto EdgeX Foundry

Desde su lanzamiento se realizan dos revisiones formales del proyecto al año Ilustración 10.

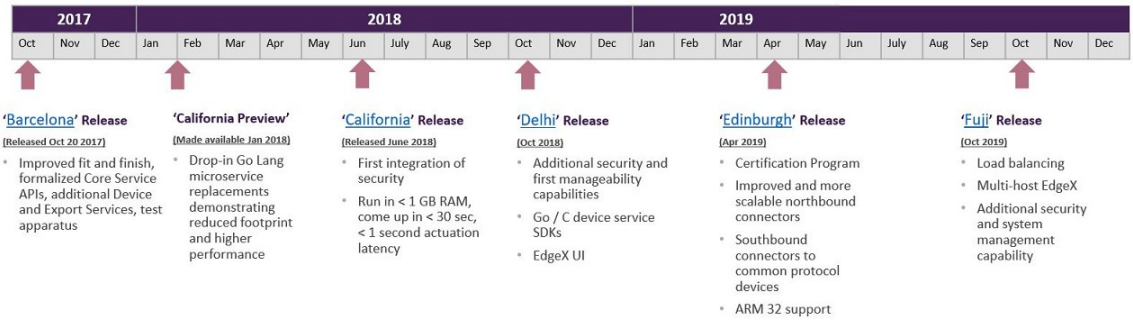

Ilustración 10. Revisiones formales del proyecto EdgeX Foundry

2.3.4 Arquitectura

EdgeX Foundry se define a través de una arquitectura modular basada en microservicios [25] que se comunican a través de APIs, lo que permite convertir aplicaciones complejas en procesos simples, interoperables y reutilizables, que se pueden modificar e implementar de forma independiente, y que pueden ser desplegados con facilidad. Las arquitecturas basadas en microservicios permiten un escalado eficiente y horizontal en función de la demanda.

El código del proyecto se puede obtener desde los repositorios de GitHub [26], los microservicios que ofrece EdgeX se pueden desplegar desde varios puntos de vista:

- Visión de los contribuyentes: Es un enfoque dirigido a desarrolladores, estos pueden descargar el código del proyecto y generar e implementar servicios que pueden ser desplegados en las plataformas elegidas.
- Visión de los usuarios: Está dirigido a usuarios que no tienen la necesidad de construir sus propios servicios, así que se pueden descargar las imágenes de contenedores de EdgeX y desplegarlas y ejecutarlas en plataformas que tengan Docker [27] instalado lo que les proporciona las capacidades mínimas de una plataforma edge.
- Visión híbrida: Es una mezcla de las dos anteriores, algunos servicios pueden ser implementados y desplegados utilizando el código del proyecto, y otros servicios se desplegarán y ejecutarán usando las imágenes de contenedores de Docker que ofrece EdgeX.

La arquitectura de EdgeX Foundry se ha definido en base a una serie de principios:

- Debe ser independiente del hardware, del sistema operativo, de sensores y protocolos, y permitir la distribución de funcionalidades a través de microservicios en cualquier nivel de IoT desde los dispositivos edge hasta la nube.
- Debe ser extremadamente flexible. Cualquier parte de la plataforma se podrá actualizar, reemplazar o incrementar con microservicios u otros componentes de software. Escalará en función de la demanda y de la capacidad de los dispositivos.
- Debe proporcionar capacidad de almacenamiento y enrutamiento.
- Debe propiciar la “inteligencia” cerca de los dispositivos edge para poder abordar preocupaciones referentes a la latencia, consumo de ancho de banda y almacenamiento.
- Debe ser segura y fácil de administrar.

La Ilustración 11 muestra la organización de estos microservicios que definen la arquitectura, el sistema se distribuye en cuatro capas o niveles (***Core*** *Services,* ***Supporting*** *Services,* ***Export*** *Services* y ***Device*** *Services*), y en dos servicios subyacentes a todo el sistema (***Security*** y ***Management***).

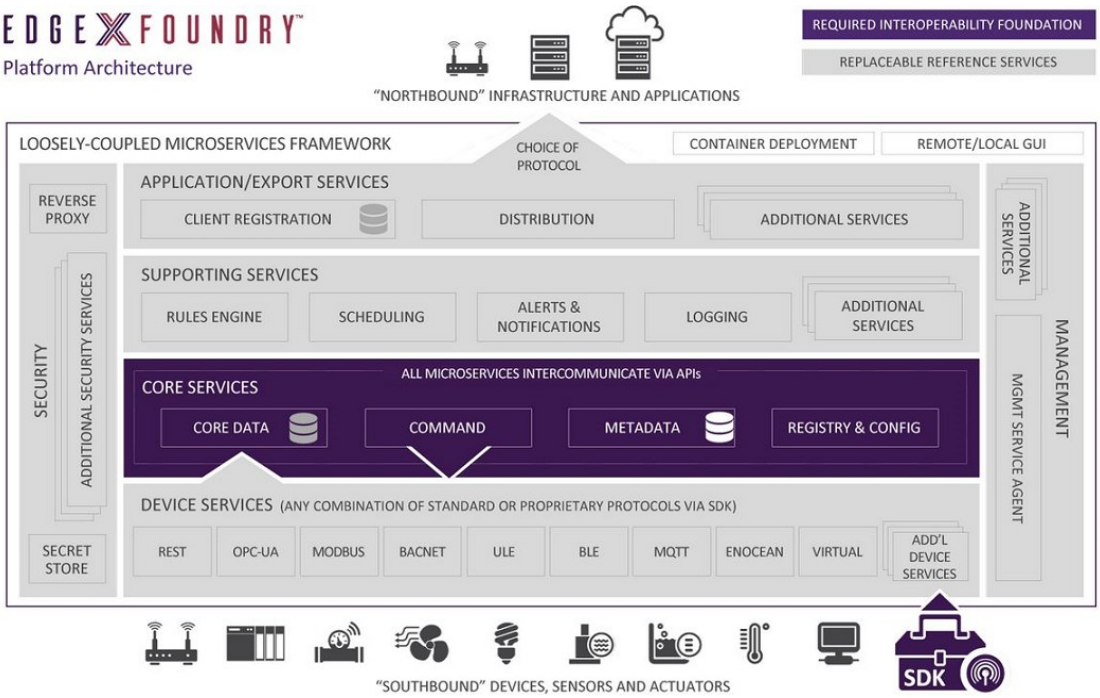

Ilustración 11. Arquitectura de EdgeX Foundry

En el “Southbound” o “lado sur” se encuentran todos los objetos de IoT, la parte física, que engloba a todo elemento que se comunica con sensores, actuadores y otros dispositivos de IoT para recopilar información de ellos. En el extremo superior, “Northbound” o “lado norte” se encuentra las infraestructuras remotas, este lado incluye la parte de red que se comunica con la nube, donde se almacenan y analizan los datos para convertirlos en información.

Las capas de la arquitectura proporcionan un motor de transformación Ilustración 12, los datos provenientes de sensores y otros dispositivos se traducen y se les da formato, posteriormente son entregados a aplicaciones y sistemas en la nube, teniendo en cuenta los protocolos y estructuras seleccionadas por el cliente.

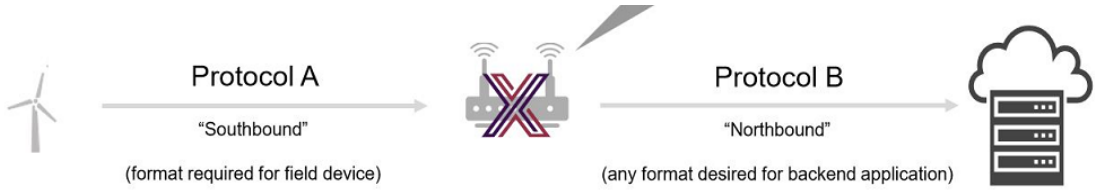

Ilustración 12. EdgeX Foundry como motor de transformación de datos

En este aspecto, hay que tener en cuenta que es necesario que los datos recolectados que llegan al lado norte hayan sufrido una serie de transformaciones:

- Se han filtrado (eliminación de lecturas erróneas, …).
- Se han transformado a un modelo de datos común.
- Se han enriquecido (metadatos, localización, …).
- Se han formateado (XML, JSON, CSV, …).
- Se han comprimido y encriptado (AES, ZIP, GZIP, ...).

Cada uno de los niveles que forma la arquitectura de EdgeX Foundry Ilustración 11 proporcionan una serie de microservicios y funciones:

- **Device Services:** Los microservicios de este nivel comunican y recopilan datos de sensores, actuadores y otros dispositivos de IoT a través de los protocolos nativos de cada elemento, una vez recopilados son convertidos a una estructura de datos común de EdgeX Foundry para ser enviados al nivel superior o a otros microservicios. La plataforma proporciona un kit de desarrollo SDK para facilitar la creación de servicios de dispositivos propios.

  Ejemplos de lo que hacen los servicios de dispositivos pueden ser:

  - Un servicio de dispositivo BACNet convierte las lecturas de temperatura y humedad recogidas por un dispositivo BACNet en una estructura de datos común de EdgeX Foundry.
  - Un servicio de dispositivo recibe y traduce comandos desde otros servicios de EdgeX o sistemas empresariales y los traslada a los dispositivos de IoT en un lenguaje que puedan entender. Por ejemplo, proteger equipos industriales si se detecta sobrecalentamiento, el servicio recibirá la solicitud de apagar un motor controlado por un PLC Modbus, la solicitud genérica de "apagar" se convertirá en un comando serial Modbus que el motor controlado por el PCL entenderá y ejecutará.

- **Core Services:** Este nivel separa la capa del lado norte de la capa del lado sur. Entre sus funciones podemos destacar:
  - Proporciona almacenamiento persistente temporal y un servicio de administración para los datos recopilados por sensores y dispositivos de IoT en el nivel inferior.
  - Ofrece un servicio que facilita y controla las solicitudes y peticiones que se envían del lado norte al lado sur.
  - Administra la información de dispositivos y sensores conectados a EdgeX Foundry, conoce el tipo y organización de los datos que generan y cómo comunicarse con ellos. Proporciona la capacidad de aprovisionar nuevos dispositivos y vincularlos con sus servicios de dispositivo.
  - Proporciona un registro de microservicios y la configuración de estos.
- **Supporting Services:** Este nivel abarca un gran número de microservicios que proporcionan "inteligencia" y análisis cerca de los dispositivos edge. Pensemos en la protección de maquinaria industrial por sobrecalentamiento, capas inferiores recolectarán y transformarán a una estructura común los niveles de temperatura, en este nivel se analizarán "localmente", si se sobrepasa un umbral establecido, se emitirá una orden que pasa al nivel inferior en forma de comando para "apagar" el equipo. Todo ello, sin tener que enviar la información a infraestructuras remotas en la nube para su análisis.
- **Export Services:** Una de las características de EdgeX Foundry es la capacidad de operar independientemente de otros sistemas cuando sea necesario, a través de gateways que a menudo puedan actuar en entornos aislados o desconectados del "lado norte" durante largos periodos de tiempo. Los microservicios de este nivel realizan las siguientes actividades:
  - Permitir a los clientes registrar los datos relevantes para ser analizados en local.
  - Permitir a los clientes externos (aplicaciones y servicios en la nube) registrarse como consumidores de los datos que recopilan los sensores y otros dispositivos de IoT.
  - Determinar qué datos serán filtrados, cómo serán formateados, encriptados y comprimidos (JSON, XML, AES, GZIP, ZIP…) y cómo se entregarán (REST, MQTT).

    - Enrutar parte de los datos y de la “inteligencia” recolectada por niveles inferiores hacia la nube o sistemas remotos.

Los servicios subyacentes a todo el sistema, **Security** y **System Management,** proporcionan:

- Mecanismos de protección para datos, sensores y otros objetos de IoT administrados por EdgeX Foundry.
- Administración central para iniciar/detener/reiniciar los servicios de EdgeX y métricas que permiten monitorizar dichos servicios.

Como se ha podido comprobar la arquitectura propuesta por EdgeX Foundry se presenta como una alternativa muy interesante para aquellas organizaciones que quieran implementar un sistema de IoT siguiendo el modelo de computación propuesto por Edge Computing.

3 Prueba de concepto

EdgeX Foundry es un colección de microservicios que pueden ser desplegados para proporcionar las capacidades de una plataforma edge mínima. El código fuente de estos microservicios puede ser descargado para que los desarrolladores interesados puedan construir y desplegar sus propios componentes. Los usuarios que no estén interesados en construir sus propios servicios tienen la opción de descargar las imágenes de EdgeX Foundry y ejecutarlas en contenedores Docker.

Con esta prueba de concepto se pretende validar desde un punto de vista práctico la documentación [28] y las imágenes de contenedores Docker del proyecto EdgeX Foundry, analizando la funcionalidad de los microservicios que el proyecto pone a disposición de cualquier interesado.

3.1 Requisitos previos

La documentación dirigida a aquellos usuarios que quieran obtener las imágenes de Docker de EdgeX Foundry y ejecutarlas establece los siguientes requisitos mínimos para el sistema en el que se desplegarán dichas imágenes:

- **Memoria:** Mínimo 1 GB
- **Espacio en disco duro:** Mínimo 3GB para ejecutar los contenedores de EdgeX Foundry, pero la estimación de espacio dependerá en gran medida del tiempo que se desee retener los datos recogidos por sensores y otros dispositivos.
- **Sistema operativo:** En este aspecto el proyecto indica que EdgeX Foundry se ha probado con éxito en muchos sistemas operativos, entre los que se incluyen, Windows (ver 7 - 10) , Ubuntu Desktop (ver 14-16), Ubuntu Server (ver 14) , Ubuntu Core (ver 16) y Mac OS X 10.

Como EdgeX Foundry es un sistema completamente independiente del sistema operativo y del hardware para la prueba de concepto se ha usado una máquina virtual creada con Oracle VM VirtualBox con las siguientes características.

Sistema Operativo: CentOS Linux release 7.6.1810 (Core)

Memoria: 2GB

Disco duro: 15GB

Número de CPU: 1

Una vez que el usuario ha decidido en qué sistema ejecutará las imágenes de EdgeX Foundry debe realizar las siguientes acciones:

1. Instalar Docker y Docker Compose
2. Descargar el fichero EdgeX Foundry Compose
3. Ejecutar EdgeX Foundry

3.2 Instalación de Docker y Docker Compose

La documentación del proyecto EdgeX Foundry nos remite a la documentación del proyecto Docker donde se especifican los pasos a seguir para la instalación de ambos componentes. Docker está disponible en dos versiones:

- Community Edition (CE): Edición para usuarios y desarrolladores que quieran empezar a experimentar con aplicaciones basadas en contenedores.
- Enterprise Edition (EE): Edición empresarial para la creación, ejecución y distribución de aplicaciones empresariales basadas en contenedores.

Para la prueba de concepto se utilizará la versión Docker CE, los pasos seguidos para la instalación en Centos a través de los repositorios de Docker se resumen a continuación.

Instalación de Docker CE en Centos 7

```
// Instalación de los paquetes necesarios que no incluye la instalación mínima de Centos7
[root@edgex ~]# yum install yum-utils.noarch

// Configuración de los repositorios de Docker
[root@edgex ~]# yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo

// Se habilitan repositorios edge y test de Docker (opcional)
[root@edgex ~]# yum-config-manager --enable docker-ce-edge
[root@edgex ~]# yum-config-manager --enable docker-ce-test

// Se instala Docker CE
[root@edgex ~]# yum install docker-ce.x86_64
```

Una vez instalado el Docker CE Ilustración 13 se procede con la instalación de Docker Compose, que es una herramienta que permite definir y administrar un conjunto de contenedores que se relacionan entre ellos, esta herramienta utiliza el fichero docker-compose.yml en el que se especifican todas las imágenes de contenedores que definen una aplicación. En este aspecto hay que tener en cuenta que las aplicaciones basadas en microservicios utilizan

múltiples contenedores, cada microservicio se ejecuta en un contenedor de forma aislada.

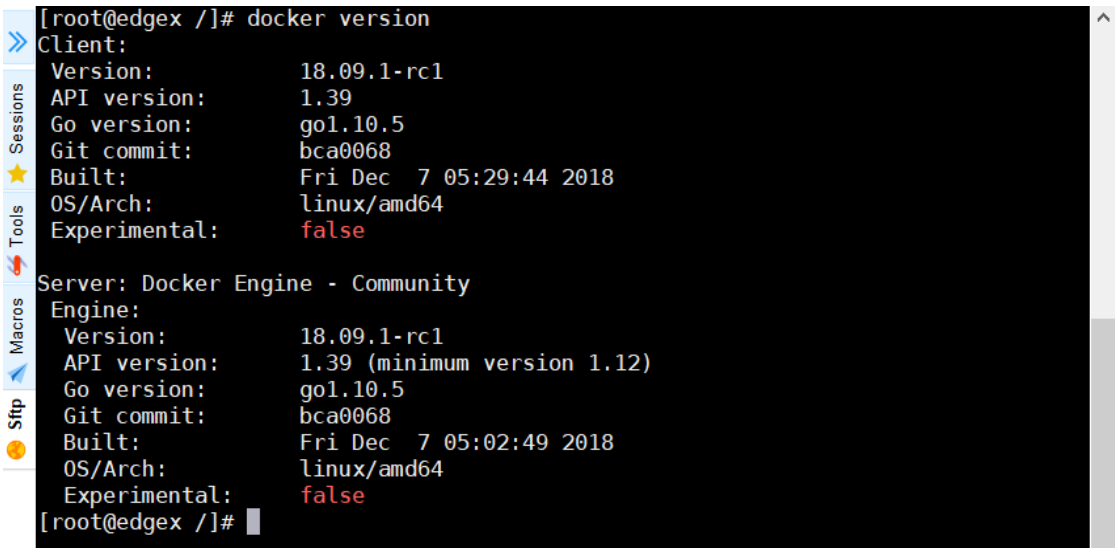

Ilustración 13. Instalación Docker CE

Los pasos seguidos para la instalación de Docker Compose se resumen a continuación:

Instalación de Docker Compose en Linux

```
// Descargamos la última versión de Docker Compose
[root@edgex ~]# curl -L
"https://github.com/docker/compose/releases/download/1.23.1/docker-compose-$(uname -
s)-$(uname -m)" -o /usr/local/bin/docker-compose

// Asignamos permisos de ejecución al binario
[root@edgex ~]# chmod +x /usr/local/bin/docker-compose
```

Una vez descargado y configurado verificamos la correcta instalación de Docker Compose Ilustración 14.

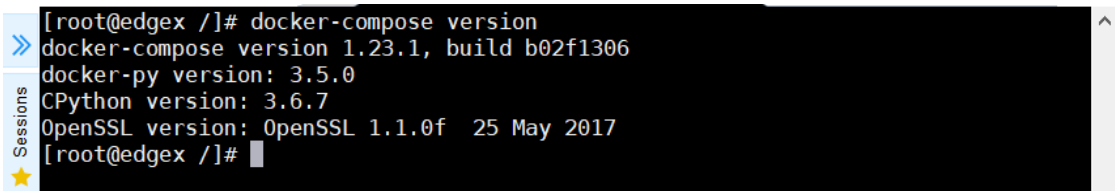

Ilustración 14. Instalación Docker Compose

Como se ha podido verificar la instalación de Docker es un proceso sencillo que no requiere de un conocimiento profundo del producto.

3.3 Obtener EdgeX Foundry Compose

Una vez que Docker ha sido instalado el siguiente paso es obtener el fichero Docker Compose de EdgeX Foundry Ilustración 15 disponible en GitHub [26].

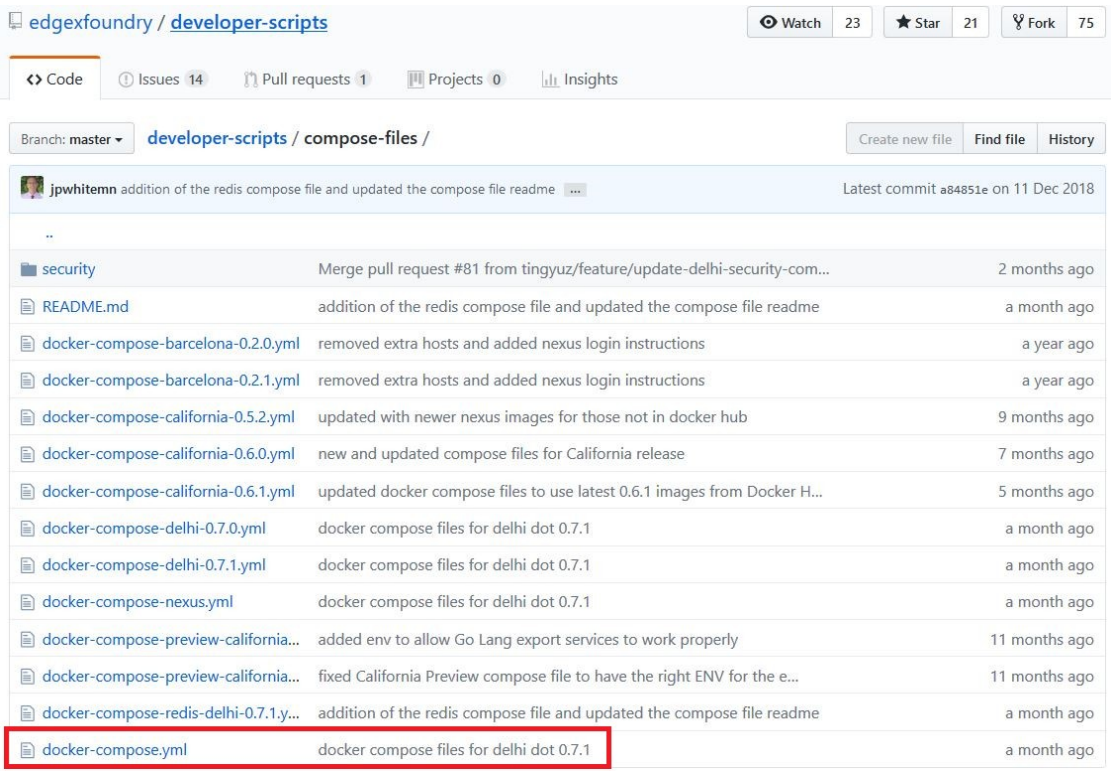

Ilustración 15. Fichero EdgeX Foundry Compose

Este fichero contiene:

- Las imágenes de contenedores Docker que deben descargarse
- El orden en que deben iniciarse los contenedores
- Los parámetros bajos los cuales se deben ejecutar los contenedores

Recoge la definición de todos los microservicios que forman la arquitectura de EdgeX Foundry y cada uno de ellos se desplegará en su propio contenedor de Docker. Los usuarios más avanzados, podrán descargarse estos contenedores de forma individual, todas las imágenes del proyecto se encuentran disponibles en el repositorio público Docker Hub.

Como se puede ver en la ilustración Ilustración 15 existen diferentes versiones del fichero Compose, cada una de ellas correspondientes a las diferentes

revisiones oficiales del proyecto que se han ido realizando Ilustración 10. Se recomienda usar la más reciente, que en este caso es Delhi (versión 0.7.1), pero el usuario puede descargase cualquiera de las disponibles y renombrar el fichero a docker-compose.yml. Para la prueba de concepto se ha descargado la versión más reciente Ilustración 16.

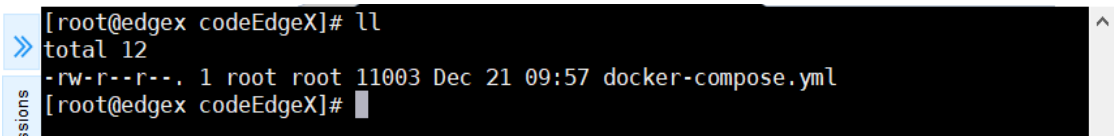

Ilustración 16. Fichero EdgeX Foundry Compose Delhi (version 0.7.1)

3.4 Ejecutar EdgeX Foundry

Una vez descargado el fichero EdgeX Foundry Compose con las definiciones de todos lo microservicios que forman la arquitectura base procedemos a descargar y ejecutar las imágenes de Docker en nuestro sistema.

Descarga y ejecución de las imágenes de Docker de EdgeX Foundry

```
// Se descargan las imágenes a local
[root@edgex testEdgeX]# docker-compose pull

// Se inician los contenedores descargados en segundo plano
[root@edgex testEdgeX]# docker-compose up -d

// Se verifica que los contenedores se han iniciado correctamente
[root@edgex testEdgeX]# docker-compose ps
```

En la Ilustración 17 se muestran los contenedores una vez han sido descargados e iniciados.

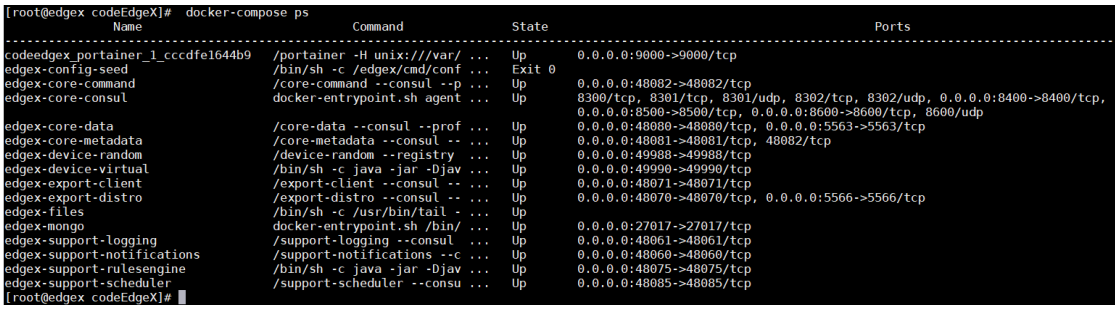

Ilustración 17. Contenedores Docker proyecto EdgeX Foundry

Con el propósito de entender cómo EdgeX funciona se muestra a continuación un breve resumen de los contenedores que se han desplegado e iniciado en mi máquina virtual.

Contenedor Docker	Puerto de conexión	Breve descripción
edgex-mongo	27017	Instancia de MongoDB, es la base de datos predeterminada para todo EdgeX.
edgex-core-consul	8500	EdgeX usa Consul para registrar los servicios y su configuración.
edgex-core-data	48080	Core Data, almacenamiento persistente para los datos recogidos por sensores y otros dispositivos.
edgex-core-metadata	48081	Core Metadata, conoce los dispositivos y sensores conectados al sistema y cómo comunicarse con ellos.
edgex-core-command	48082	Core Command, permite enviar comandos o acciones a dispositivos y sensores en nombre de otros servicios, aplicaciones o sistemas externos.
edgex-support-scheduler	48085	Support Scheduling, permite iniciar diversos eventos o acciones en un horario programado, como el borrado de datos antiguos.
edgex-support-logging	48061	Support Logging, servicio central de registro para todos los microservicios.
edgex-support-notifications	48060	Support Notifications, servicio central de alertas y notificaciones para todos los microservicios.
edgex-support-rulesengine	48075	Support Rules Enginee, microservicios que utilizan el sistemas de gestión de reglas Drools, supervisan los datos entrantes de sensores y dispositivos en busca de lecturas que estén dentro de rangos que requieran la activación de acciones inmediatas.
edgex-export-client	48071	Export Client, permite al cliente registrase como destinatario de los datos que llegan del Core Data. El

		cliente puede ser un sistema remoto en la nube, al que los datos serán enviados.
edgex-export-distro	48070	Export Distribution, recibe datos desde Core Data, a través de una cola de mensajes, a continuación los filtra, transforma y formatea en base a los requisitos del cliente, y finalmente los distribuye a su punto final mediante el protocolo seleccionado.
edgex-device-virtual	49990	Software que simula el comportamiento de un sensor con el propósito de explorar la funcionalidad de EdgeX cuando no se dispone de datos reales.

EdgeX proporciona una serie de comandos y utilidades que nos permiten verificar el estado y funcionamiento de los servicios que ofrece la plataforma una vez que los contenedores han sido desplegados e iniciados.

- Verificar con el servicio de registro y configuración de EdgeX que todos los servicios se han iniciado correctamente:

  ```
  http://<hostname>:8500
  ```

- Verificación de los logs de cualquier contenedor:

  ```
  docker-compose logs -f --tail 50 [compose name]
  Ejemplo: docker-compose logs f --tail 50 coredata
  ```

- Verificar la disponibilidad de los microservicios a través del servicio “ping”:

  ```
  http://<hostname>:<service port>/api/v1/ping
  ```

- Verificar el número de eventos recogidos:

  ```
  http://<hostname>:48080/api/v1/event/count
  ```

- Verificar los dispositivos conectados:

  ```
  http://<hostname>:48081/api/v1/device
  ```

- Verificar eventos/lecturas de un dispositivo concreto:

  ```
  http://<hostname>:48080/api/v1/event/device/<id>/<count limit>
  ```

El servicio de registro de EdgeX es una utilidad muy interesante que nos permite chequear el estado de todos los contenedores desplegados y la configuración de cada uno de ellos. La máquina virtual que se ha creado para el despliegue de las imágenes de contenedores Docker de EdgeX tiene la IP 192.168.1.131, así el acceso al servicio de registro de EdgeX se realizará a través de la URL: http://192.168.1.131:8500.

Como se observa en la Ilustración 18, la herramienta nos muestra el estado de todos los contenedores desplegados, seleccionando cualquier servicio, en este caso edgex-core-data, vemos con más detalle su estado de salud. Así, con esta utilidad he podido verificar que todos mis servicios se han registrado correctamente y están en ejecución, es una forma rápida de comprobar el estado de nuestro entorno EdgeX.

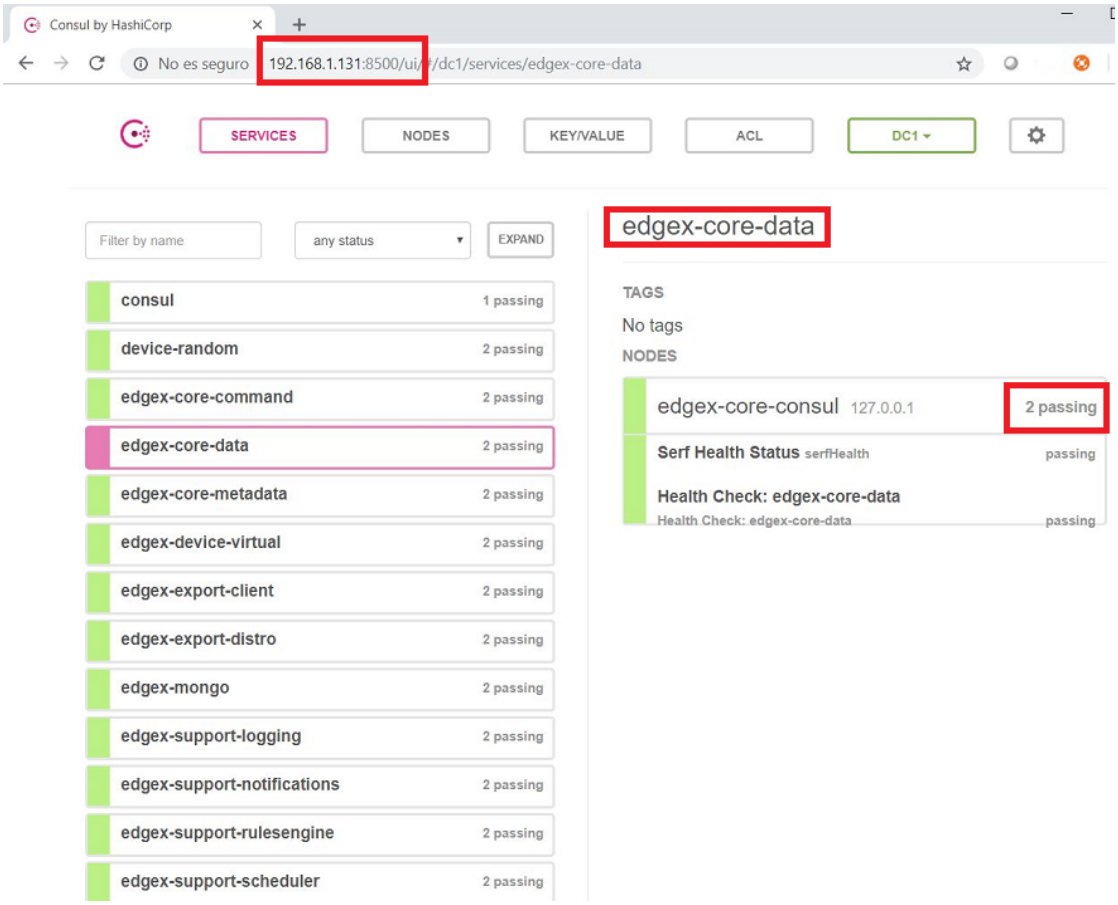

Ilustración 18. Servicio de registro y configuración de EdgeX

De la misma forma, si detenemos uno de los contenedores, comprobamos que el servicio pasa a un estado de fallo al no estar en ejecución Ilustración 19.

```
[root@edgex codeEdgeX]# docker-compose stop device-virtual
Stopping edgex-device-virtual ... done
```

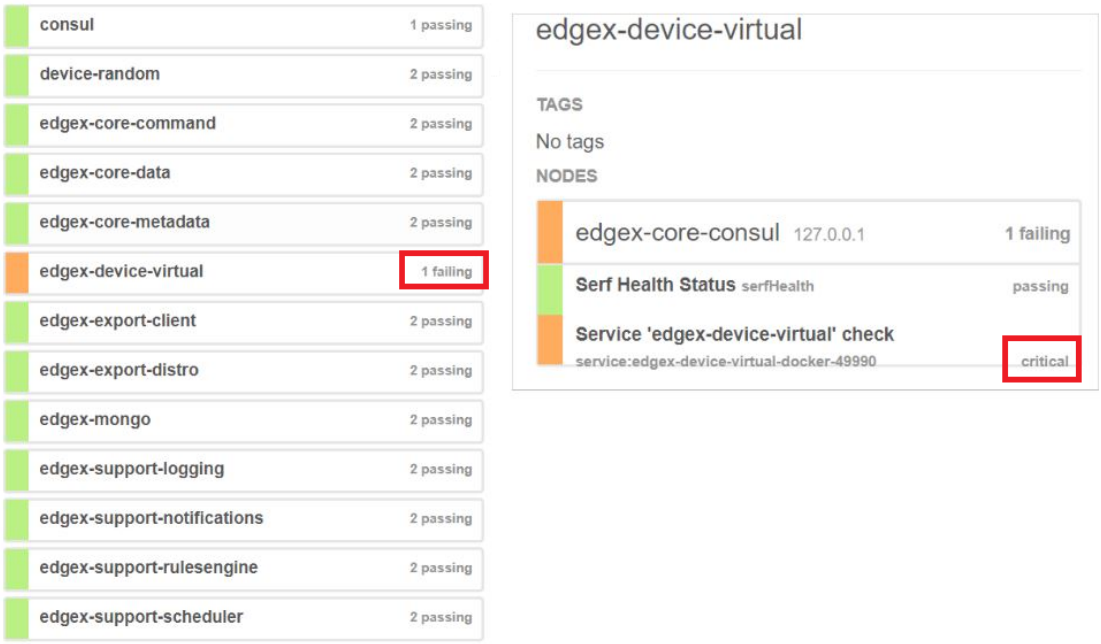

Ilustración 19. Chequeo servicio device-virtual de EdgeX

Otro aspecto a tener en cuenta del servicio de registro y configuración de EdgeX es que nos proporciona toda la configuración de cualquier servicio, la podemos visualizar a través de la pestaña “KEY/VALUE” de la herramienta Ilustración 20, así de manera centralizada podemos no sólo monitorizar el estado de nuestra plataforma sino también acceder a la configuración de cualquier aplicación.

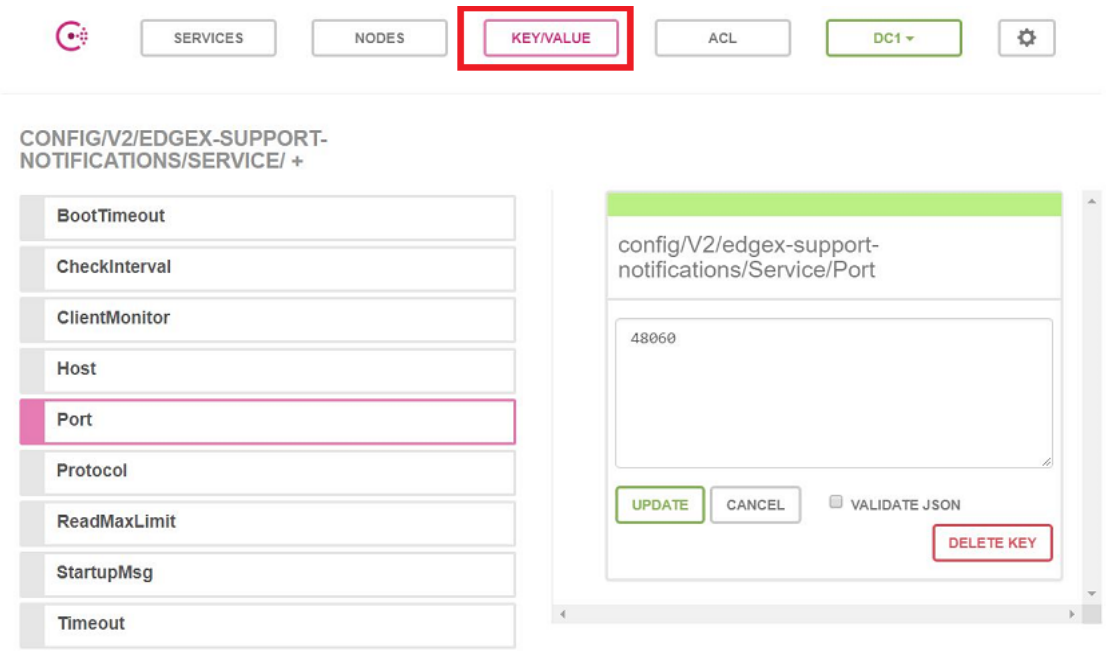

Ilustración 20. Puerto de acceso al servicio Support-Notifications de EdgeX

Cada uno de los microservicios de EdgeX ha sido diseñado para responder a una solicitud HTTP de “ping”, lo que permite chequear la disponibilidad de dichos microservicios a través de peticiones HTTP al recurso “ping” de la API: **http://[host]/[puerto]/api/vi/ping**.

Esta funcionalidad se puede probar a través de cualquier cliente que nos permita hacer este tipo de peticiones, para esta prueba de concepto se ha usado el cliente Postman, verificando que los microservicios responden correctamente a peticiones “ping” Ilustración 21.

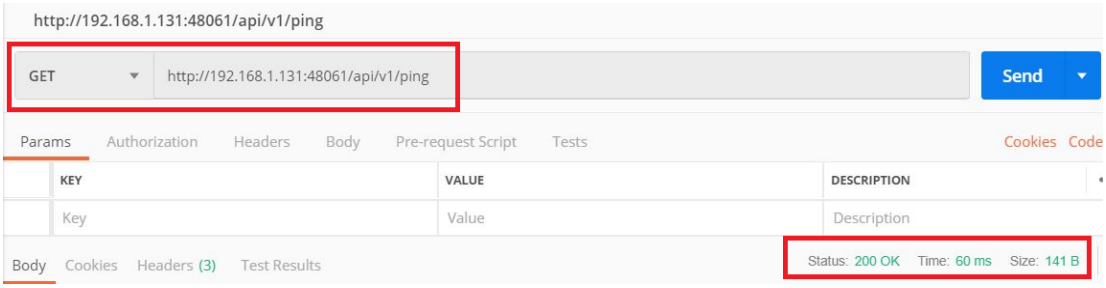

Ilustración 21. Petición HTTP "ping" a la API del microservicio Support Logging

Como he podido verificar, siguiendo la documentación proporcionada por EdgeX Foundry se pueden descargar e iniciar sin mayor dificultar los microservicios que el proyecto pone a disposición de los usuarios, estos proporcionan una plataforma mínima con capacidad para transformar y analizar los datos recolectados por sensores y dispositivos de IoT, sin embargo, tras el primer contacto con la plataforma si considero necesario que los usuarios que quieran dar sus primeros pasos dentro del proyecto tengan unas nociones básicas de desarrollo web y de uso de Docker.

3.5 Testeo de la funcionalidad del sistema

El proyecto EdgeX Foundry proporciona un servicio de dispositivo virtual Ilustración 22 que simula el comportamiento de los sensores, lo que nos permite probar la funcionalidad del sistema cuando no se disponen de datos reales, este servicio es el que se usará en esta prueba de concepto. El servicio que genera eventos y lecturas de forma aleatoria no viene activado por defecto, así que lo primero que debemos hacer es editar el fichero docker-compose.yml y des-comentar la líneas correspondientes al servicio “device-random” Ilustración 23.

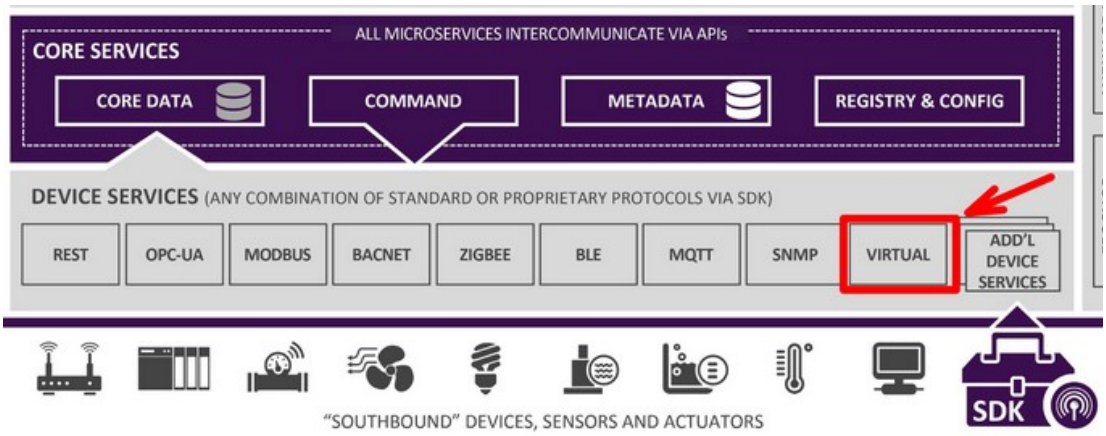

Ilustración 22. Servicio de dispositivo virtual de EdgeX

```
#  device-random:
#    image: edgexfoundry/docker-device-random-go:0.7.1
#    ports:
#      - "49988:49988"
#    container_name: edgex-device-random
#    hostname: edgex-device-random
#    networks:
#      - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#    depends_on:
#      - data
#      - command
```

Ilustración 23. Servicio Device-Random de EdgeX

A continuación iniciamos el contenedor correspondiente:

```
[root@edgex codeEdgeX]# docker-compose up -d device-random
```

y verificamos que arranca correctamente Ilustración 24:

```
[root@edgex codeEdgeX]# docker-compose ps
          Name                        Command               State                                                    Ports
-------------------------------------------------------------------------------------------------------------------------------------------------
codeedgex_portainer_1_cccdfe1644b9   /portainer -H unix:///var/ ...   Exit 1
edgex-config-seed                    /bin/sh -c /edgex/cmd/conf ...   Exit 0
edgex-core-command                   /core-command --consul --p ...   Up       0.0.0.0:48082->48082/tcp
edgex-core-consul                    docker-entrypoint.sh agent ...   Up       8300/tcp, 8301/tcp, 8301/udp, 8302/tcp, 8302/udp,
                                                                               0.0.0.0:8500->8500/tcp, 0.0.0.0:8600->8600/tcp, 8
edgex-core-data                      /core-data --consul --prof ...   Up       0.0.0.0:48080->48080/tcp, 0.0.0.0:5563->5563/tcp
edgex-core-metadata                  /core-metadata --consul --  ...   Up       0.0.0.0:48081->48081/tcp, 48082/tcp
edgex-device-random                  /device-random --registry  ...   Up       0.0.0.0:49988->49988/tcp
edgex-device-virtual                 /bin/sh -c java -jar -Djav ...   Up       0.0.0.0:49990->49990/tcp
edgex-export-client                  /export-client --consul      ...   Up       0.0.0.0:48071->48071/tcp
```

Ilustración 24. Servicio Device Random en ejecución

Una vez que el contenedor se ha iniciado, el servicio de dispositivo Ilustración 11, registrará automáticamente en el sistema EdgeX un dispositivo llamado *Random-Integer-Generator01* que comenzará a enviar lecturas aleatorias. Estas lecturas se enviarán al nivel superior, al servicio Core Data para su

almacenamiento temporal. Para verificar que esta ingesta de datos se está realizando correctamente podemos verificar los logs del contenedor core-data:

```
[root@edgex codeEdgeX]# docker-compose logs -f --tail=10 data

Attaching to edgex-core-data
edgex-core-data   | INFO: 2018/12/28 15:49:00 Posting Event:
{"id":"","pushed":0,"device":"Random-Integer-
Generator01","created":0,"modified":0,"origin":1548517740002,"schedule":null,"event":
null,"readings":
[{"id":"","pushed":0,"created":0,"origin":1548517740002,"modified":0,"device":"Rando
m-Integer-Generator01","name":"RandomValue_Int32","value":"-560686752"}]}
edgex-core-data   | INFO: 2018/12/28 15:49:00 Posting Event:
{"id":"","pushed":0,"device":"Random-Integer-
Generator01","created":0,"modified":0,"origin":1548517740007,"schedule":null,"event":
null,"readings":
[{"id":"","pushed":0,"created":0,"origin":1548517740003,"modified":0,"device":"Random
-Integer-Generator01","name":"RandomValue_Int16","value":"-20691"}]}
edgex-core-data   | INFO: 2018/12/28 15:49:00 Putting event on message queue
edgex-core-data   | INFO: 2018/12/28 15:49:00 Putting event on message queue
edgex-core-data   | INFO: 2018/12/28 15:49:00 Putting event on message queue
edgex-core-data   | INFO: 2018/12/28 15:49:05 Posting Event:
{"id":"","pushed":0,"device":"Random-Integer-
Generator01","created":0,"modified":0,"origin":1548517745004,"schedule":null,"event":n
ull,"readings":
[{"id":"","pushed":0,"created":0,"origin":1548517745004,"modified":0,"device":"Random
-Integer-Generator01","name":"RandomValue_Int32","value":"-826232447"}]}

....
```

Como vemos las lecturas recogidas por mi dispositivo *Random-Integer-Generator01* las está recibiendo correctamente el servicio core-data. Podemos también hacer una llamada a la API de este servicio para verificar el número de eventos que ha ido recolectando:

```
[root@edgex codeEdgeX]# curl http://192.168.1.131:48080/api/v1/event/count
43835
[root@edgex codeEdgeX]# curl http://192.168.1.131:48080/api/v1/event/count
43844
[root@edgex codeEdgeX]# curl http://192.168.1.131:48080/api/v1/event/count
44022
...
```

Mientras el dispositivo *Random-Integer-Generator01* esté operativo, todos los eventos y lecturas que genera se envían al core-data para su almacenamiento temporal. Con el cliente Postman podemos hacer una llamada a la API

correspondiente del servicio core-data y solicitarle una de las lecturas de nuestro dispositivo Ilustración 25, de esta forma podemos concluir que las lecturas generadas por mi dispositivo virtual se están registrando correctamente en la plataforma EdgeX desplegada, vemos también como estás lecturas se encapsulan / transforman en una estructura de datos común de EdgeX.

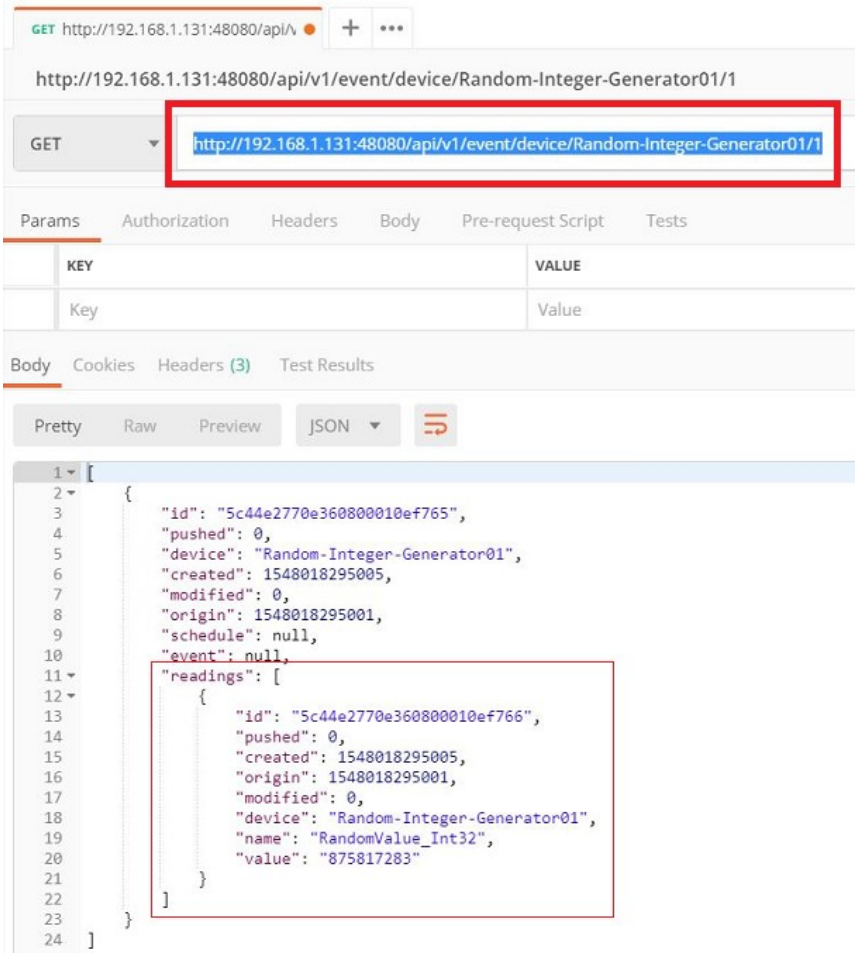

Ilustración 25. Lectura almacenada por el servicio core-data

Por otro lado, el servicio core-metadata conoce los dispositivos y sensores conectados al sistema, así que una llamada a la API correspondiente nos indicará que nuestro dispositivo *Random-Integer-Generator01* está conectado a nuestro entorno EdgeX Ilustración 26.

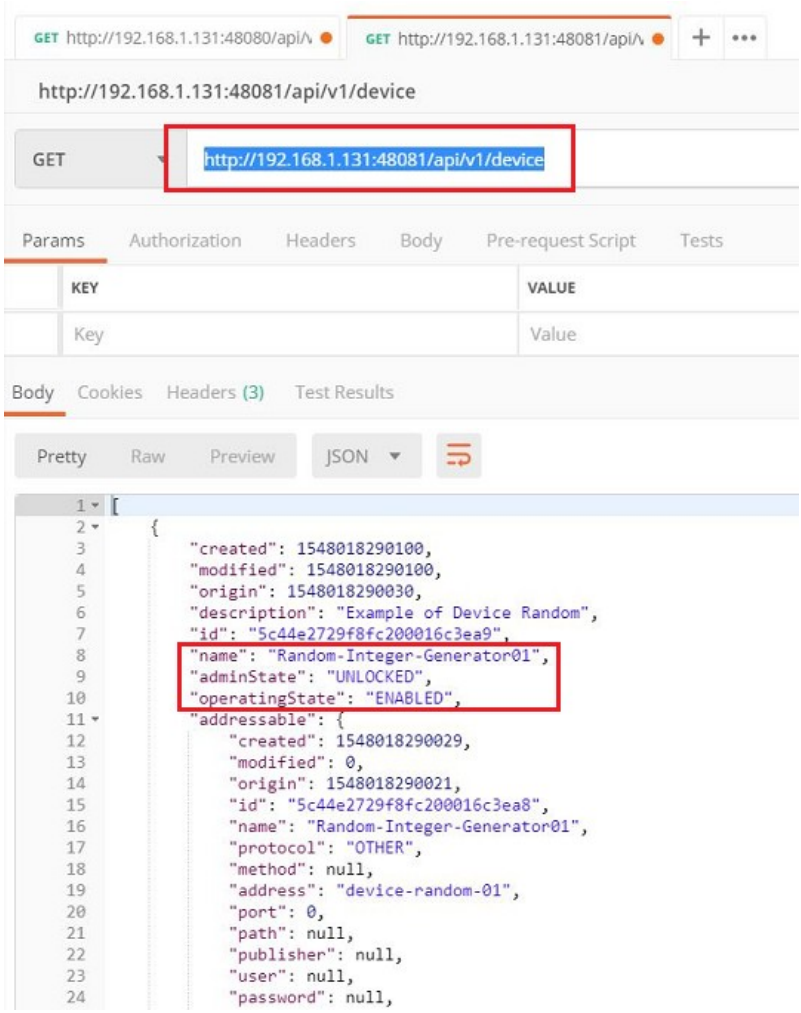

Ilustración 26. Dispositivos conocidos por el servicio core-metadata

El control de los dispositivos mediante comandos es otra funcionalidad que ofrece la plataforma EdgeX a través del microservicio core-command. Una llamada a la API correspondiente:

http://192.168.1.131:48082/api/v1/device/name/Random-Integer-Generator01

nos devuelve los comandos a los que podemos llamar para controlarlo.

Si pensamos en conectar nuestras aplicaciones a la plataforma desplegada, el servicio edgex-export-client permite que un cliente se registre como destinatario de los datos que llegan del Core Data. En este contexto, un cliente puede ser una aplicación remota en la nube a la que el sistema EdgeX enviará los datos recolectados por el dispositivo que le indiquemos. Para probar esta funcionalidad que ofrece la plataforma, enviaremos los datos que recolecta

nuestro dispositivo, *Random-Integer-Generator01,* a un servidor público de Internet (broker.hivemq.com) utilizando el protocolo MQTT.

El registro de un cliente como destinatario de los datos que llegan del Core Data se realiza a través de una llamada a la API correspondiente mediante el método POST, indicando que el servidor externo será el receptor de los datos, tal como se muestra a continuación:

```
curl -X POST -d '{
   "name":"QuickStartExport",
   "addressable":{
      "name":"HiveMQBroker",
      "protocol":"tcp",
      "address":"broker.hivemq.com",
      "port":1883,
      "publisher":"EdgeXExportPublisher",
      "topic":"EdgeXQuickStartGuide"
   },
   "format":"JSON",
   "filter":{
      "deviceIdentifiers":["Random-Integer-Generator01"]
   },
   "enable":true,
   "destination":"MQTT_TOPIC"
}' http://192.168.1.131:48071/api/v1/registration
```

Una vez que el dispositivo se ha registrado como exportador de datos, a través de un cliente MQTT podremos verificar que los datos están llegando correctamente al servidor externo, para la prueba de concepto he usado el cliente Mosquitto, verificando que el comportamiento de la plataforma es el esperado:

```
[root@edgex codeEdgeX]# mosquitto_sub -h broker.hivemq.com -p 1883 -t
EdgeXQuickStartGuide

{"id":"5c4865640e36080001374a0a","pushed":0,"device":"Random-Integer-
Generator01","created":1548248420005,"modified":0,"origin":1548248420003,"schedu
le":null,"event":null,"readings":
[{"id":"5c4865640e36080001374a0b","pushed":0,"created":1548248420005,"origin":1
548248420002,"modified":0,"device":"Random-Integer-
Generator01","name":"RandomValue_Int8","value":"-64"}]}
{"id":"5c4865640e36080001374a0c","pushed":0,"device":"Random-Integer-
Generator01","created":1548248420008,"modified":0,"origin":1548248420001,"schedu
le":null,"event":null,"readings":
[{"id":"5c4865640e36080001374a0d","pushed":0,"created":1548248420008,"origin":1
548248420001,"modified":0,"device":"Random-Integer-
Generator01","name":"RandomValue_Int32","value":"1263092215"}]}
...
```

Con esta prueba de concepto se ha podido verificar que la instalación y despliegue del sistema EdgeX es un proceso sencillo que no presenta mayor dificultad, además se ha verificado la funcionalidad del conjunto de microservicios que proporciona EdgeX y que nos ofrecen la capacidad mínima de una plataforma edge.

Utilizando un sensor virtual que generaba lecturas de forma aleatoria, se ha podido verificar como el sensor se registraba correctamente en nuestra plataforma, las lecturas recolectadas por el sensor eran enviadas al servicio correspondiente para su almacenamiento temporal. Se comprobó que dichas lecturas se encapsulaban / transformaban en una estructura común de datos de EdgeX Foundry, y que esos datos podían ser exportados hacia infraestructuras remotas en Internet. Todo ello de forma sencilla, a través de llamadas a las APIs de los microservicios que forman la plataforma, en este aspecto hay que indicar que la documentación que facilita el proyecto es bastante clara y guía al usuario en todo el proceso de despliegue y evaluación del sistema. Tanto la documentación como el código del proyecto están en un proceso continúo de revisión y mejora.

4 Conclusiones

Este trabajo me ha permitido entender los problemas actuales a los que se enfrentan los sistemas de IoT y como tecnologías emergentes como Edge y Fog Computing intentan dar solución a estos problemas, disminuyendo las latencias inherentes a la red en la que operan estos sistemas, y los costos asociados a servicios en la nube. Por otro lado, he podido entender que el despliegue de este tipo de sistemas no es sencillo y por eso en muchas ocasiones no es fácil determinar por dónde empezar. Hablamos de sistemas formados por plataformas heterogéneas con diversidad de sistemas operativos, elementos hardware de diferentes fabricantes, diferentes soluciones en la nube, diferentes protocolos de comunicación, … por lo que se hace necesario soluciones que permitan la interoperabilidad de todos estos elementos. He podido comprobar como EdgeX Foundry contribuye a la estandarización de IoT facilitando la adopción de estos sistemas. En este aspecto, se considera que los objetivos planteados inicialmente se han logrado siguiendo la metodología propuesta.

4.1 Trabajos futuros

El proyecto EdgeX Foundry proporciona una extensa plataforma de código y documentación en la que no se ha podido profundizar debido a la corta duración del proyecto. En este aspecto, quedan abiertos diferentes trabajos futuros que pasan por conectar dispositivos reales a la plataforma para validar su comportamiento y profundizar en como los diferentes microservicios que forman la plataforma interactúan entre ellos.

5 Glosario

Actuador Dispositivo que puede transformar energía eléctrica, hidráulica o neumática en energía de salida utilizable para crear un efecto sobre un determinado proceso automatizado. Funciona proporcionando fuerza para que otro dispositivo actúe. El actuador recibe las ordenes de un regulador o controlador.

AES Advanced Encryption Standard. Uno de los algoritmos de cifrado más utilizados. Adoptado por el gobierno de de EEUU como un estándar de cifrado.

Amazon Es una compañía estadounidense de comercio electrónico y servicios de computación en la nube a todos los niveles con sede en la ciudad estadounidense de Seattle, Estado de Washington.

API Application Programming Interface. Es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes.

Apache Apache es un software de servidor web gratuito y de código abierto mantenido y desarrollado por Apache Software Foundation.

BACNet Building Automation and Control Networks. Es un protocolo de comunicación de datos diseñado para comunicar entre sí a los diferentes aparatos electrónicos presentes en los edificios actuales.

CDN Red de Distribución de Contenido, conjunto de servidores ubicados en diferentes puntos de una red que contienen copias locales de ciertos contenidos (vídeos, imágenes, música, documentos, ...) que están almacenados en otros servidores generalmente alejados geográficamente, de forma que sea posible servir dichos contenidos de manera más eficiente.

Centos	Es un sistema operativo de código abierto basado en la distribución Red Hat Enterprise Linux.
Centro de Datos	Espacio donde se concentran los recursos necesarios para el procesamiento de la información de una organización.
Cisco	Multinacional estadounidense líder en soluciones de networking.
Cloud Computing	Es una tecnología que permite ofrecer servicios de computación bajo demanda (software, almacenamiento, procesamiento de datos, …) a través de Internet.
Consul	Herramienta para el descubrimiento dinámico de microservicios. Es un sistema distribuido open source creado por Hashicorp para el registro de servicios y la configuración de los mismos. Dispone de una API que permite a los servicios el registro/desregistro en su catálogo. Consul proporciona un chequeo constante de la salud de los servicios, marcando aquéllos que no pasan el mismo.
Contenedores de software	Entornos ligeros que proporcionan a las aplicaciones los archivos, las variables y las bibliotecas que necesitan para ejecutarse.
CSV	Comma-Separated Values. Tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.
Dell	Multinacional estadounidense líder en soluciones tecnológicas.
DevOps	Es una práctica de ingeniería de software que tiene como objetivo unificar el desarrollo y las operaciones de tecnología de la información.

Docker Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software.

Docker Compose Es una herramienta que permite simplificar el uso de Docker, generando scripts que facilitan el diseño y la construcción de servicios.

Docker Hub Repositorio público en la nube, similar a GitHub, para distribuir de contenidos. Está mantenido por la propia Docker y hay multitud de imágenes, de carácter gratuito, que se pueden descargar y así no tener que hacer el trabajo desde cero al poder aprovechar "plantillas". También permite crear repositorios propios privados e, incluso, dispone de una tienda.

Drools Es un sistema de gestión de reglas de negocio para Java.

Escalabilidad Es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad.

Framework Marco de referencia para el desarrollo y/o la implementación de una aplicación.

Gateway Es un dispositivo que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.

Gateway de IoT Es dispositivo físico o definido por software que hace de punto de conexión entre los sistemas remotos (nube) y los dispositivos de IoT. El tráfico de datos que va de los dispositivos de IoT hacia la nube y viceversa pasan por un gateway, en ocasiones representa el lugar en el que los datos pueden sufrir un pequeño procesamiento antes de ser enviados a infraestructuras remotas en la nube.

GitHub Es una plataforma creada para facilitar el desarrollo colaborativo de software, permite alojar proyectos en la web gratuitamente, por lo general de forma pública, aunque se pueden alojar de modo privado, si se paga una pequeña suscripción mensual. Ofrece al desarrollador toda la potencia y agilidad del sistema de control de versiones Git, más un interesante set de herramientas añadidas

Google Multinacional estadounidense especializada en productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías.

GZIP Algoritmo de compresión sin pérdida de información, reduce el tamaño de los archivos sustituyendo las cadenas más frecuentes por cadenas más cortas.

Hardware Engloba a todos los componentes físicos de una tecnología: cables, circuitos, placa base, disco duro, tarjetas de red, … y cualquier otro componente que sea tangible.

HTTP Hypertext Transfer Protocol, es un protocolo de red que permite la transferencia de información entre diferentes servicios, generalmente páginas web y sus componentes.

Interoperabilidad La interoperabilidad es la capacidad de dos o más sistemas o componentes para intercambiar información y usar la información que se han intercambiado.

IP Número que identifica, de manera lógica y jerárquica, a una interfaz de red de un dispositivo que utilice el protocolo IP, encargado de la comunicación de datos a través de una red.

JSON JavaScript Object Notation. Estándar basado en texto plano para el intercambio de información, se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas,

	JSON es independiente de cualquier lenguaje de programación, así los servicios que comparten información por éste método, no necesitan hablar el mismo idioma,
Latencia	Es el tiempo necesario para que un paquete de información sea transmitido de un punto a otro de la red. Puede verse como el tiempo que tarda un servicio basado en la red en responder a la solicitud de un usuario. Es un factor determinante para medir la velocidad de una red.
Microservicio	Es un enfoque que permite desarrollar una aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, generalmente mediante APIs a través de HTTP.
Modbus	Protocolo de comunicaciones estándar para la conexión de dispositivos electrónicos industriales
MongoDB	Es un sistema de base de datos NoSQL orientado a documentos de código abierto. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.
MQTT	Message Queue Telemetry Transport. Es un protocolo usado para la comunicación machine-to-machine (M2M) en "Internet of Things". Está orientado a la comunicación de sensores, debido a que consume muy poco ancho de banda y puede ser utilizado en dispositivos con pocos recursos.
Postman	Herramienta que permite realizar peticiones HTTP a cualquier API. Esta dirigida a desarrolladores web y a todo aquel que tenga que interactuar con una API.
Plug-and-play	Se refiere a la capacidad de un sistema de configurar

	automáticamente los dispositivos al conectarlos. Permite poder enchufar un dispositivo y utilizarlo inmediatamente, sin preocuparte de la configuración.
REST	Representational State Transfer. Es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON
SDN	Redes definidas por software, utilizan software en lugar de dispositivos especializados para el aprovisionamiento y la gestión de los servicios de aplicaciones y de red, lo que permite la movilidad y la entrega de aplicaciones programables, escalables y bajo demanda.
Sensor	Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas que pueden ser cuantificadas y manipuladas. Las variables de instrumentación pueden ser: temperatura, intensidad lumínica, distancia, aceleración, movimiento, presión, humedad, ...
XML	Extensible Markup Language. Es una especificación / lenguaje de programación diseñado para definir, validar, interpretar y compartir formatos de documentos en la Web

6 Bibliografía

[1] *Wikipedia: Internet de la cosas.*
Consultado: 28 de Septiembre del 2018
https://es.wikipedia.org/wiki/Internet_de_las_cosas

[2] *Leading the IoT. Gartner.*
Consultado: 28 de Septiembre del 2018
https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf

[3] *The Emergence of Edge Computing, Mahadev Satyanarayanan, Carnegie Mellon University*
Consultado: 2 de Octubre del 2018
http://elijah.cs.cmu.edu/DOCS/satya-edge2016.pdf

[4] *Cisco: Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*
Consultado: 2 de Octubre del 2018
https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[5] *EdgeXFoundry: The Linux Foundation Project.*
Consultado:4 de Octubre del 2018
https://www.edgexfoundry.org/

[6] *Wikipedia: Kevin Ashton*
Consultado: 12 de Octubre del 2018
https://en.wikipedia.org/wiki/Kevin_Ashton

[7] *RFID Journal: That 'Internet of Things' Thing by Kevin Ashton, 2009*
Consultado: 12 de Octubre del 2018
https://www.rfidjournal.com/articles/view?4986

[8] *Cisco: El valor de IoT: cómo pasar de conectar cosas a obtener información*
Consultado: 22 de Octubre del 2018
https://www.cisco.com/c/dam/assets/global/ES/offers/datacenter/potential/dc-05-attaining-iot-value-wp-cte-es-eu.pdf

[9] *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper*
Consultado: 25 de Octubre del 2018
https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[10] *Wikipedia: Content delivery network*
Consultado: 2 de Noviembre del 2018
https://en.wikipedia.org/wiki/Content_delivery_network

[11] *Wikipedia: Cloud Computing*
Consultado: 30 de Octubre del 2018

https://en.wikipedia.org/wiki/Cloud_computing

[12] *White Paper: The Internet of Things Reference Model*
Consultado: 30 Octubre del 2018
http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf

[13] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are, White Paper Cisco*
Consultado: 2 de Noviembre del 2018
https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[14] *Agile Application-Aware Adaptation for Mobility*, Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, Kevin R. Walker, 1997
Consultado: 19 Noviembre del 2018
https://www.researchgate.net/publication/220910032_Agile_Application-Aware_Adaptation_for_Mobility

[15] *Open Edge Computing*
Consultado: 8 de Octubre del 2018
http://openedgecomputing.org/lel.pdf

[16] *Edge Computing Consortium*
Consultado: 8 de Octubre del 2018
http://en.ecconsortium.org/

[17] *Wikipedia: Micro Data Center*
Consultado: 19 de Noviembre del 2018
https://en.wikipedia.org/wiki/Micro_data_center

[18] *OpenFog Consortium*
Consultado: 8 de Octubre del 2018
https://www.openfogconsortium.org/

[19] *Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review*
Consultado: 22 de Noviembre del 2018
https://www.sciencedirect.com/science/article/pii/S016412121730256X?via%3Dihub

[20] *The Linux Foundation*
Consultado: 26 de Noviembre del 2018
https://www.linuxfoundation.org/

[21] *Join EdgeX Foundry*
Consultado: 2 de Diciembre del 2018
https://www.edgexfoundry.org/about/members/join/

[22] *Contributor Suggestions (how can you be a part of EdgeX Foundry?), EdgeX Foundry Wiki*
Consultado: 14 de Diciembre del 2018
https://wiki.edgexfoundry.org/pages/viewpage.action?pageId=4063305

[23] *Technical Work in the EdgeX Foundry Project*
Consultado: 10 de Diciembre del 2018
https://wiki.edgexfoundry.org/display/FA/Technical+Work+in+the+EdgeX+Foundry+Project

[24] *EdgeX Foundry Project Wiki*
Consultado: 10 de Diciembre del 2018
https://wiki.edgexfoundry.org/

[25] *Aquitectura de microservicios, Wikipedia*
Consultado: 10 de Diciembre del 2018
https://es.wikipedia.org/wiki/Arquitectura_de_microservicios

[26] *Repositories EdgeX Foundry Project, GitHub*
Consultado: 14 de Diciembre del 2018
https://github.com/edgexfoundry/

[27] *Docker Documentation*
Consultado: 10 de Diciembre del 2018
https://docs.docker.com/

[28] *Welcome to the EdgeX Foundry documentation*
Consultado: 17 de Diciembre del 2018
https://docs.edgexfoundry.org/index.html

7 Anexos

7.1 Instalación de Docker CE y Docker Compose

Para la instalación de Docker CE en Centos 7 se han seguido las instrucciones recogidas en la Web del proyecto: https://docs.docker.com/install/linux/docker-ce/centos/, y para la instalación de Docker Compose en Linux se han seguido las instrucciones recogidas en la Web del proyecto: https://docs.docker.com/compose/install/.

7.2 Contenido del fichero EdgeX Foundry Compose

Contenido fichero docker-compose.yml para EdgeX Foundry

```
#
/*******************************************************************************
# * Copyright 2018 Dell Inc.
# *
# * Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
# * in compliance with the License. You may obtain a copy of the License at
# *
# * http://www.apache.org/licenses/LICENSE-2.0
# *
# * Unless required by applicable law or agreed to in writing, software distributed under the License
# * is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
# * or implied. See the License for the specific language governing permissions and limitations under
# * the License.
# *
# * @author: Jim White, Dell
# * EdgeX Foundry, "Latest"
# * added: Dec 10, 2018
#
*******************************************************************************/

version: '3'
volumes:
  db-data:
  log-data:
  consul-config:
  consul-data:
  portainer_data:

services:
  volume:
```

```
  image: edgexfoundry/docker-edgex-volume:0.6.0
  container_name: edgex-files
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data

consul:
  image: consul:1.1.0
  ports:
    - "8400:8400"
    - "8500:8500"
    - "8600:8600"
  container_name: edgex-core-consul
  hostname: edgex-core-consul
  networks:
    edgex-network:
      aliases:
          - edgex-core-consul
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - volume

config-seed:
  image: edgexfoundry/docker-core-config-seed-go:0.7.1
  container_name: edgex-config-seed
  hostname: edgex-core-config-seed
  networks:
    edgex-network:
      aliases:
          - edgex-core-config-seed
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - volume
    - consul

mongo:
  image: edgexfoundry/docker-edgex-mongo:0.6.0
  ports:
    - "27017:27017"
  container_name: edgex-mongo
  hostname: edgex-mongo
  networks:
```

```
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - volume

logging:
  image: edgexfoundry/docker-support-logging-go:0.7.1
  ports:
    - "48061:48061"
  container_name: edgex-support-logging
  hostname: edgex-support-logging
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - config-seed
    - mongo
    - volume

notifications:
  image: edgexfoundry/docker-support-notifications-go:0.7.1
  ports:
    - "48060:48060"
  container_name: edgex-support-notifications
  hostname: edgex-support-notifications
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - logging

metadata:
  image: edgexfoundry/docker-core-metadata-go:0.7.1
  ports:
    - "48081:48081"
  container_name: edgex-core-metadata
  hostname: edgex-core-metadata
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
```

```
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - logging

data:
  image: edgexfoundry/docker-core-data-go:0.7.1
  ports:
    - "48080:48080"
    - "5563:5563"
  container_name: edgex-core-data
  hostname: edgex-core-data
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - logging

command:
  image: edgexfoundry/docker-core-command-go:0.7.1
  ports:
    - "48082:48082"
  container_name: edgex-core-command
  hostname: edgex-core-command
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - metadata

scheduler:
  image: edgexfoundry/docker-support-scheduler-go:0.7.1
  ports:
    - "48085:48085"
  container_name: edgex-support-scheduler
  hostname: edgex-support-scheduler
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - metadata
```

```
export-client:
  image: edgexfoundry/docker-export-client-go:0.7.1
  ports:
    - "48071:48071"
  container_name: edgex-export-client
  hostname: edgex-export-client
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - data
  environment:
    - EXPORT_CLIENT_MONGO_URL=edgex-mongo
    - EXPORT_CLIENT_DISTRO_HOST=export-distro
    - EXPORT_CLIENT_CONSUL_HOST=edgex-config-seed

export-distro:
  image: edgexfoundry/docker-export-distro-go:0.7.1
  ports:
    - "48070:48070"
    - "5566:5566"
  container_name: edgex-export-distro
  hostname: edgex-export-distro
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
    - consul-config:/consul/config
    - consul-data:/consul/data
  depends_on:
    - export-client
  environment:
    - EXPORT_DISTRO_CLIENT_HOST=export-client
    - EXPORT_DISTRO_DATA_HOST=edgex-core-data
    - EXPORT_DISTRO_CONSUL_HOST=edgex-config-seed
    - EXPORT_DISTRO_MQTTS_CERT_FILE=none
    - EXPORT_DISTRO_MQTTS_KEY_FILE=none

rulesengine:
  image: edgexfoundry/docker-support-rulesengine:0.7.0
  ports:
    - "48075:48075"
  container_name: edgex-support-rulesengine
  hostname: edgex-support-rulesengine
  networks:
    - edgex-network
  volumes:
    - db-data:/data/db
    - log-data:/edgex/logs
```

```
    - consul-config:/consul/config
    - consul-data:/consul/data

#############################################################################
# Device Services
#############################################################################

  device-virtual:
    image: edgexfoundry/docker-device-virtual:0.6.0
    ports:
      - "49990:49990"
    container_name: edgex-device-virtual
    hostname: edgex-device-virtual
    networks:
      - edgex-network
    volumes:
      - db-data:/data/db
      - log-data:/edgex/logs
      - consul-config:/consul/config
      - consul-data:/consul/data
    depends_on:
      - data
      - command

#  device-random:
#    image: edgexfoundry/docker-device-random-go:0.7.1
#    ports:
#      - "49988:49988"
#    container_name: edgex-device-random
#    hostname: edgex-device-random
#    networks:
#      - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#    depends_on:
#      - data
#      - command

#  device-mqtt:
#    image: edgexfoundry/docker-device-mqtt-go:0.7.1
#    ports:
#      - "49982:49982"
#    container_name: edgex-device-mqtt
#    hostname: edgex-device-mqtt
#    networks:
#      - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
```

```
#    depends_on:
#      - data
#      - command

#  device-modbus:
#    image: edgexfoundry/docker-device-modbus-go:0.7.1
#    ports:
#      - "49991:49991"
#    container_name: edgex-device-modbus
#    hostname: edgex-device-modbus
#    networks:
#      - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#    depends_on:
#      - data
#      - command

#   device-bluetooth:
#     image: nexus3.edgexfoundry.org:10004/docker-device-bluetooth:0.6.0
#     ports:
#       - "49988:49988"
#       - "5000:5000"
#     container_name: edgex-device-bluetooth
#     hostname: edgex-device-bluetooth
#     privileged: true
#     network_mode: "host"
#     cap_add:
#       - NET_ADMIN
# #    networks:
# #      - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#     depends_on:
#       - data
#       - command

#   device-snmp:
#     image: nexus3.edgexfoundry.org:10004/docker-device-snmp:0.6.0
#     ports:
#       - "49989:49989"
#     container_name: edgex-device-snmp
#     hostname: edgex-device-snmp
#     networks:
#       - edgex-network
#    volumes:
#      - db-data:/data/db
#      - log-data:/edgex/logs
```

```
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#    depends_on:
#      - data
#      - command

#  device-fischertechnik:
#    image: nexus3.edgexfoundry.org:10004/docker-device-fischertechnik:0.6.0
#    ports:
#      - "49985:49985"
#    container_name: edgex-device-fischertechnik
#    networks:
#      - edgex-network
#   volumes:
#     - db-data:/data/db
#     - log-data:/edgex/logs
#     - consul-config:/consul/config
#     - consul-data:/consul/data
#    privileged: true
#    depends_on:
#      - data
#      - command

#  device-bacnet:
#    image: nexus3.edgexfoundry.org:10004/docker-device-bacnet:0.6.0
#    ports:
#      - "49986:49986"
#      - "5002:5002"
#    container_name: edgex-device-bacnet
#    hostname: edgex-device-bacnet
#    networks:
#      - edgex-network
#   volumes:
#     - db-data:/data/db
#     - log-data:/edgex/logs
#     - consul-config:/consul/config
#     - consul-data:/consul/data
#   depends_on:
#      - data
#      - command

############################################################################
# UIs
############################################################################
#  ui:
#    image: edgexfoundry/docker-edgex-ui-go:0.1.1
#    ports:
#      - "4000:4000"
#    container_name: edgex-ui-go
#    hostname: edgex-ui-go
#    networks:
#      - edgex-network
#    volumes:
#      - db-data:/data/db
```

```
#      - log-data:/edgex/logs
#      - consul-config:/consul/config
#      - consul-data:/consul/data
#    depends_on:
#      - data
#      - command

#############################################################################
# Tooling
#############################################################################

  portainer:
    image:  portainer/portainer
    ports:
      - "9000:9000"
    command: -H unix:///var/run/docker.sock
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data
    depends_on:
      - volume

networks:
  edgex-network:
    driver: "bridge"
...
```