Gestión inteligente de datos en tiempo real

Edge computing



Contenidos

Objetivos

1. Diferencia entre *local*, *edge* y *cloud computing*
    1.1. *Edge computing*
2. Contenedores
    2.1. Segmentar en diferentes servidores
    2.2. Segmentar en máquinas virtuales
    2.3. Segmentar en contenedores
3. Contenedores Docker
    3.1. Imágenes en Docker
4. Distribución Industrial Edge
    4.1. Industrial Edge Hub (IEH)
    4.2. Industrial Edge Management (IEM)
    4.3. Industrial Edge App Publisher (IEAP)
    4.4. Industrial Edge Device (IED)

Referencias bibliográficas



Objetivos

A la finalización de este tema, el alumno será capaz de:

- Diferenciar entre computación *edge* y *cloud*.
- Conocer cómo funcionan los contenedores.
- Conocer la distribución de Industrial Edge.

1. Diferencia entre *local*, *edge* y *cloud computing*

Otra posibilidad que aparece es el término *fog*, que pretende aunar las ventajas del *edge* (latencias bajas de adquisición de datos) y los beneficios del *cloud* (tener una gran capacidad de computación de datos al no depender directamente del *hardware* y ser escalable).

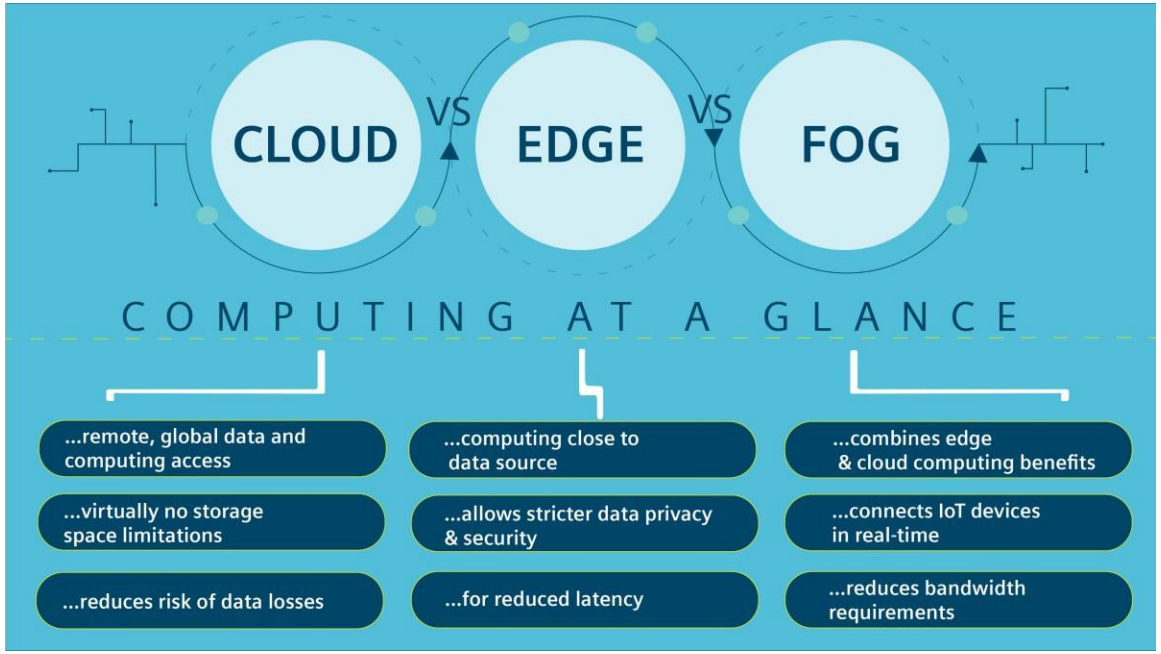


Figura 1. *Edge*, *fog* y *cloud*. Fuente: <https://www.siemens-advanta.com/blog/how-solve-edge-computing-vs-cloud-computing-debate>

1. Diferencia entre *local*, *edge* y *cloud computing*

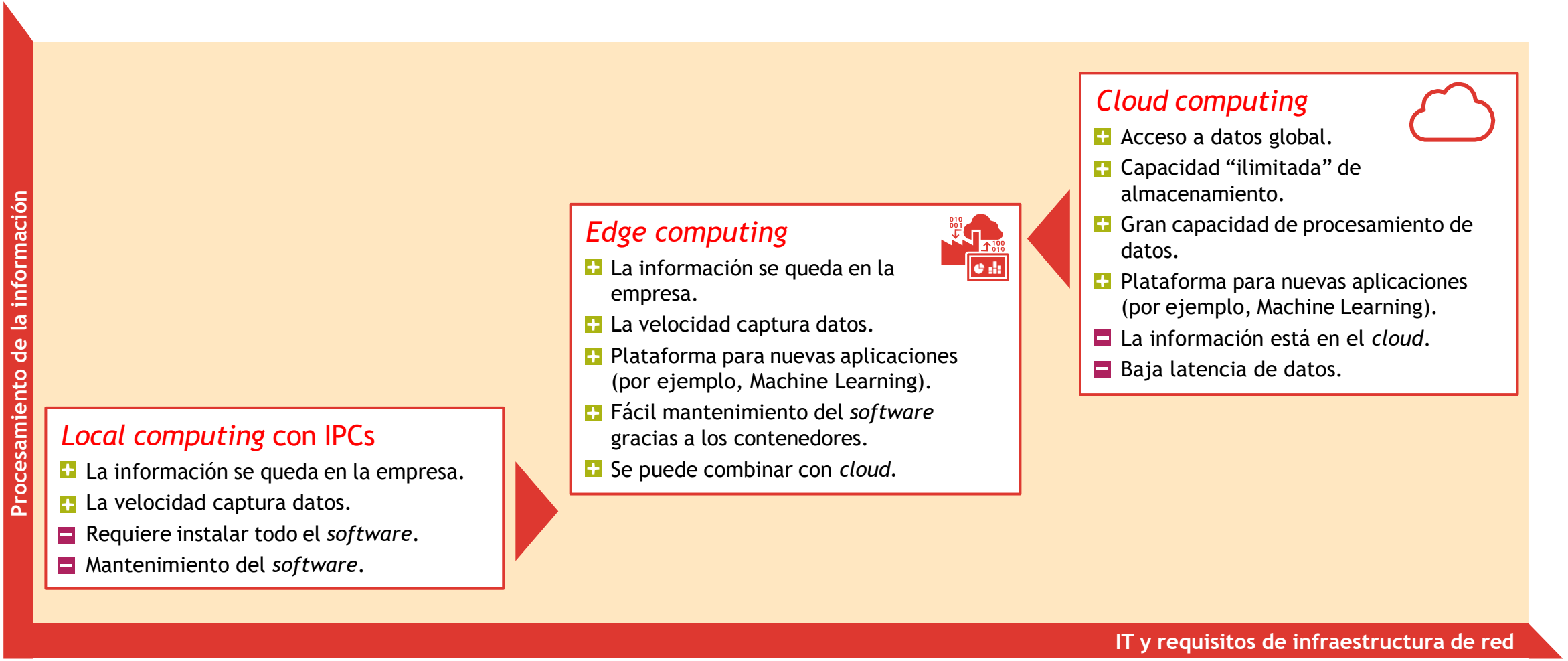


Figura 2. *Local*, *edge* y *cloud computing*. Elaborada por Joaquín Sanz Ferrándiz.

1. Diferencia entre *local*, *edge* y *cloud computing*

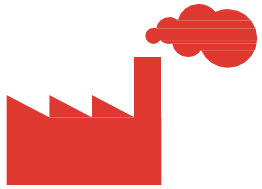
1.1. *Edge computing*



Recolecta datos con alta latencia, los analiza, almacena y los reenvía desde cualquier equipo industrial.



Despliega e integra las aplicaciones de forma más rápida y fiable gracias a los contenedores en el nivel de campo.



Gestiona dispositivos y aplicaciones de forma centralizada en todas las plantas industriales.



Decide individualmente qué aplicaciones implementar en las instalaciones.

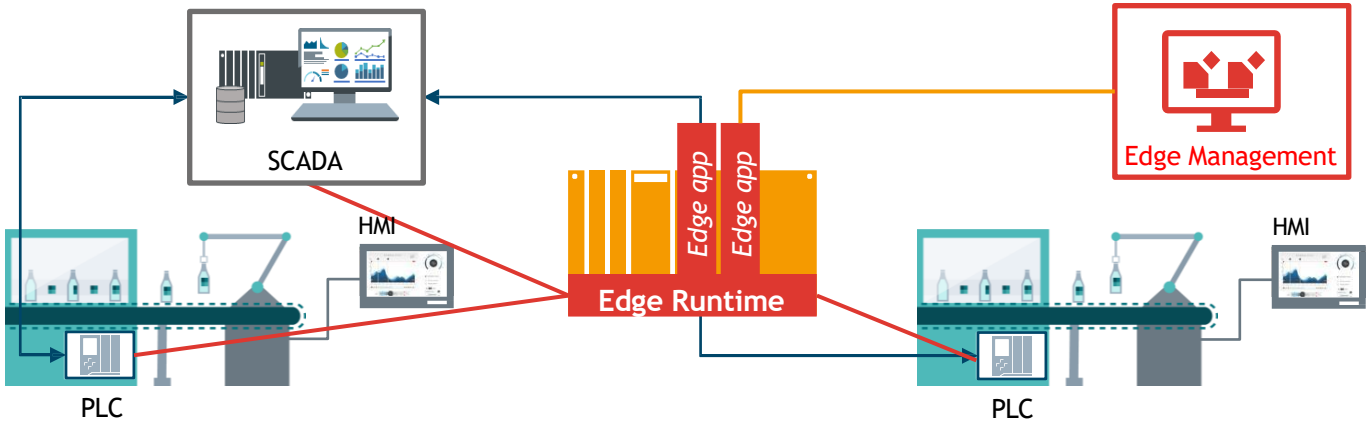


Figura 3. *Edge computing*. Adaptada de: <https://www.automation.siemens.com>

2. Contenedores

El concepto de contenedor surge para evitar problemas como el que existe a día de hoy cuando se ejecutan diferentes aplicaciones en un mismo ordenador. Estos problemas son:

- Sistema operativo (SO): al tener que ejecutarse todas las aplicaciones sobre un mismo servidor, todas ellas tienen que compartir el mismo SO. Con la evolución que existe en el mercado esto supone un gran problema.
- Librerías (DLLs): conforme se van instalando diferentes aplicaciones estas requieren una serie de librerías, lo que puede ocasionar incompatibilidad entre ellas.
- Recursos: conforme se van ejecutando las aplicaciones estas van demandando recursos y es posible que puedan llegar a acaparar los recursos que requiere otra aplicación. Esto conlleva a una ralentización del sistema o incluso, en caso de fallo, puede bloquear el servidor y dejar de ejecutar el resto de aplicaciones.



Figura 4. Aplicaciones en un servidor. Elaborada por Joaquín Sanz Ferrándiz.

2. Contenedores

2.1. Segmentar en diferentes servidores

Como primera solución que se pretendió aplicar hace unos años, se procedió a duplicar servidores en función de la tarea que fuesen a ejecutar.

- Servidor para SCADA.
- Servidor para SW de programación antiguo.
- Servidor para SW programación actual.

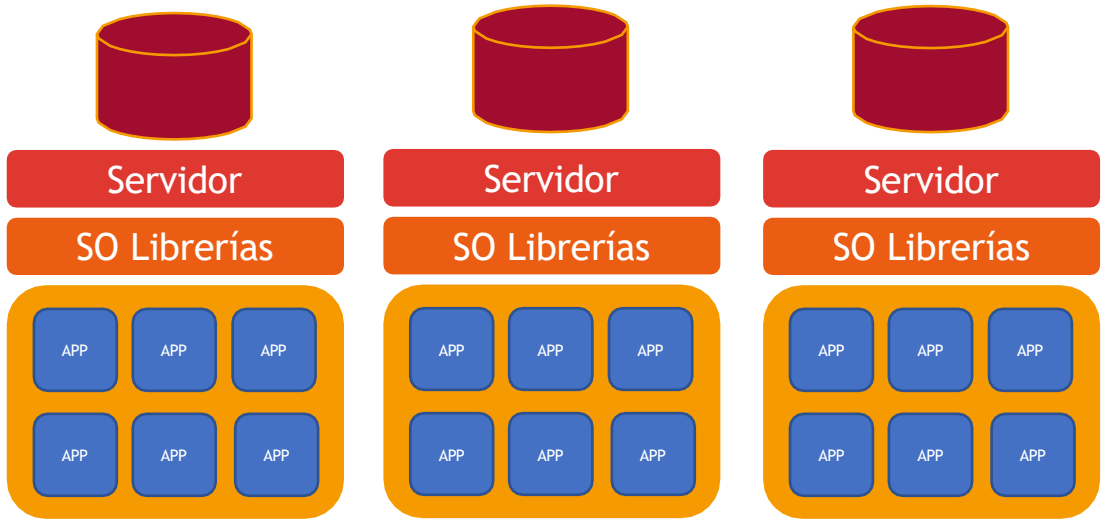


Figura 5. Diferentes aplicaciones en varios servidores. Elaborada por Joaquín Sanz Ferrándiz.

Importante

Como es comprensible, este tipo de soluciones, además de no aprovechar al máximo las capacidades de cada servidor (en algunos casos los servidores ejecutaban una o ninguna aplicación), son un derroche de dinero y de recursos de planta.

2. Contenedores

2.2. Segmentar en máquinas virtuales

Las máquinas virtuales lo que pretenden es emular un servidor entero dentro de otro físico, de tal manera que al final se utiliza un único servidor y se aprovechan mucho mas los recursos.

Esta opción aporta mucha versatilidad, ya que se pueden dedicar unos recursos específicos a cada máquina virtual o, incluso si hay escasez de recursos, se puede llevar esta máquina virtual a otro servidor.

El gran inconveniente de las máquinas virtuales es que, como se observa en la imagen, se duplican los recursos: cada una de estas máquinas virtuales tiene su propio sistema operativo y esto hace que sean más pesadas y consuman más recursos de lo deseado.

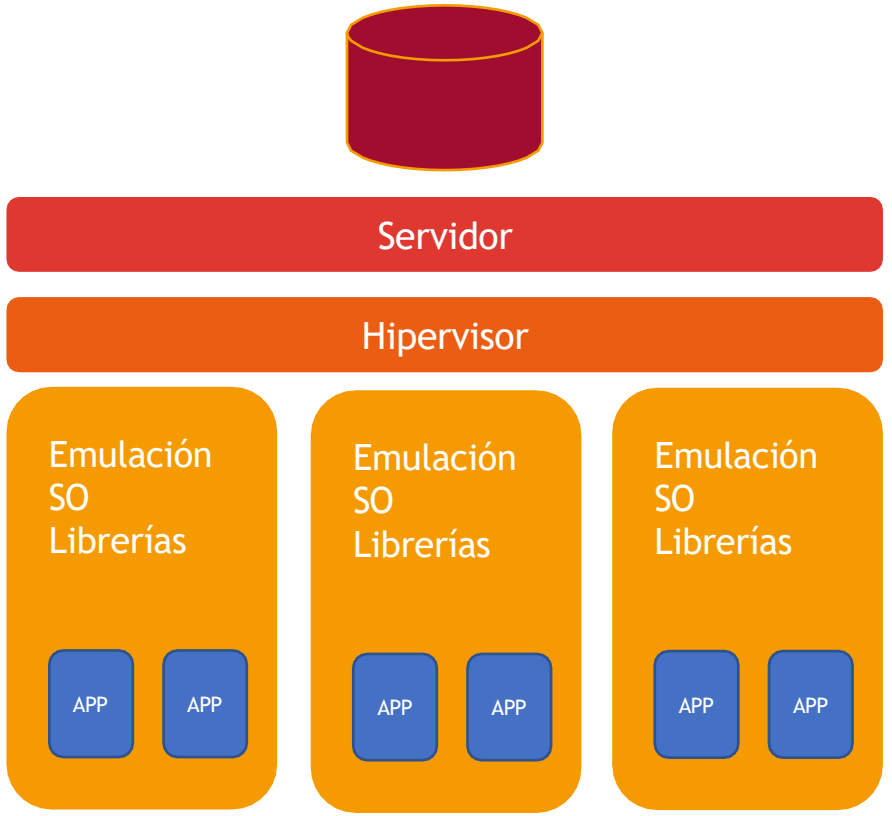


Figura 6. Máquinas virtuales. Elaborada por Joaquín Sanz Ferrándiz.

2. Contenedores

2.3. Segmentar en contenedores

Por último, la solución que existe a día de hoy son los contenedores. Lo que hace el servidor es destinar diferentes recursos o darles accesos a zonas del servidor de modo que estos nunca puedan rebasar esos límites.

Gracias esta solución lo que se consigue es eliminar ese sistema operativo que tanto ocupaba y, por consiguiente, lograr unas soluciones mucho mas livianas. Además, cada una de estas aplicaciones tendrá sus propias librerías para poder trabajar perfectamente.



Importante

Como se ha podido observar durante estas tres soluciones, lo que se pretende desde un principio es lo mismo, correr diferentes aplicaciones. Conforme se va avanzando, lo que se logra es optimizar al máximo los recursos de los servidores además de ejecutarlos mucho más rápido.



Figura 7. Contenedores. Elaborada por Joaquín Sanz Ferrándiz.

3. Contenedores Docker

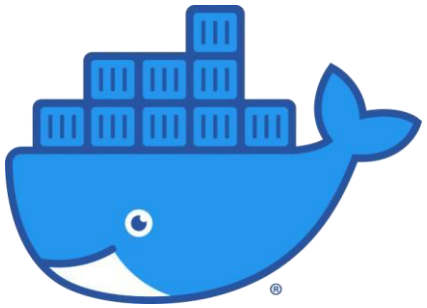


Figura 8: Docker. Fuente: <https://www.docker.com>

Se crearon en el año 2013 aprovechando la tecnología de contenedores en Linux. En la actualidad funcionan tanto en Windows como en Mac OSX además del ya mencionado Linux. La tecnología Docker es única ya que se centra en los requisitos de la aplicación estableciendo una independencia del servidor. Docker es utilizado a nivel global por diferentes empresas debido a:

- Carácter estándar: Docker *file* es un estándar de contenedor e incluye la información sobre cómo va a ser el contenedor, la base de la que se parte, su contenido, librerías y comportamiento una vez sea ejecutado.
- Fácil distribución: distribuyendo estos Docker *files* se puede construir desde cero un contenedor, algo muy sencillo ya que es un archivo de texto.
- Despliegue rápido: Docker aprovecha al máximo el espacio utilizado y al instalar únicamente las librerías necesarias se consigue un arranque muy rápido.
- Alta densidad: permite optimizar mucho los recursos de los servidores al utilizar solo los recursos necesarios.

3. Contenedores Docker

3.1. Imágenes en Docker

Las imágenes de Docker son la base de la que se parte a la hora de crear un contenedor propio. Estas pueden ser de diferentes tipos, desde un sistema operativo completo a una aplicación sencilla.

Un registro Docker es básicamente un repositorio de imágenes desde el que se pueden descargar imágenes oficiales realizadas por los propios programadores de *software* u otras en las cuales la gente hace modificaciones. La descarga de las imágenes es muy sencilla, básicamente se busca en el navegador la imagen necesaria o el *software* y se descarga.



Enlace

Docker Hub es el registro más famoso y grande del mundo, en el que se pueden encontrar más de 100.000 imágenes:

<https://hub.docker.com/>

4. Distribución Industrial Edge

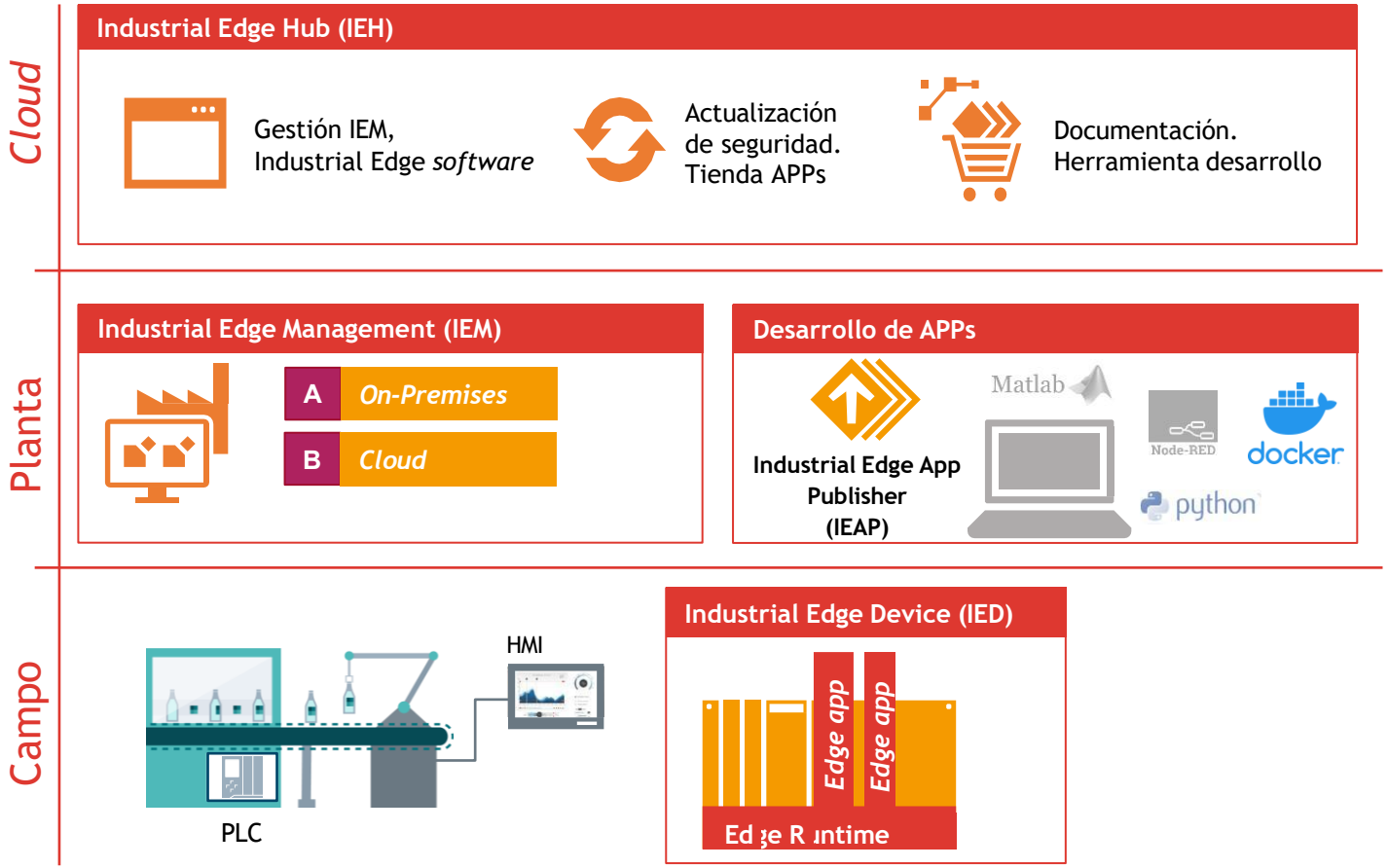


Figura 9. Distribución Industrial Edge. Adaptada de: <https://www.automation.siemens.com>

De aquí en adelante se utilizarán diferentes abreviaturas para hacer referencia a los distintos componentes de la distribución de *edge*:

- **IEH**: Industrial Edge Hub, basado en *cloud* se encuentran APPs, genera IEMs…
- **IEM**: Industrial Edge Management, se transfiere la información obtenida al IED.
- **IEAP**: Industrial Edge APP Publisher, herramienta para poder crear APPs propias.
- **IED**: Industrial Edge Device, dispositivo *edge* sobre el cual se vuelcan las distintas APPs.

4. Distribución Industrial Edge

4.1. Industrial Edge Hub (IEH)

El Industrial Edge Hub es el portal central de toda la plataforma *edge*. Está localizado en la nube y desde aquí se pueden realizar las siguientes acciones:

- Documentación: descarga documentación necesaria para manejo y operación.
- *Software*: se puede descargar la imagen del IEM o el IEAP.
- Registro IEM: se pueden crear los diferentes gestores o ver los que se han generado, además aporta información sobre si está conectado o no.
- *Market*: repositorio de aplicaciones y conectores SIMATIC global.
- *Library*: repositorio de aplicaciones y conectores SIMATIC descargados en el IEH, desde el que se pueden transferir las aplicaciones a los distintos IEM.

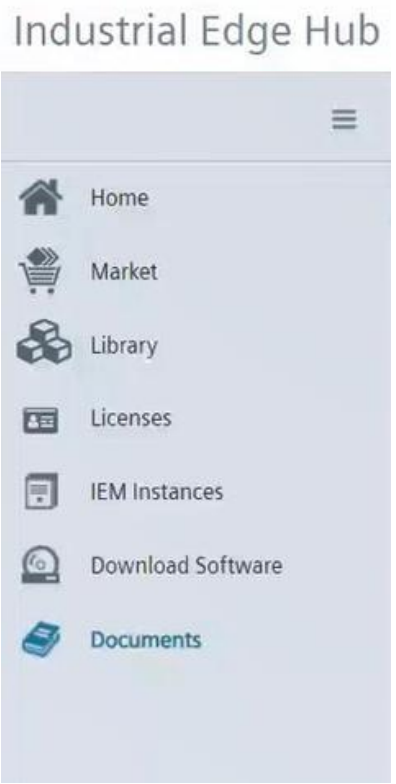


Figura 10. Industrial Edge Hub. Elaborada por Joaquín Sanz Ferrándiz.

4. Distribución Industrial Edge

4.2. Industrial Edge Management (IEM)

El Industrial Edge Management es el gestor de aplicaciones y de equipos *edge*. Desde este se pueden realizar:

- Catálogo: en él se pueden observar todas las aplicaciones y conectores cargados en el propio IEM desde el IEH y transferirlos a los dispositivos *edge*.
- Dispositivos *edge*: aparecen todos los dispositivos *edge* que controla nuestro IEM, además de ver información como consumos de CPU, aplicaciones ejecutándose…
- Aplicaciones: se encuentran las aplicaciones desarrolladas, propias y transferidas por el IEAP.
- Aplicaciones instaladas: se realizan actualizaciones a todos los dispositivos que tengan la aplicación y se configuran los conectores.
- Grupos: se pueden definir usuarios y decidir qué aplicaciones o dispositivos controlan, además de diferentes roles para una mayor versatilidad.

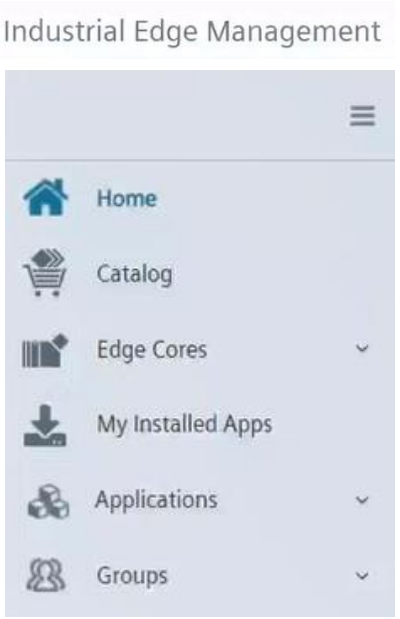


Figura 11. Industrial Edge Management. Elaborada por Joaquín Sanz Ferrándiz.

4. Distribución Industrial Edge

4.3. Industrial Edge App Publisher (IEAP)

El Industrial Edge App Publisher es el desarrollador de aplicaciones y se utiliza para transferir aplicaciones creadas por un usuario al IEM.

4. Distribución Industrial Edge

4.4. Industrial Edge Device (IED)

El Industrial Edge Device es el propio dispositivo *edge*:

- Apps: aplicaciones instaladas en el propio dispositivo, que se pueden desinstalar, arrancar, comprobar si hay nuevas versiones, ver los recursos utilizados…
- Estadísticas: información general sobre las aplicaciones instaladas, memoria del equipo, IPs…
- Conectividad: el propio Industrial Edge tiene varios conectores para comunicar con otros equipos.
    - S7 Connector: conecta a través de enlace S7 y OPC UA.
    - Industrial Edge Databus: Broker MQTT para comunicar entre aplicaciones.
    - Cloud Connector: envío de datos al *cloud*.



Figura 12. Industrial Edge Device. Elaborada por Joaquín Sanz Ferrándiz.

Referencias bibliográficas

Docker. <https://www.docker.com/>

Docker Hub. <https://hub.docker.com/>

Edge computing

@Copyright Universidad Europea de Madrid. Todos los derechos reservados.