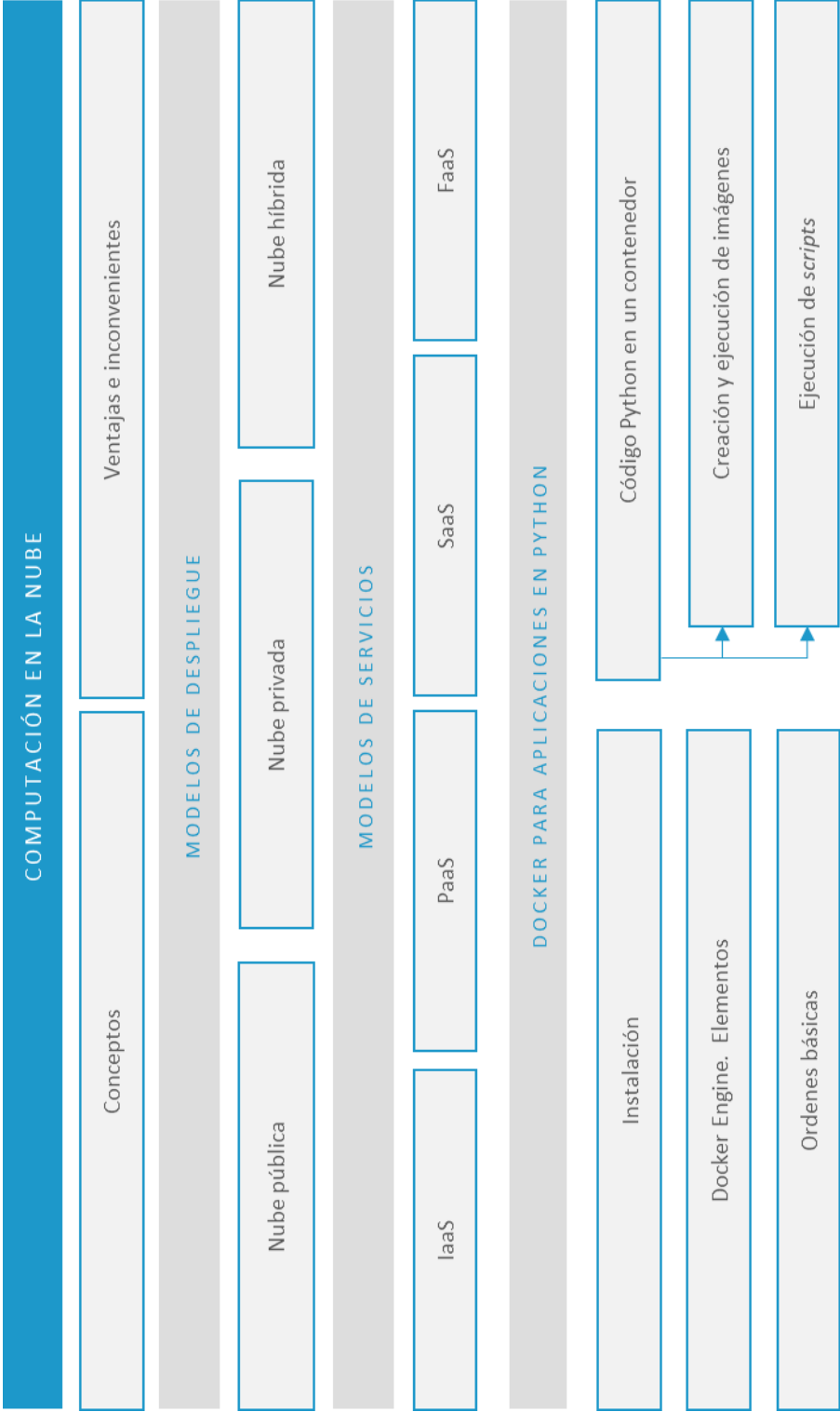


Programación Científica y HPC

Computación en la nube

Índice

Esquema	3
Ideas clave	4
10.1. Introducción y objetivos	4
10.2. Modelos de despliegue	6
10.3. Categorización: IaaS, PaaS, FaaS, SaaS	7
10.4. Ventajas e inconvenientes de la nube	10
10.5. Docker para aplicaciones en Python	12
10.6. Referencias bibliográficas	20
A fondo	21
Test	22



Esquema

10.1. Introducción y objetivos

En términos generales, la computación en la nube, conocida como *cloud computing*, describe una nueva **forma de computación basada en la red y que se lleva a cabo a través de Internet**.

El *cloud computing* es una propuesta más avanzada de servicios, que **permite el acceso de forma remota a distintos paquetes de software**, da la posibilidad de contar con un almacenamiento de archivos y **realizar el procesamiento de datos en procesadores en la nube**. De esta forma, se contará con una serie de servicios y recursos *hardware*, *software* y de red escalables que, en general, cuentan con la capacidad necesaria para que no se requieran instalaciones en la computadora u ordenador personal, ni realizar un mantenimiento. Con esto, **se ahorra en costes de inversión en equipos, unidades de almacenamiento y software**, además del coste de **mantenimiento**, ya que en realidad se «alquilan» cuando se necesitan.

Estas plataformas ocultan la complejidad y los detalles de la infraestructura subyacente a los usuarios y las aplicaciones, ya que proporcionan una interfaz gráfica muy sencilla o API.

La computación en la nube utiliza una capa de red para conectar los dispositivos de los usuarios con los recursos disponibles. En este tipo de sistemas, por tanto, se cuenta con un entorno o nube en el que se ejecutarán las aplicaciones y que **agrupará los recursos en red para que sean compartidos**.

En definitiva, la computación en la nube ofrece servicios haciendo uso de una **conectividad a gran escala y accesible a través de Internet**.

En estos sistemas se pueden distinguir dos partes:

- ▶ La **interfaz de usuario o *front end***, que facilita el acceso al usuario. En esta parte se cuenta con la red y la aplicación que permite acceder al servicio en la nube y que, por tanto, dará acceso a la nube.
- ▶ Y el **entorno que facilita los servicios o *back end***. En este caso, se puede decir que el entorno estará constituido por:
 - Los sistemas de almacenamiento de información.
 - Las aplicaciones disponibles.
 - El servidor central que administrará el funcionamiento del sistema. Para esto, se usará una capa de red y una serie de protocolos que facilitan las conexiones entre los distintos equipos de la red.

Con todo lo mencionado, estas plataformas pueden ofrecer:

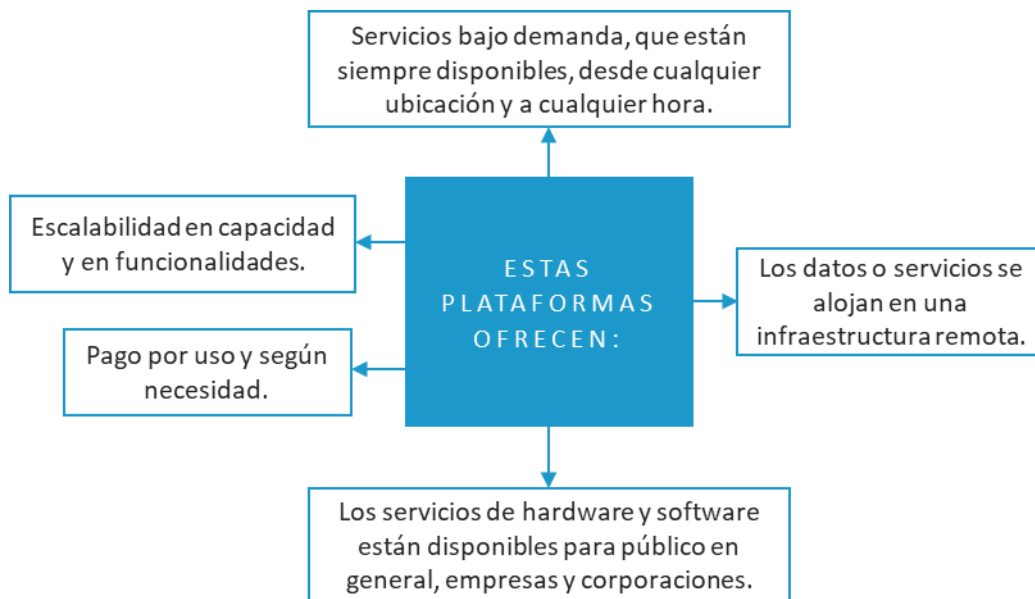


Figura 1. Estas plataformas ofrecen. Fuente: elaboración propia.

Los modelos de funcionamiento de la computación en nube son los siguientes:

- ▶ Modelos de despliegue.
- ▶ Modelos de servicio.

10.2. Modelos de despliegue

Estos modelos se definen como los tipos de nube disponibles que se distinguen por la **cantidad de gestión que requieren del cliente y por su nivel de seguridad**.

Nube pública

Proporciona el acceso a sus recursos, sistemas y servicios, con fácil accesibilidad para **cualquier tipo de usuario que quiera obtener los servicios** ofertados. Toda la **infraestructura informática se encuentra en el sitio del proveedor**. En este caso, es importante que el proveedor sea de confianza, ya que no se tiene control sobre otros posibles clientes que estén compartiendo la infraestructura que proporciona.

En este tipo se cuenta, entre otras, con [Amazon Web Services](#) (AWS), [Azure](#) de Microsoft, [Google Cloud Compute Engine](#) o [Oracle Cloud Infrastructure](#) (OCI).

Nube privada

Proporciona **servicios de uso exclusivo** para cada cliente, es decir, los sistemas y servicios son accesibles desde una organización. El alojamiento de **la nube puede estar en la propia organización o en el centro de datos del proveedor**. Esta nube proporciona un nivel más alto de seguridad y control. Los usuarios de este tipo de nubes suelen ser organizaciones y entidades en general de gran volumen.

Nube híbrida

Es una solución que está adquiriendo gran relevancia en la actualidad, debido a su gran potencial. En este caso, se contratan una serie de **servicios que se pueden ir ampliando según las necesidades**. Básicamente, los clientes alojan sus aplicaciones más críticas en la nube privada y, de esta forma, cuentan con más seguridad y control. Las demás aplicaciones no críticas se alojan en la nube pública.

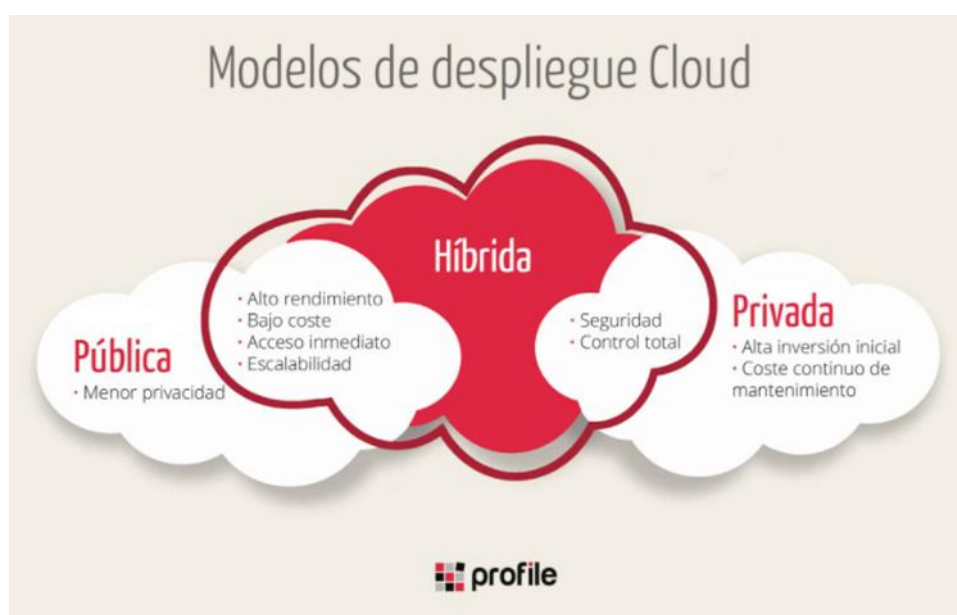


Figura 2. Modelos de despliegue. Fuente: <https://profile.es/blog/servicios-cloud-que-es-iaas-saas-y-paas/>

10.3. Categorización: IaaS, PaaS, FaaS, SaaS

En este apartado se expondrán lo que se conoce como los **modelos de servicio**, que es el modelo de referencia en el que se basa la computación en nube y representan las distintas categorías de servicios disponibles para el *cloud computing*.

Hay tres modelos principales de servicios en la nube para los que se usan acrónimos bastante extendidos:

Infraestructura como servicio (IaaS)

Permite el acceso a recursos fundamentales como máquinas físicas, virtuales y almacenamiento de datos y redes a demanda. La infraestructura es administrada por el proveedor, pero el cliente generalmente es responsable de la instalación, configuración, protección y mantenimiento del *software* que se instala en los recursos de la infraestructura, como pueden ser las bases de datos, *middleware* y algunas aplicaciones. **Está orientada a los administradores de sistemas y los arquitectos de red.** El cliente tiene más control, es un modelo más escalable y en el que la seguridad es mayor. El nivel de abstracción es más bajo lo que la hace menos intuitiva para el usuario final.

Plataforma como servicio (PaaS)

Proporciona al cliente el entorno de ejecución de aplicaciones y herramientas de desarrollo y despliegue. Esto **permite la implementación y gestión de**, entre otras, **aplicaciones para móviles y para web.** Este tipo de solución incluye además la infraestructura y otras soluciones que permiten soportar el ciclo de vida de las aplicaciones como compilación, implementación, pruebas, mantenimiento y administración. **El modelo está orientado al uso por desarrolladores de aplicaciones y cuenta con una arquitectura multinivel altamente escalable.**

Software como servicio (SaaS)

En este modelo se **alojan las aplicaciones del cliente en su entorno** de forma que ejecuta sus aplicaciones en la nube, es decir, pone a disposición básicamente recursos *software*. El cliente cuenta con una suite de servicios, es decir, una infraestructura de cómputo donde **las aplicaciones están siempre actualizadas y gestionadas por el proveedor.** Posibilita que los clientes pueden ampliar o reducir los servicios que tienen contratados de acuerdo con sus necesidades. Es un **modelo centrado en el**

usuario final, un poco **menos escalable y seguro** que los otros, pero con un nivel de abstracción y facilidad de uso mayor.

En la Figura 3 se pueden ver los modelos con sus características.

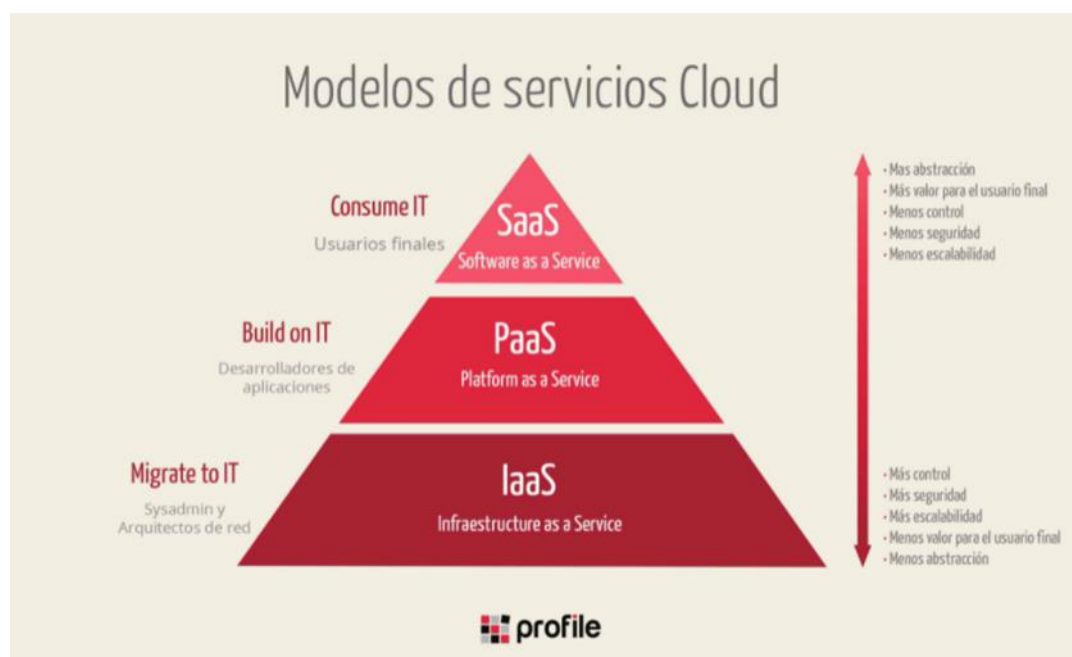


Figura 3. Modelos de servicios en la nube. Fuente: <https://profile.es/blog/servicios-cloud-que-es-iaas-saas-y-paas>

Por último, otro modelo es funciones como servicio (**FaaS**). Es una arquitectura que **permite la ejecución a través de contenedores efímeros que se crean en el momento del uso**. Favorece el desarrollo de arquitecturas que se basan en microservicios y facilita los despliegues continuos. Se conocen como arquitecturas *serverless*, no porque no haya un servidor, sino porque los servidores son un elemento más de la infraestructura. Solo se generan gastos cuando se ejecuta la función correspondiente.

Un contenedor es un paquete lógico en el que las aplicaciones pueden independizarse del entorno de ejecución.

10.4. Ventajas e inconvenientes de la nube

A continuación, se enumeran las ventajas e inconvenientes del uso *del cloud computing*, aunque podemos adelantar que hay más ventajas siempre que las necesidades de computación y almacenamiento de información justifiquen su uso y el pago por los servicios ofertados.

Ventajas

Flexibilidad

Permite que **las demandas de los usuarios puedan ser atendidas de forma inmediata y que éstas puedan variar en distintos momentos**. Los servicios contratados pueden ser de distinta naturaleza, como capacidad de procesamiento o capacidad de almacenamiento, pudiendo modificarse en cualquier momento. También es flexible la cantidad de tiempo contratada para el uso que puede ser por horas, días, semanas o meses.

Escalabilidad

La cantidad de servicios disponibles permiten satisfacer las demandas crecientes de los clientes, es decir, el sistema crece y se adapta a las necesidades.

Confianza

A menudo se construyen arquitecturas con redundancia, lo que garantiza un alto porcentaje de acceso a los servidores, aunque se produzcan fallos de *hardware*.

Copias de seguridad (*backup*) y recuperación

Estas cuestiones pueden ser manejadas por los proveedores de *cloud computing* para la realización de copias periódicas, a demanda de una parte o la totalidad de los recursos o datos de información. La recuperación también se gestiona de forma eficiente.

Actualizaciones

Las actualizaciones de *software* son controladas por los proveedores, lo que posibilita contar siempre con las versiones más modernas de estos.

Hiperconvergencia

Combinan en un solo sistema redes y almacenamiento y configuran un pequeño centro de datos, lo que facilita el trabajo para organizaciones multisede, posibilitando el acceso compartido a la información y el trabajo colaborativo, al también compartir aplicaciones.

Seguridad

Protección de la integridad de las aplicaciones, los datos y las infraestructuras. Los riesgos de seguridad son mayores en las nubes públicas, pero a medida que se impone este tipo de servicios, los proveedores han avanzado en este sentido. Las acciones van dirigidas a asegurar la confidencialidad de la información y garantizar el mantenimiento y el borrado de los datos cuando acabe el servicio, sin que por ello se vea afectado el rendimiento. De cualquier forma, la seguridad perfecta no existe, con lo que se debe trabajar y analizar de manera especial los procesos y las soluciones que se deben aplicar.

Reducción de costes

Con el uso de la nube se elimina la necesidad de hacer inversiones grandes en hardware y en software porque son, en su mayoría, proporcionadas por la infraestructura de la nube.

Fomento de la innovación

Se ofrecen nuevas aplicaciones y servicios que permiten el acceder al uso de nuevas tecnologías como la inteligencia artificial (IA) y facilidades de internet de las cosas (IoT, por sus siglas en inglés).

Inconvenientes

Algunos de los inconvenientes con los que nos podemos encontrar son:

Conectividad

Es importante contar con una buena conectividad a internet por parte del cliente para el acceso a los servicios.

Calidad y disponibilidad del servicio

Los servicios que proporciona la nube están disponibles de forma permanente y son servicios de calidad porque cuentan con un soporte y mantenimiento experto.

10.5. Docker para aplicaciones en Python

Docker es una **herramienta de creación de contenedores que se usa para crear entornos de aplicación aislados y reproducibles**. Es muy popular entre los desarrolladores de Python. Con este se pueden desarrollar aplicaciones y desplegarlas, tanto localmente, como en la nube, acelerando el proceso de paso a producción.

El propósito de los contenedores es la **separación de los procesos**, de modo que puedan ejecutarse de manera independiente sobre una infraestructura, al mismo tiempo que se conserva la seguridad con la que se contaría en sistemas separados.

En realidad, Docker es también una **plataforma que permite ejecutar contenedores con aplicaciones preempaquetadas**. El sistema se usa para empaquetar aplicaciones con todas sus dependencias, para que pueda ser ejecutada en plataformas diferentes. Posteriormente, el despliegue es rápido y se puede repetir.

Instalación

Docker debe ser instalado previamente para poder usarlo de forma conveniente. Al instalarlo se obtiene el *daemon* de Docker, el cliente Docker y el Docker *compose*.

En la sección A fondo podrán encontrar el enlace de la página oficial para poder descargarlo.

El motor de Docker: Docker Engine

Docker Engine **proporciona una arquitectura cliente-servidor que crea, envía y ejecuta contenedores Docker**. En la Figura 4 se muestran los elementos principales del motor y los elementos que maneja, que se enumeran a continuación de la ilustración.

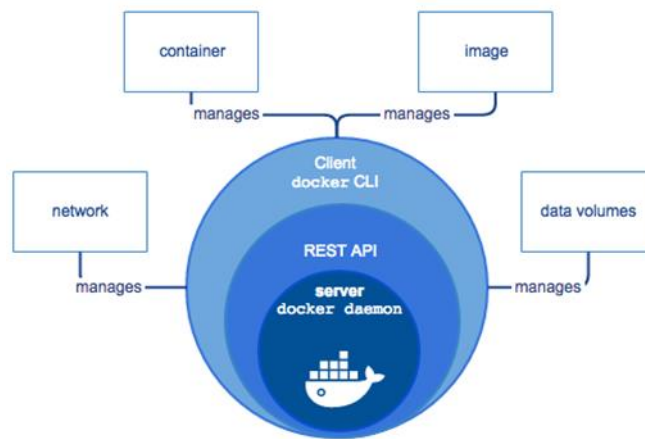


Figura 4. Elementos de Docker Engine. Fuente:
<https://www.arquitectoit.com/docker/arquitectura-docker/>

- ▶ Un servidor, que es un proceso demonio de ejecución continua.
- ▶ Una API REST, que especifica las interfaces que los programas pueden utilizar para hablar con el demonio y darle instrucciones.
- ▶ Un cliente de interfaz de línea de comandos (CLI).

En Docker los elementos que se manejan principalmente son:

- ▶ **Las imágenes.** Son plantillas que captura el estado del contenedor, por lo que es una representación estática de la aplicación y sus dependencias.
- ▶ **Contenedor.** Es una imagen en ejecución con un estado determinado.

En A fondo se proporciona un enlace a un repositorio de imágenes Docker compartidas, Docker Hub.

Ordenes básicas en Docker

Básicamente existen varias acciones que se ejecutan en la línea de comandos.

Construir una imagen

Las imágenes se crean en archivo de texto plano, en el que se describe cómo se debe configurar la imagen Docker con una serie de órdenes. Estos archivos se conocen como *dockerfile*. Entre otras cosas contiene:

- ▶ Código.
- ▶ Bibliotecas que se van a usar.
- ▶ Variables de entorno.
- ▶ Archivos de configuración.

Se construyen con la orden: `docker build -t nombre_imagen`. El punto especifica el directorio actual, como el contexto de construcción de la imagen y qué debe ser el directorio que contiene el *dockerfile*.

Ejecutar la imagen

Se recomienda el uso de `-rm` para eliminar el contenedor después de su uso: `docker run -rm nombre_imagen`.

Listar imágenes y listar contenedores

Las ordenes en este caso son:

- ▶ `docker images`
- ▶ `docker ps -a`

Borrar imágenes y contenedores

Borrar imágenes y contenedores usando su identificador, que se obtiene del listado de imágenes y de contenedores:

- ▶ `docker rmi imagen_id`
- ▶ `docker rm container_id`

Ejecución de código Python en un contenedor Docker

Para ejecutar Python en un contenedor Docker se puede ejecutar la siguiente orden en el terminal, que descargará la imagen `python:rc` desde Docker Hub, iniciará un contenedor y ejecutará Python dentro de ese contenedor. Las opciones `-it` son necesarias para ejecutar el contenedor de forma interactiva. Con la etiqueta `rc` se descarga la última versión de desarrollo de Python.

```
$ docker run -it --rm python:rc
```

Figura 5. Orden para la ejecución de Python en un contenedor Docker. Fuente: elaboración propia.

Crear una *dockerfile* para Python, crear la imagen y ejecutarla

Se debe tener en cuenta que **el contenedor es un entorno aislado**. Por lo tanto, normalmente no es necesario usar un entorno virtual dentro del contenedor.

En este caso se pueden instalar los paquetes necesarios con `pip`, incluyendo la orden en un *dockerfile*.

Ejemplo 1. Obtenido de Build Your Python Image, 2021

Se crea un archivo *dockerfile* con las siguientes directrices y ordenes:

- ▶ La primera línea es una directiva de análisis sintáctico opcional para que versiones más antiguas actualicen el analizador.
- ▶ En la segunda línea se importa la imagen oficial de Python con todas las herramientas y paquetes necesarios para la ejecución de una aplicación.
- ▶ En la siguiente línea se crea un directorio de trabajo. Se usarán rutas relativas basadas en el directorio de trabajo.
- ▶ Se instalan los paquetes `pip` para instalar todas las dependencias. Antes se copia el archivo de dependencias en la imagen con `COPY`. Se copia el archivo «requirements.txt» en el directorio de trabajo.
- ▶ Se añade el código fuente a la imagen con `COPY`, que toma todos los archivos del directorio actual y los copia en la imagen.
- ▶ Se indica a Docker la orden que se quiere ejecutar cuando la imagen se ejecute en el contenedor. Se indica el `-host=0.0.0.0` para que sea visible externamente.

El resultado es que se ha creado un directorio llamado `python-docker` y se crea una aplicación Python usando el *framework* Flask. Se crea un *dockerfile* que contiene las órdenes para construir una imagen.

```
# syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD [ "python3", "-m", "flask", "run", "--host=0.0.0.0" ]
```

Figura 6. Código para crear una *dockerfile* para Python, crear la imagen y ejecutarla. Fuente: elaboración propia.

La estructura de directorios de la aplicación Python es:

```
python-docker
|___ app.py
|___ requirements.txt
|___ Dockerfile
```

Figura 7. Estructura de directorios. Fuente: elaboración propia.

Construir la imagen:

```
$ docker build --tag python-docker
```

Figura 8. Orden para la construcción de una imagen Docker. Fuente: elaboración propia.

Ejecutar la imagen:

```
$ docker run -it --rm python-docker
```

Figura 9. Orden para la ejecución de la imagen Docker. Fuente: elaboración propia.

Ejecutar un *script* de Python usando Docker

Hay dos formas generales de ejecutar *scripts* de Python en un contenedor Docker:

- **Montar un directorio local como un volumen en el contenedor Docker.** Es la opción recomendada cuando se están realizando pruebas, ya que cuando se realizan cambios en el *script* no se necesitará reconstruir la imagen Docker.

Los volúmenes constituyen los mecanismos que se usan, preferentemente, en los contenedores Docker para la persistencia de los datos generados y usados por el contenedor.

- **Copiar el *script* en el contenedor Docker.**

Montar un directorio

Para montar un directorio como un volumen se usa la opción `-v`.

Ejemplo 2. Montar un directorio volumen y ejecutar un *script* de Python

Se supone que se ha implementado un *script* para mostrar los elementos de la sucesión de Fibonacci, `fibonacci.py` y se tiene en el directorio `/home/documents/pscript`. La orden que se debe ejecutar es:

```
$ docker run --rm -v /home/documents/pscript:/app rp python /app/fibonacci.py
```

Figura 9. Orden para montar un directorio y ejecutar el *script* Fibonacci de Python. Fuente: elaboración propia.

Con la opción `-v /home/documents/pscript:/app` el directorio local se monta como `/app` dentro del contenedor. A continuación se ejecuta el *script* con `rp python /app/fibonacci.py`.

Copiar el script en el contenedor Docker

Si se va a desplegar el *script* en otra máquina se puede copiar en el contenedor añadiendo algunas directrices en *dockerfile*.

Ejemplo 3. Copia de script en el contenedor usando *dockerfile*

```
FROM python:3.7.5-slim
WORKDIR /app
RUN python -m pip install //se instalan los módulos necesarios
COPY Fibonacci.py
CMD["python","fibonacci.py"]
```

En este vídeo podrán ver las características de la **computación en la nube**, modelos de despliegue y modelos de servicio.



Accede al vídeo

10.6. Referencias bibliográficas

Cuervo, M. (2 de diciembre de 2019). *Docker Engine* [Imagen]. Arquitecto IT. [Arquitectura Docker – Arquitecto IT](#)

Martín, M. J. (23 de marzo de 2018). *Pirámide de servicios Cloud: IaaS, PaaS y SaaS* [Imagen]. Profile. [Servicios Cloud: ¿Qué es IaaS, SaaS y PaaS? \(profile.es\)](#)

Martín, M. J. (23 de marzo de 2018). *Tipos de despliegue Cloud: nube pública, nube privada y nube híbrida* [Imagen]. Profile. [Servicios Cloud: ¿Qué es IaaS, SaaS y PaaS? \(profile.es\)](#)

Get Docker

Docker Docs (<https://docs.docker.com/get-docker/>)

En esta página oficial se ofrecen las instrucciones para la instalación de Docker en distintos sistemas operativos.

Docker Hub

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

Espacio para la creación, gestión y entrega de las aplicaciones de contenedores Docker.

Docker tutorial

Docker Tutorials [Tutoriales]. TecAdmin. [Docker Tutorials - TecAdmin](#)

Tutoriales para el uso de los contenedores de Docker.

1. La computación en la nube es:
 - A. Es un servicio de *hosting*.
 - B. Es un servicio gratuito para computación de alto rendimiento.
 - C. Es un servicio solo para almacenamiento de datos en la nube
 - D. Ninguna de las anteriores es verdadera.

2. La nube pública:
 - A. Proporciona el acceso a recursos de forma gratuita
 - B. Proporciona el acceso de forma más segura porque la infraestructura no es compartida por otros clientes.
 - C. Proporciona servicios en infraestructura compartida por todo tipo de clientes.
 - D. Ninguna de las anteriores es verdadera

3. La nube privada:
 - A. Proporciona el acceso a recursos de forma gratuita.
 - B. Proporciona el acceso de forma exclusiva, lo que proporciona más seguridad.
 - C. Proporciona servicios en infraestructura compartida por todo tipo de clientes.
 - D. Ninguna de las anteriores es verdadera.

4. El *cloud computing*:
 - A. Tiene el inconveniente de que no cuenta con escalabilidad para satisfacer demandas crecientes de clientes.
 - B. Tiene la ventaja de que cuenta con escalabilidad para satisfacer demandas crecientes de clientes.
 - C. Tiene el inconveniente de no gestionar la hiperconvergencia.
 - D. Ninguna de las anteriores es verdad.

5. La hiperconvergencia:
- A. No es soportada por la computación en la nube, al no poder gestionar distintos tipos de recurso y servicios de forma simultánea.
 - B. Solo es soportada para el acceso compartido a los datos.
 - C. Es soportada por la computación en la nube, al gestionar distintos tipos de recurso y servicios de forma simultánea.
 - D. Ninguna de las anteriores es verdadera.
6. SaaS:
- A. Es una infraestructura que permite el acceso a recursos de procesamiento, almacenamiento y redes gestionados por el proveedor.
 - B. Proporciona herramientas para el desarrollo.
 - C. Proporciona una *suite* de servicios para computación y ejecución de aplicaciones.
 - D. Ninguna de las anteriores es verdadera.
7. PaaS:
- A. Es una infraestructura que permite el acceso a recursos de procesamiento, almacenamiento y redes gestionados por el proveedor.
 - B. Proporciona una *suite* de servicios para computación y ejecución de aplicaciones.
 - C. Proporciona herramientas para el desarrollo.
 - D. Ninguna de las anteriores es verdadera.
8. SaaS:
- A. Es una infraestructura que permite el acceso a recursos de procesamiento, almacenamiento y redes gestionados por el proveedor.
 - B. Proporciona una *suite* de servicios para computación y ejecución de aplicaciones.
 - C. Proporciona herramientas para el desarrollo.
 - D. Ninguna de las anteriores es verdadera.

9. Una imagen Docker es:
- A. Una visualización de datos procesados.
 - B. Es una instancia de un contenedor en ejecución.
 - C. Son plantillas que capturan el estado del contenedor.
 - D. Ninguna de las anteriores es verdadera.
10. Un contenedor Docker es:
- A. Una visualización de datos procesados.
 - B. Es una instancia de una imagen en ejecución.
 - C. Son plantillas que capturan el estado del contenedor en la ejecución.
 - D. Ninguna de las anteriores es verdadera.