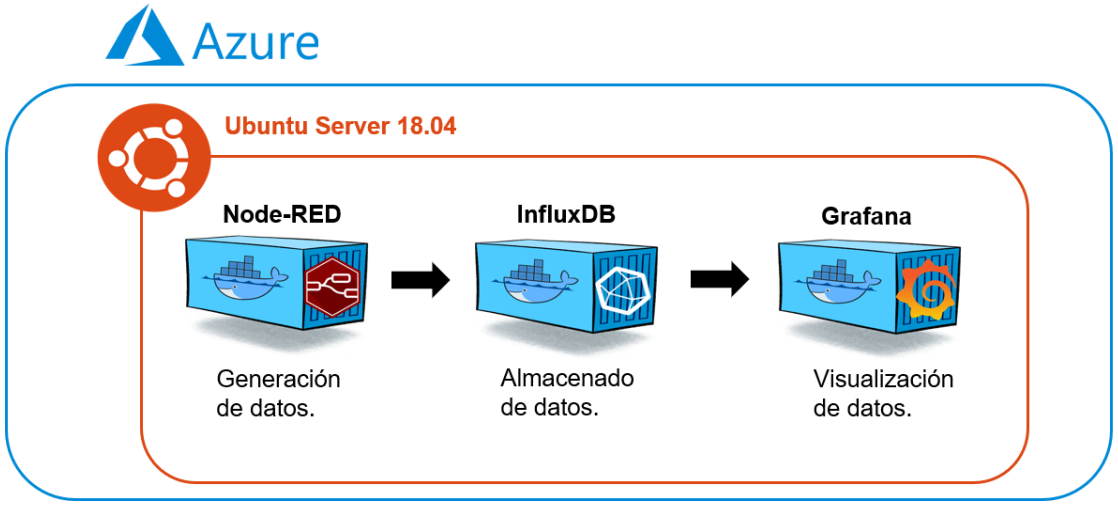
Introducción a Docker

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

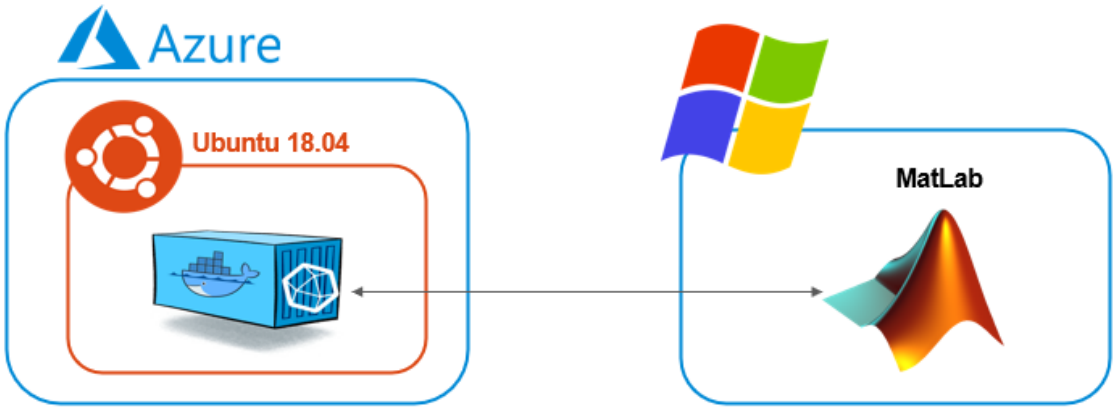
1. Motivación

Implementación a llevar a cabo en el proyecto de la asignatura permitirá aplicar los conceptos teóricos que van a ser presentados a continuación.

PRIMERA PARTE DEL PROYECTO



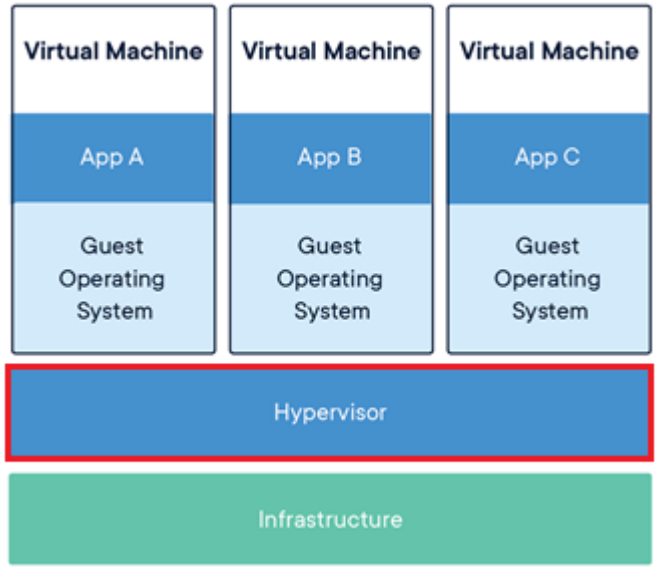
SEGUNDA PARTE DEL PROYECTO



2. Tecnología previa a Docker

En una implementación anterior a Docker, se utilizaba una máquina virtual para cada programa para garantizar:

- Reducir el riesgo de errores causados por incompatibilidades.
- Garantizar la máxima seguridad posible.
- Una mejor gestión de los recursos físicos disponibles, ya que el ***Hypervisor*** se encarga de dividirlos según las necesidades de las máquinas virtuales que estén funcionando en cada momento.



Una implementación de este tipo supone las siguientes consecuencias:

- Se necesitan tantas máquinas virtuales (cada una con su sistema operativo) como programas queramos integrar en una solución tecnológica.
- El coste de iniciar y mantener máquinas virtuales repercute en la escalabilidad (manteniendo un compromiso prestaciones vs coste).

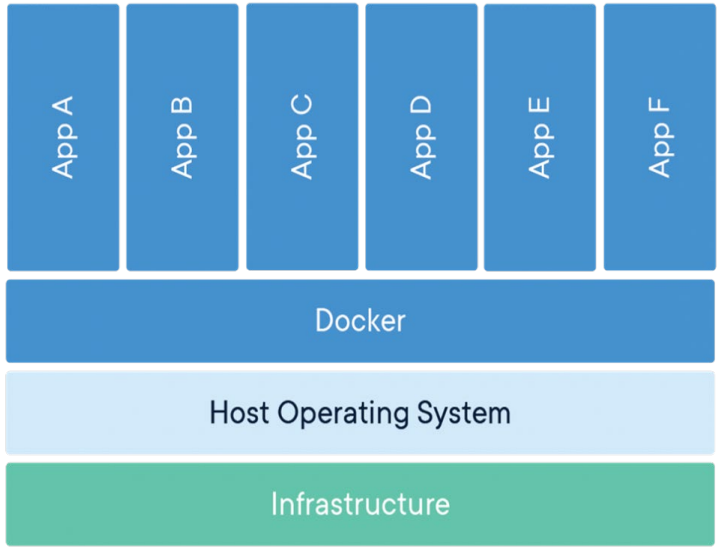
3. Tecnología Docker

La tecnología Docker aporta la posibilidad de usar los contenedores como máquinas virtuales extremadamente livianas y modulares. Un contenedor (o *container*) es una unidad de software que empaqueta código para que las aplicaciones funcionen como procesos aislada.

Ello permite mantener las ventajas de la tecnología anterior, añadiendo:

- La seguridad de trabajar siempre con instalaciones limpias y aisladas gracias a que cada contenedor se genera como una nueva instancia cada vez que se arranca.
- La posibilidad de construir imágenes *custom* garantizando una configuración específica.
- Un registro público donde los desarrolladores cuelgan sus imágenes y los usuarios pueden descargarlas.

Todo ello favorece enormemente la escalabilidad de las soluciones y la automatización del proceso de diseño y puesta en marcha.



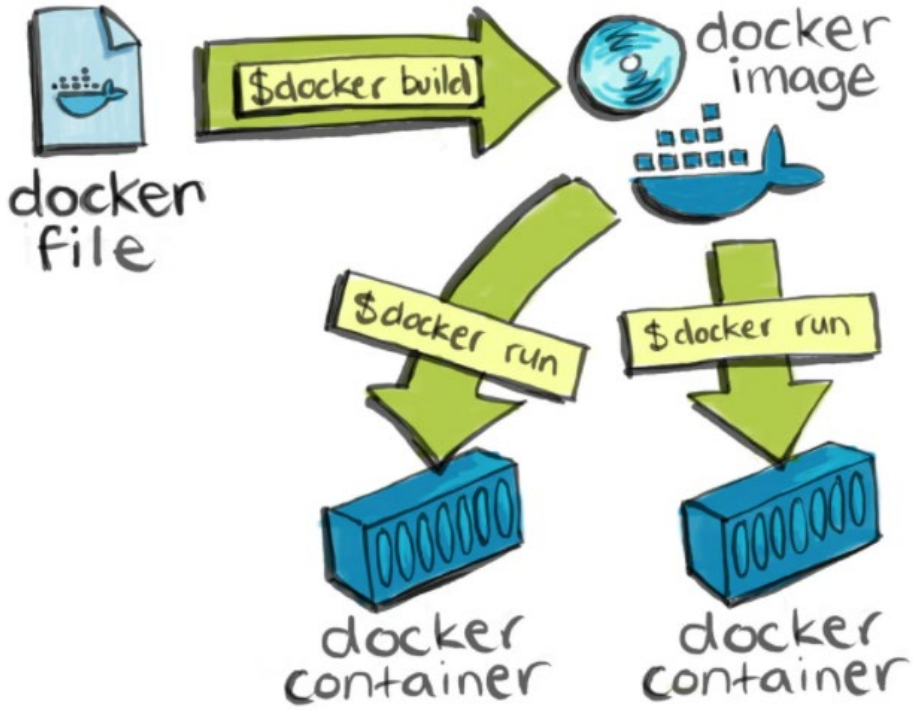
4. Archivo DockerFile

La tecnología Docker permite definir un archivo DockerFile para cada contenedor, es decir, para cada software. En este archivo se puede definir (utilizando una sintaxis concreta) una serie de instrucciones que se ejecutarán en el momento de creación de una imagen.

Las instrucciones pueden hacer referencia a aspectos como:

- Instalar unos paquetes concretos sobre la imagen base.
- Añadir un archivo en una ruta específica dentro del contenedor.
- Exponer un puerto de comunicaciones.

Una vez escrito, la imagen se puede generar mediante el comando *build*.



5. Docker Compose

Es una herramienta para definir y ejecutar soluciones multi-contenedor, que utiliza un archivo YAML para configurar las distintas aplicaciones y lleva a cabo el proceso de creación y arranque de los contenedores con solo un comando (*docker-compose up*). La configuración implica que se especifique, entre otros:

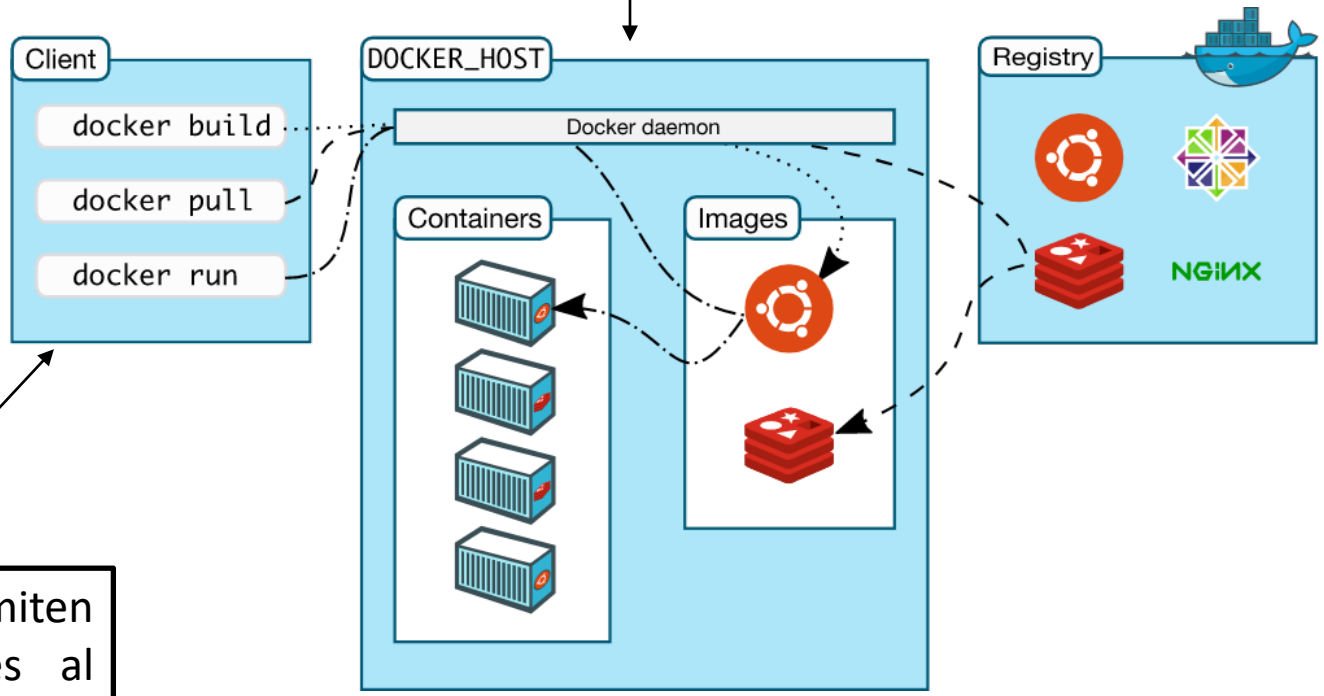
- La imagen que se utilizará. Docker Compose puede utilizar imágenes genéricas u otras que hayan sido previamente generadas a partir de un DockerFile según las necesidades de una solución tecnológica concreta.
- La definición de las variables de entorno necesarias.
- La correspondencia entre los puertos de comunicaciones entre el contenedor y la máquina virtual

```yaml
version: '3'

services:
        node-red:
             image: nodered/node-red:latest
             environment:
                - TZ=Europe/Amsterdam
             ports:
                - "1880:1880"
             networks:
                network1:
                     ipv4_address: 172.16.238.4
             volumes:
                - node-red-data:/data
        web:
                image: grafana/grafana
                ports:
                 - "3002:3000"
                environment:
                - GF_SECURITY_ADMIN_USER=admin
                - GF_SECURITY_ADMIN_PASSWORD=supersecretpassword
                volumes:
                - grafana-storage:/var/lib/grafana
                - ./grafana-provisioning/:/etc/grafana/provisioning
                networks:
                        network1:
                                ipv4_address: 172.16.238.3

        database:
                image: influxdb
                ports:
                 - "8086:8086"
                volumes:
                 - ./var/lib/influxdb
                environment:
                - INFLUXDB_DB=mydb
                - INFLUXDB_ADMIN_USER=admin
                - INFLUXDB_ADMIN_PASSWORD=supersecretpassword
                networks:
                     network1:
                             ipv4_address: 172.16.238.2
```

En el ejemplo a la derecha de esta diapositiva podéis ver un archivo YAML muy parecido al que vais a utilizar para generar los tres contenedores que utilizaréis en el proyecto.

6. Implementación de la tecnología Docker

Software **host** que gestiona contenedores e imágenes.



Comandos que permiten enviar instrucciones al programa des del software **cliente.**

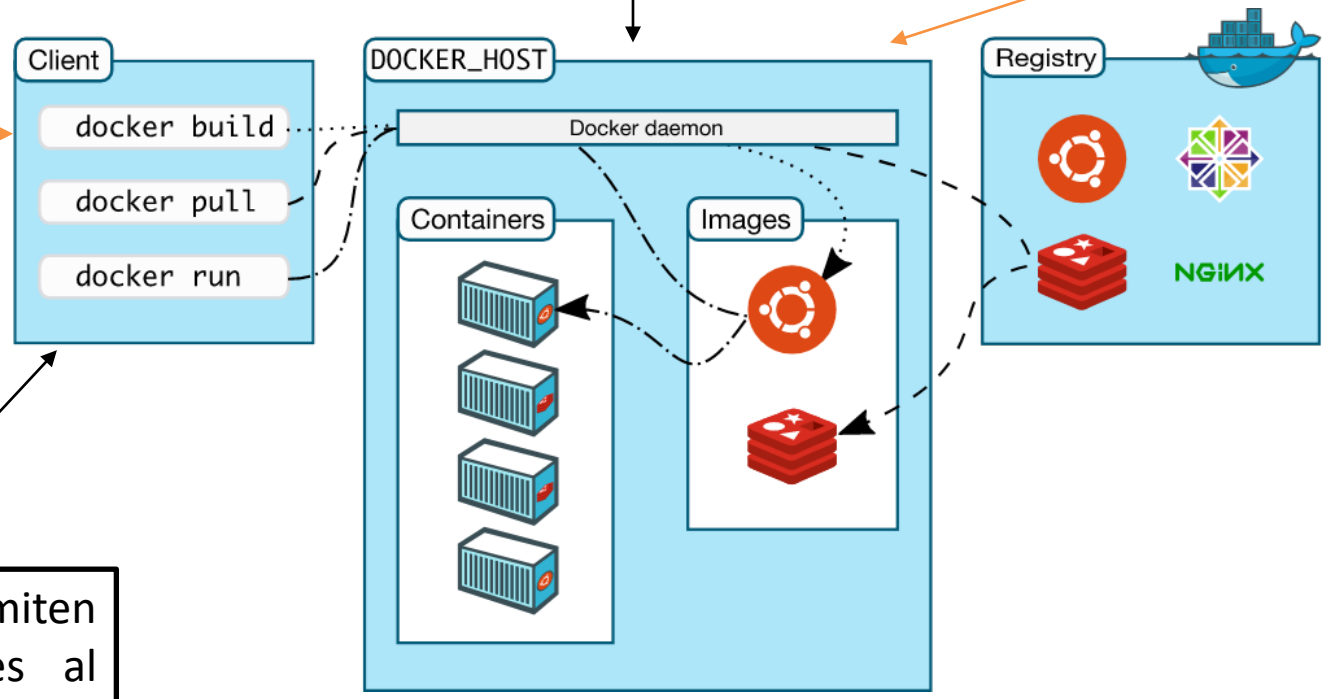
Registro público de imágenes **Docker Hub.**

6. Implementación de la tecnología Docker

Ejecutaréis estos comandos en la propia VM cuando os conectéis a ella.

Software **host** que gestiona contenedores e imágenes.

Estará instalado en la VM de Azure que generaréis.



Comandos que permiten enviar instrucciones al programa des del software **cliente.**

Registro público de imágenes **Docker Hub.**