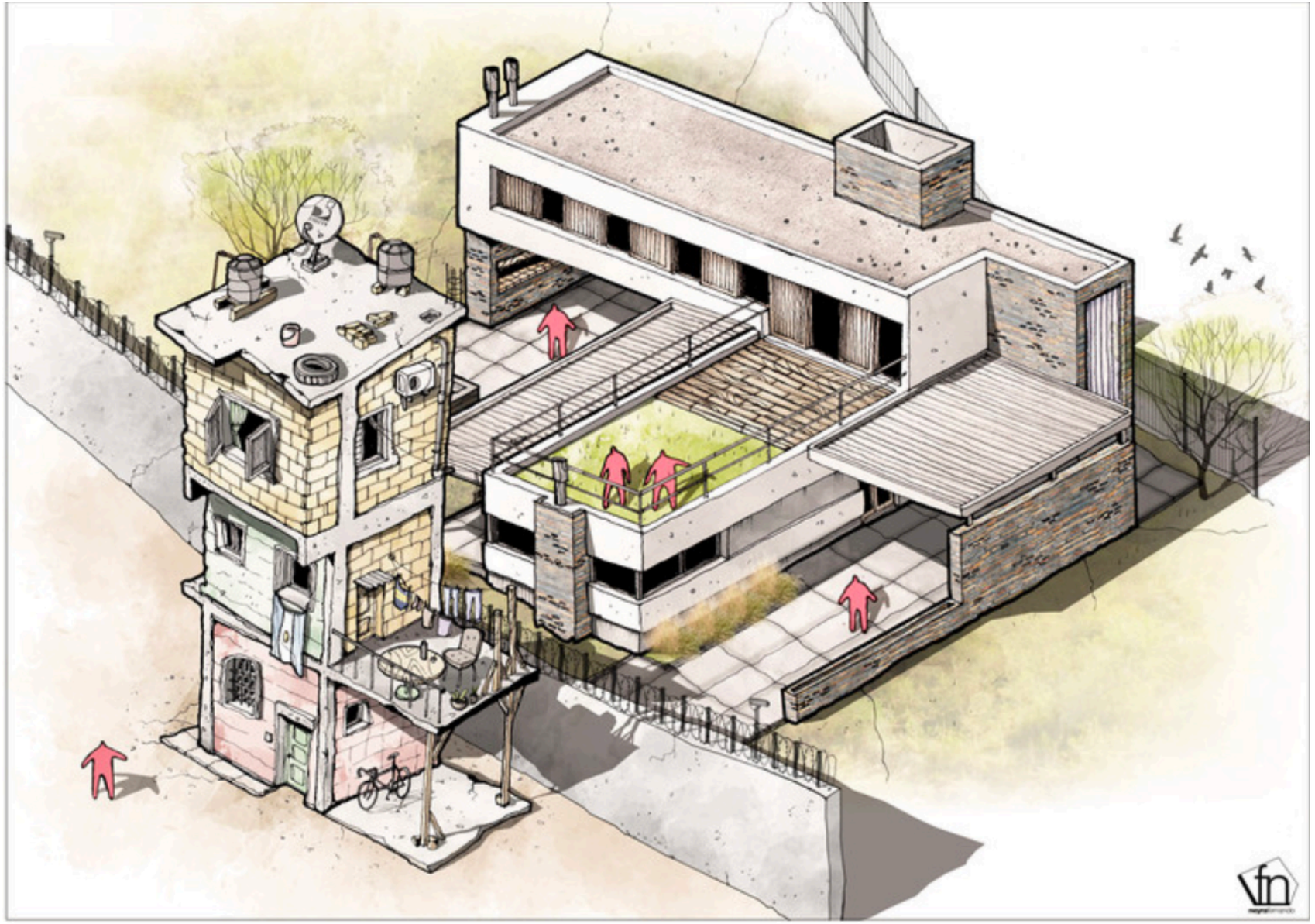
Ingeniería de Software

21 - Arquitectura de Software

IIC2143-3

Josefa España

jpespana@uc.cl



Josefa España

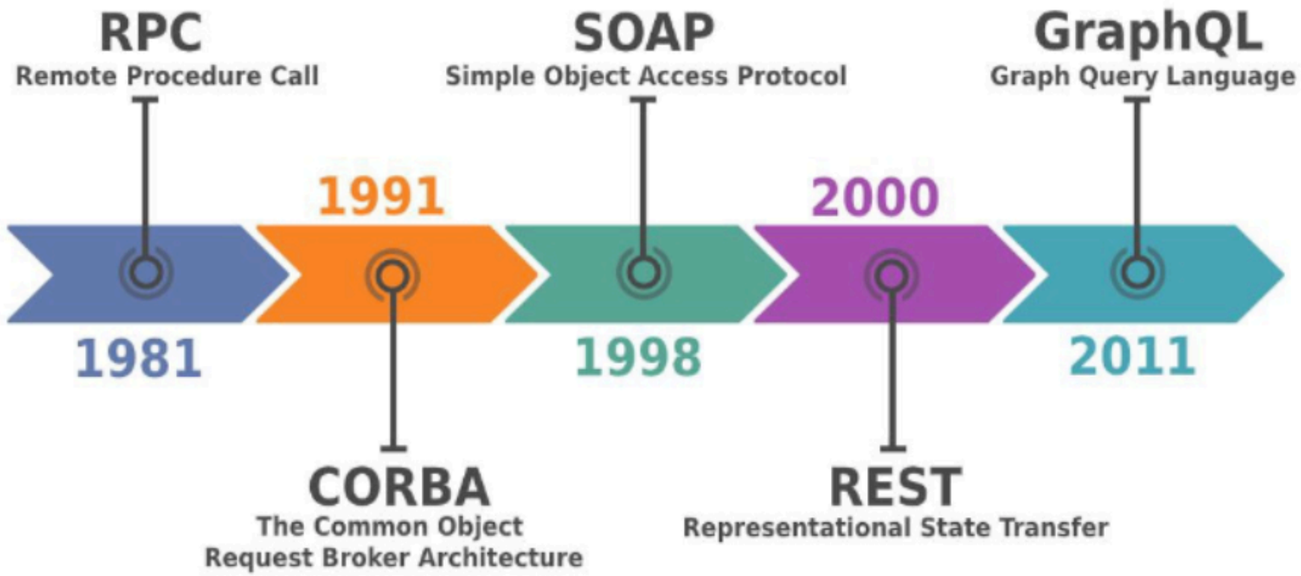
¿Qué es Arquitectura de Software?

- Centrarse en la estructura del desarrollo.
- Anticiparse a decisiones costosas.
- Hacer explícitas las decisiones para tener una buena calidad.

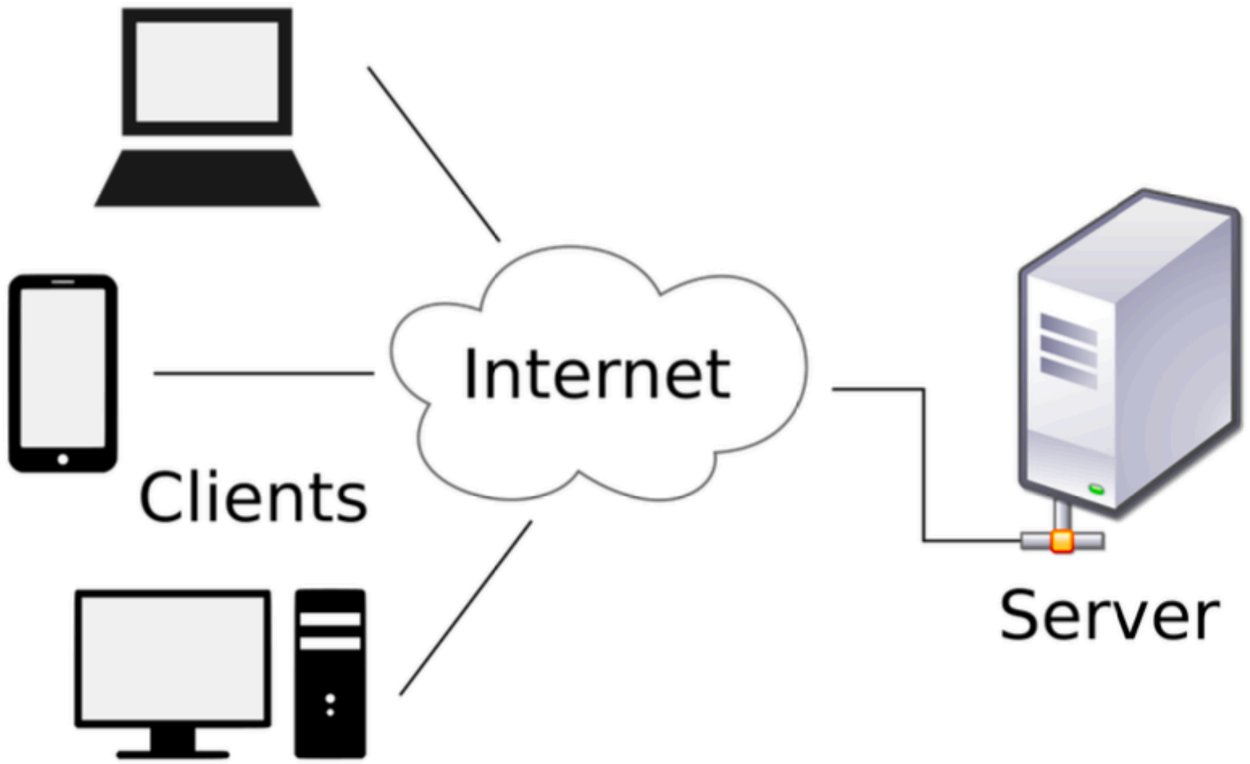
Requisitos funcionales en el siglo 21

- Sea desarrollado y mantenido por muchos años (**Maintainability**)
- Soporte a millones de usuarios (**Scalability**)
- Disponibilidad 24/7 los 365 días del año (**Reliability**)
- Queremos que tenga buena latencia (**Efficiency**)

Un poco de historia



Cliente Servidor


Internet
Clients
Server

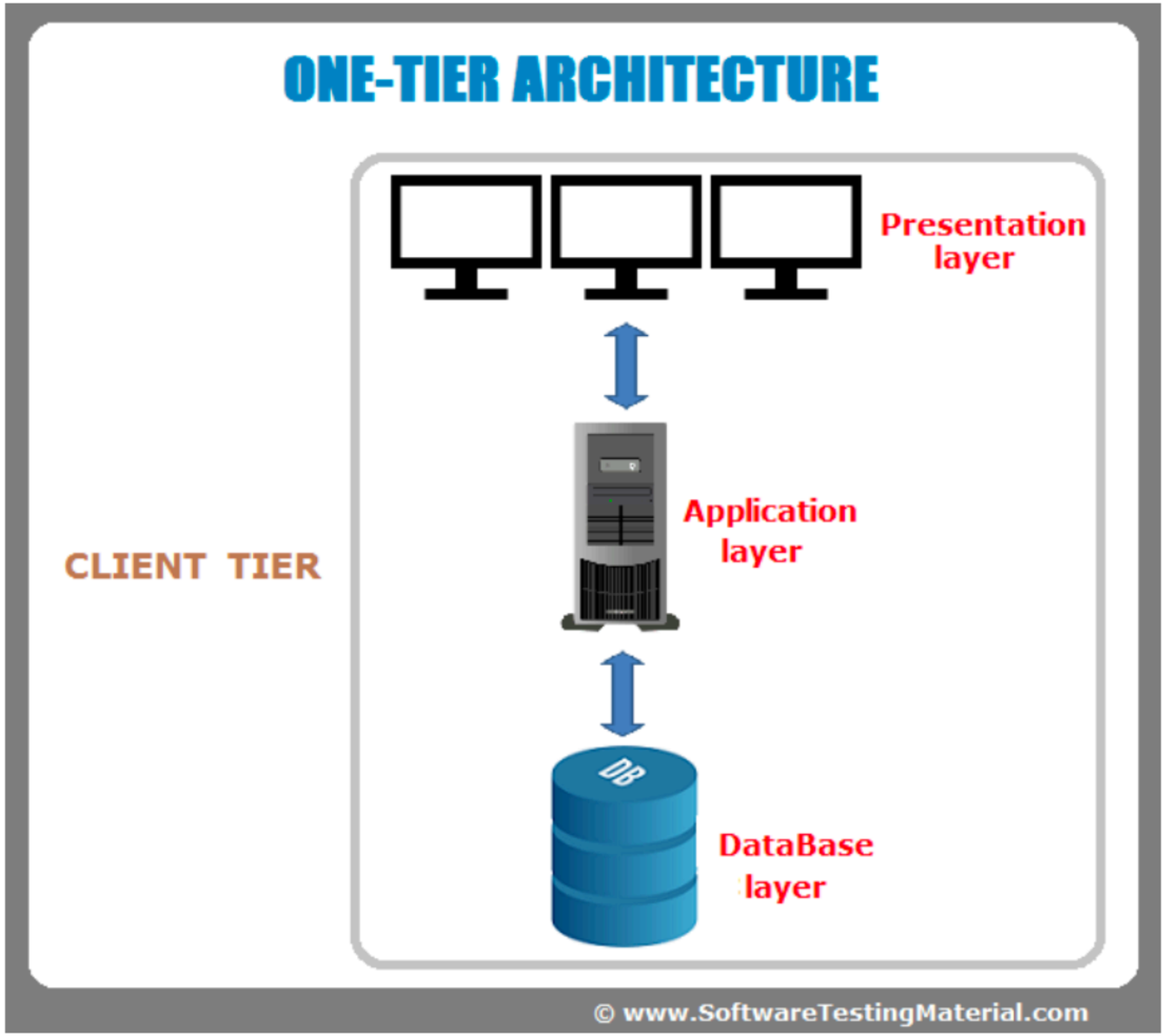
Clientes

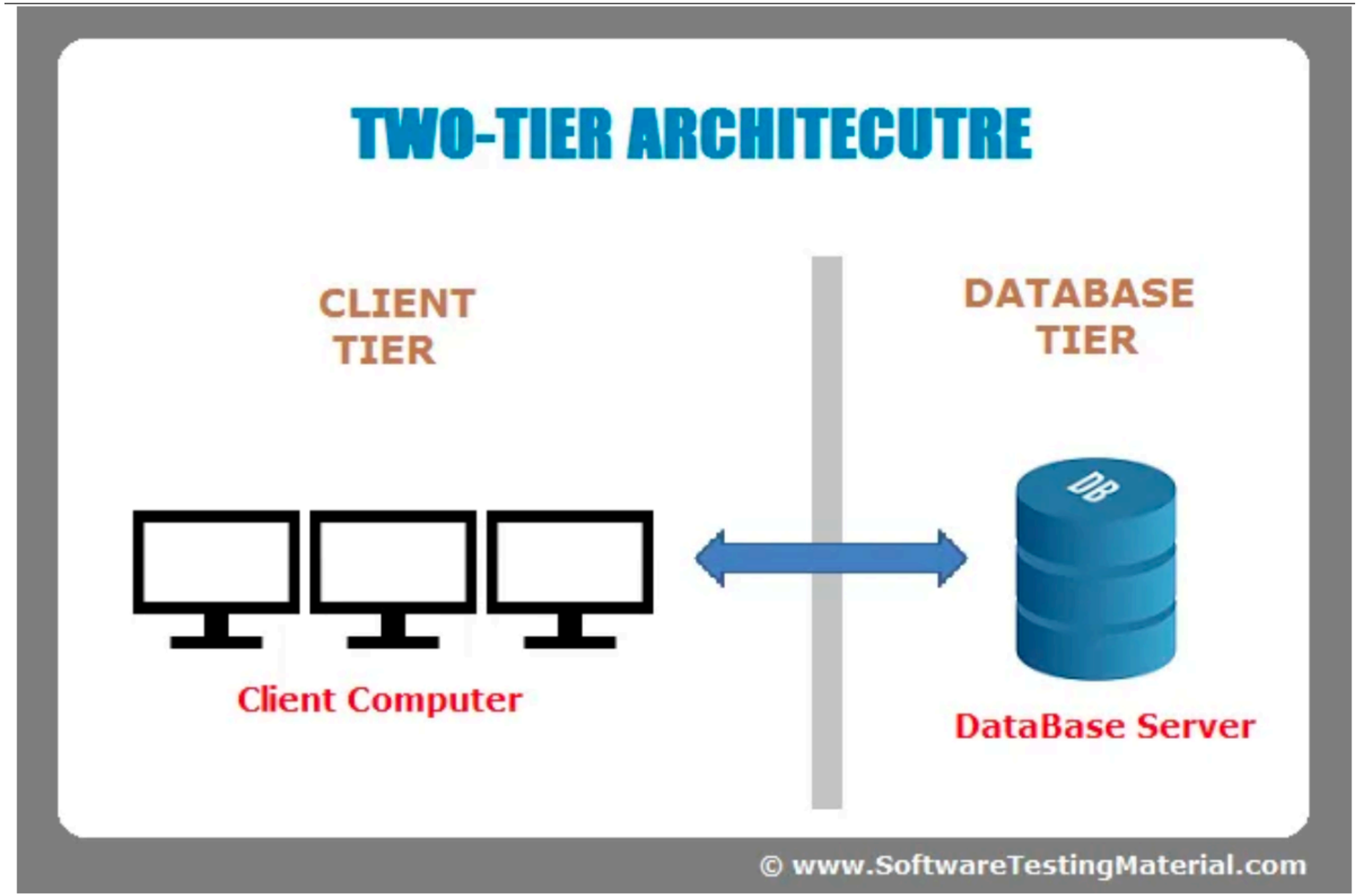
- Una máquina o un programa que tiene la capacidad y una forma de enviar solicitudes (requests) a través de internet.
- No necesariamente es un browser.
- Un computador puede tener varios clientes.

POSTMAN

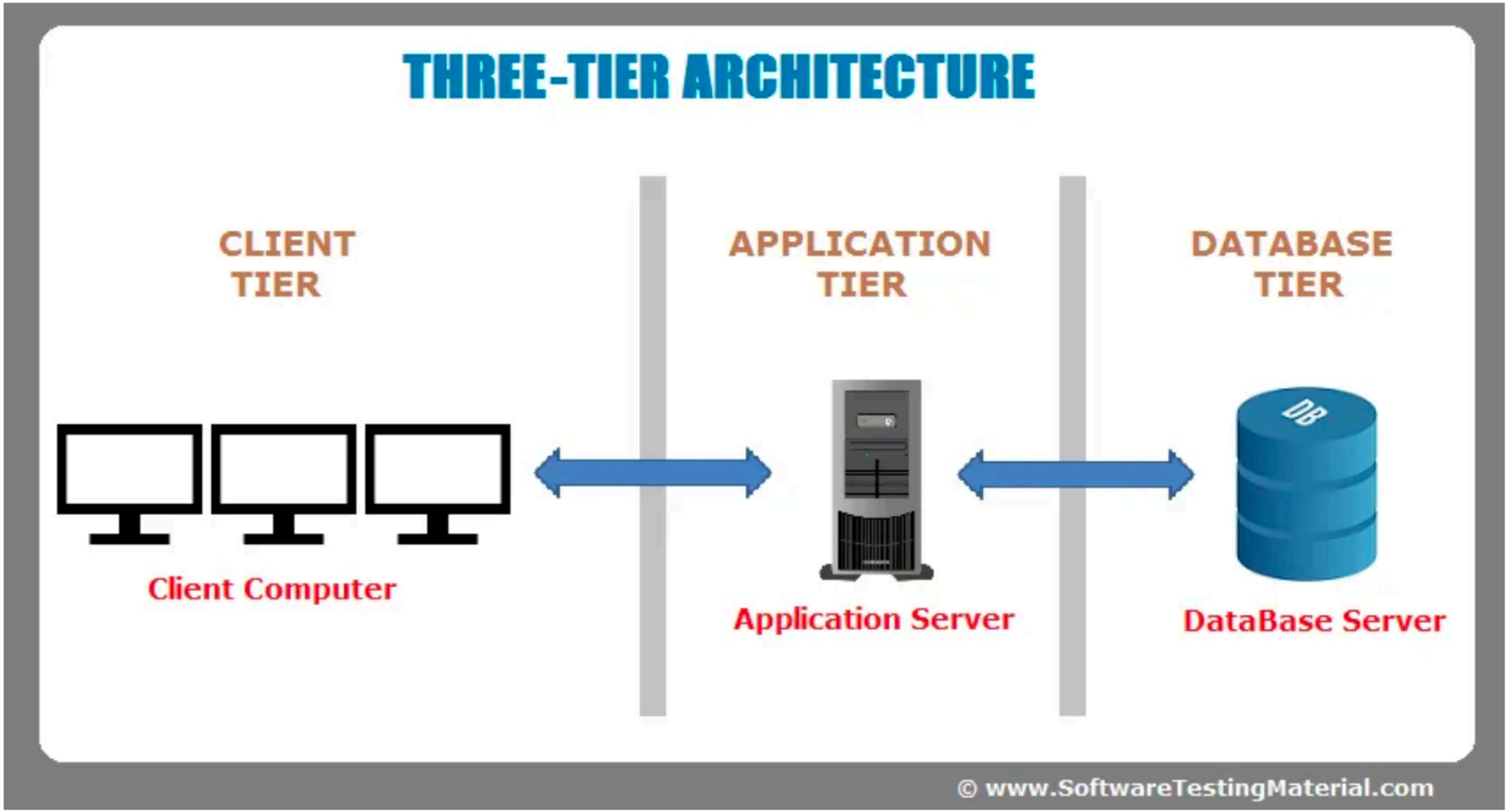
Servidor

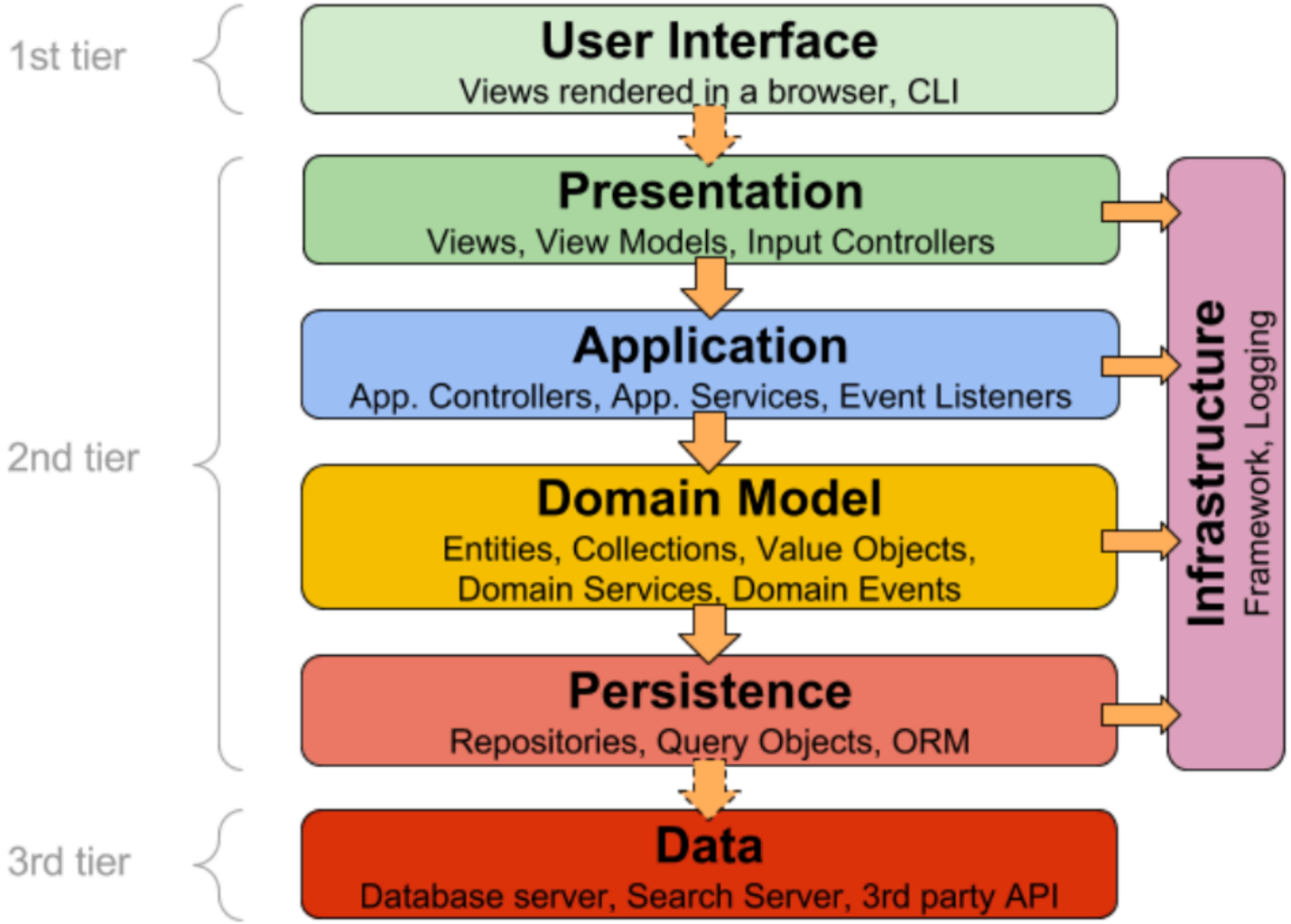
- No es necesariamente un dispositivo (computadora).
- Las computadoras de alto rendimiento son llamadas servidores porque ejecutan programas que dan servicios.
- Un servidor puede atender múltiples clientes al mismo tiempo.


ONE-TIER ARCHITECTURE
Presentation layer
Application layer
CLIENT TIER
DB
DataBase layer
© www.SoftwareTestingMaterial.com

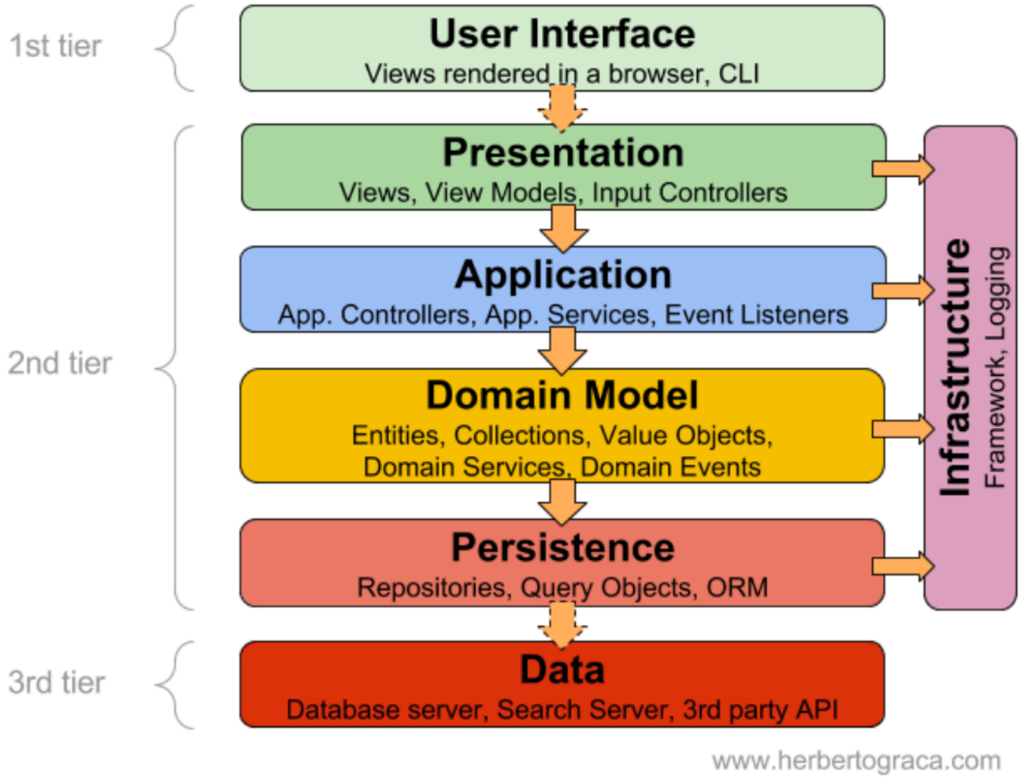

TWO-TIER ARCHITECUTRE
CLIENT TIER
DATABASE TIER
DB
Client Computer
DataBase Server
© www.SoftwareTestingMaterial.com

Arquitectura de Capas

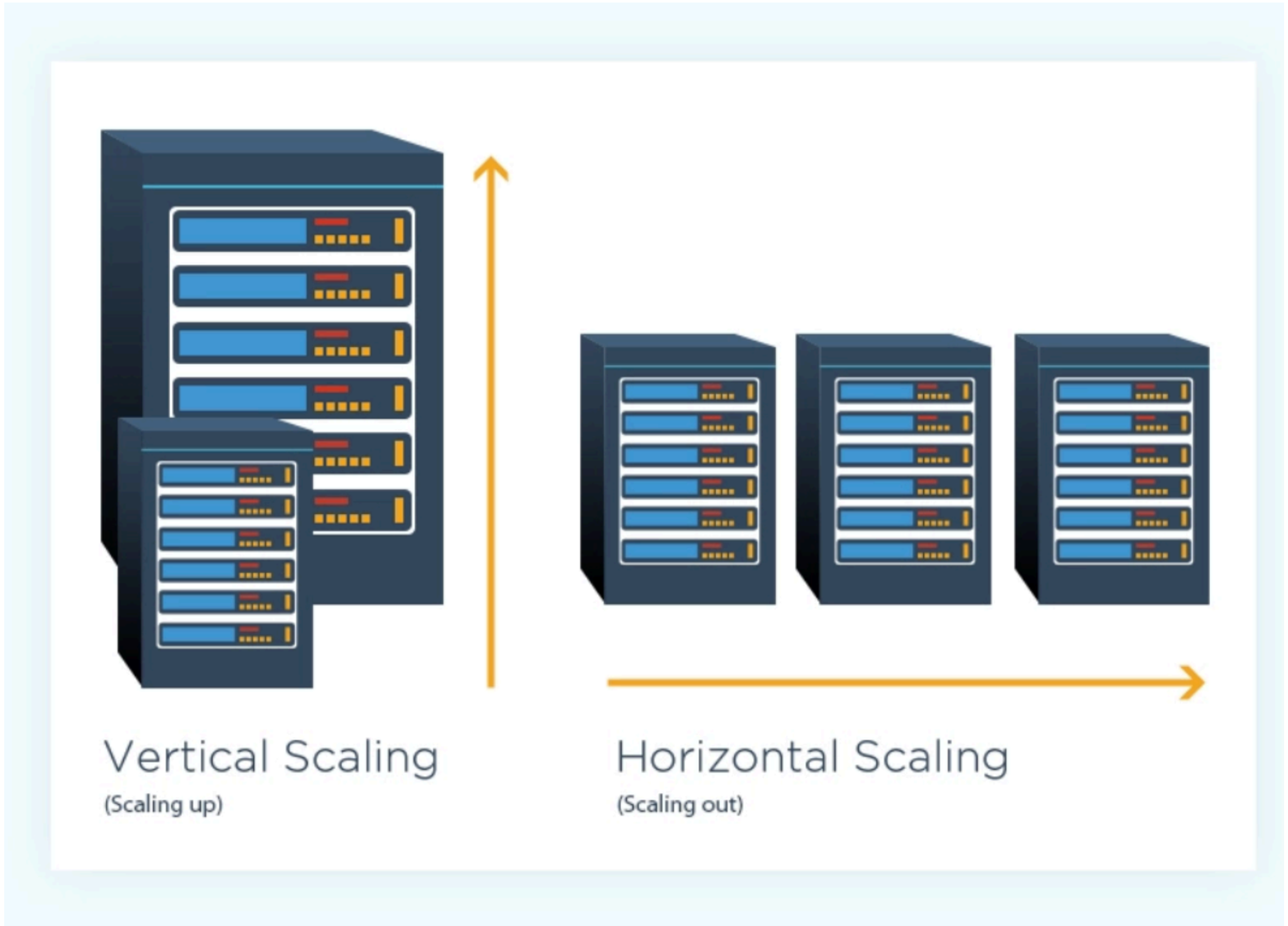

THREE-TIER ARCHITECTURE
CLIENT TIER
APPLICATION TIER
DATABASE TIER
DB
Client Computer
Application Server
DataBase Server
© www.SoftwareTestingMaterial.com


1st tier
User Interface
Views rendered in a browser, CLI
2nd tier
Presentation
Views, View Models, Input Controllers
Application
App. Controllers, App. Services, Event Listeners
Domain Model
Entities, Collections, Value Objects,
Domain Services, Domain Events
Persistence
Repositories, Query Objects, ORM
Infrastructure
Framework, Logging
3rd tier
Data
Database server, Search Server, 3rd party API
www.herbertograca.com

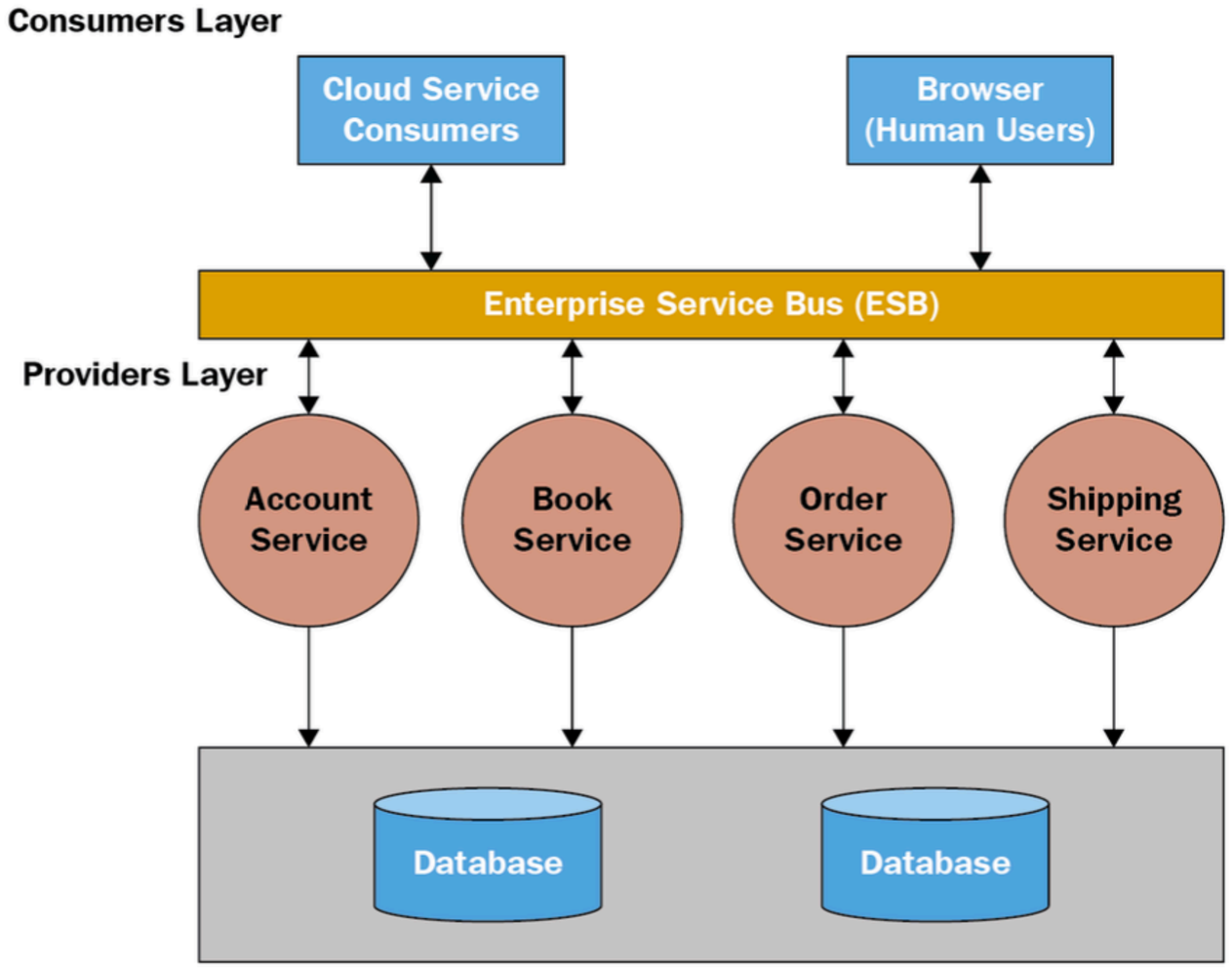
¿Podemos escalar?



- 100 usuarios: funcionando ✅
- 1000 usuarios: funcionando ✅
- 1000000 usuarios: no necesariamente.

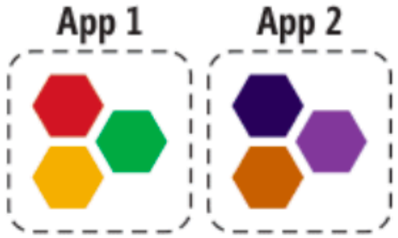

Vertical Scaling
(Scaling up)
Horizontal Scaling
(Scaling out)

Arquitectura Orientada a Servicios


Consumers Layer
Cloud Service Consumers
Browser (Human Users)
Enterprise Service Bus (ESB)
Providers Layer
Account Service
Book Service
Order Service
Shipping Service
Database
Database

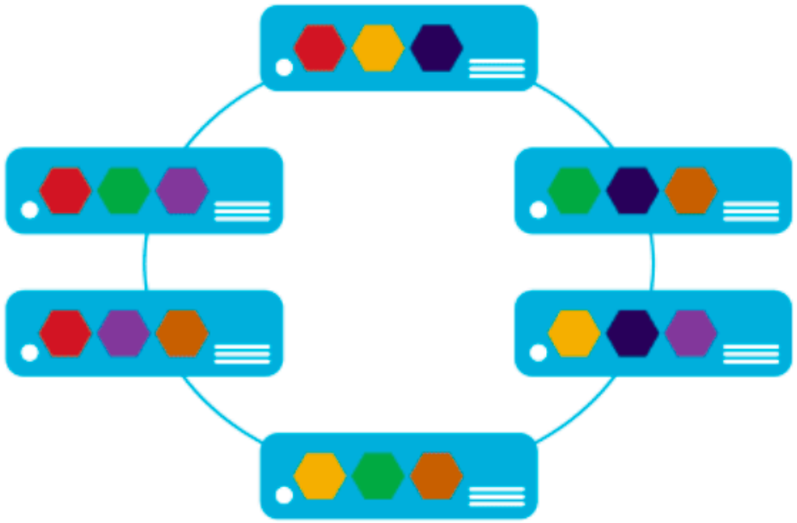
Arquitectura Orientada a Micro-Servicios

Microservices Approach



A microservice approach segregates functionality into small autonomous services.

And scales out by **deploying independently** and replicating these services across servers/VMs/containers.

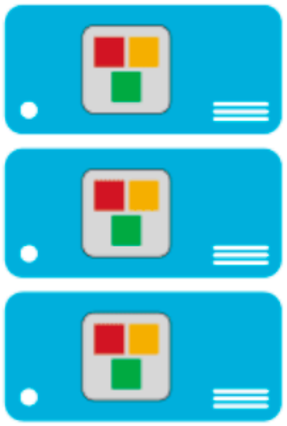


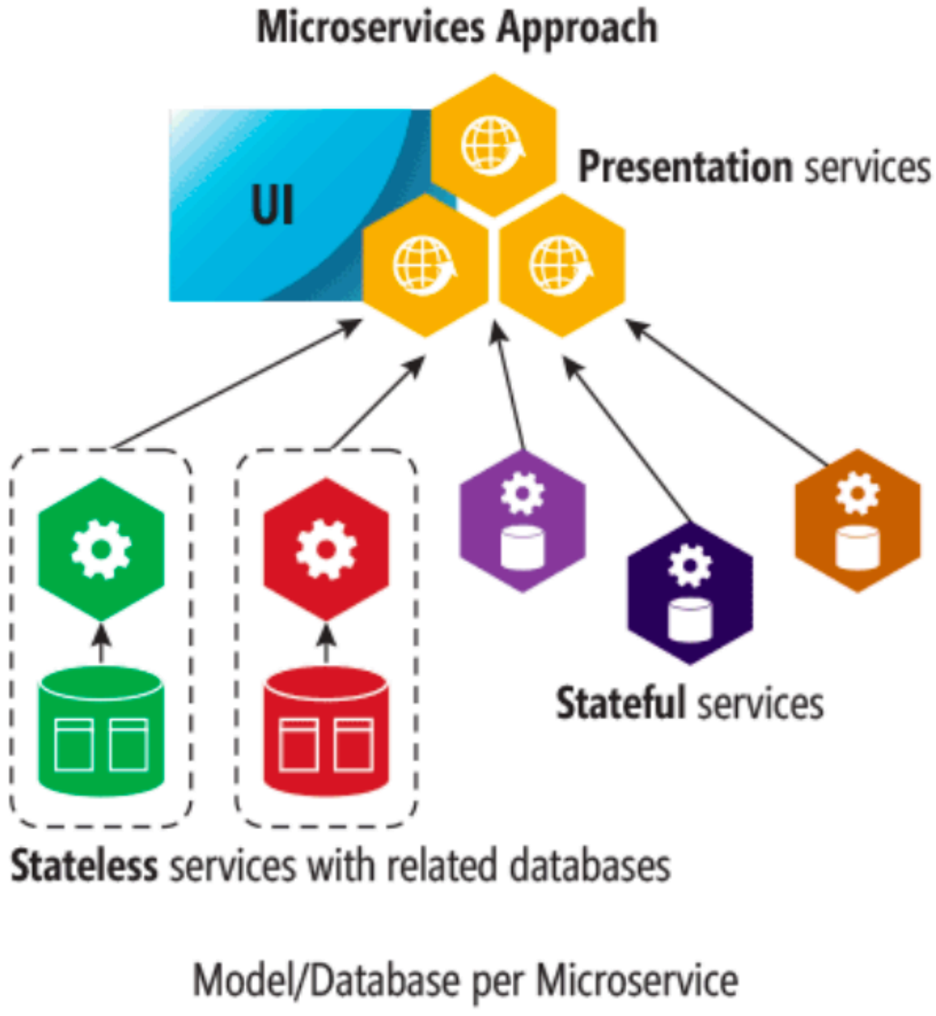
VS.

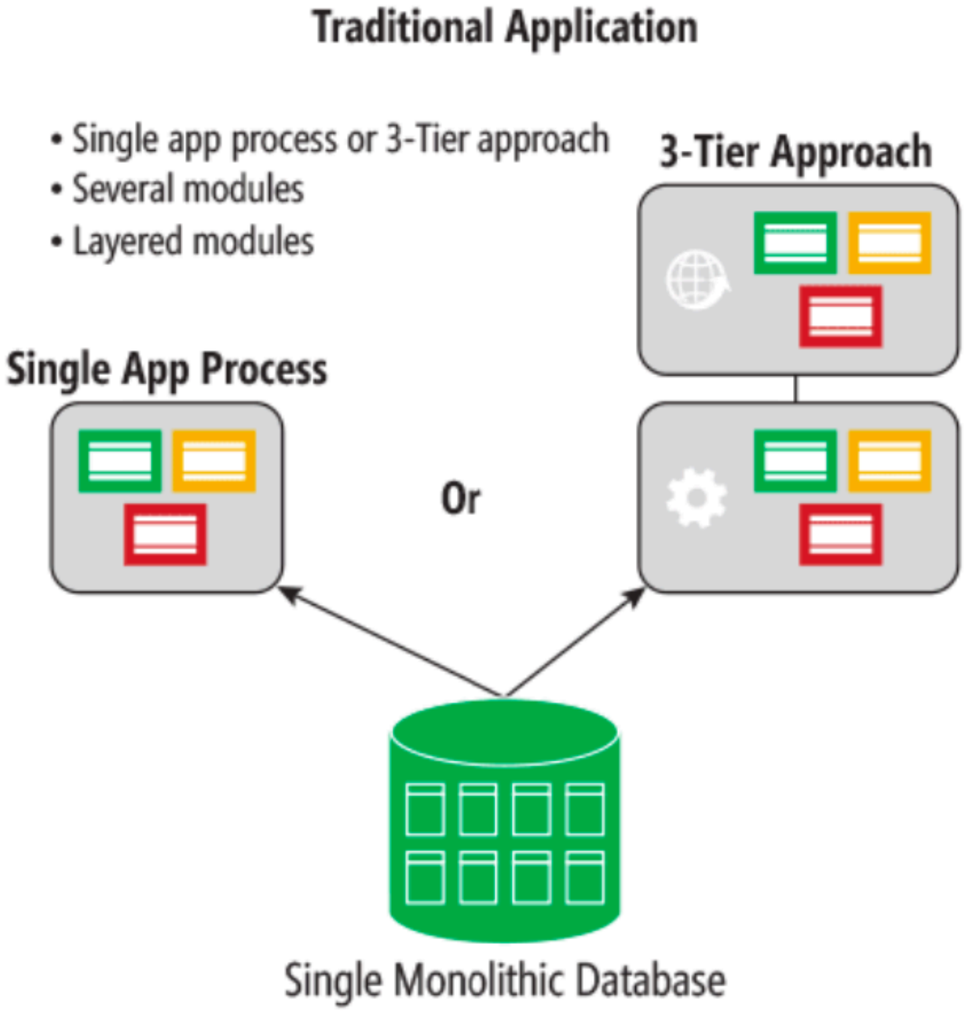
Traditional Approach

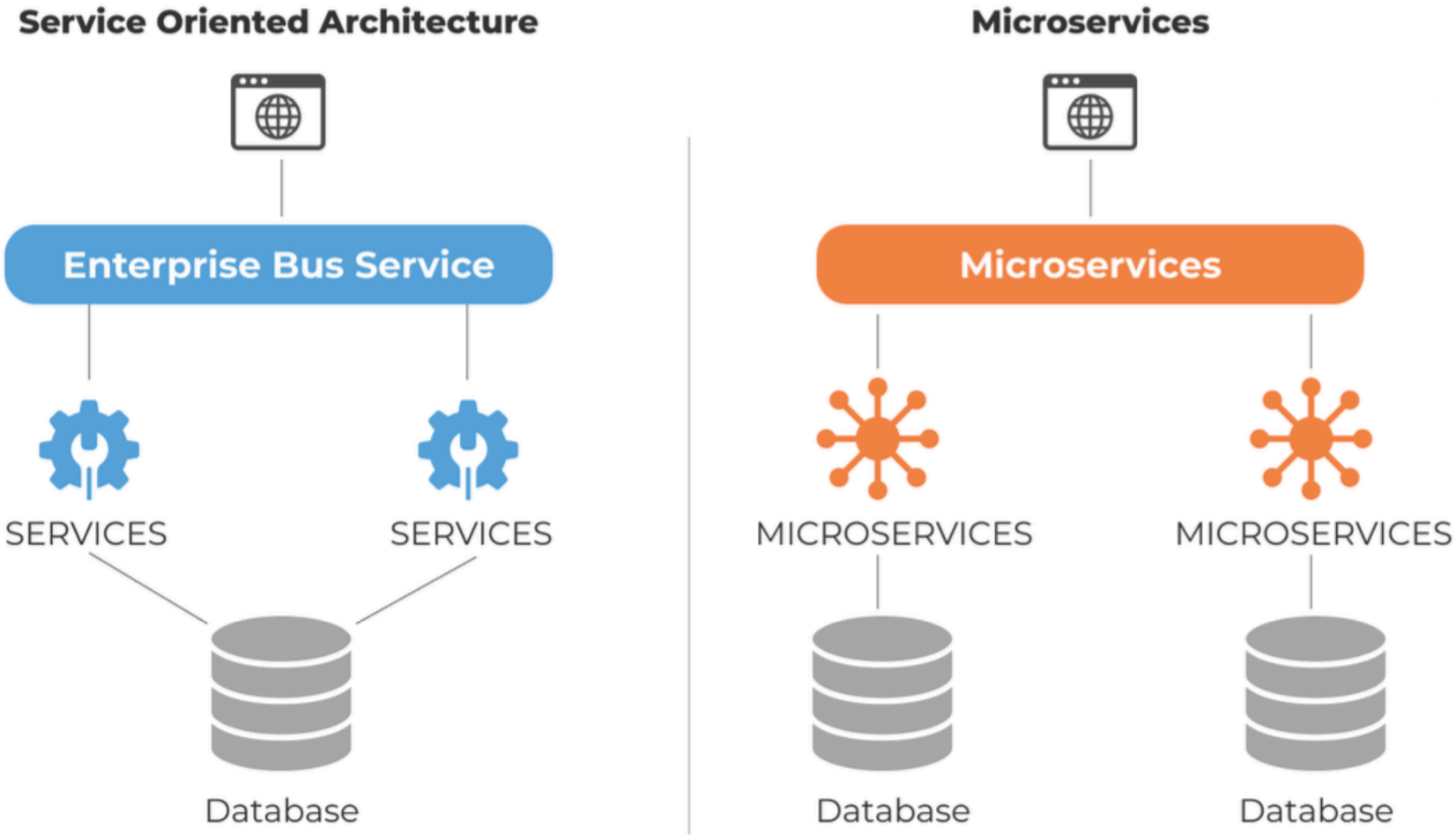
A traditional application (Web app or large service) usually has most of its functionality within a single process (usually internally layered, though).

And scales by cloning the whole app on multiple servers/VMs/containers.



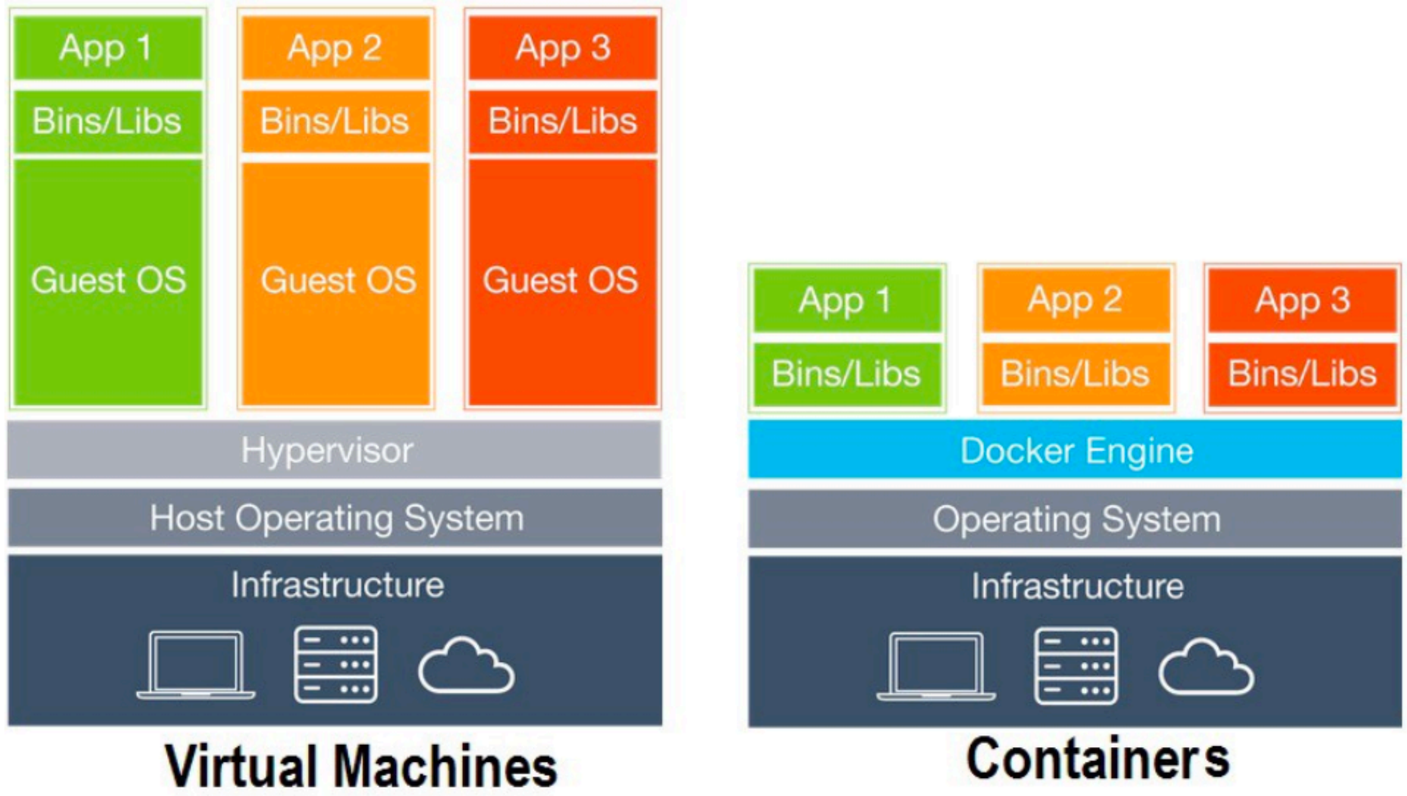

Microservices Approach
UI
Presentation services
Stateful services
Stateless services with related databases
Model/Database per Microservice

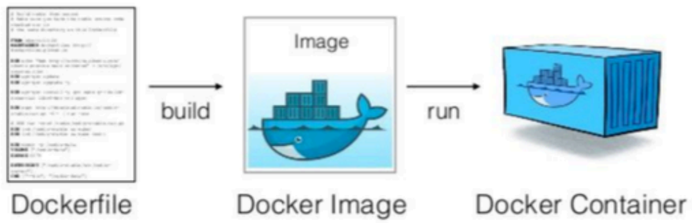

Traditional Application
• Single app process or 3-Tier approach
• Several modules
• Layered modules
3-Tier Approach
Single App Process
Or
Single Monolithic Database


Service Oriented Architecture
Enterprise Bus Service
SERVICES
SERVICES
Database
Microservices
Microservices
MICROSERVICES
MICROSERVICES
Database
Database

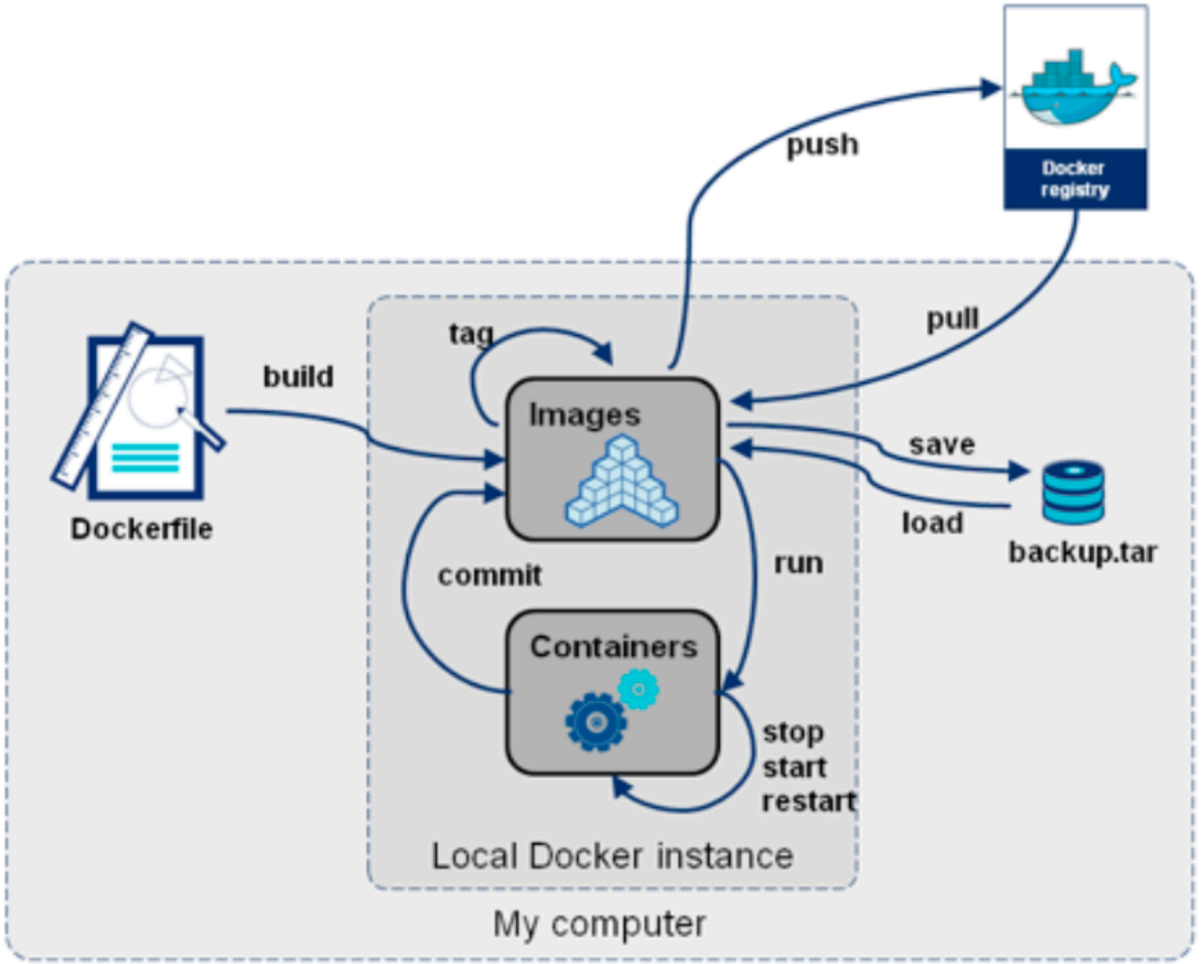
¿Cómo implementar una arquitectura con micro-servicios?

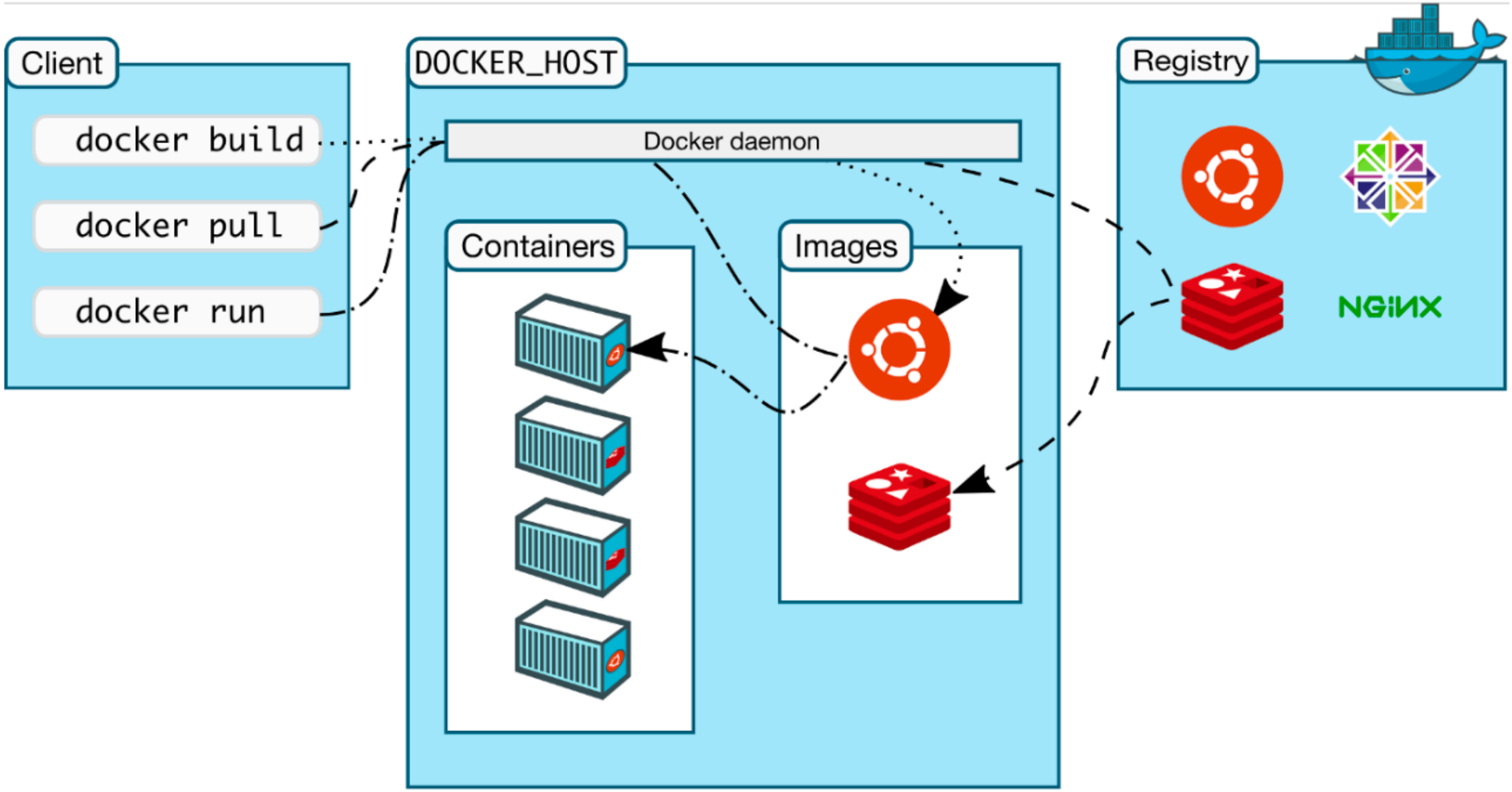
Containers (Docker)

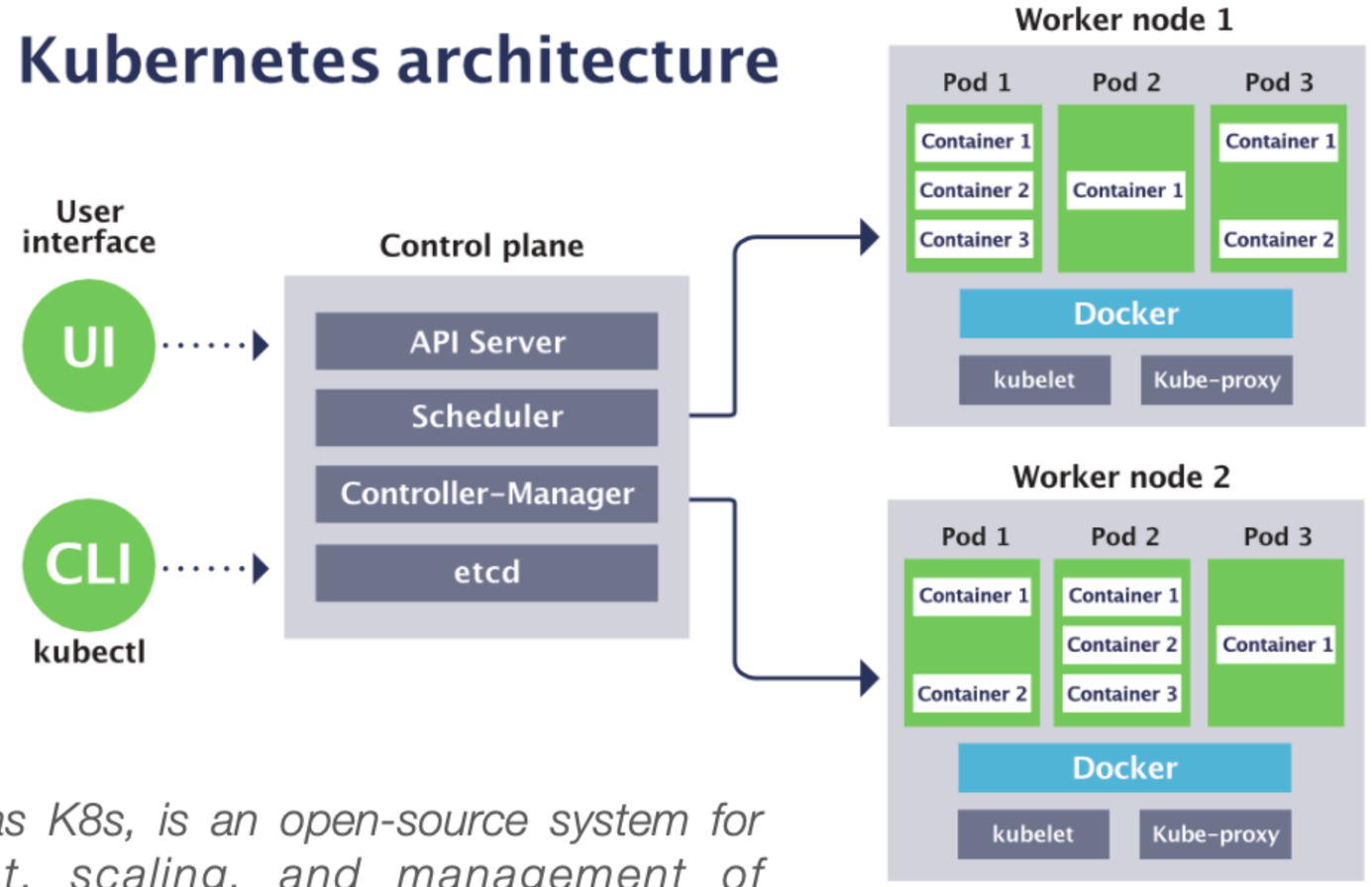




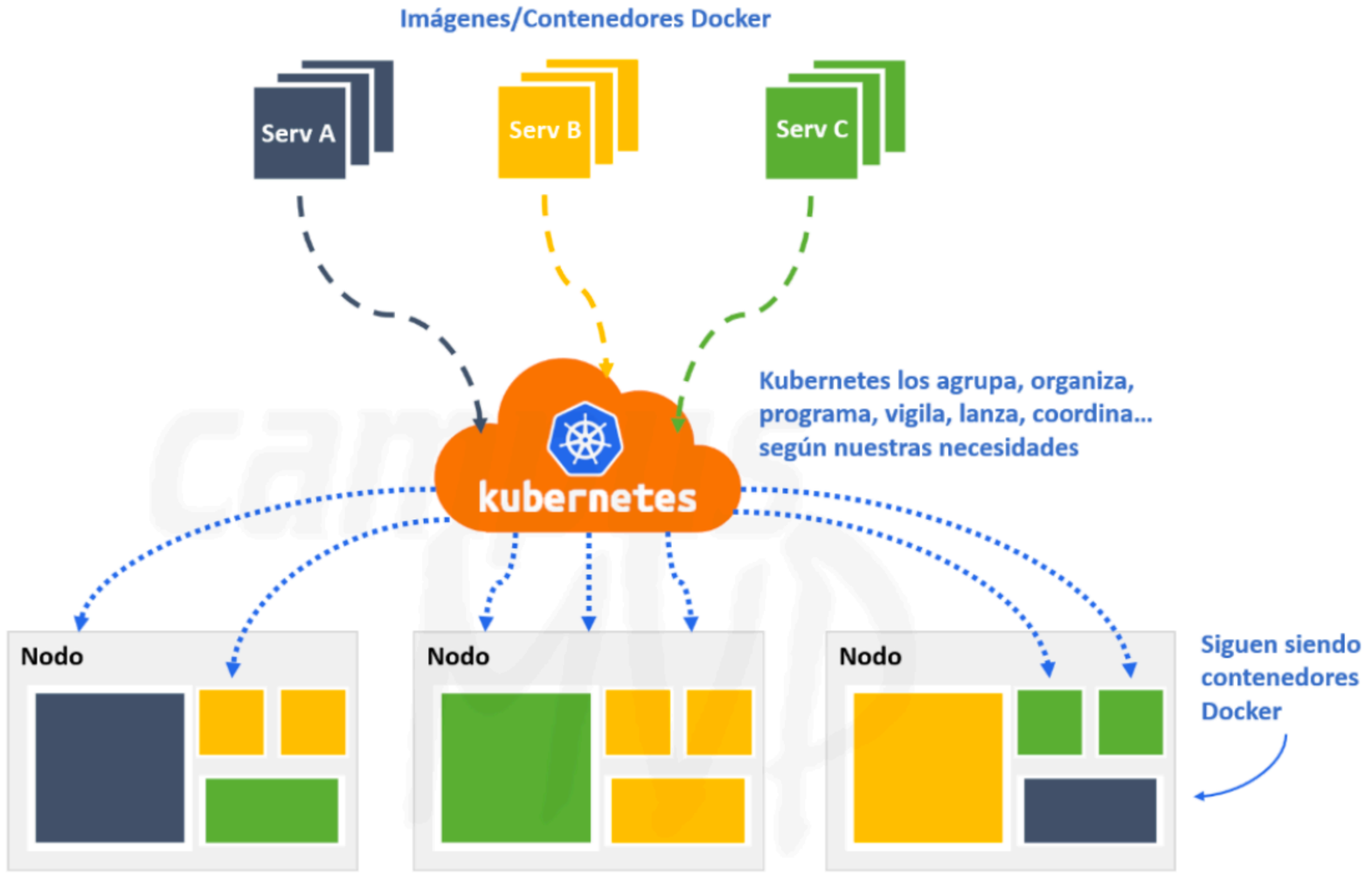

```
# Especificamos la imágen de la que se heredará

FROM python:3.6-alpine

# Definimos algunas variables de ambiente

ENV LIBRARY_PATH=/lib:/usr/lib

ENV PYTHONUNBUFFERED 1

# Ejecutamos algunos comandos para aprovisionar la imágen

RUN mkdir /code

WORKDIR /code

ADD requirements.txt /code/

RUN pip install -r requirements.txt

# Copiamos el código

ADD . /code/

# Exponemos el puerto por donde se interactuará con la aplicación

EXPOSE 8000

# Definimos el comando de entrada a la aplicación

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```


Docker registry
push
pull
tag
build
Images
save
load
backup.tar
Dockerfile
commit
run
Containers
stop
start
restart
Local Docker instance
My computer

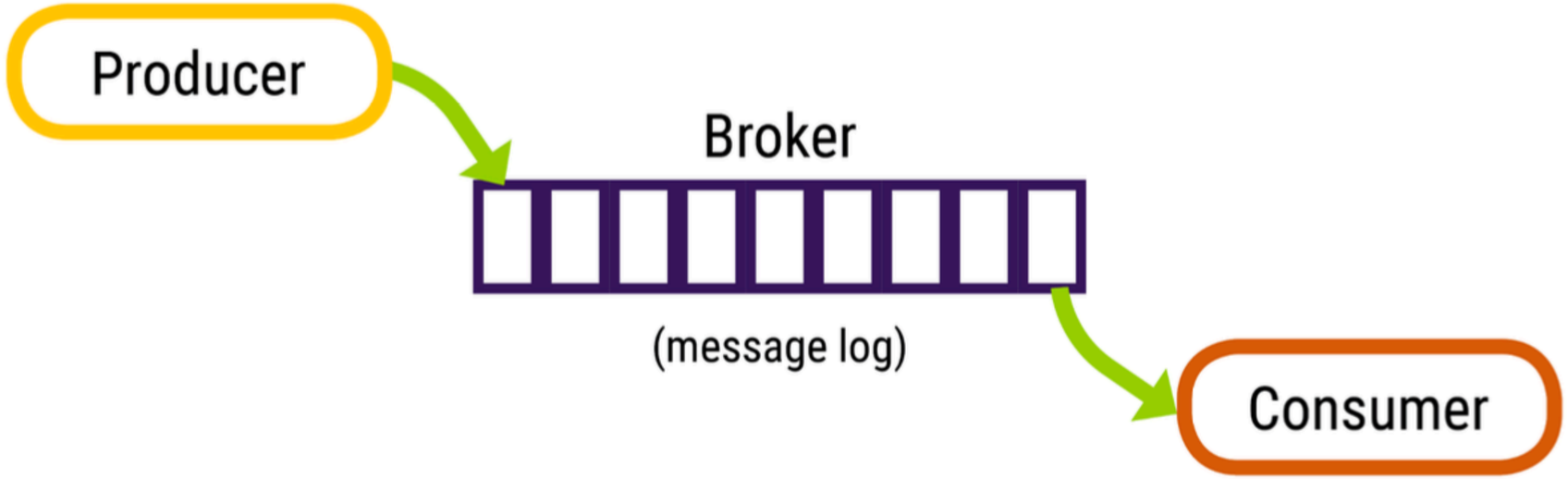

Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

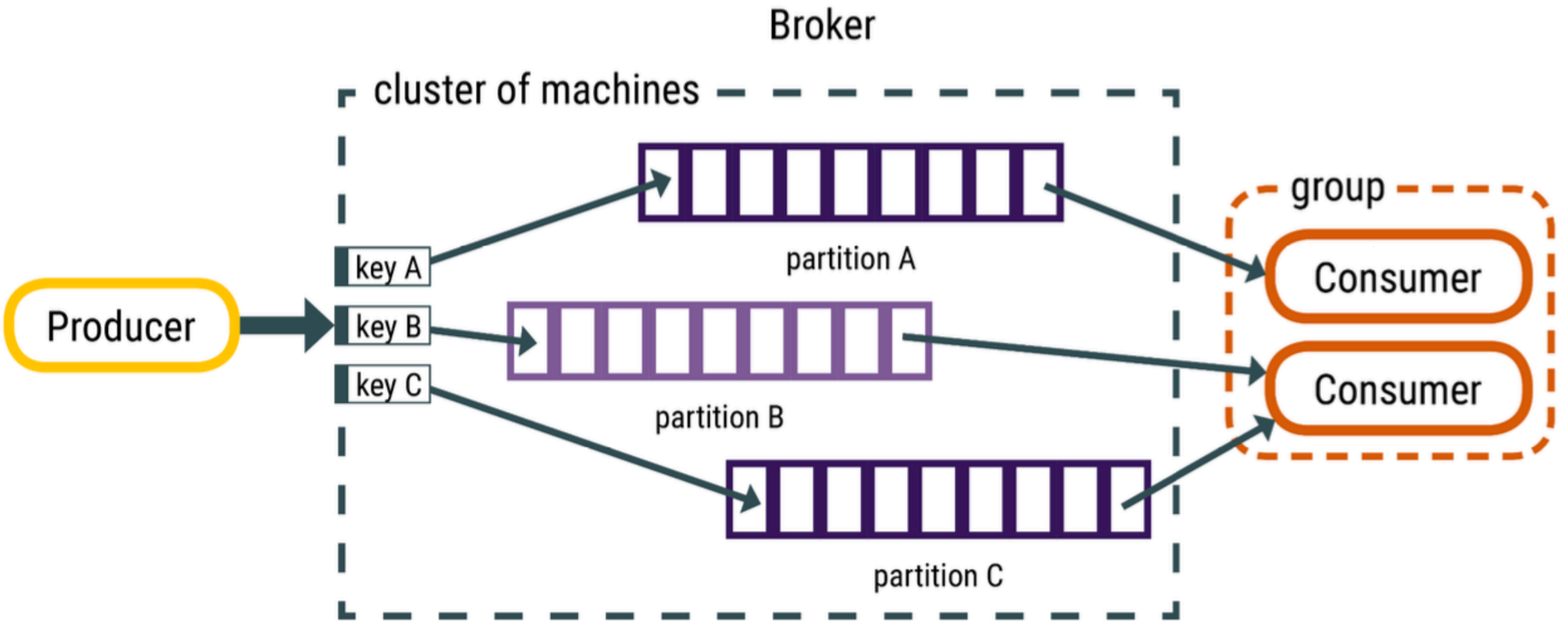


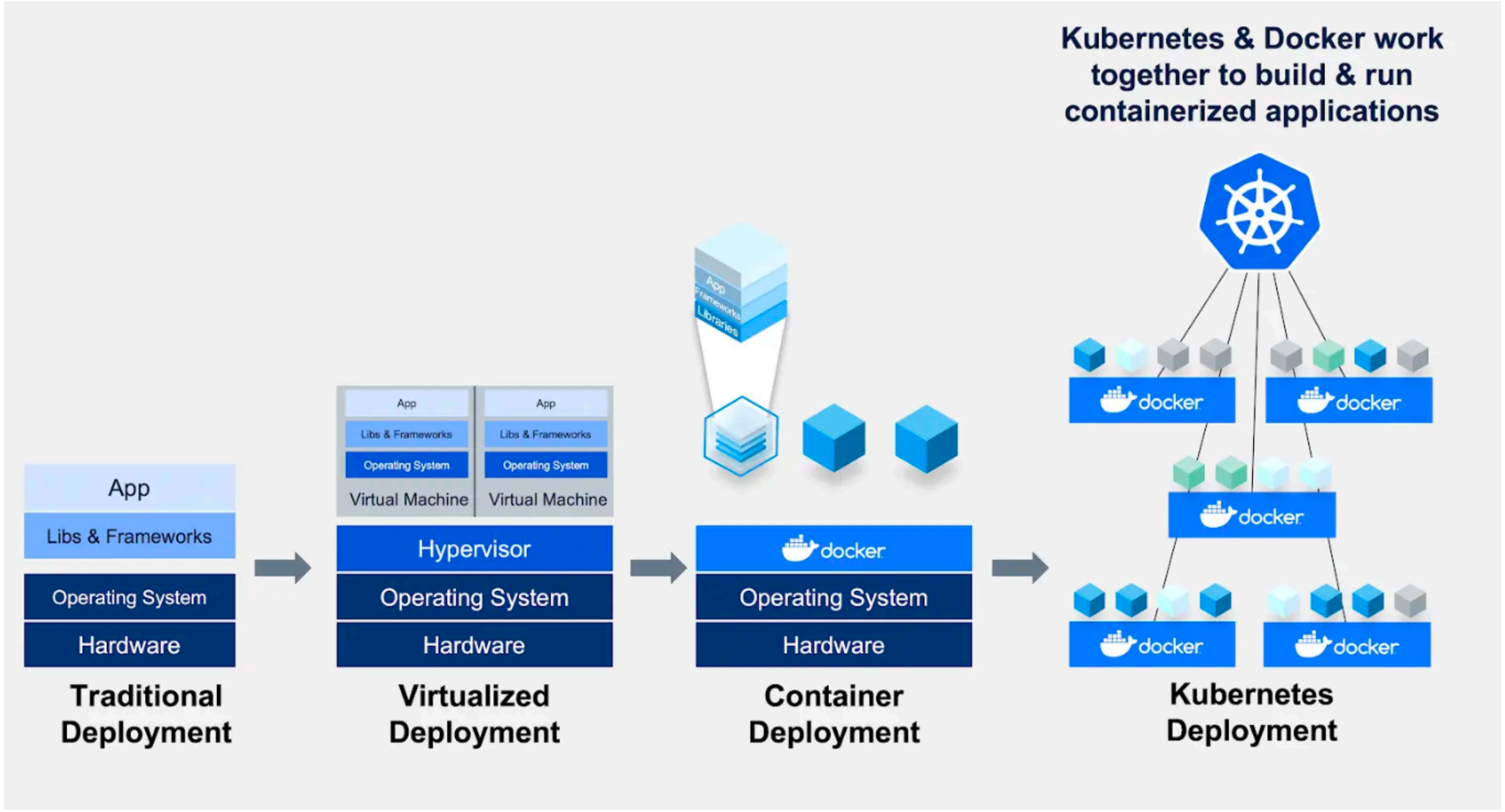
Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of ***containerized*** *applications.*


Imágenes/Contenedores Docker
Serv A
Serv B
Serv C
kubernetes
Kubernetes los agrupa, organiza, programa, vigila, lanza, coordina... según nuestras necesidades
Nodo
Nodo
Nodo
Siguen siendo contenedores Docker


ORACLE
salesforce
workday.
kafka
SAP


Producer
Broker
(message log)
Consumer


Broker
cluster of machines
key A
key B
key C
Producer
partition A
partition B
partition C
group
Consumer
Consumer


Kubernetes & Docker work together to build & run containerized applications
App
Libs & Frameworks
Operating System
Hardware
Traditional Deployment
App
Libs & Frameworks
Operating System
Virtual Machine
App
Libs & Frameworks
Operating System
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
Frameworks
Libraries
docker
Operating System
Hardware
Container Deployment
docker
docker
docker
docker
docker
Kubernetes Deployment

Bibliografía

- Apuntes Sistemas Operativos y Redes 2023-1
- Documentación Docker
- Documentación Kubernetes

¿Consultas?

Ingeniería de Software

21 - Arquitectura de Software

IIC2143-3

Josefa España

jpespana@uc.cl