HELLO!

JOSE LUIS GOMEZ ORTEGA

- PhD Computer Science
- Msc Engineering with Management
- Diplomatura Ingenieria Industrial

Currently:
Data Science Lead instructor @ TheBridge/EDEM

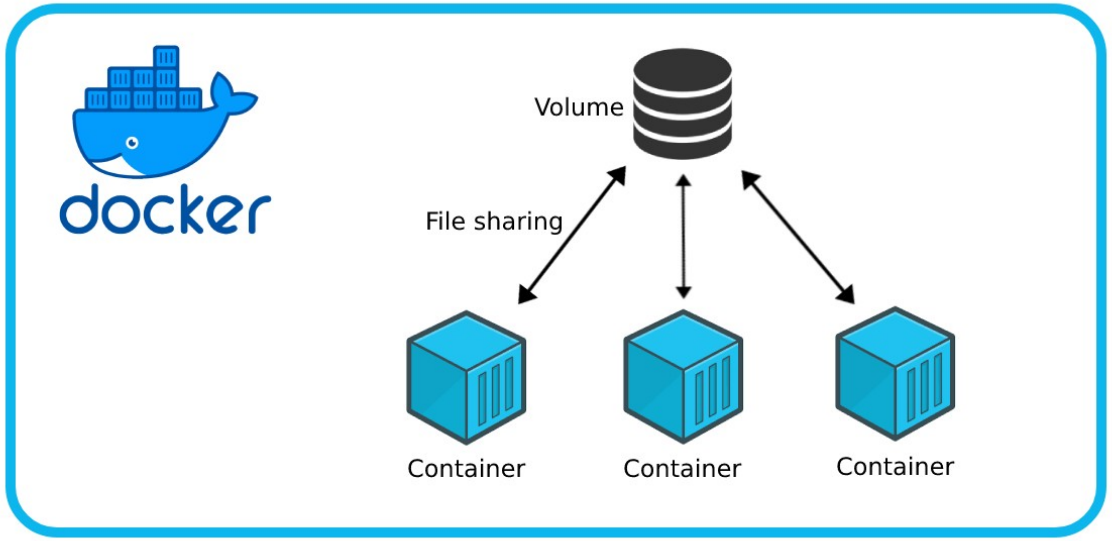
https://www.linkedin.com/in/jlgortega/

jose.lgo.datascience@gmail.com

QUICK ROUND?

Setting the Stage for Containerization

Welcome to the Docker Introduction Session



AGENDA

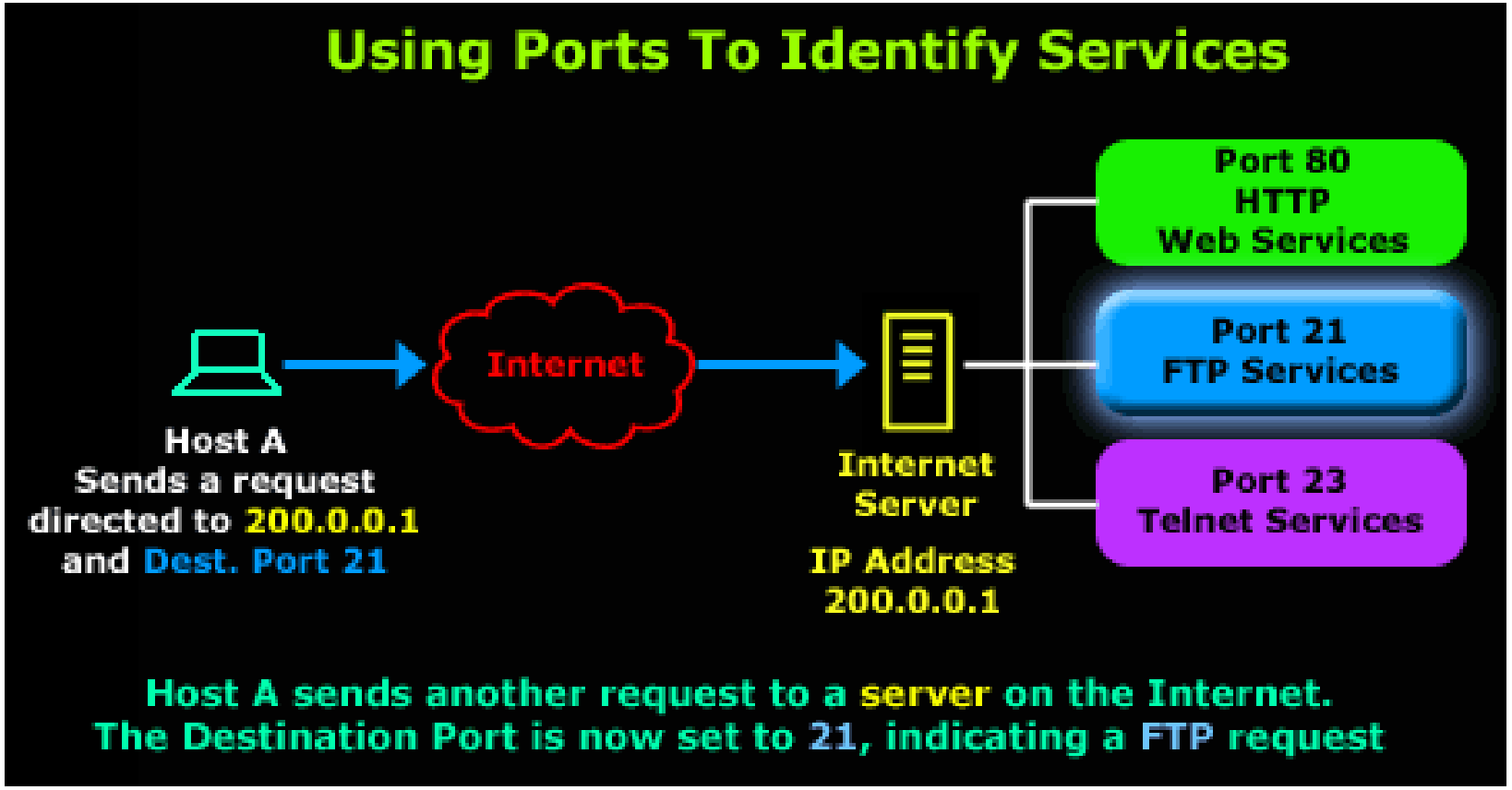
TOPICS:

- Services and Ports in Operating Systems
- DevOps Principles
- Modern Application Architectures
- Virtual Machines and Hypervisors
- Virtualization in the Cloud
- Introduction to Docker

Understanding Services in Operating Systems:

Definition: Programs running in the background, providing functionalities to other software or users.

Examples: Web servers, database servers, email servers.

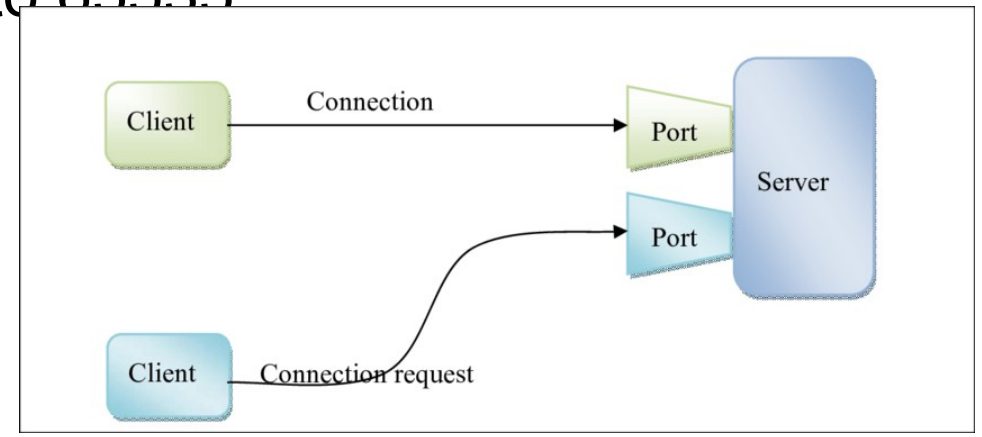


PORTS

The Role of Ports in Networking

Definition: Logical endpoints for network communication, identifying specific processes or services.

Port Numbers: Range from 0 to 65535



Port	Service name	Transport protocol
20, 21	File Transfer Protocol (FTP)	TCP
22	Secure Shell (SSH)	TCP and UDP
23	Telnet	TCP
25	Simple Mail Transfer Protocol (SMTP)	TCP
50, 51	IPSec	
53	Domain Name System (DNS)	TCP and UDP
67, 68	Dynamic Host Configuration Protocol (DHCP)	UDP
69	Trivial File Transfer Protocol (TFTP)	UDP
80	HyperText Transfer Protocol (HTTP)	TCP
110	Post Office Protocol (POP3)	TCP

SERVICES AND PORTS

Service Binding: Services "listen" on specific ports for incoming requests.

Client Communication: Clients connect using the service's port number.

```
C:\Users\Ort\Desktop\folder>python -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::ffff:127.0.0.1 - - [07/Oct/2024 19:44:25] "GET / HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [07/Oct/2024 19:44:28] "GET /nombre.ipynb HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [07/Oct/2024 19:44:46] "GET /.ipynb_checkpoints/ HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [07/Oct/2024 19:44:49] "GET /.git/ HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [07/Oct/2024 19:44:56] "GET /ubc7djom1jeqih3ttuxl.png HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [07/Oct/2024 19:45:41] "GET / HTTP/1.1" 200 -
```

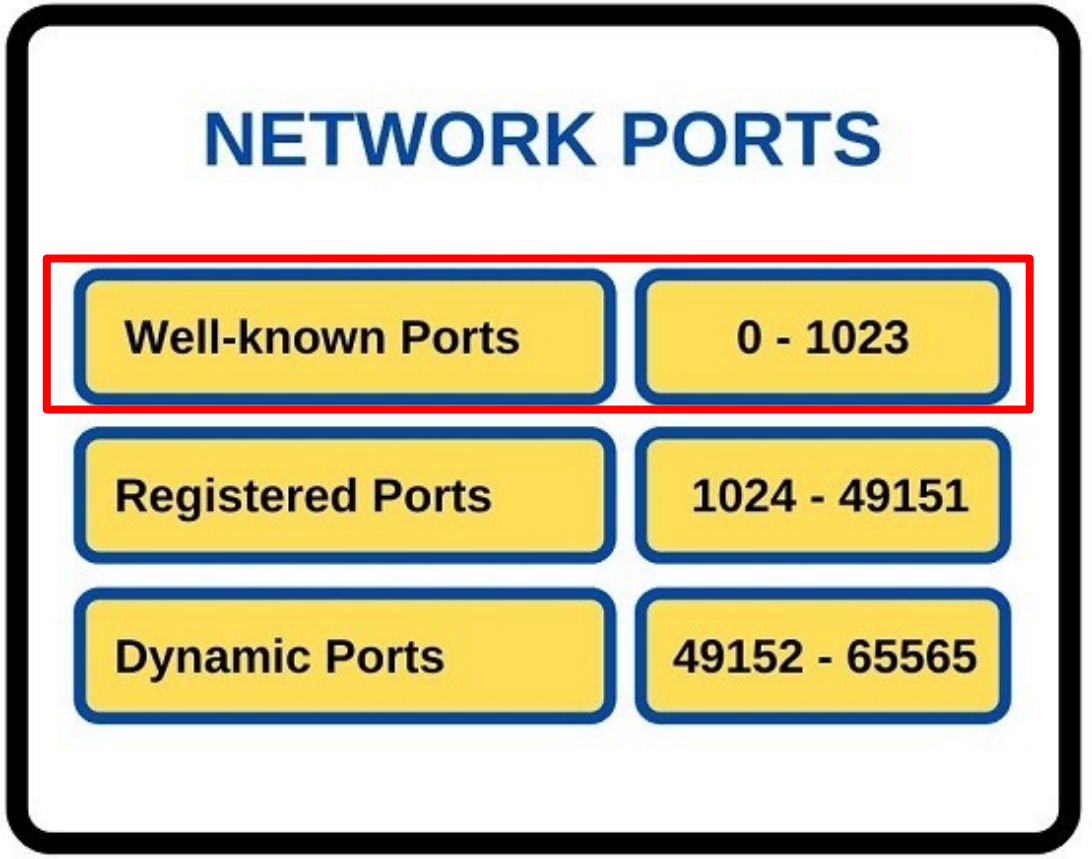
PORT RANGE I

Puertos Bien Conocidos (Well-Known Ports): 0 - 1023

- Estos puertos están reservados para servicios y aplicaciones que son ampliamente utilizados y estandarizados a nivel mundial.
- Son asignados y administrados por la IANA (Internet Assigned Numbers Authority).

- Requieren privilegios administrativos para ser utilizados en muchos sistemas operativos.
- Están asociados a servicios fundamentales de Internet

- Puerto 53: DNS (Domain Name System)
- Puerto 80: HTTP (HyperText Transfer Protocol)
- Puerto 110: POP3 (Post Office Protocol v3)
- Puerto 143: IMAP (Internet Message Access Protocol)
- Puerto 443: HTTPS (HTTP Secure)



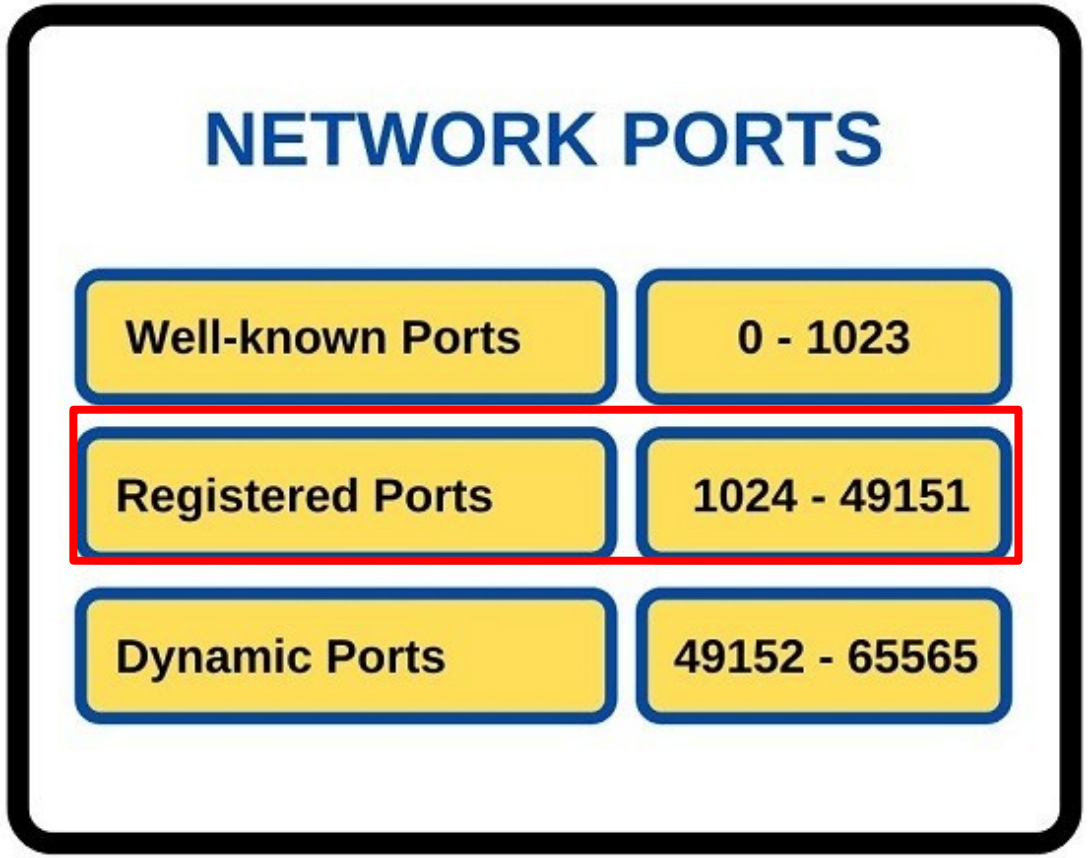
PORT RANGE II

Puertos Registrados (Registered Ports): 1024 - 49151

- Estos puertos son asignados por la IANA a aplicaciones y servicios específicos para evitar conflictos.
- No son tan estandarizados como los puertos bien conocidos, pero permiten identificar aplicaciones comunes.

- Generalmente no requieren privilegios administrativos para su uso.
- Son utilizados por aplicaciones de usuario o servicios que necesitan un puerto fijo.

- Puerto 1433: Microsoft SQL Server
- Puerto 1521: Oracle Database
- Puerto 3306: MySQL
- Puerto 5432: PostgreSQL
- Puerto 6379: Redis
- Puerto 8080: Servidores web alternativos o en desarrollo
- Puerto 27017: MongoDB



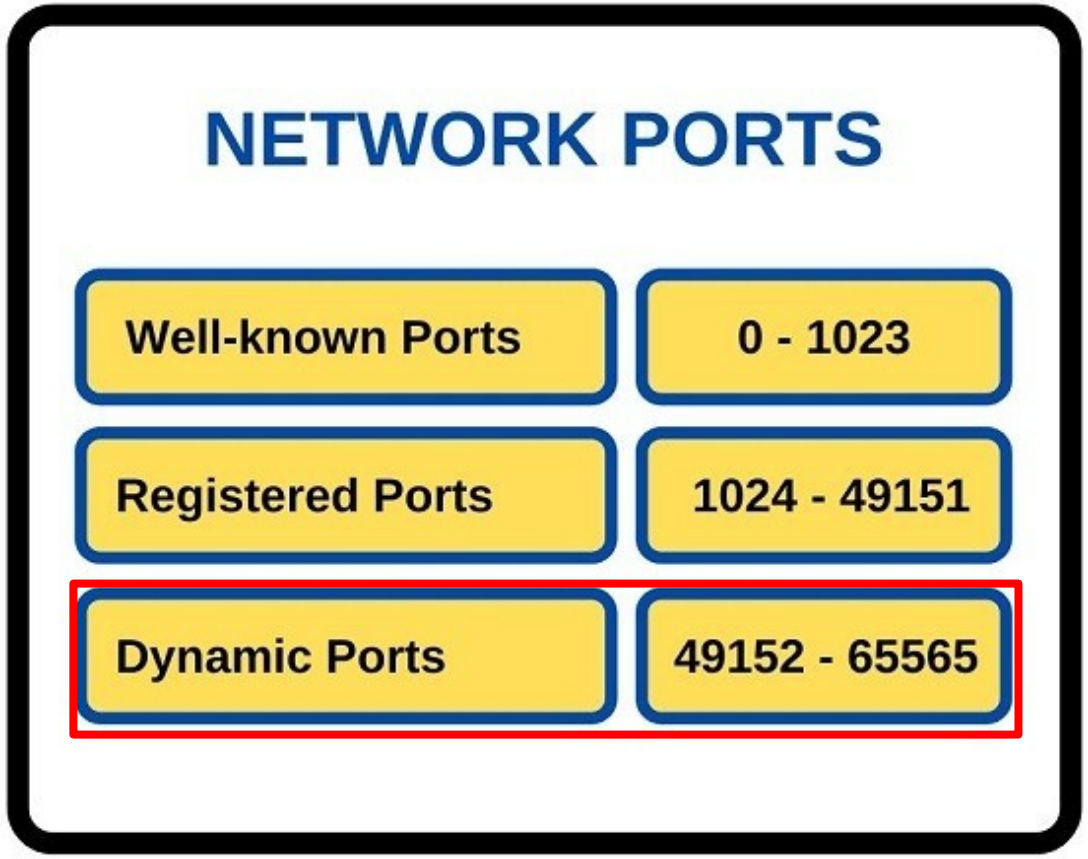
PORT RANGE III

Puertos Dinámicos o Privados (Dynamic or Private Ports): 49152 - 65535

- Estos puertos no están asignados a ningún servicio o aplicación específica.
- Son utilizados temporalmente por aplicaciones para establecer conexiones efímeras.

- Asignados dinámicamente por el sistema operativo cuando una aplicación necesita comunicarse en red.
- No requieren registro ni administración por parte de la IANA.

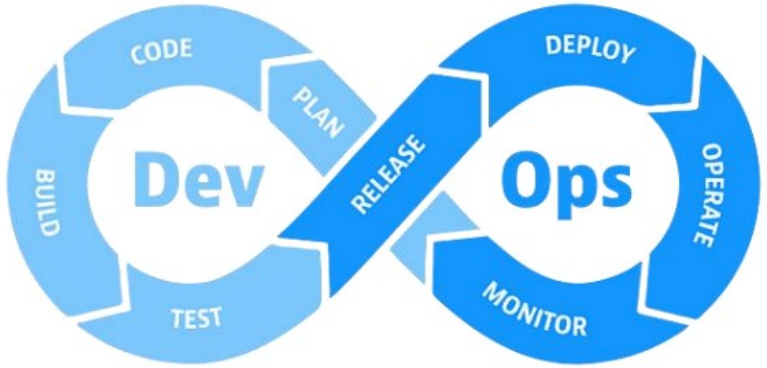
- Puertos Efímeros: Cuando un cliente inicia una conexión a un servidor (por ejemplo, al navegar por Internet), el sistema operativo asigna un puerto dinámico para esa sesión.
- Aplicaciones Peer-to-Peer: Pueden utilizar puertos dinámicos para establecer conexiones directas entre usuarios.



EXAMPLE – PYTHON SERVER

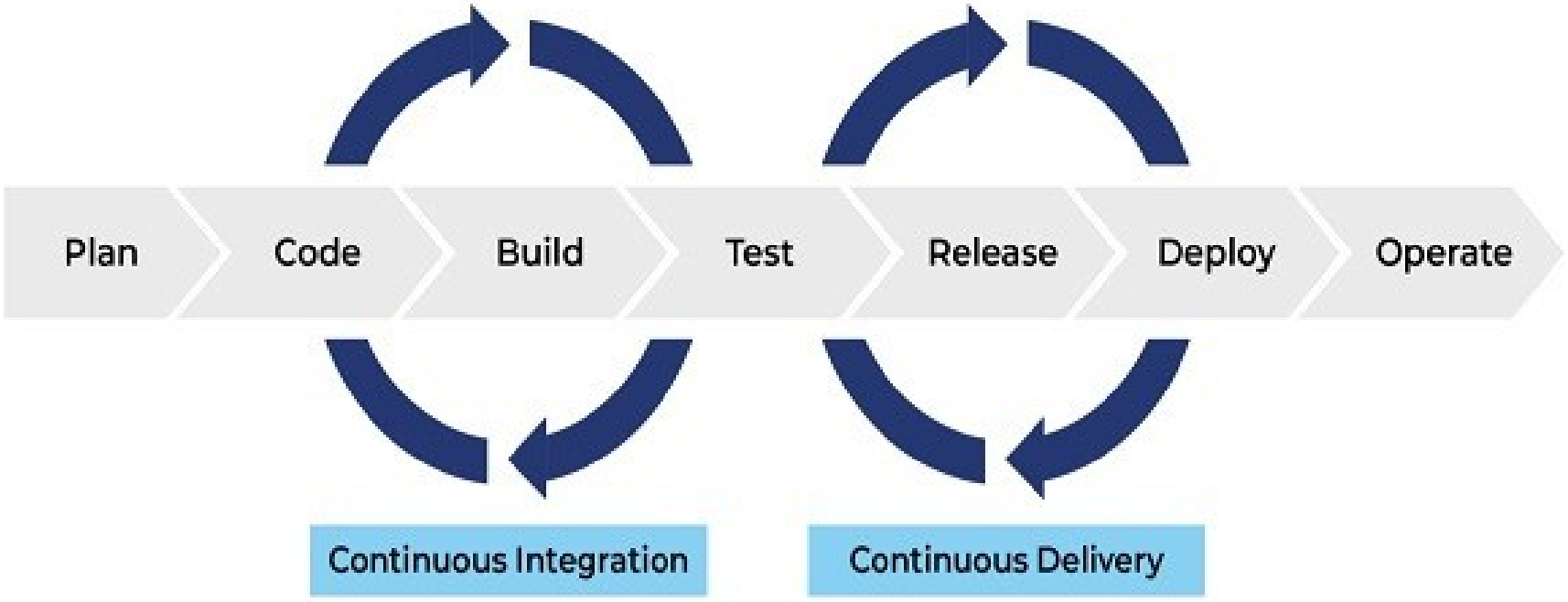
DEVOPS

- **DevOps** ("Development" and "Operations") improve communication, collaboration, integration and automation between Developers and Other IT Professionals
- DevOps focuses teams to have the **same goal**: Release Quality Software Fast

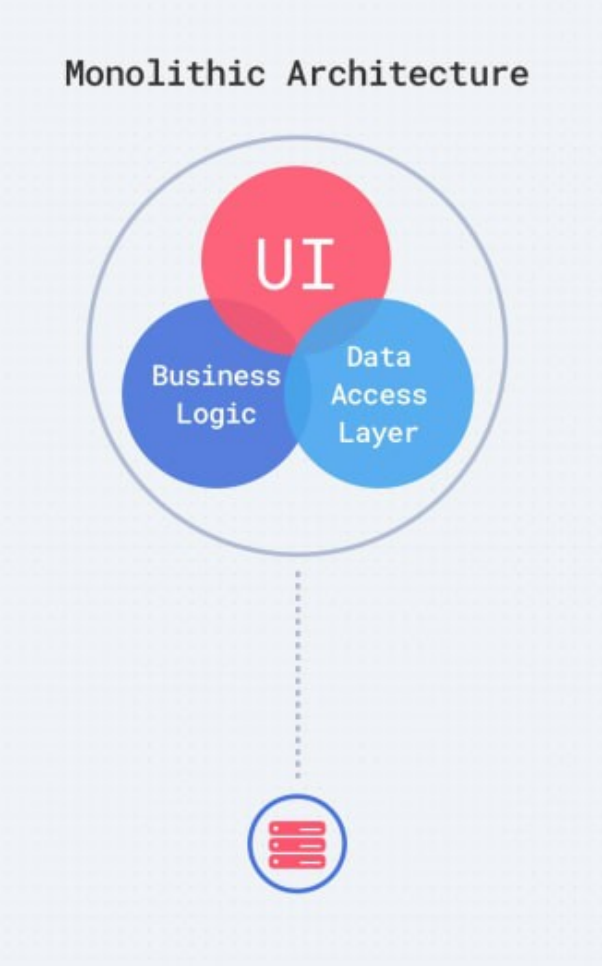


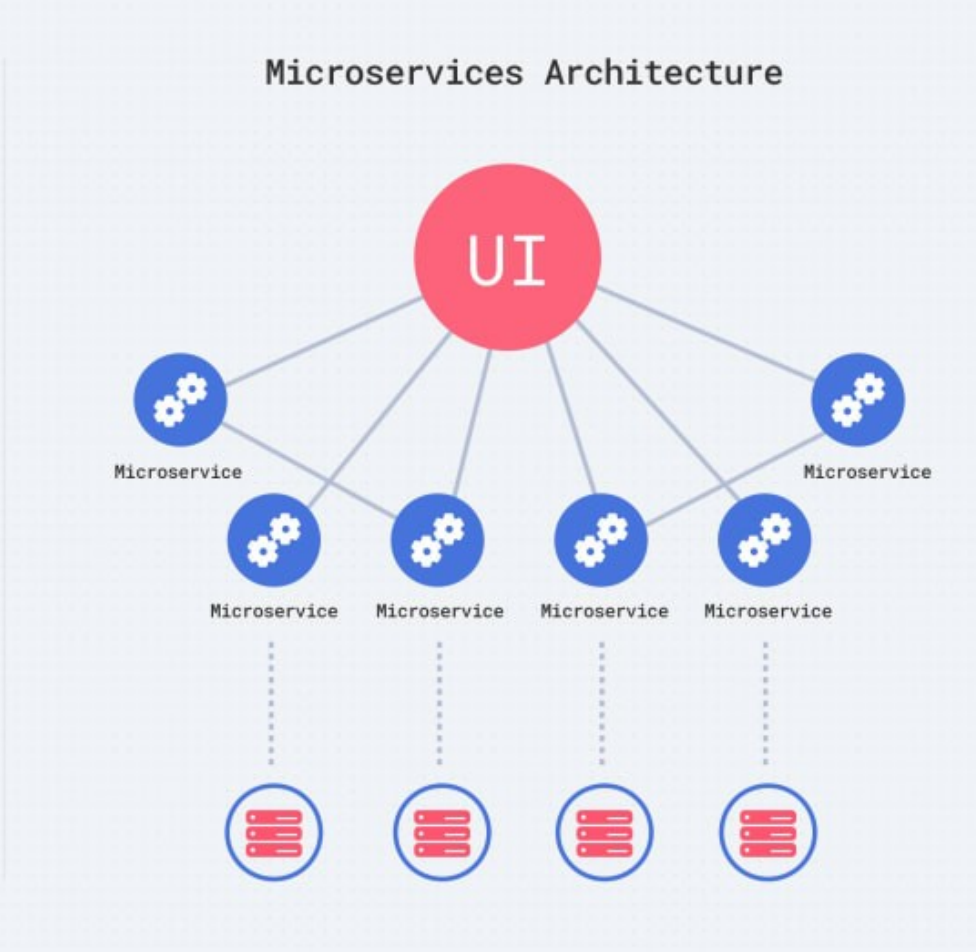
KEY DEVOP PRACTICES

- **Continuous Integration (CI)**: Frequent code integrations into a shared repository.
- **Continuous Delivery (CD)**: Automating the software release process.
- **Infrastructure as Code (IaC)**: Managing and provisioning infrastructure through code rather than manual processes.

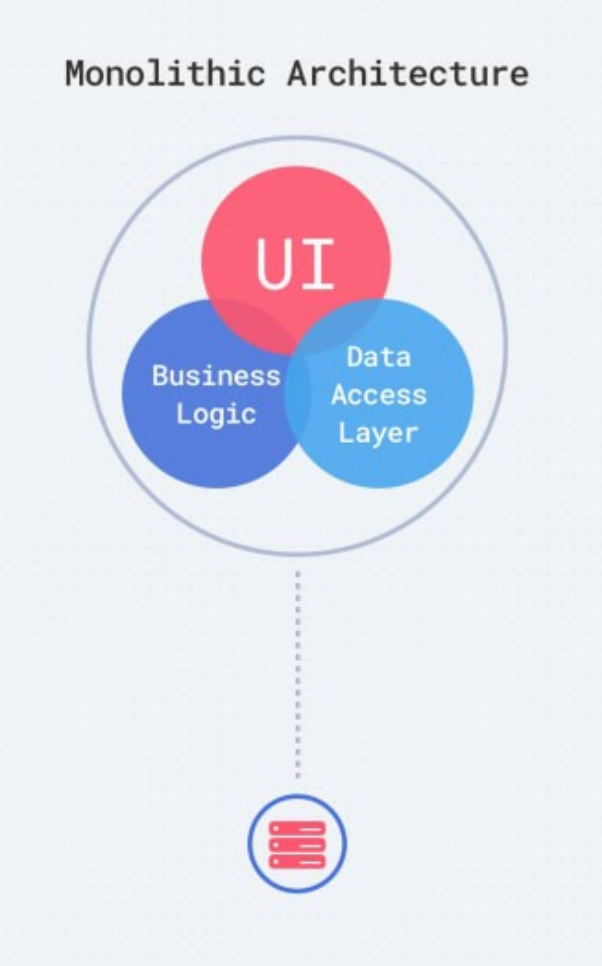


MONOLITHIC/MICROSERVICES





MONOLITHIC/MICROSERVICES



- Everything from user interface, business codes and database calls is included in the same codebase.

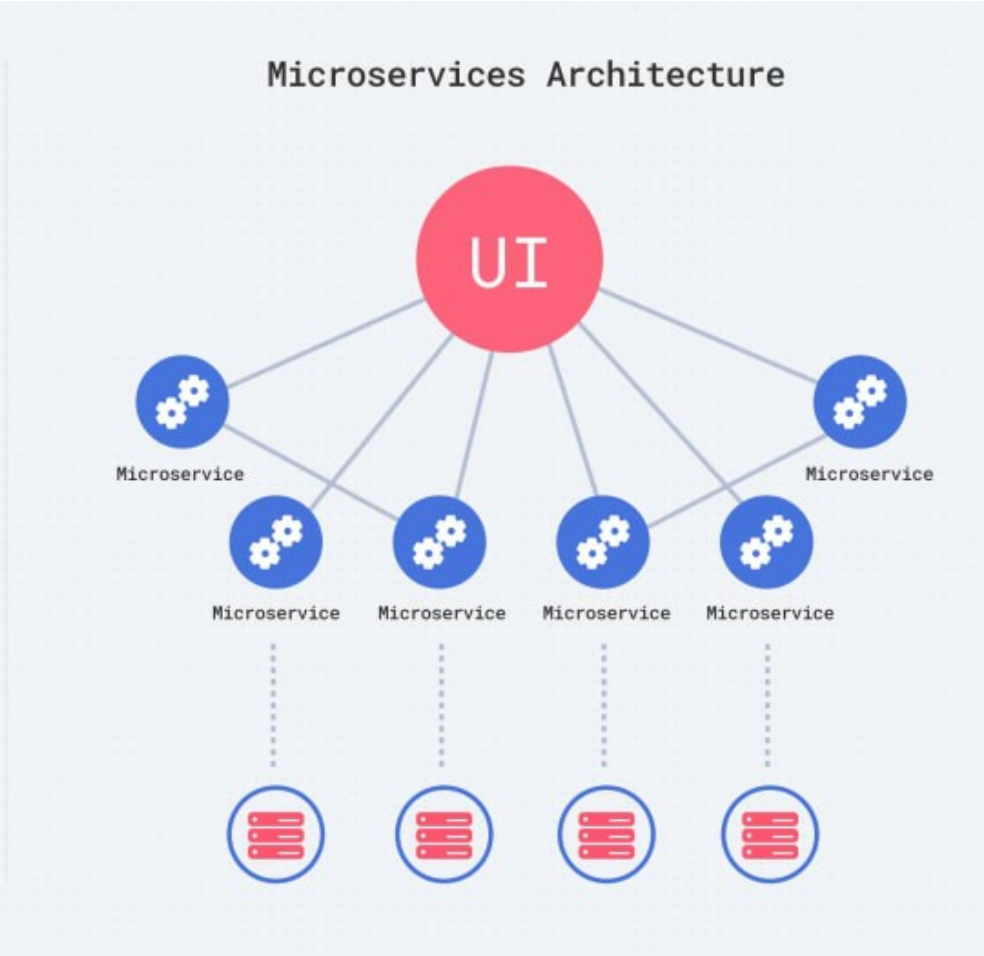
- All application concerns are contained in a single big deployment

- Disadvantages:
  - Difficult to scale
  - Difficult to test
  - Difficult to evolve and add new feature

MONOLITHIC/MICROSERVICES

- Microservices are an architectural style that structures an application as a collection of services that are:
  - Highly maintainable and testable
  - Loosely Coupled
  - Independently deployable
  - Organized around business capabilities
  - Owned by a small team

- The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications.

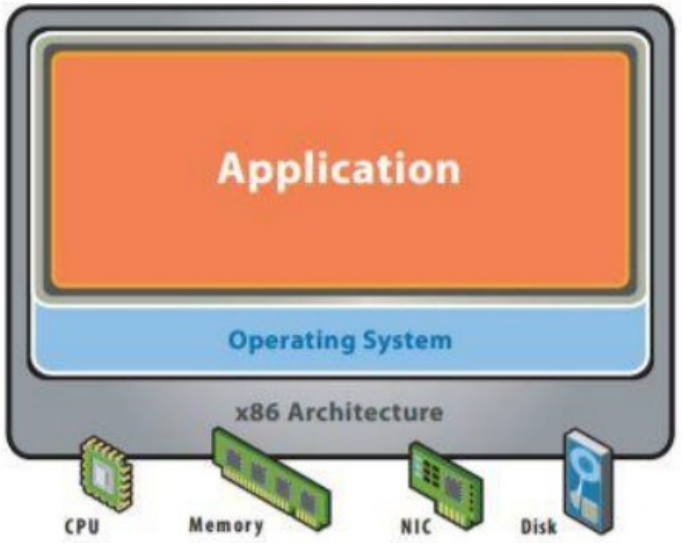


VIRTUALIZATION

Before Virtualization

- Servers would traditionally run one application on one server with one operating system
  - Even one or more applications and an operating system would run on their own unique physical server

- Expensive hardware were being purchased, but not used
  - Depending on application, most of resources were unused



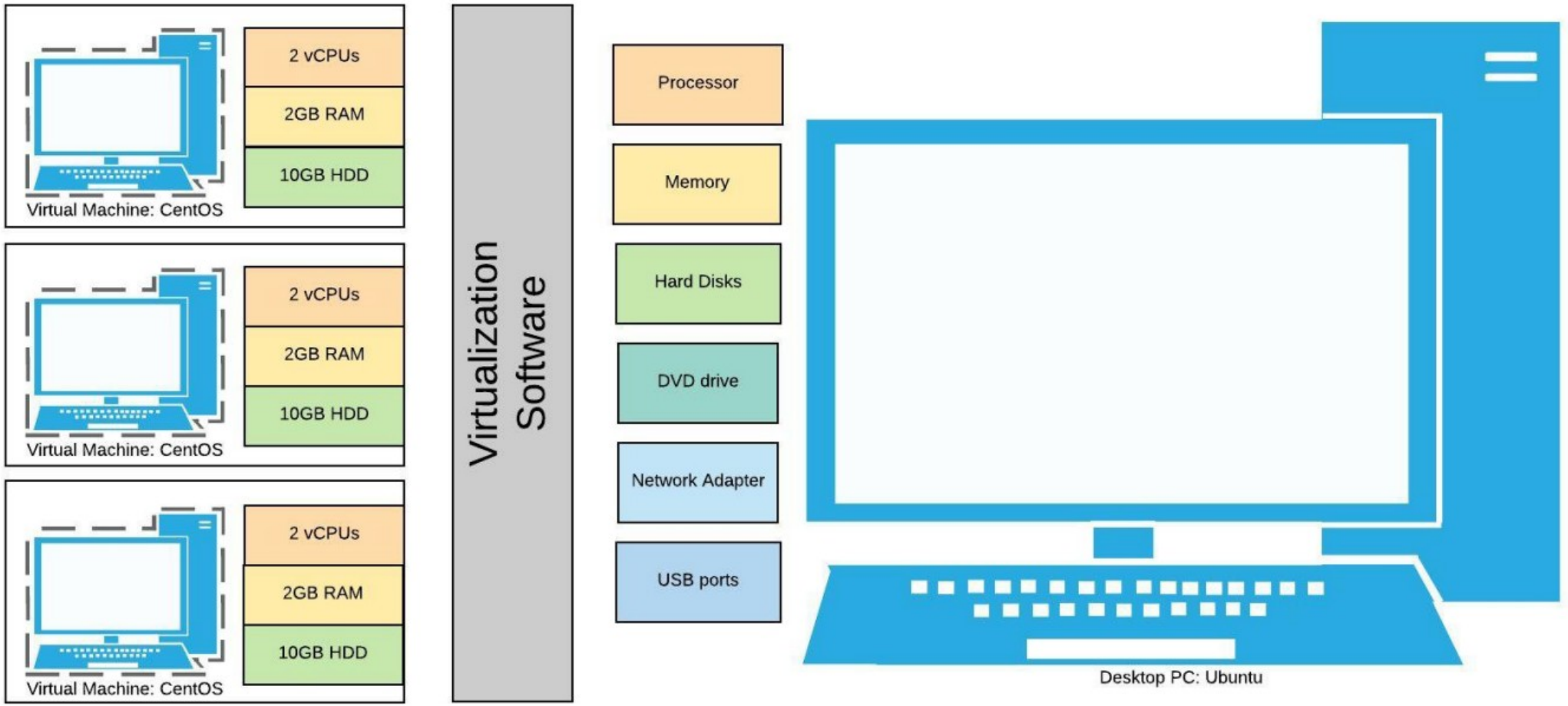
VIRTUALIZATION

Before Virtualization

- It was not unusual to see a physical server using less than five percent, or even ten percent, of its CPU and/or memory
- Multiple applications in a single OS, in one operating system have an impact in terms of security

WHAT IS VIRTUALIZATION

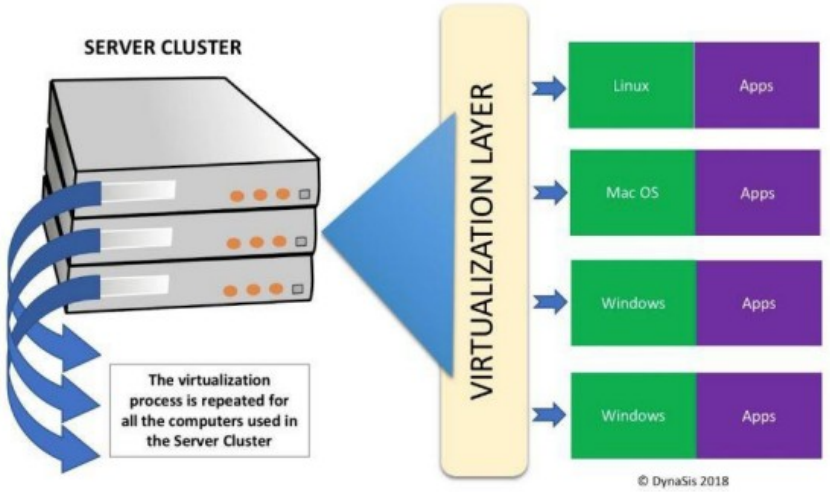
Hardware Virtualization: a Desktop Virtualization Example



WHY VIRTUALIZATION

- Issues with traditional systems:
  - Running multiple applications on same machine often creates conflict
  - Underutilized resources
  - Inflexible and costly infrastructure
  - Software and hardware tightly coupled

- Virtualization lets you run more applications on fewer physical servers.
  - Rather than **one** application running on **one** server with **one** operating system, **multiple** VMs run **multiple** apps and operating systems on **one** physical server



HYPERVISOR

- A hypervisor is computer software, firmware or hardware that creates and runs virtual machines

- It's a process that separates a computer's operating system and applications from the underlying physical hardware
  - Even though VMs can run on the same physical hardware, they are still logically separated from each other
  - That means that if one VM experiences an error, crash or malware attack, it doesn't extend to other VMs on the same machine

Microsoft Hyper-v

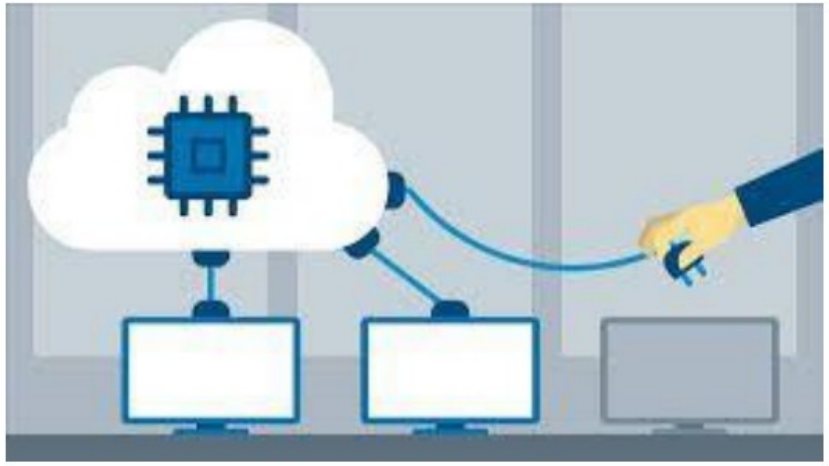
vmware

HYPERVISOR

	VMware	VirtualBox	Hyper-V
Facilidad de uso	Medio	Fácil	Complicado
Rendimiento	Bueno	Medio	Bueno
Instantáneas	Si	Si	No
Compartir archivos	Si	Si	Si, pero complicado.
Integración con Windows	Si	Si	No
Cifrado	Si	Si (a través de Guest Additions)	Si
Sistemas compatibles	Windows, Linux, macOS	Windows, Linux, macOS	Windows y Linux (este con limitaciones)
Precio	Gratis / De pago	Gratis	Gratis
Otros	Excelente seguridad	OpenSource	Solo en Windows 10 Pro Soporte WSL y WSL2 W

VIRTUALIZATION GCP

- **Google Compute Engine - GCE**
  - It's the Infrastructure as a Service (IaaS)
  - Enables users to lanch Virtual Machines on demand
  - VMs can be launched from the **standard images** or **custom images**
  - An image is a persistent disk that contains the OS and root file system that is necessary for starting an instance



RECAP AND KEY TAKEAWAYS

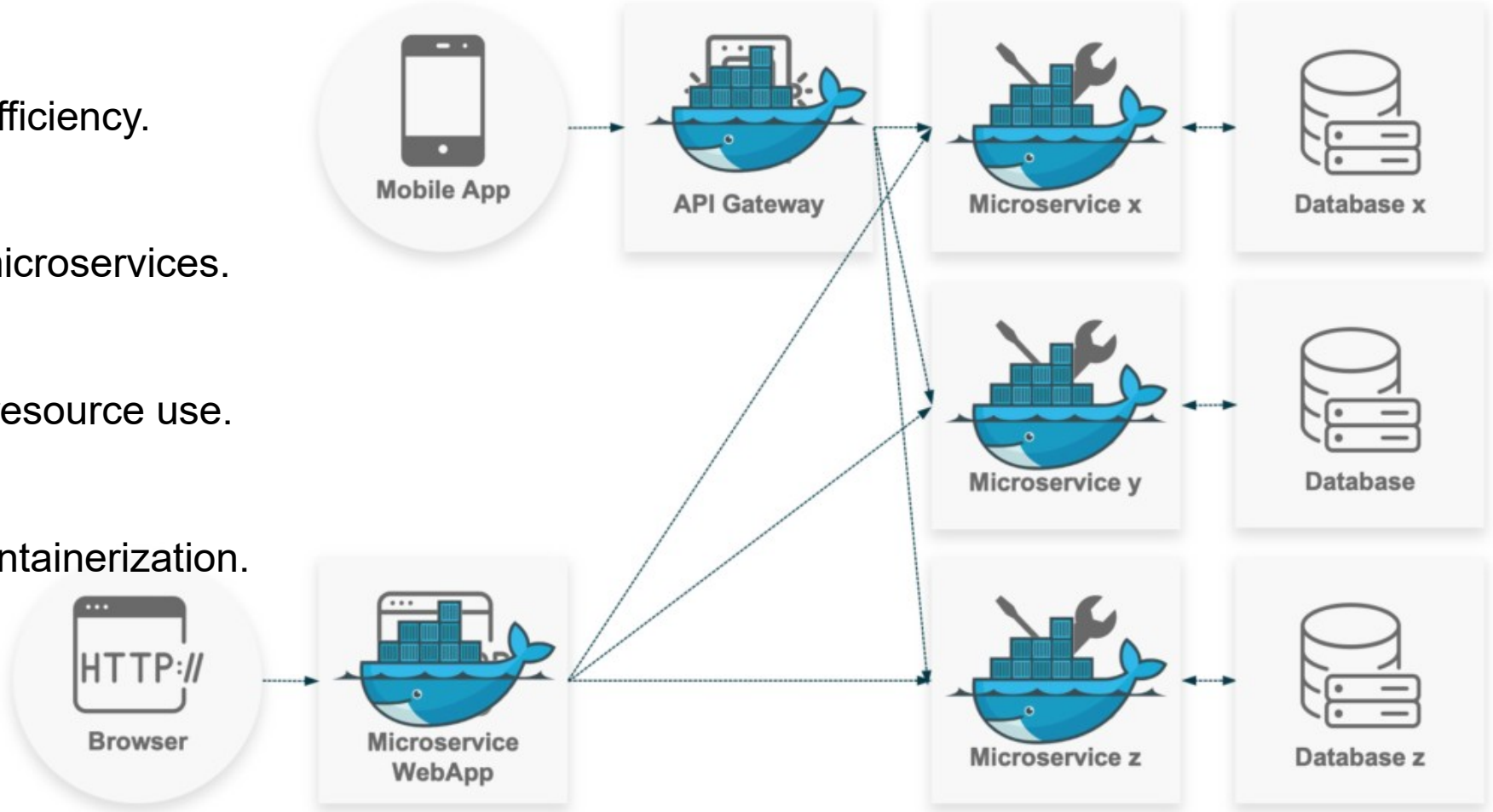
Services and Ports:
Foundation of network communication.

DevOps:
Enhancing collaboration and efficiency.

Application Architectures:
Transition from monolithic to microservices.

Virtualization:
Enabling flexible and efficient resource use.

Preparing for Docker:
Understanding the need for containerization.



DIVING INTO DOCKER

Why Docker?:

- Solves many limitations of VMs.
- Simplifies deployment and scaling.

Docker in DevOps:

- Integral to modern CI/CD pipelines.

What's Ahead:

Installing Docker.
Running your first container..