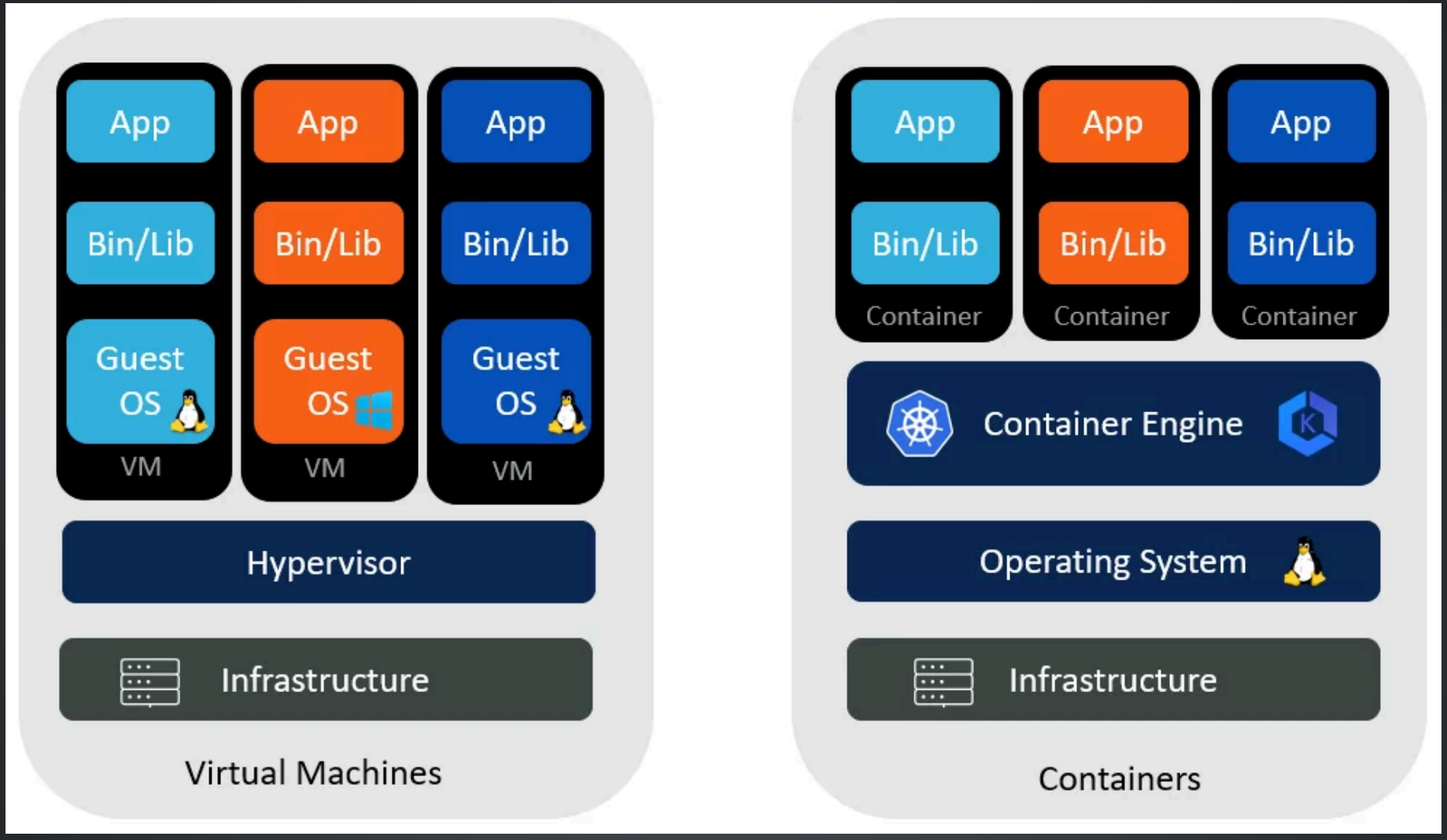
DSA - DOCKER

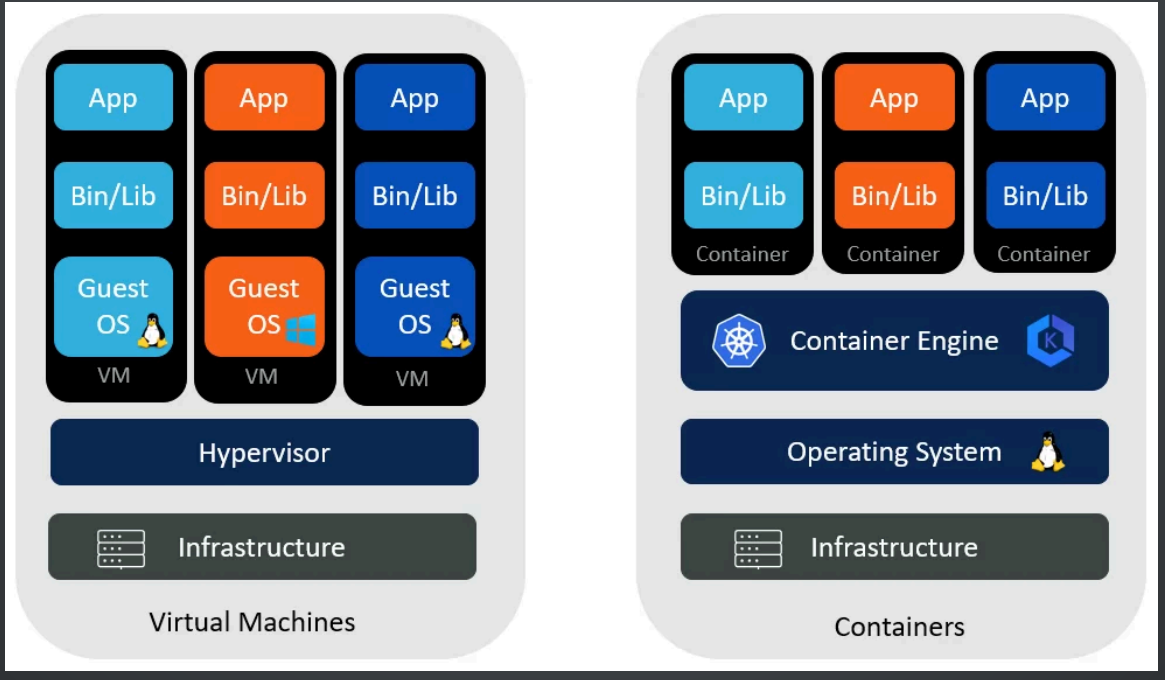
CONTENEDORES VS. MÁQUINAS VIRTUALES

- En el mundo de la virtualización y el desarrollo de software, existen dos enfoques principales: **máquinas virtuales** y **contenedores**.
- Los contenedores han surgido como una alternativa ligera y eficiente a las máquinas virtuales tradicionales.


App
Bin/Lib
Guest OS
VM
App
Bin/Lib
Guest OS
VM
App
Bin/Lib
Guest OS
VM
Hypervisor
Infrastructure
Virtual Machines
App
Bin/Lib
Container
App
Bin/Lib
Container
App
Bin/Lib
Container
Container Engine
Operating System
Infrastructure
Containers

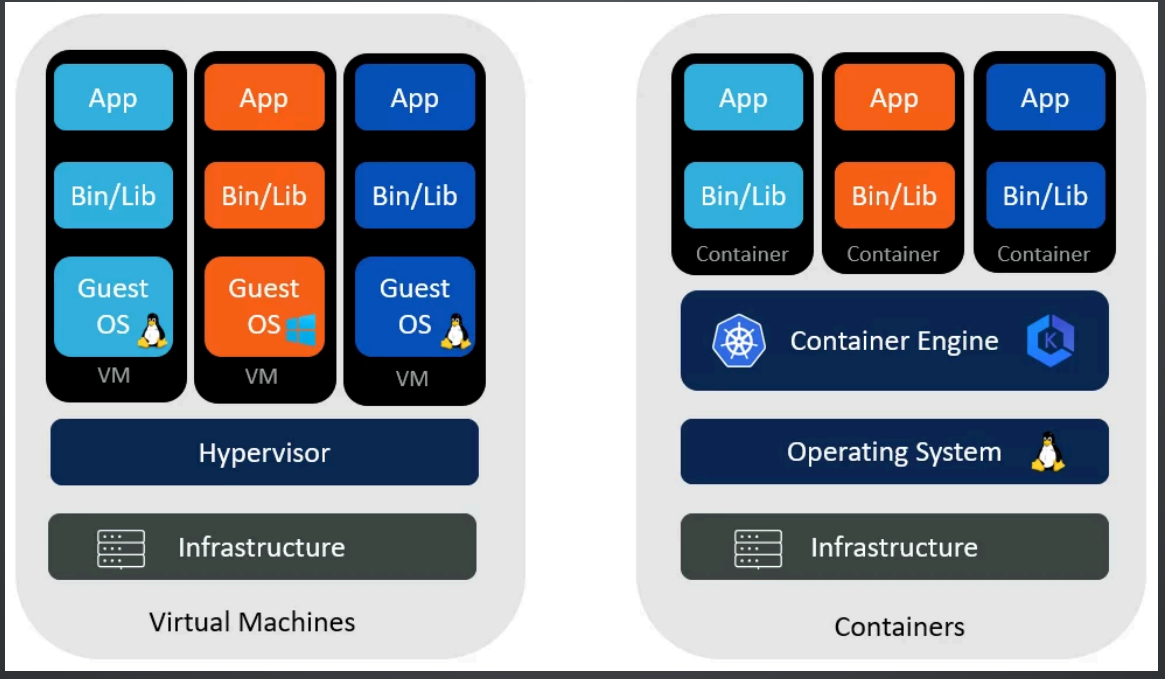
MÁQUINAS VIRTUALES (VMS)

- Las **máquinas virtuales** son entornos de computación completos que se ejecutan sobre un hardware físico.
- Cada VM incluye un sistema operativo completo y recursos virtuales dedicados como CPU, RAM y almacenamiento.
- La virtualización completa del sistema operativo a través de un **hipervisor** permite la ejecución de múltiples sistemas operativos en un solo servidor.



CONTENEDORES

- Los **contenedores** son entornos de ejecución ligeros y portátiles que comparten el kernel del sistema operativo del host.
- Cada contenedor encapsula una aplicación y sus dependencias, pero no incluye un sistema operativo completo como en el caso de las máquinas virtuales.
- Los contenedores proporcionan aislamiento a nivel de proceso y utilizan recursos de manera más eficiente que las VMs.



CASOS DE USO

- **Desarrollo Ágil**: Los contenedores permiten a los equipos de desarrollo crear, probar y desplegar aplicaciones de manera rápida y consistente.
- **Entornos Aislados**: Las imágenes utilizadas por los contenedores ofrecen aislamiento de la aplicación y sus dependencias. Esto favorece prácticas de DevOps.
- **Despliegue Escalable**: La escalabilidad horizontal de los contenedores facilita el despliegue de aplicaciones en entornos de nube y microservicios.

DOCKER

- **Docker** es una plataforma de contenedores que simplifica el proceso de creación, distribución y ejecución de contenedores.
- Docker ha revolucionado la forma en que empaquetamos, distribuimos y ejecutamos nuestras aplicaciones.
- Dos conceptos fundamentales en Docker son las **imágenes** y los **contenedores**.

IMÁGENES DOCKER

- Una **imagen Docker** es un paquete ligero y autónomo que contiene todo lo necesario para ejecutar una aplicación: código, runtime, librerías, variables de entorno y configuraciones.
- Se crean a partir de un archivo llamado Dockerfile que especifica los pasos para construir la imagen.
- Las imágenes se pueden almacenar localmente o en repositorios compartidos como DockerHub. Esto permite que las mismas puedan ser reutilizadas fácilmente.

DOCKERFILE

- Es un archivo de configuración que se utiliza para crear imágenes.
- En dicho archivo indicamos qué es lo que queremos que tenga la imagen, y los distintos comandos para instalar las herramientas.

CONTEINERIZANDO WGET

```
FROM alpine

RUN apk update
RUN apk add wget
RUN rm -rf /var/cache/apk/*

WORKDIR /root

ENTRYPOINT [ "wget" ]

CMD [ "--help" ]
```

```
# Construcción de la imagen
docker build -t dsa_wget .

# Inspección de las imagenes locales
docker images

# Ejecutar el comando conteinerizado (sale la ayuda)
docker run dsa_wget

# Ejecutar el comando conteinerizado para descargar la pagina de la UNLP
docker run dsa_wget https://unlp.edu.ar
```

DONDE ESTÁ LA DESCARGA REALIZADA?

```
FROM alpine

RUN apk update
RUN apk add wget
RUN rm -rf /var/cache/apk/*

WORKDIR /root

ENTRYPOINT [ "wget" ]

CMD [ "--help" ]
```

```
nico in ~/catedras/dsa/tmp/wget λ ls
Dockerfile
nico in ~/catedras/dsa/tmp/wget λ docker run dsa_wget https://unlp.edu.ar
--2024-04-21 15:58:30--  https://unlp.edu.ar/
Resolving unlp.edu.ar (unlp.edu.ar)... 163.10.0.135, 2800:340:0:64::135
Connecting to unlp.edu.ar (unlp.edu.ar)|163.10.0.135|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

     0K .......... .......... .......... .......... ..........  995K
    50K .......... .......... .......... .......... .......... 2.01M
   100K .......... .......... .......... .......... .......... 8.82M
   150K .......... .......... .......... .......... ..........  166M
   200K .......... .......... .......... .......... .......... 2.26M
   250K .......... .......... .......... ..........             147M=0.1s

2024-04-21 15:58:31 (2.77 MB/s) - 'index.html' saved [297014]
```

```
nico in ~/catedras/dsa/tmp/wget λ ls
Dockerfile
nico in ~/catedras/dsa/tmp/wget λ
```

VOLÚMENES

- No es una buena práctica guardar los datos persistentes dentro de un contenedor de Docker.
- Los volúmenes son espacios de almacenamiento fuera de los contenedores.
- Así podremos crear y borrar contenedores sin preocuparnos por que se borren los datos.
- Concepto analogo a las carpetas compartidas en VMs.
- También se pueden utilizar para compartir datos entre contenedores.

```
# Ejecutar docker usando un volumne
mkdir salida
docker run -v ./salida:/root dsa_wget https://unlp.edu.ar
```

```
nico in ~/catedras/dsa/tmp/wget λ mkdir salida
nico in ~/catedras/dsa/tmp/wget λ ls
Dockerfile  salida
nico in ~/catedras/dsa/tmp/wget λ docker run -v ./salida:/root dsa_wget https://unlp.edu.ar
--2024-04-21 16:59:10--  https://unlp.edu.ar/
Resolving unlp.edu.ar (unlp.edu.ar)... 163.10.0.135, 2800:340:0:64::135
Connecting to unlp.edu.ar (unlp.edu.ar)|163.10.0.135|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

     0K .......... .......... .......... .......... .......... 1.60M
    50K .......... .......... .......... .......... ..........  817K
   100K .......... .......... .......... .......... ..........  151M
   150K .......... .......... .......... .......... .......... 1.59M
   200K .......... .......... .......... .......... ..........  153M
   250K .......... .......... .......... ..........            15.9M=0.1s

2024-04-21 16:59:12 (2.26 MB/s) - 'index.html' saved [297014]

nico in ~/catedras/dsa/tmp/wget λ ls salida
index.html
nico in ~/catedras/dsa/tmp/wget λ
```

COMPOSE

- En ocasiones es necesario enlazar contenedores entre si, por ejemplo: frontend, backend y db
- Docker Compose es una herramienta que permite simplificar el uso de contenedores Docker cuando se necesita armar soluciones que agrupan distintos contenedores.
- A partir de archivos YAML es mas sencillo crear contendores, conectarlos, habilitar puertos, volumenes, etc

UN EJEMPLO POSIBLE

```
docker-compose.yml

- app web
  se expone en el puerto 8000

- servidor de base de datos
  usado internamente por los otros contenedores

- phpmyadmin
  se expone en el puerto 8080
```

COMANDOS UTILES

```
# Dado un archivo docker-compose.yml en la carpeta actual
# Levantar servicios asociados en el archivo docker-compose.yml
docker compose up -d

# Inspeccionar contenedores / puertos expuestos
docker compose ps


# Inspeccionar logs de contenedores
docker compose logs
```

COMANDOS UTILES

```
# Crear archivo docker-compose.yml
nico in ~/catedras/dsa/tmp/teoria-2-compose λ cat docker-compose.yml

services:

  dvwa_web:
    image: cytopia/dvwa:php-${PHP_VERSION:-8.1}
    restart: unless-stopped
    ports:
      - "8000:80"
    networks:
      - dvwa-net
    environment:
      - MYSQL_HOSTNAME=dvwa_db
      - MYSQL_DATABASE=dvwa
      - MYSQL_USERNAME=dvwa
      - MYSQL_PASSWORD=p@ssw0rd

  dvwa_db:
    image: mariadb:10.1
    hostname: dvwa_db
    volumes:
      - dvwa_db_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: dvwa
      MYSQL_USER: dvwa
      MYSQL_PASSWORD: p@ssw0rd
    restart: unless-stopped
    networks:
      - dvwa-net
```