

# CS2032 - Cloud Computing (Ciclo 2024-2)

## Virtualización con contenedores

### Semana 3 - Taller 1: Contenedores

---

ELABORADO POR: GERALDO COLCHADO

# Contenido

## Contenedores

1. **Objetivo del taller 1**
2. Ejercicio 1: Instalar docker en Ubuntu
3. Ejercicio 2: Comandos básicos de docker
4. Ejercicio 3: Contenedor de página web
5. Ejercicio 4: Contenedor de Api REST python3
6. Cierre

# Objetivo del taller 1:

## Contenedores

---

- Aprender a instalar docker en ubuntu Linux
- Aprender los comandos básicos de docker
- Aprender a crear una imagen, ejecutar y detener un contenedor de página web
- Aprender a crear una imagen, ejecutar y detener un contenedor de Api REST python3

# Contenido

## Contenedores

1. Objetivo del taller 1
2. **Ejercicio 1: Instalar docker en Ubuntu**
3. Ejercicio 2: Comandos básicos de docker
4. Ejercicio 3: Contenedor de página web
5. Ejercicio 4: Contenedor de Api REST python3
6. Cierre

# Ejercicio 1:

## Instalar docker en ubuntu

---

Nota: La Amazon Machine Image (AMI) “amazon/Cloud9Ubuntu” ya viene con docker instalado.

- Paso 1: Ingresar por ssh a la máquina virtual “MV Desarrollo” con ubuntu Linux
- Paso 2: Verificar si docker ya está instalado:  
\$ docker -v  
Docker version 27.1.1, build 6312585

En caso no tuviera docker instalado puede seguir este manual:  
<https://docs.docker.com/engine/install/ubuntu/>

# Contenido

## Contenedores

1. Objetivo del taller 1
2. Ejercicio 1: Instalar docker en Ubuntu
3. **Ejercicio 2: Comandos básicos de docker**
4. Ejercicio 3: Contenedor de página web
5. Ejercicio 4: Contenedor de Api REST python3
6. Cierre

# Ejercicio 2:

## Comandos básicos de docker

---

Comando	Descripción
\$ docker build	Construir una imagen desde un Dockerfile
\$ docker run	Ejecuta una imagen en un nuevo contenedor
\$ docker images	Lista las imágenes almacenadas
\$ docker ps	Lista los contenedores en ejecución
\$ docker stop	Detiene la ejecución de un contenedor
\$ docker start	Inicia un contenedor detenido
\$ docker rm	Elimina un contenedor detenido
\$ docker exec	Ejecuta un comando en un contenedor en ejecución

# Contenido

## Contenedores

1. Objetivo del taller 1
2. Ejercicio 1: Instalar docker en Ubuntu
3. Ejercicio 2: Comandos básicos de docker
4. **Ejercicio 3: Contenedor de página web**
5. Ejercicio 4: Contenedor de Api REST python3
6. Cierre



# Ejercicio 3:

## Contenedor de página web

---

- **Paso 1:** Abrir puerto 8080 en máquina virtual “MV Desarrollo”
- **Paso 2:** Agregue el archivo Dockerfile al repositorio websimple en github
- **Paso 3:** Cree el directorio /home/ubuntu/contenedores/ e ingrese
- **Paso 4:** Descargar de github el repositorio websimple con git clone

# Ejercicio 3:

## Contenedor de página web

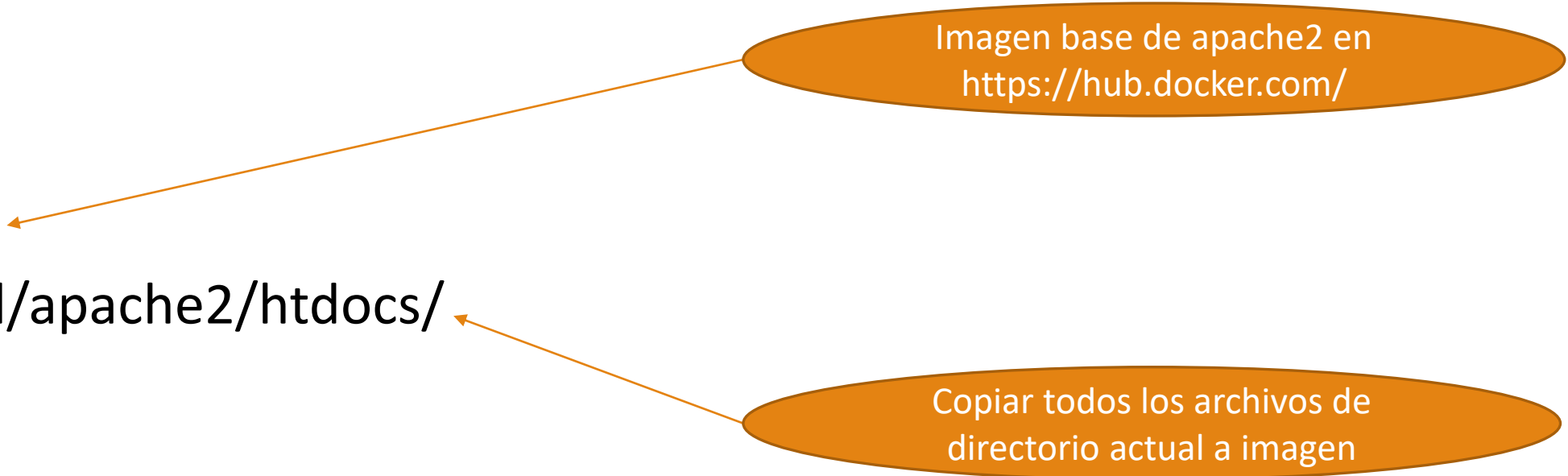
---

### Dockerfile

FROM httpd:2.4

COPY . /usr/local/apache2/htdocs/

Imagen base de apache2 en  
<https://hub.docker.com/>



Copiar todos los archivos de  
directorio actual a imagen

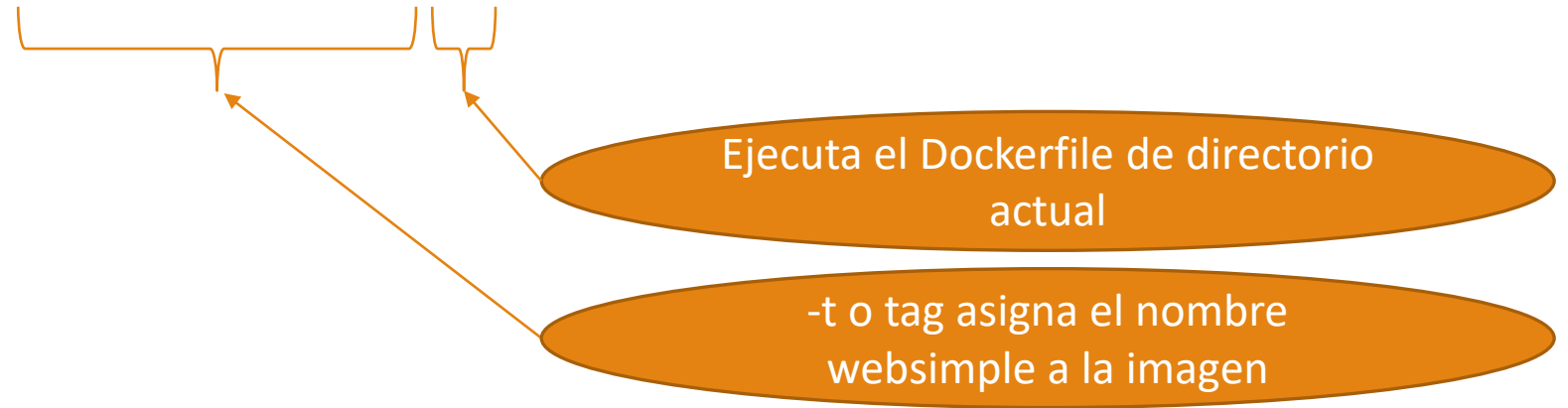
# Ejercicio 3:

## Contenedor de página web

---

- **Paso 5:** En /home/ubuntu/contenedores/websimple/ ejecutar el comando para **construir la imagen**:

\$ docker build -t websimple .



# Ejercicio 3:

## Contenedor de página web

---

- **Paso 6:** Ejecute este comando para **ejecutar en un contenedor** la imagen creada anteriormente:

\$ docker run -d -p 8080:80 websimple

Ejecutar contenedor en segundo plano y mostrar el container ID

Nombre de la imagen para ejecutar en un contenedor

Relaciona el puerto 8080 de la máquina virtual con el Puerto 80 del contenedor

Nota: Para detener la ejecución use este comando: \$ **docker stop CONTAINER ID**  
Obtener CONTAINER ID con \$ **docker ps**

# Ejercicio 3:

## Contenedor de página web

---

- **Paso 7:** Pruebe la página web en un navegador:  
**dirección-IP:8080**



## **Página web del Curso Cloud Computing**

### **Taller de máquina virtual**

Los alumnos son:

1. PEREZ, JUAN
2. COLCHADO, GERALDO
3. LAPADULA, GIANLUCA
4. TORERO, MELVIN

# Contenido

## Contenedores

1. Objetivo del taller 1
2. Ejercicio 1: Instalar docker en Ubuntu
3. Ejercicio 2: Comandos básicos de docker
4. Ejercicio 3: Contenedor de página web
5. **Ejercicio 4: Contenedor de Api REST python3**
6. Cierre

# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 1:** Subir el archivo Dockerfile a repositorio api-students de github
- **Paso 2:** Ingrese a directorio /home/ubuntu/contenedores/
- **Paso 3:** Descargue el repositorio api-students con git clone

Nota: Entender a detalle el archivo Dockerfile que tiene los comandos para construir automáticamente la imagen

# Ejercicio 4:

## Contenedor de Api REST python3

---

### Dockerfile

```
FROM python:3-slim  
WORKDIR /programas/api-students  
RUN pip3 install flask  
COPY . .  
RUN python3 db.py  
CMD [ "python3", "./app.py" ]
```

Imagen base de python3 delgada  
en <https://hub.docker.com/>

Directorio a crear en imagen

Instalar flask en imagen

Copiar todos los archivos de  
directorio actual a imagen

Crear la base de datos en imagen

Ejecutar el api REST de estudiantes  
cuando se ejecute la imagen en  
contenedor



# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 4:** Ingrese al directorio `/home/ubuntu/contenedores/api-students/` y ejecute este comando para **construir la imagen:**

`$ docker build -t api-students .`



Ejecuta el Dockerfile de directorio actual

-t o tag asigna el nombre api-students a la imagen

# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 5:** Ejecute este comando para **ejecutar en un contenedor** la imagen creada anteriormente:

\$ docker run -p 8000:8000 api-students

Nombre de la imagen para ejecutar en un contenedor

Relaciona el puerto 8000 de la máquina virtual con el Puerto 8000 del contenedor

Nota: Con CTRL + C podemos detener la ejecución de este contenedor

# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 6:** Ejecute este comando para ejecutar el contenedor en segundo plano:

`$ docker run -d -p 8000:8000 api-students`

Ejecutar contenedor en segundo plano y mostrar el container ID

Nombre de la imagen para ejecutar en un contenedor

Relaciona el puerto 8000 de la máquina virtual con el Puerto 8000 del contenedor

Nota: Para detener la ejecución use este comando:

`$ docker stop CONTAINER ID`

Obtener CONTAINER ID con  
`$ docker ps`

Nota: Para volver a iniciar use este comando:

`$ docker start CONTAINER ID`

Obtener CONTAINER ID con  
`$ docker ps -a`

Nota: Para eliminar un contenedor detenido use este comando:

`$ docker rm CONTAINER ID` (Obtener CONTAINER ID con `$ docker ps -a`)

# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 7:** Abra otra sesión ssh y ejecute este comando para **entrar al contenedor en modo comando:**

```
$ docker exec -it bd2d1cd689e2 /bin/bash
```

-i es de interactive  
-t es de Allocate a pseudo-TTY

Ejecutar el intérprete de  
comandos

Reemplazar por el CONTAINER ID  
(\$ docker ps)

# Ejercicio 4:

## Contenedor de Api REST python3

---

```
:~ $ docker exec -it bd2d1cd689e2 /bin/bash
root@bd2d1cd689e2:/programas/api-students# ls -l
total 20
-rw-rw-r-- 1 root root 131 Aug 28 00:01 Dockerfile
-rw-rw-r-- 1 root root 2788 Aug 24 03:55 app.py
-rw-rw-r-- 1 root root 346 Aug 24 03:51 db.py
-rw-r--r-- 1 root root 8192 Aug 28 00:02 students.sqlite
root@bd2d1cd689e2:/programas/api-students# exit
exit
:~ $
```

# Ejercicio 4:

## Contenedor de Api REST python3

---

- **Paso 8:** Pruebe el api REST de estudiantes con <https://www.postman.com/>

# Contenido

## Contenedores

1. Objetivo del taller 1
2. Ejercicio 1: Instalar docker en Ubuntu
3. Ejercicio 2: Comandos básicos de docker
4. Ejercicio 3: Contenedor de página web
5. Ejercicio 4: Contenedor de Api REST python3
6. **Cierre**

# Cierre:

## Contenedores - Qué aprendimos?

---

- Instalar docker en ubuntu Linux
- Comandos básicos de docker
- Crear una imagen, ejecutar y detener un contenedor de página web
- Crear una imagen, ejecutar y detener un contenedor de Api REST python3



# Gracias

Elaborado por docente: Geraldo Colchado