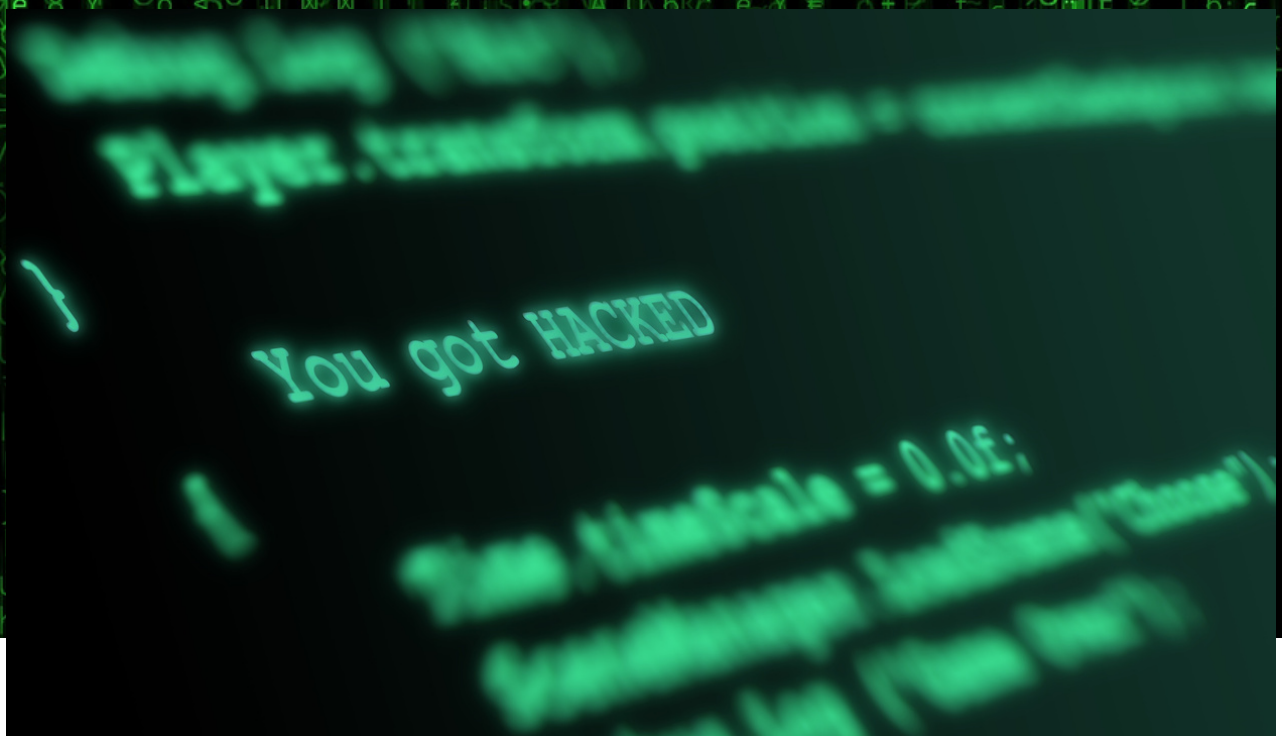


DOCKER en acción: Dominando los fundamentos.

Fondo evolución digital. | Olumni | Fundación EducaMás | Prográmate Academy | SIMPLON.co

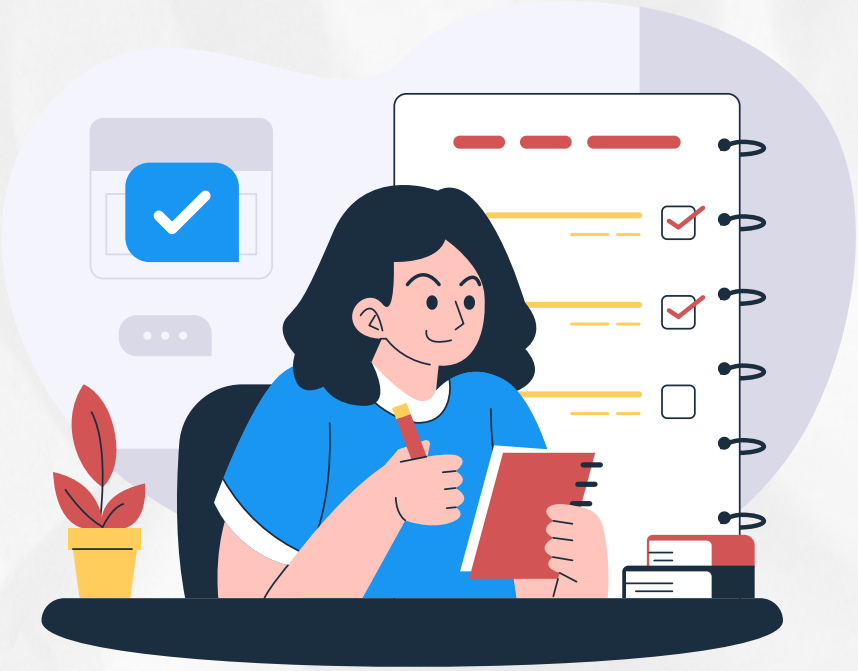
IMPORTANTE

Página Docker

Se estara utilizando la herramienta Docker, una aplicación que permitira el entendimiento de la clase gratis. A continuación el link de descarga:

https://docs.docker.com/desktop/

CONCEPTOS BASICOS



- Imágenes de Docker
- Creación de contenedores
- Relación entre imágenes y contenedores
- Registros de Docker
- Docker Hub

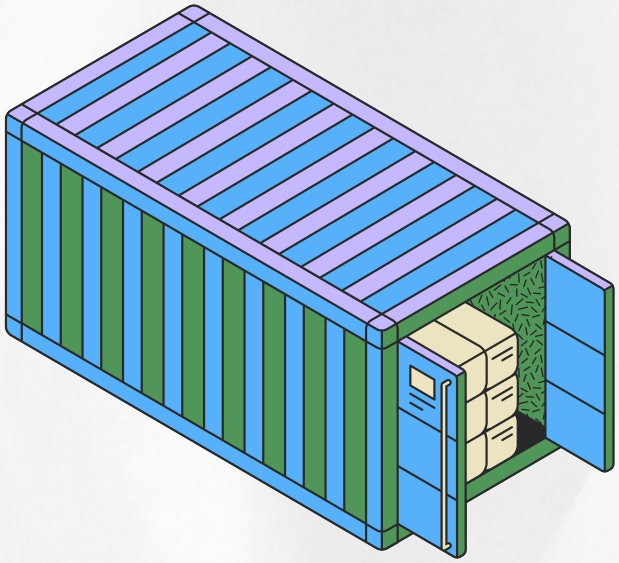
IMÁGENES DE DOCKER

Una imagen de Docker es una plantilla o plantilla de solo lectura que contiene todo lo necesario para ejecutar una aplicación, incluye el código, las bibliotecas, las dependencias y las configuraciones. Las imágenes se utilizan como base para crear contenedores.

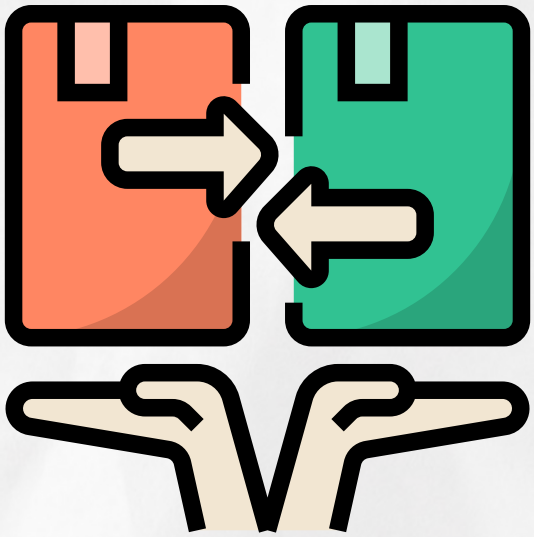
CREACIÓN DE CONTENEDORES

Los contenedores son instancias en tiempo de ejecución de una imagen de Docker. Se pueden crear y ejecutar múltiples contenedores a partir de una sola imagen.

Cada contenedor es aislado y contiene su propio entorno de ejecución, lo que significa que puede tener aplicaciones y dependencias independientes.



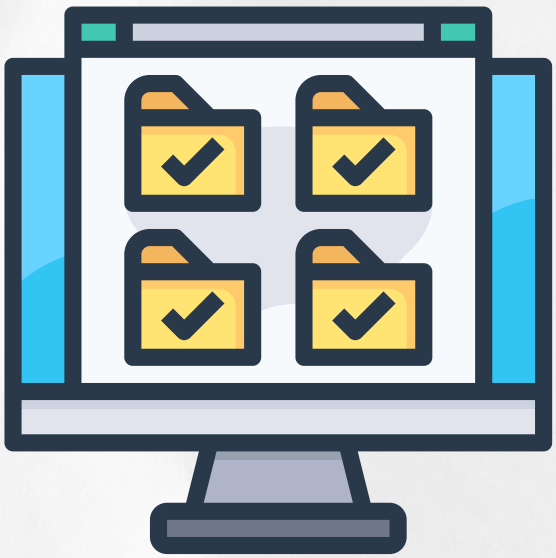
RELACIÓN ENTRE IMÁGENES Y CONTENEDORES:



Las imágenes de Docker son plantillas o moldes mientras que los contenedores son las instancias en tiempo de ejecución creadas a partir de esas plantillas. Cada contenedor tiene una copia de la imagen subyacente, sin embargo, pueden tener configuraciones y estados individuales.

REGISTROS DE DOCKER:

Los registros de Docker son repositorios donde se almacenan y comparten imágenes de Docker. El registro más popular es Docker Hub, un repositorio público donde se encontra una amplia gama de imágenes. También existen registros privados que se pueden almacenar y compartir imágenes dentro de la organización.



¿DOCKER HUB?

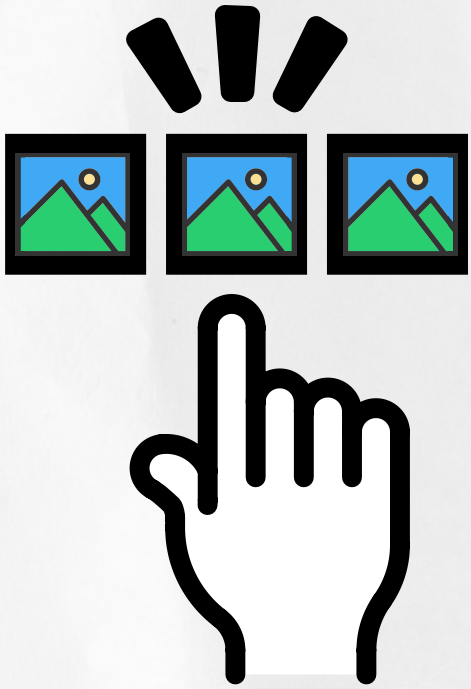
DOWNLOAD

Docker Hub es un registro de imágenes públicas donde se encuentran imágenes creadas por la comunidad y empresas.

Permite buscar imágenes específicas, ver sus descripciones, verificaciones de integridad y descargarlas en la máquina para su uso.

¿POR QUE USAR DOCKER?

Recuerda que las imágenes de Docker son livianas, portátiles y se pueden compartir fácilmente. Esto significa que puedes utilizar imágenes existentes de Docker Hub o crear tus propias imágenes personalizadas para tus aplicaciones.



REQUERIMIENTOS MAC

- Se recomienda la versión 11 o más reciente. Esto incluye Big Sur (11), Monterey (12) o Ventura (13).
- Recomendamos actualizar la última versión de macOS.
- Se requieren al menos 4 GB de RAM.
- No se debe tener instalado VirtualBox anterior a la versión 4.3.30, ya que no es compatible con Docker Desktop.

INSTALACIÓN EN WINDOWS

- Descarga el instalador de Docker para macOS desde el sitio oficial de Docker (https://www.docker.com/products/docker-desktop).
- Ejecuta el instalador descargado y sigue las instrucciones del asistente de instalación.
- Una vez completada la instalación, abre la aplicación Docker desde Launchpad o la carpeta Aplicaciones.
- Docker Desktop se iniciará y, una vez que el icono de Docker en la barra de menú se vuelva estable, Docker estará listo para su uso.
- Verifica la instalación abriendo una terminal y ejecutando el siguiente comando:
  docker run hello-world
- Si la instalación fue exitosa, verás un mensaje que indica que Docker está funcionando correctamente.

INSTALACIÓN EN LINUX

- Actualiza el gestor de paquetes:

```
sudo apt-get update
```

- Instala los paquetes necesarios para permitir que apt utilice repositorios a través de HTTPS:

```
sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
```

- Agrega la clave GPG oficial de Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

- Agrega el repositorio de Docker a las fuentes de apt:

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

INSTALACIÓN EN LINUX

- Actualiza nuevamente el gestor de paquetes:

sudo apt-get update

- Instala Docker:

sudo apt-get install docker-ce docker-ce-cli containerd.io

- Verifica la instalación ejecutando el siguiente comando:

sudo docker run hello-world

EJERCICIOS

1. Dockerfile

Dockerfile

```
1  FROM python:3.10
2
3  WORKDIR /app
4  COPY requirements.txt /app/requirements.txt
5
6  RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt
7
8  COPY . /app
9
10 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

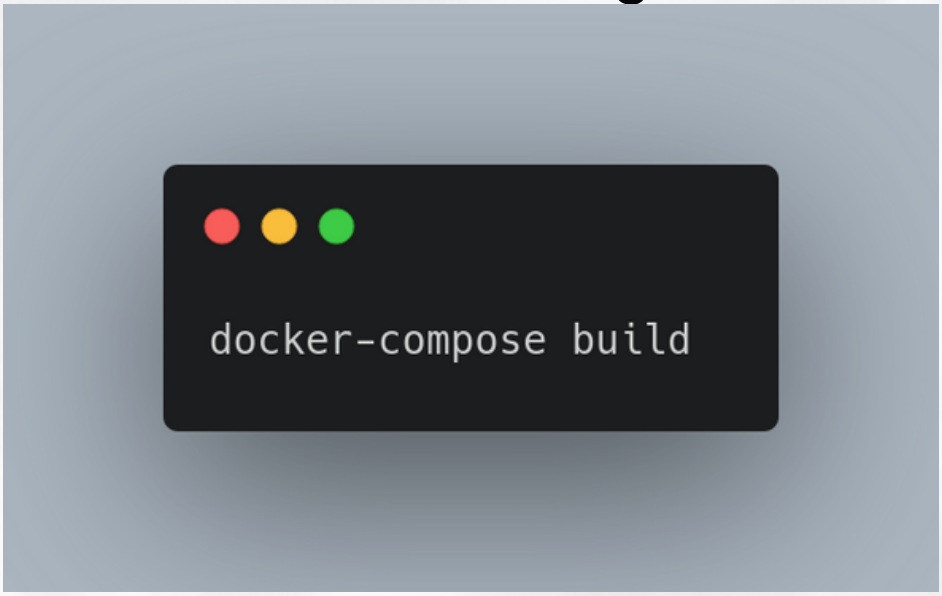
2. docker-compose.yml

Dockerfile M | docker-compose.yml

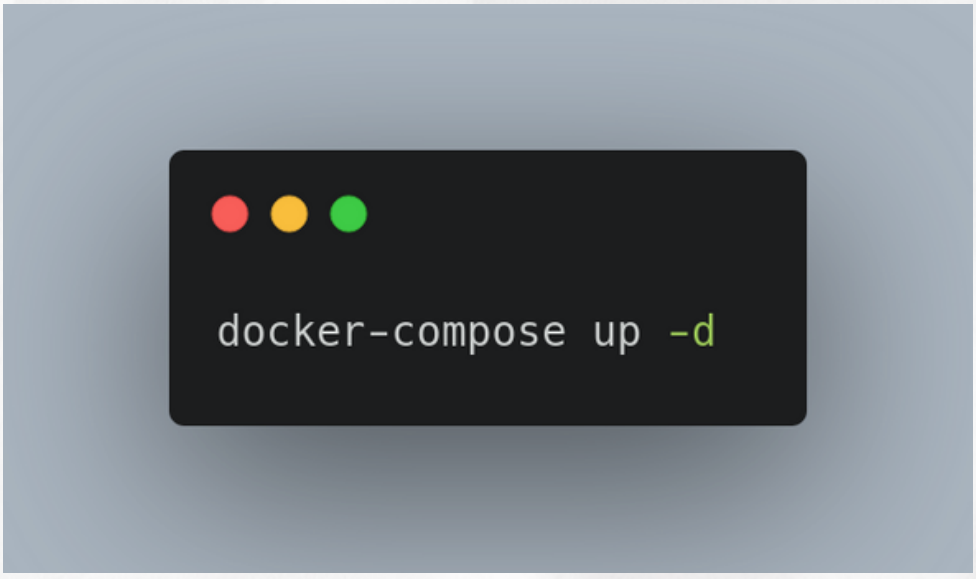
```
1  services:
2    web-server:
3      build:
4        context: .
5        dockerfile: Dockerfile
6      volumes:
7        - .:/app
8      ports:
9        - '80:80'
```

EJERCICIOS

3. Vamos a crear la imagen

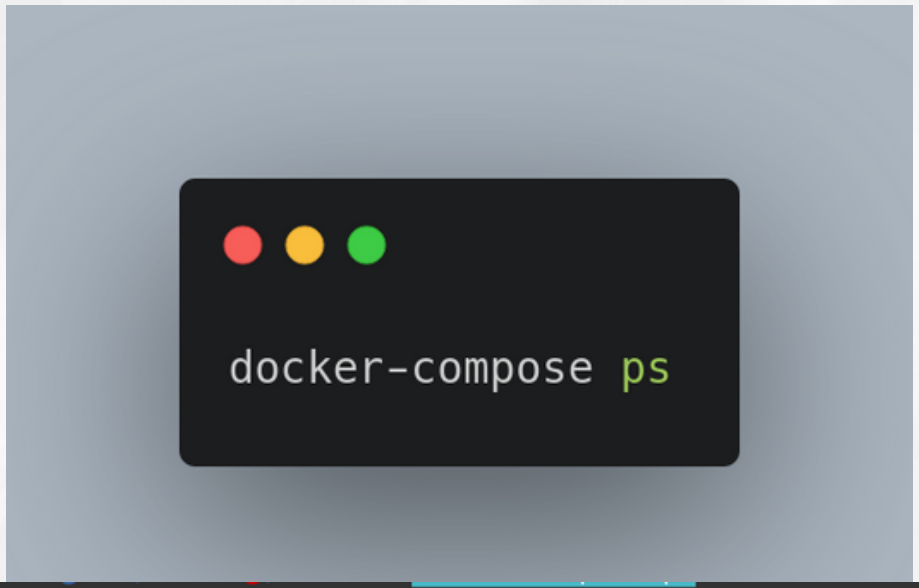


4. Vamos a levantar el contenedor



EJERCICIOS

5. revisamos que el contenedor este levantado




```
        Name                      Command               State          Ports
-------------------------------------------------------------------------------
my-movie-app-c9_web-      uvicorn main:app              Up      0.0.0.0:80->80/tcp,:::
server_1                  --host 0. ...                         80->80/tcp
```

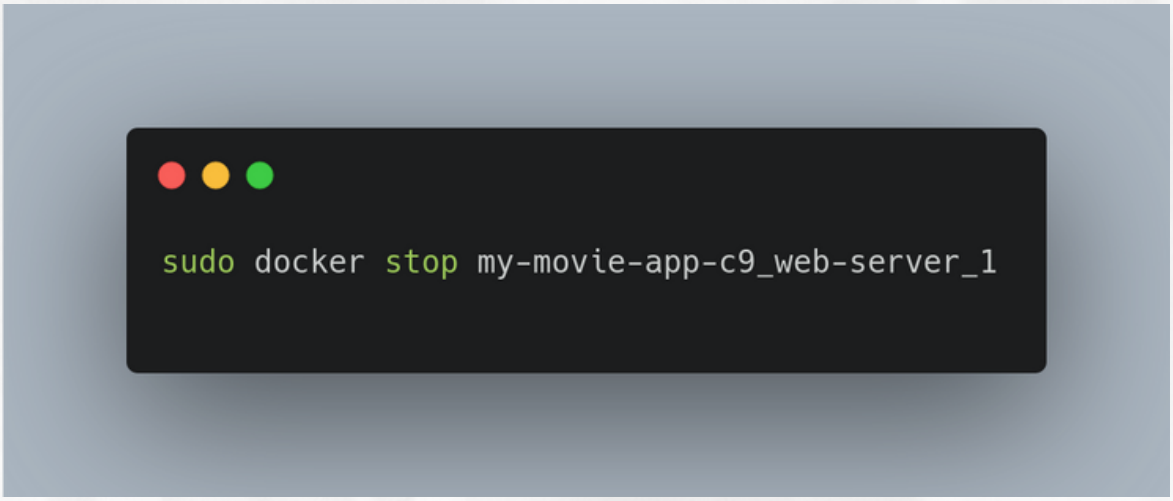
EJERCICIOS

6. Vamos a mirar el servidor

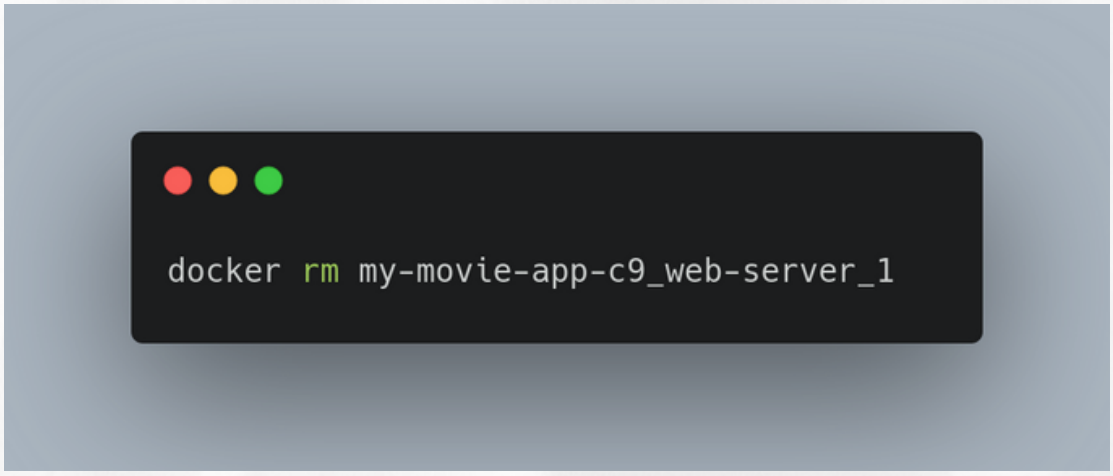
En este caso tenemos una aplicación de fastAPI

EJERCICIOS

7. Vamos a detener el contenedor



8. Vamos a eliminar el contenedor



Recuerda detener los contenedores que no estas usando por que gastan capacidad en tu PC

PROYECTO A CONSTRUIR

Conéctate las clases y trabajemos juntos:

Sesión 1

Sesión 2

```
fromTime = 0
toTime = 150
animLength = toTime - fromTime + 1

# prompt user for directory
filePath = c4d.storage.SaveDialog()
filePath, objName = os.path.split(filePath)
objName = objName + "_"
filePath = filePath + "\\"
# Check for confirmation
questionDialogText = "Obj Sequence will be saved as:\n\n"\
    "" + filePath + objName + "####.obj\n\n"\
    "from frame " + str(fromTime) + " to " + str(toTime) + " for " + str(animLength) + " frames.\n
proceedBool = c4d.gui.QuestionDialog(questionDialogText)

if proceedBool == True:

    # Loop through animation and export frames
    for x in range(0,animLength):

        # change frame, redraw view
        moveTime = c4d.BaseTime(fromTime,docFps) + c4d.BaseTime(x,docFps)
        doc.SetTime(moveTime)
        c4d.EventAdd(c4d.EVENT_FORCEREDRAW)
        c4d.DrawViews(c4d.DRAWFLAGS_FORCEFULLREDRAW)

        # progress bar
        c4d.StatusSetText("Exporting " + str(x) + " of " + str(animLength))
        c4d.StatusSetBar(100.0*x/animLength)

        # add buffer 0001
        bufferedNumber = str(doc.GetTime().GetFrame(docFps))
        if len(bufferedNumber)
```

Conoce nuestros programas:

https://educamas.com.co/

Fondo evolución digital. | Olumni | Fundación EducaMás | Prográmate Academy | SIMPLON.co