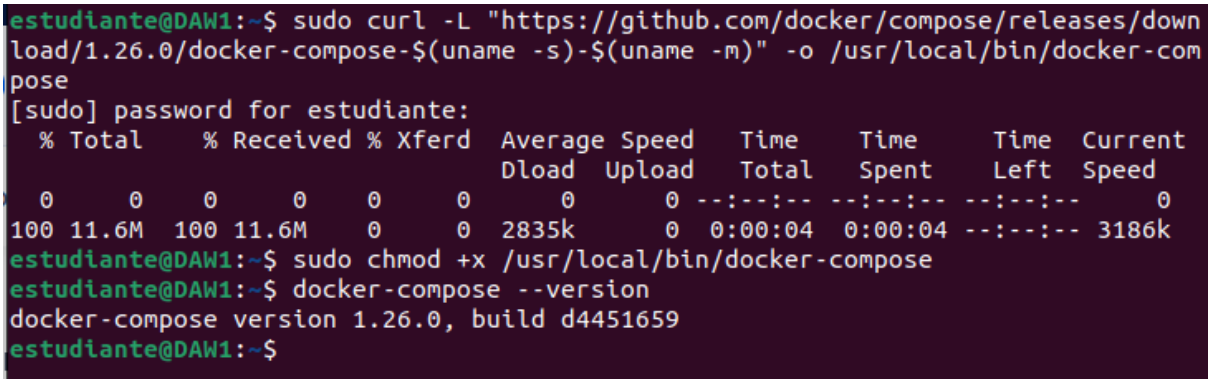
**1. ¿Que es Docker Compose? ¿Qué utilidades tiene? Instálalo y comprueba que está instalado**

Docker Compose es una herramienta para definir y ejecutar aplicaciones de Docker de varios contenedores.

En Compose, se usa un archivo **YAML** para configurar los servicios de la aplicación. Después, con un solo comando, se crean y se inician todos los servicios de la configuración.

Docker Compose se encargará de ejecutar todas las acciones necesarias para mantener ese estado




**2. Haz un glosario de comandos de Docker Compose**

Siempre que usemos los comandos de docker-compose debemos hacerlo en el mismo directorio que el archivo docker-compose.yml}]

- **docker-compose up :** levantar toda la aplicación (todos los contenedores que contiene)

- **docker-compose start contenedor :** levantar un contenedor concreto

- **docker-compose stop contenedor** : parar un contenedor concreto

- **docker-compose down contenedor** : para frenar y eliminar contenedores

- **docker-compose ps** : lista los contenedores que están corriendo

[

- **docker-compose –versión :** comprobar la versión instalada de docker compose

- **docker-compose build** : generar la imagen

**3. ¿Qué es un fichero YAML? ¿Qué estructura tiene? Investiga y detállalo. Haz también un glosario de YAML.**

Este formato de serialización de datos se encarga de almacenar archivos de configuración y se puede usar junto con todos los lenguajes de programación. **YAML** no propone etiquetas solo formato e identificadores mínimos, centrándose realmente en los datos.

Cada **YAML** comienza con --- que denota el inicio de un archivo YAML.

Los strings no deben estar entre comillas, aunque también es válido.

Los strings de líneas múltiples pueden ser escritos como un 'bloque literal' (usando pipes |) o como un 'bloque doblado' (usando >)

La indentación se usa para anidar elementos:

```
un_mapa_indentado:
    llave: valor
    otra_llave: otro valor
    otro_mapa_indentado:
        llave_interna: valor_interno
```

Las llaves de los mapas no requieren ser strings necesariamente.

```
0.25: una llave numérica
```

Las colecciones en YAML usan la indentación para delimitar el alcance y cada elemento de la colección inicia en su propia línea.

Las secuencias (equivalentes a listas o arreglos) se ven así:

```
- Amarillo
- Verde
- Azul
```

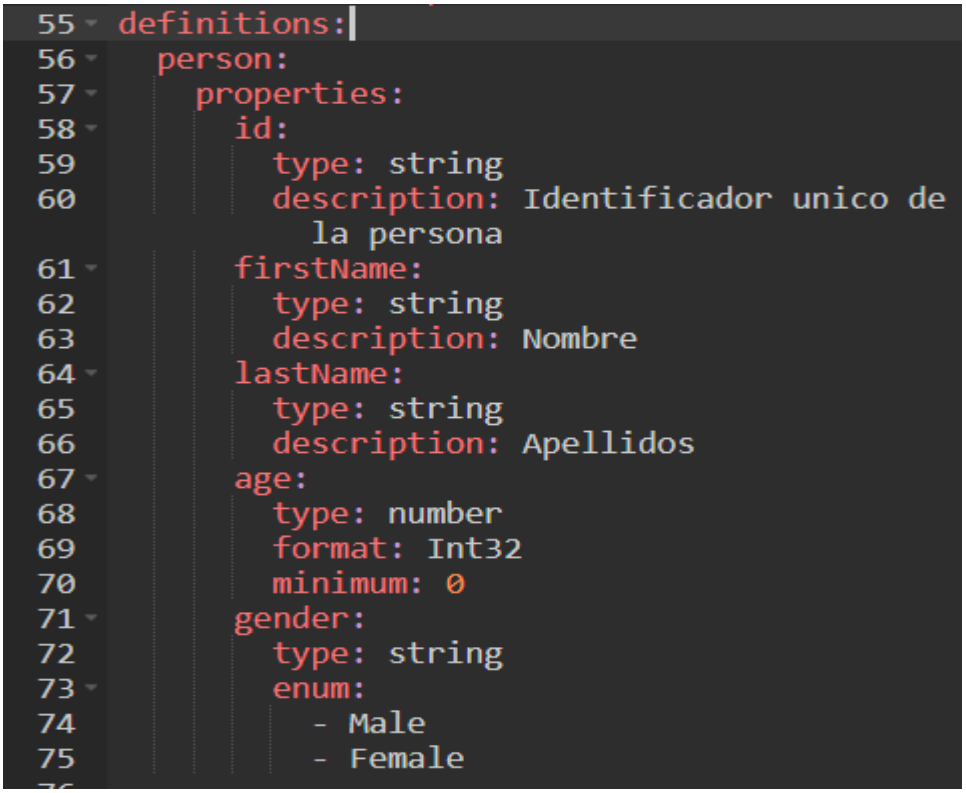
Las secuencias pueden tener distintos tipos en su contenido.

```
secuencia_combinada:
    - texto
    - 5
    - 0.6
    - llave: valor
   -
        - Esta es una secuencia
        - ...dentro de otra secuencia
```

Dado que todo JSON está incluído dentro de YAML, también puedes escribir mapas con la sintaxis de JSON y secuencias:

```
mapa_de_json_1: {"llave": "valor"}
mapa_de_json_2:
  llave: valor
```

Ejemplos:



```
definitions:
  person:
    properties:
      id:
        type: string
        description: Identificador unico de
          la persona
      firstName:
        type: string
        description: Nombre
      lastName:
        type: string
        description: Apellidos
      age:
        type: number
        format: Int32
        minimum: 0
      gender:
        type: string
        enum:
          - Male
          - Female
```

```
---

MALE: FALSE

GPA: 3.61

ISSUES: NULL

NAME: “BIGYAN”

AGE: 16
```

**4. Crea un archivo de configuración para una aplicación que contiene un único servicio:**

**- Imagen: httpd:2.4 (del servidor web Apache):**

**steps:**

- **name**: 'gcr.io/httpd:2.4r/Apache''

**- Puerto:** el host de Docker publicará el puerto 80 y hará una redirección con el puerto 80 del contenedor.

**- Bind mount:** lo creamos entre el directorio actual del host de Docker y el directorio *usr*/local/apache2/htdocs/ del contenedor (que es el directorio que utiliza el servidor web para servir el contenido que encuentre en su interior.

```
version: '3.7'
services:
  apache:
    image: httpd:2.4
    container_name: my-apache-app
    ports:
    - '8080:80'
    volumes:
    - ./website:/usr/local/apache2/htdocs
```

**5. Consulta la lista de contenedores que están en ejecución y explica la salida. ¿Qué diferencia hay entre “docker ps y docker compose ps”?**

```
estudiante@DAW1:~/compose-demo$ docker-compose ps
    Name              Command          State                  Ports
-------------------------------------------------------------------------------
my-apache-app   httpd-foreground   Up      0.0.0.0:8080->80/tcp,:::8080->80/tcp
estudiante@DAW1:~/compose-demo$
```

**Name:** Se refiere al nombre de nuestra app.
**Command:** Nos indica el tipo de servidor web que estamos utilizando en este caso utilizamos httpd de apache.
**State:** Nos indica el estado de nuestra app.
**Ports:** Nos aparece los puertos utilizados y los redireccionamientos del mismo.

**docker ps** enumera todos los contenedores en ejecución en el motor docker.
**docker-compose ps** enumera los contenedores relacionados con las imágenes declaradas en **docker-compose file.**

**6. Muestra la salida de docker-compose.yml**

```yaml
version: '3.7'
services:
  apache:
    image: httpd:2.4
    container_name: my-apache-app
    ports:
    - '8080:80'
    volumes:
    - ./website:/usr/local/apache2/htdocs
```

**7. Visiona el video relativo a docker compose del módulo 7.4**