Label Studio con Docker en Windows

Guía enfocada en la instalación en sistemas con Windows como sistema operativo. El proceso es muy similar en Mac y Linux, pero algunos comandos pueden variar un poco. En Mac también hay un Docker Desktop, pero en Linux se lleva totalmente desde la terminal.

Hay varias maneras de instalar Label Studio, pero la más limpia y sencilla es con Docker, por eso es la que se describe en esta guía. Para ver otras maneras de instalación se puede recurrir a la documentación oficial (https://labelstud.io/guide/install.html).

Paso 1: Instalar Docker en Windows

Para instalar Docker en Windows, tenemos que ir al siguiente enlace:

https://docs.docker.com/desktop/setup/install/windows-install/

Descargamos esta versión:

Docker Desktop for Windows - x86_64

(también se puede descargar desde Microsoft Store, pero aquí lo haremos con el exe)

Ejecutamos el exe. Si en algún momento nos da a elegir, elegiremos WSL 2 en vez de Hyper-V, pero aparentemente por defecto ya siempre funciona con WSL 2. Nos hará hacer log out y log in cuando termine la instalación.

Al volver a iniciar sesión nos saltará con la ventana de Docker.

Aceptamos los términos y condiciones.

No hace falta que pongamos correo, le damos a skip.

Ya tenemos Docker. Necesitaremos tener Docker ejecutándose en segundo plano. Podemos ver los programas ejecutándose en segundo plano en la barra de tareas, abajo del todo a la derecha. Pulsamos la flechita desplegable hacia arriba y nos saldrá el icono de Docker. Seguramente se pondrá automática en modo aplicaciones de inicio, así que cada vez que reiniciemos el ordenador, se lanzará automáticamente. Se puede cambiar este comportamiento en ajustes/apps/startup (o inicio).

Si no está el icono de la ballenita, simplemente iniciamos la aplicación Docker. Incluso si luego la cerramos normalmente, seguirá la ballenita en la barra de tareas, lo que significa que sigue ejecutándose en segundo plano. Si la queremos cerrar de segundo plano, es darle con el clic derecho y cerrar.

Docker desktop es una manera visual de trabajar con Docker. Os puede ser útil, pero realmente todo lo de Docker se puede trabajar desde la terminal.

POWERSHELL

Usaremos **PowerShell** en vez de CMD (Command Prompt o Símbolo del sistema).

Esto garantiza compatibilidad total con los comandos que podemos necesitar usar.

Podemos buscar PowerShell en la barrita de búsqueda. Recomiendo fijarlo en la barra de tareas.

Dejo aquí algunos comandos útiles relacionados con Docker que podemos escribir en la PowerShell:

docker --version

Nos dirá qué versión tenemos de Docker, y es una manera de comprobar que todo funciona correctamente.

docker ps

Nos muestra los contenedores que tenemos ejecutándose ahora mismo. Cada contenedor tiene asociado un proceso con un id. Esto nos ayuda a identificarlo y poder detenerlo manualmente.

docker ps -a

Nos enseña todos los contenedores, incluso los que están detenidos.

docker stop <ID_o_Nombre>

Sustituimos lo que hay entre el menor que y mayor que (incluyendo los símbolos) por el id que nos sale asociado al contenedor. También podemos usar el nombre que tenga asociado en vez. Al crear un contenedor, se le puede asociar un nombre.

docker start <ID_o_Nombre>

Sirve para arrancar un contenedor detenido (que vemos con *docker ps -a*). Aplica lo mismo del id o nombre.

docker rm <ID_o_Nombre>

Con esto borramos un contenedor en caso de que no tenga autoborrado. Hay que detenerlo primero, ya que no se pueden borrar contenedores en ejecución.

docker images

Sirve para listar todas las imágenes. Las imágenes son como "recetas" o "instrucciones", mientras que un contenedor es aplicar esa receta para generar un "plato", o aplicación tangible. Se podría decir que la imagen es el archivo de instalación, y el contenedor es ya la aplicación instalada en el dispositivo.

docker pull <imagen>

Este comando sirve para descargar una imagen de su repositorio. No la ejecuta, solo la descarga. Esto no crea su contenedor.

docker rmi <ID_imagen>

Se usa para borrar una imagen que ya no queramos. Al listar las imágenes nos sale su id, que usamos para borrarla.

Paso 2: Instalar Label Studio

Aquí tenemos la web del Docker de Label Studio:

https://hub.docker.com/r/heartexlabs/label-studio

Lo primero, iniciamos **PowerShell**. Vamos a escribir lo siguiente:

pwd

Esto nos dirá en qué ubicación estamos. Parece que normalmente nos dejan en nuestro usuario, algo similar a esto:

C:\Users\yanco

Podemos trabajar aquí mismo, es una buena ubicación. Lo primero, vamos a descargar la imagen. Escribimos:

docker pull heartexlabs/label-studio:latest

Esto nos descarga la imagen. Si ahora escribimos *docker images*, nos saldrá la imagen de Label Studio. Esta es una manera de verla, pero también podemos verla visualmente desde Docker desktop, en la sección de imágenes.

Para lanzar el contenedor, hay varias opciones adicionales. A mí me gusta lanzarlo con autoborrado, para que no se queden los contenedores almacenados cuando no están en uso. Si quitamos *--rm* del comando, no se autoborrará tras detener el docker. Con *--name* le estamos asignando nosotros un nombre arbitrario (label-studio en este caso) y no es obligatorio. La opción *-it* sirve para que se vean los logs en el PowerShell y podamos detenerlo cuando queramos. La opción más importante es *-v*, porque lo que hace es crear una carpeta con persistencia de datos, y lo que sigue le indica las carpetas que va a crear en la dirección actual, por eso hicimos *pwd* (print working directory), para asegurarnos de dónde estamos y dónde se va a crear la carpeta. Así, podemos cerrar y borrar el contenedor tranquilamente, porque todos nuestros datos del proyecto se almacenan en esa carpeta.

docker run --rm -it --name label-studio -p 8080:8080 -v ${pwd}/mydata:/label-studio/data heartexlabs/label-studio:latest

Tras ello, ya tenemos el contenedor/aplicación ejecutándose. Tenemos que mantener esa ventana del PowerShell sin cerrarse. La podemos minimizar. Nos enseña logs y con **control c** podemos detener el contenedor.

En la propia terminal nos da una dirección de localhost. La podemos copiar y pegar (cuidado, hay que usar **control + shit + c** para poder copiar cosas de PowerShell, porque **control c** tal cual recordemos que es para detener el contenedor (y en general detener procesos en la terminal). Hemos establecido nosotros la dirección de localhost, 8080. Entonces ponemos en el navegador la siguiente dirección (sirven ambas en teoría):

http://0.0.0.0:8080/

http://localhost:8080/

Esto debería dirigirnos automáticamente a la aplicación de Label Studio que estamos ejecutando localmente. No van los datos a ningún sitio, aunque sea un navegador web. Entonces no es relevante poner un correo real, nos lo podemos inventar, es solo para manejar los usuarios internamente y siempre de manera local y privada. Eso sí, conviene acordarse de nuestras credenciales.

Ahora tenemos una nueva carpeta creada, que se llama **mydata**. Estará en el directorio donde hayamos ejecutado la aplicación. Lo vimos con *pwd*

(C:\Users\yanco). Esta carpeta persiste. Si volvemos a ejecutar el mismo Docker con la opción que vimos arriba, si ya existe la carpeta, no tiene que volver a crearla, y seguimos conservando todos nuestros datos.

Cuando ya hayamos terminado de usar la aplicación, vamos a la terminal de PowerShell que teníamos minimizada, en la que lanzamos el contenedor, y le damos a **control c**. Con esto ya se detiene el programa, pero se conserva la carpeta de trabajo con todos nuestros archivos. *Rinse and repeat*.

Permisos:

Si hay error estilo "Access Denied", es un tema de permisos. Podemos ir a Docker Desktop/ajustes/fuentes (resources en inglés). Aquí, en Compartir archivos (*File sharing*), comprobamos si nuestra carpeta está incluida. Si no lo está, o escribimos manualmente la dirección de nuestra carpeta (C:\Users\yanco\mydata), o le damos a navegar (*browse*) para encontrarla. Luego le damos al botón "+" de la derecha, y aplicar (*Apply*) abajo a la derecha.

Archivo bat:

Para que no tengáis que estar escribiendo manualmente en la PowerShell el comando para ejecutar el contenedor de Label Studio, lo podéis meter en un archivo **.bat** (Windows), que con doble clic sobre él se ejecuta. Para Mac o Linux, se puede hacer con un archivo **.sh**. Si le dais el comando a vuestra IA de confianza, os lo generará. Especificadle dónde lo queréis. En esta guía lo estamos haciendo tal cual en la carpeta del usuario. Os adjunto un archivo 1_**label-studio.bat** para que lo podáis ejecutar fácilmente (primero hace *pull* de la imagen, si ya está disponible no se descarga de nuevo; y luego ya corre el contenedor). O sea, si ya tenéis instalado Docker y ejecutándose en segundo plano, se puede usar directamente el archivo bat, pero conviene aprender un poco cómo funciona.

Paso 3: Instalar Backend (avanzado)

Para poder conectar nuestra aplicación con modelos para hacer preanotación, tenemos que descargarnos el repositorio oficial. Podemos hacerlo de dos formas, pero ya que estamos, hacemos la versión avanzada.

Forma 1:

Lo primero, necesitamos tener instalado git en el sistema. Para ello, nos iremos a la siguiente dirección:

https://git-scm.com/install/windows

Nos descargamos el primero que sale, “Click here to download”, en la pestaña de Windows. Le daremos al archivo de instalación. Todas las opciones por defecto están correctas, le damos a todo que sí. Ya tenemos git en nuestro sistema. Si teníamos la PowerShell abierta, la cerramos para que se apliquen los cambios.

Abrimos el PowerShell. Le damos a *pwd* para saber dónde estamos. Si estamos en el mismo que la última vez, es un buen sitio para descargarlo. Usaremos el siguiente comando:

git clone https://github.com/HumanSignal/label-studio-ml-backend

Ya tenemos la carpeta que contiene todo lo necesario para hacer preanotación con modelos. Estará en la misma ubicación que **mydata**. Se debería llamar así la carpeta:

label-studio-ml-backend

Forma 2:

Vamos a esta dirección:

https://github.com/HumanSignal/label-studio-ml-backend

Aquí le damos al botón verde que pone **<> Code** que tiene una flechita hacia abajo. Pinchamos, y le damos a “Download ZIP”. Lo llevamos a la ubicación que queramos de manera normal (navegando visualmente por las carpetas), pero recomiendo la misma de antes (C:\Users\yanco). Lo descomprimimos y ya la tenemos. Realmente ambos métodos son indistintos.

Paso 4: Conectar modelos (avanzado)

Tesseract:

El dataset de ejemplo que vamos a utilizar en el taller se encuentra en el siguiente enlace a Zenodo:

https://zenodo.org/records/17902212

Se trata de imágenes de documentos históricos en español de entre los siglos XVI y XIX, si no recuerdo mal. Este tipo de textos suelen dar problemas con OCR convencionales, así que conviene manejar las expectativas usando Tesseract tal cual.

Para esta sección, he utilizado de referencia el siguiente ejemplo de la documentación de la web oficial:

https://labelstud.io/guide/ml_tutorials/tesseract#Interactive-bounding-boxes-OCR-using-Tesseract

Debemos tener activado en segundo plano Docker, y ya tener ejecutándose Label Studio. Iniciamos un proyecto y ponemos los datos pertinentes. Del dataset de prueba que nos hemos descargado antes, en la carpeta Los101-Gresel/PastReader-Gresel/ground_truth_data/jpg, importamos, por ejemplo, 5 imágenes. En **Labeling Setup**, le damos a **Custom template**. Aquí, borramos lo que hay en la cajita y pegamos lo siguiente:

```
<View>
  <Image name="image" value="$ocr" zoom="true" zoomControl="false"
      rotateControl="true" width="100%" height="100%"
      maxHeight="auto" maxWidth="auto"/>

  <RectangleLabels name="bbox" toName="image" strokeWidth="1" smart="true">
    <Label value="Label1" background="green"/>
    <Label value="Label2" background="blue"/>
    <Label value="Label3" background="red"/>
  </RectangleLabels>

  <TextArea name="transcription" toName="image"
  editable="true" perRegion="true" required="false"
  maxSubmissions="1" rows="5" placeholder="Recognized Text"
  displayMode="region-list"/>
</View>
```

Tras ello, le daremos a guardar. Ahora, tenemos que ir Label Studio, y clicar a las rayitas de arriba a la izquierda. Vamos a **Organization**. Aquí, arriba a la derecha está la opción **API Tokens Settings**. Desactivamos Personal **Access Tokens**, y activamos **Legacy Tokens**.

A continuación, pulsamos arriba a la derecha, en nuestro perfil, y le pulsamos en **Account & Settings**. Clicamos en Legacy Token, y copiamos el Access Token (un código de 40 caracteres). Tiene este aspecto:

2e83ffff1a522a73db30695affd8ec907cce6a05

Ahora vamos a la siguiente dirección con el propio explorador de archivos visual. Está dentro de la carpeta de ML backend que nos descargamos anteriormente:

C:\Users\yanco\label-studio-ml-backend\label_studio_ml\examples\tesseract

Aquí, tenemos que modificar el archivo llamado:

docker-compose.yml

Borramos el contenido de ese archivo y pegamos lo siguiente:

```
version: "3.8"

services:
 tesseract:
  container_name: tesseract
  image: heartexlabs/label-studio-ml-backend:tesseract-master
  init: true
  build:
   context: .
  ports:
   - 9090:9090
  volumes:
   - "./data/server:/data"
  env_file:
   - .env
  extra_hosts:
   - "host.docker.internal:host-gateway"
```

Además, crearemos en la misma ubicación un archivo que se tiene que llamar:

.env

Este tipo de archivos que empiezan por punto son invisibles en el Explorador de Archivos por defecto. Para poder verlos, le damos a **vista**, **mostrar**, y clicamos en **elementos ocultos** (*hidden ítems*). Podemos darle a clic derecho en esa carpeta, darle **nuevo**, y a **documento de texto**. Le ponemos el nombre **.env**, y abrimos el archivo. Copiamos lo siguiente, y pondremos nuestro token que aparece en Label Studio:

LABEL_STUDIO_HOST=http://host.docker.internal:8080
LABEL_STUDIO_API_KEY=**2e83ffff1a522a73db30695affd8ec907cce6a05**
LABEL_STUDIO_ACCESS_TOKEN=**2e83ffff1a522a73db30695affd8ec907cce6a05**
***¡¡Sustituir esos números por el token que obtuvimos previamente!!**

Tras ello, abrimos la PowerShell, y nos vamos a la misma ubicación:

cd label-studio-ml-backend/label_studio_ml/examples/tesseract

Una vez aquí, introducimos el siguiente comando:

docker compose up

Ahora tendremos corriendo Tesseract en segundo plano en esa terminal de PowerShell. Podemos minimizarla.

A continuación, nos vamos a nuestro proyecto de Label Studio, en el menú donde podemos ver las imágenes que tenemos subidas. Vamos a **Settings**, arriba a la derecha. Luego, le damos a **Model**. Aquí le damos a **Connect Model**. Lo podemos llamar Tesseract, por ejemplo. En Backend URL pondremos la siguiente dirección que apunta hacia nuestro modelo en local que tenemos ejecutándose:

http://host.docker.internal:9090

Le damos a **Validate and save**. Veremos que nos aparece ahí como conectado. Le damos a **save**.

Volvemos a nuestro proyecto, elegimos cualquier imagen, y veremos que ya preanota el OCR al elegir cajitas.