1. Introducción:

Un Dockerfile es un archivo de configuración en formato de texto plano utilizado para definir los pasos para construir una imagen de Docker. Es un componente crítico del ecosistema de Docker, que permite a los desarrolladores automatizar el proceso de creación de imágenes de contenedor reproducibles y consistentes. Los Dockerfiles se utilizan para encapsular la configuración y dependencias de una aplicación o servicio, lo que facilita empaquetar, distribuir e implementar software en forma de contenedores.

El Dockerfile incluye típicamente una serie de instrucciones, cada una en una línea separada, que dictan cómo debe construirse la imagen. El motor de Docker interpreta estas instrucciones y construye la imagen capa por capa. Cuando se crea un nuevo contenedor a partir de la imagen, el sistema de archivos se construye en base a estas capas, lo que lo hace eficiente y rápido.

A continuación se muestra un ejemplo de un Dockerfile para crear un servidor de Jupyter Notebook utilizando la imagen oficial de Jupyter Docker Stacks:

In [1]:
```
!cat Dockerfile
```

```
FROM jupyter/base-notebook:latest

WORKDIR /home/jovyan/work

COPY . /home/jovyan/work

EXPOSE 8889

RUN pip install pandas matplotlib

CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8889","--no-browser"]
```

2. Componentes clave de un Dockerfile:

- **2.1 Imagen Base:**

El Dockerfile comienza especificando la imagen base sobre la cual se construirá la imagen personalizada. La imagen base es el punto de partida para la imagen de tu aplicación y contiene el sistema operativo y otro software o bibliotecas preinstaladas. La imagen base se define utilizando la instrucción FROM. Por ejemplo:

In [ ]:
```
FROM jupyter/minimal-notebook
```

- **2.2 Configuración del Entorno:**

El Dockerfile puede incluir configuraciones del entorno que definan variables, especifiquen directorios de trabajo y establezcan otros comportamientos en tiempo de ejecución. La

In [ ]:
```
ENV NODE_ENV=production
```

- **2.3 Copiar Archivos:**

La instrucción COPY te permite copiar archivos y directorios desde el sistema anfitrión al sistema de archivos de la imagen. Este paso es esencial para agregar el código de tu aplicación y cualquier recurso necesario a la imagen. Por ejemplo:

In [ ]:
```
COPY . /app
```

- **2.4 Ejecución de Comandos:**

La instrucción RUN se utiliza para ejecutar comandos dentro del sistema de archivos de la imagen. Esto se utiliza comúnmente para instalar paquetes, dependencias y otro software necesario para que tu aplicación funcione.

Por ejemplo, para una aplicación Node.js, podrías ejecutar el siguiente comando para instalar las dependencias de Node.js:

In [ ]:
```
RUN npm install
```

- **2.5 Exposición de Puertos:**

La instrucción EXPOSE se utiliza para especificar en qué puertos debe escuchar el contenedor. No publica los puertos en realidad, sino que sirve como documentación para los desarrolladores y usuarios de la imagen.

Por ejemplo, para indicar que el contenedor escucha en el puerto 3000, agregarías la siguiente línea:

In [ ]:
```
EXPOSE 3000
```

- **2.6 Comando de Ejecución del Contenedor:**

La instrucción ***CMD*** o ***ENTRYPOINT*** define el comando que se debe ejecutar cuando se inicia un contenedor a partir de la imagen. Por lo general, este es el proceso principal de tu aplicación.

Por ejemplo, para una aplicación Node.js, podrías usar:

In [ ]:
```
CMD ["node", "app.js"]
```

Una vez que hayas definido tu Dockerfile, puedes usar el comando ***docker build*** para construir la imagen de Docker basada en las instrucciones proporcionadas en el archivo. Esto creará una imagen de contenedor que contiene todas las configuraciones, dependencias y código

especificados necesarios para ejecutar tu aplicación.

Los Dockerfiles son una herramienta poderosa para crear imágenes de contenedor consistentes y reproducibles. Permiten a los desarrolladores definir el entorno exacto necesario para sus aplicaciones, lo que facilita compartir y distribuir el software en diferentes entornos y sistemas.

3. Crear un Dockerfile paso a paso:

Vamos a crear un Dockerfile paso a paso para un servidor de Jupyter Notebook utilizando la imagen oficial de Jupyter Docker Stacks:

- **3.1: Elegir la Imagen Base**

Utilizaremos la imagen oficial de Jupyter Notebook de Jupyter Docker Stacks como imagen base. Esta imagen incluye la instalación básica de Jupyter Notebook. Elige la etiqueta (tag) de la imagen base según tus requisitos. En este ejemplo, utilizaremos la etiqueta ***base-notebook***.

In [ ]:
```
FROM jupyter/base-notebook:latest
```

- **3.2: Establecer el Directorio de Trabajo**

Es una buena práctica establecer el directorio de trabajo dentro del contenedor donde se ubicarán tus archivos de Jupyter Notebook. Lo estableceremos en ***/home/jovyan/work***, que es el directorio predeterminado para los archivos de Jupyter Notebook en la imagen base.

In [ ]:
```
WORKDIR /home/jovyan/work
```

- **3.3: Copiar Archivos de Notebooks**

A continuación, copiaremos los archivos de Jupyter Notebook desde el directorio local (donde se encuentra el Dockerfile) al directorio de trabajo del contenedor. Asegúrate de que tus archivos de Jupyter Notebook (por ejemplo, archivos ***.ipynb***) estén presentes en el mismo directorio que el Dockerfile.

In [ ]:
```
COPY . /home/jovyan/work
```

- **3.4: Exponer Puerto**

El Jupyter Notebook se ejecuta de forma predeterminada en el puerto 8888, pero puede haber casos en los que se use este puerto, por lo que lo cambiaremos a 8889. Para acceder a él desde la máquina anfitriona, necesitamos exponer este puerto desde el contenedor.

In [ ]:
```
EXPOSE 8889
```

- **3.5: Instalar Paquetes Adicionales (Opcional)**

Si tus Jupyter Notebooks requieren paquetes o bibliotecas adicionales de Python, puedes instalarlos utilizando pip o conda. En este ejemplo, utilizaremos pip para instalar pandas y matplotlib

In [ ]:
```
RUN pip install pandas matplotlib
```

- **3.6: Iniciar Servidor de Jupyter Notebook**

Finalmente, agregaremos el comando para iniciar el servidor de Jupyter Notebook cuando se ejecute el contenedor. La opción --ip=0.0.0.0 permite el acceso al servidor desde todas las direcciones IP, y --no-browser evita que Jupyter intente abrir un navegador dentro del contenedor.

In [ ]:
```
CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8889","--no-browser"]
```

El Dockerfile completo se verá así:

In [6]:
```
!cat Dockerfile
```

```
FROM jupyter/base-notebook:latest

WORKDIR /home/jovyan/work

COPY . /home/jovyan/work

EXPOSE 8889

RUN pip install pandas matplotlib

CMD ["jupyter", "notebook", "--ip=0.0.0.0", "--port=8889","--no-browser"]
```

To build the Docker image, save the above code into a file named Dockerfile in the same directory as your Jupyter Notebook files. Then, open a terminal or command prompt, navigate to the directory containing the Dockerfile, and run the following command:

In [7]:
```
!docker build -t jupyter_server .
```

```
[+] Building 0.0s (0/1)
[+] Building 0.1s (2/3)
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 37B
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [internal] load metadata for docker.io/jupyter/base-notebook:latest
0.0s
[+] Building 0.3s (7/8)
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 37B
0.0s
 => [internal] load .dockerignore
0.0s
```

The build process will create a Docker image with your Jupyter Notebook server configuration.

To run the Jupyter Notebook server in a container based on the image you just built, use the following command:

In [8]:
```
!docker run -p 8889:8889 jupyter_server
```

```
[I 2023-08-12 19:19:48.106 ServerApp] Package notebook took 0.0000s to import
[I 2023-08-12 19:19:48.131 ServerApp] Package jupyter_lsp took 0.0172s to imp
ort
[W 2023-08-12 19:19:48.131 ServerApp] A `_jupyter_server_extension_points` fu
nction was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_pa
ths` function was found and will be used for now. This function name will be
deprecated in future releases of Jupyter Server.
[I 2023-08-12 19:19:48.140 ServerApp] Package jupyter_server_terminals took
0.0086s to import
[I 2023-08-12 19:19:48.141 ServerApp] Package jupyterlab took 0.0000s to impo
rt
[I 2023-08-12 19:19:48.705 ServerApp] Package nbclassic took 0.0019s to impor
t
[W 2023-08-12 19:19:48.707 ServerApp] A `_jupyter_server_extension_points` fu
nction was not found in nbclassic. Instead, a `_jupyter_server_extension_path
s` function was found and will be used for now. This function name will be de
precated in future releases of Jupyter Server.
[I 2023-08-12 19:19:48.707 ServerApp] Package notebook_shim took 0.0000s to i
mport
[W 2023-08-12 19:19:48.707 ServerApp] A `_jupyter_server_extension_points` fu
nction was not found in notebook_shim. Instead, a `_jupyter_server_extension_
paths` function was found and will be used for now. This function name will b
e deprecated in future releases of Jupyter Server.
[I 2023-08-12 19:19:48.708 ServerApp] jupyter_lsp | extension was successfull
y linked.
[I 2023-08-12 19:19:48.712 ServerApp] jupyter_server_terminals | extension wa
s successfully linked.
[I 2023-08-12 19:19:48.716 ServerApp] jupyterlab | extension was successfully
linked.
[I 2023-08-12 19:19:48.720 ServerApp] nbclassic | extension was successfully
linked.
[I 2023-08-12 19:19:48.724 ServerApp] notebook | extension was successfully l
inked.
[I 2023-08-12 19:19:48.725 ServerApp] Writing Jupyter server cookie secret to
/home/jovyan/.local/share/jupyter/runtime/jupyter_cookie_secret
[I 2023-08-12 19:19:48.943 ServerApp] notebook_shim | extension was successfu
lly linked.
[I 2023-08-12 19:19:48.971 ServerApp] notebook_shim | extension was successfu
lly loaded.
[I 2023-08-12 19:19:48.973 ServerApp] jupyter_lsp | extension was successfull
y loaded.
[I 2023-08-12 19:19:48.974 ServerApp] jupyter_server_terminals | extension wa
s successfully loaded.
[I 2023-08-12 19:19:48.976 LabApp] JupyterLab extension loaded from /opt/cond
a/lib/python3.11/site-packages/jupyterlab
[I 2023-08-12 19:19:48.976 LabApp] JupyterLab application directory is /opt/c
onda/share/jupyter/lab
[I 2023-08-12 19:19:48.977 LabApp] Extension Manager is 'pypi'.
[I 2023-08-12 19:19:48.980 ServerApp] jupyterlab | extension was successfully
loaded.
[I 2023-08-12 19:19:48.983 ServerApp] nbclassic | extension was successfully
loaded.
[I 2023-08-12 19:19:48.986 ServerApp] notebook | extension was successfully l
oaded.
[I 2023-08-12 19:19:48.986 ServerApp] Serving notebooks from local directory:
/home/jovyan/work
[I 2023-08-12 19:19:48.987 ServerApp] Jupyter Server 2.7.0 is running at:
```

```
[I 2023-08-12 19:19:48.987 ServerApp] http://6786521305f5:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e (http://6786521305f5:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e)
[I 2023-08-12 19:19:48.987 ServerApp]     http://127.0.0.1:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e (http://127.0.0.1:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e)
[I 2023-08-12 19:19:48.987 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2023-08-12 19:19:48.990 ServerApp]

    To access the server, open this file in a browser:
        file:///home/jovyan/.local/share/jupyter/runtime/jpserver-7-open.html
    Or copy and paste one of these URLs:
        http://6786521305f5:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e (http://6786521305f5:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e)
        http://127.0.0.1:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e (http://127.0.0.1:8889/tree?token=bf10af5a3f4bd15a0d617b26c87563066f6683d495949e4e)
[I 2023-08-12 19:19:49.564 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
^C


  _   _                 _       _
 | | | |_ __   __ _  __| | __ _| |_ ___
 | |_| | '_ \ / _` |/ _` |/ _` | __/ _ \
  \___/| .__/ \__,_|\__,_|\__,_|\__\___|
       |_|

Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions.

https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html (https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html)

Please note that updating to Notebook 7 might break some of your extensions.

[I 2023-08-12 19:22:22.457 ServerApp] Interrupted...
```

El **-p 8889:8889** mapea el puerto 8889 del contenedor al puerto 8889 de la máquina anfitriona. Luego, puedes acceder al servidor de Jupyter Notebook abriendo un navegador web y navegando a http://localhost:8889 (http://localhost:8889) en la máquina anfitriona. Verás la interfaz de Jupyter Notebook y podrás comenzar a trabajar con tus notebooks.

In [ ]: