BIG DATA

PROYECTO FINAL

Carlos Ramírez Martín

01- Configuración

New Technology School

Tokio.

Índice

Introducción

Presentación y Objetivos

Este documento detalla el proceso de configuración e instalación de las dependencias necesarias para el proyecto final **"Big Data: Arquitectura y Análisis de Datos"**. Además, incluye una descripción de las decisiones tomadas y los estudios paralelos realizados con el objetivo de implementar la solución más óptima.

El proyecto se centra en el análisis de un conjunto de datos relacionados con el tráfico aéreo en el aeropuerto de San Francisco, California.

Para obtener una comprensión completa del proyecto, se recomienda explorar la siguiente documentación:

- **02 - Técnico:** *02_tecnico_carlos_ramirez_martin.pdf*
  - Documento técnico donde se recoge el código fuente empleado para la resolución de cada una de las tareas solicitadas.

- **03 - Científico:** *03_cientifico_carlos_ramirez_martin.pdf*
  - Informe en el que se transmiten los resultados de los análisis realizados y las conclusiones pertinentes.

- **04 - Presentación:** *04_presentacion_carlos_ramirez_martin.pdf*
  - Presentación que contiene los resultados del proyecto usando técnicas de storytelling.

Entorno de Desarrollo - Cassandra

Implementación de Docker

En el desarrollo y despliegue de nuestro proyecto, nos enfrentamos a la decisión de cómo implementar y gestionar nuestra base de datos NoSQL, Cassandra. En este contexto, la elección de la tecnología adecuada puede tener un impacto significativo en la consistencia, portabilidad y escalabilidad de nuestro entorno. En este sentido, proponemos utilizar Docker con un contenedor de Cassandra en lugar de alternativas en línea, como el servicio web proporcionado por *killercoda.com.*

A continuación, destacamos algunas de las razones fundamentales que respaldan esta elección:

- **Consistencia del Entorno:**
  - Docker garantiza un entorno de ejecución consistente en todos los sistemas compatibles con Docker, independientemente de las variaciones en el hardware o el sistema operativo. Esto elimina posibles conflictos y asegura la reproducibilidad del entorno de desarrollo y despliegue.

- **Aislamiento de Recursos:**
  - La encapsulación de Cassandra dentro de un contenedor Docker proporciona aislamiento de recursos, evitando conflictos con otros servicios o aplicaciones que puedan estar en ejecución en la máquina. Esto es esencial para prevenir interferencias y mantener un ambiente de ejecución predecible.

- **Control de Versiones:**
  - Docker facilita la gestión de versiones, permitiendo la especificación exacta de la versión de Cassandra a utilizar. Esto mejora la consistencia y simplifica la gestión de dependencias, asegurando que el proyecto funcione de manera coherente en diferentes entornos y en el futuro.

Es importante tener en cuenta que el uso gratuito de *killercoda.com* está limitado a una hora, después de la cual, los servidores en línea se reinician, resultando en la pérdida de datos. Es crucial destacar que la base de datos no es persistente y no se puede transferir a la máquina que el cuerpo docente de Tokio School utilizará para evaluar el funcionamiento del proyecto. Esta limitación podría afectar la continuidad y la accesibilidad de los datos, siendo necesario considerar alternativas más robustas y permanentes para garantizar la integridad de la información.

Instalación de Docker en macOs

Paso 1: Descarga de Docker Desktop

Diríjase al sitio oficial de Docker *https://www.docker.com/get-started/* y seleccione la opción "Download for Mac". Este proceso descargará un archivo en formato .dmg.

Paso 2: Instalación de Docker Desktop

Una vez descargado el archivo .dmg, proceda a abrirlo y arrastre el ícono de Docker a la carpeta de Aplicaciones. Este paso completará la instalación de Docker en su sistema.

Paso 3: Inicio de Docker Desktop

Vaya a la carpeta de Aplicaciones y ejecute Docker haciendo doble clic en el ícono correspondiente. Al ejecutar Docker Desktop por primera vez, es posible que se le soliciten permisos de sistema.

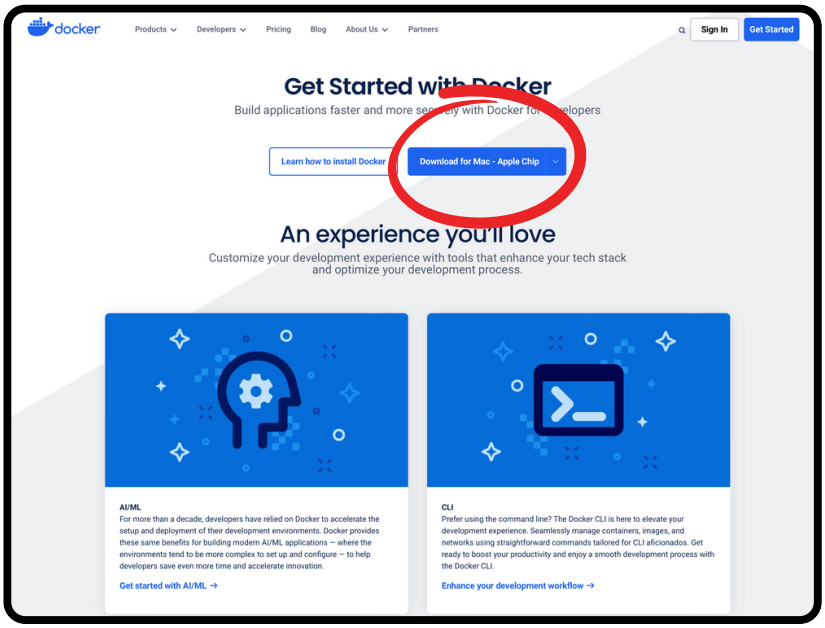
Paso 4: Docker Hub

Docker Hub es un servicio en la nube que facilita la gestión y el intercambio de imágenes de contenedores. Más adelante durante la instalación y configuración de cassandra le enseñaremos a descargar una imagen.

Paso 5: Verificación de la instalación

Abra una terminal y ejecute el siguiente comando para confirmar que Docker se ha instalado correctamente:

```
>> docker --version
```



Instalación de Docker en Windows

Paso 1: Descarga de Docker Desktop

Diríjase al sitio oficial de Docker *https://www.docker.com/get-started/* y seleccione la opción "Download for Windows". Este proceso descargará un archivo en formato .exe.

Paso 2: Instalación de Docker Desktop

Ejecute el archivo .exe descargado. Durante la instalación, se le solicitará habilitar características como Hyper-V. Acepte estas configuraciones para garantizar un funcionamiento adecuado de Docker en su sistema.

Paso 3: Inicio de Docker Desktop

Una vez completada la instalación, Docker Desktop se iniciará automáticamente. Puede encontrar el ícono de Docker en la bandeja del sistema.

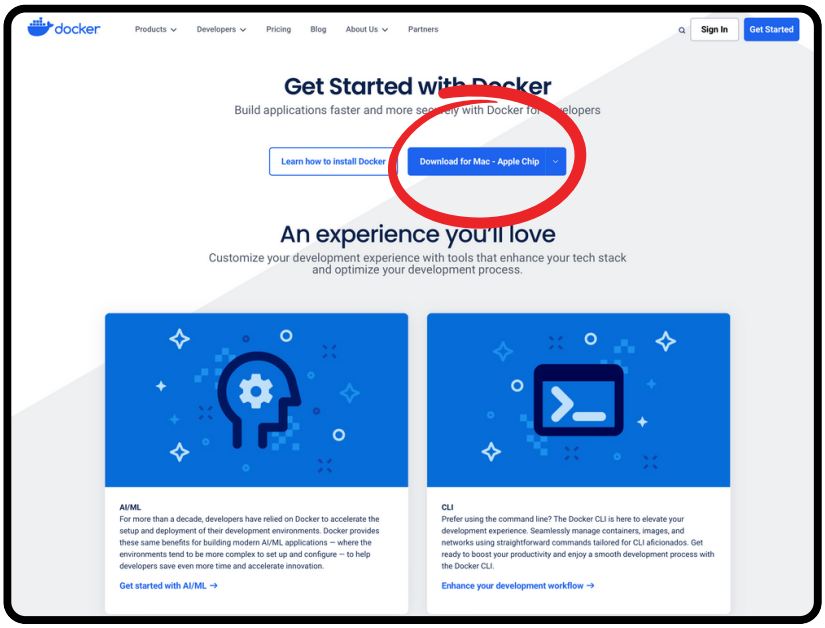
Paso 4: Docker Hub

Docker Hub es un servicio en la nube que facilita la gestión y el intercambio de imágenes de contenedores. Más adelante durante la instalación y configuración de cassandra le enseñaremos a descargar una imagen.

Paso 5: Verificación de la instalación

Abra una terminal y ejecute el siguiente comando para confirmar que Docker se ha instalado correctamente:

```
>> docker --version
```



Instalación de Cassandra 4.1.3

Se prestan varias metodologías para la obtención de una imagen de Docker, seguida por la creación de su correspondiente contenedor. Es crucial destacar que el propósito de esta guía no consiste en instruir sobre el uso específico de Docker, sino en facilitar la generación del entorno y las dependencias esenciales para que el usuario pueda efectuar la ejecución de la base de datos en su máquina.

A continuación, abra el terminal o la consola PowerShell de Linux y ejecute la siguiente instrucción:

```
>> docker run -d \
   --name cassandra_tokio \
   -p 9042:9042 \
   -v indique_ruta_local:/var/lib/cassandra \
   cassandra:4.1.3
```

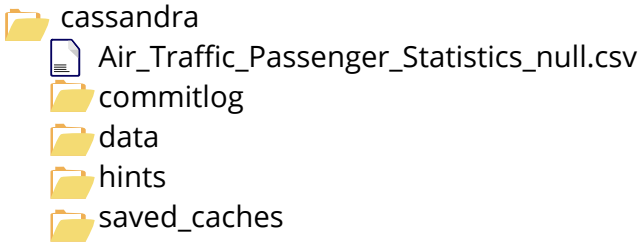
Explicación de las opciones utilizadas:

- **-d**: Ejecuta el contenedor en segundo plano (modo detach).
- **--name cassandra_tokio**: Asigna el nombre "cassandra_tokio" al contenedor.
- **-p 9042:9042**: Port Mapping al puerto 9042 del host hacia el puerto 9042 del contenedor, que es el puerto por defecto de Cassandra. Esto le permitirá acceder al contenedor desde su máquina.
- **-v indique_ruta_local:/var/lib/cassandra**: Monta un volumen llamado "indique_ruta_local" en el directorio **/var/lib/cassandra** del contenedor. Esto es necesario para persistir los datos de Cassandra fuera del contenedor.
- Finalmente, **cassandra:4.1.3** es la imagen de Cassandra y la etiqueta de versión que vamos a utilizar.

Asegúrese de reemplazar "indique_ruta_local" con el nombre que deseé darle al volumen (recomendamos crear una carpeta en el escritorio e indicar esa ruta).

Estos comandos descargarán automáticamente la imagen de Cassandra y ejecutarán un contenedor para que pueda acceder a la base de datos.
Por último, eliminé las carpetas que Docker ha creado dentro de su “indique_ruta_local” y reemplácelo por el contenido existente en la carpeta Cassandra que encontrará en /proj_tokio/data/cassandra.

La estructura de la carpeta cassandra es la siguiente:



Por último, ejecute la siguiente instrucción para acceder a Cassandra dentro del contenedor.
A partir de este momento ya tiene acceso a la tabla de datos.

```
>> docker exec -it cassandra_tokio cqlsh
```

Entorno de Desarrollo - SPARK

Implementación de Docker

En consonancia con la decisión estratégica de implementar y gestionar la base de datos NoSQL Cassandra mediante Docker, proponemos extender esta lógica al entorno de procesamiento y análisis de datos, específicamente con Apache Spark.

A continuación, presentamos aspectos esenciales que respalda la elección de utilizar Docker para ejecutar Apache Spark en este proyecto:

- **Consistencia del Entorno:**
  - Al igual que en el caso de Cassandra, Docker garantiza un entorno de ejecución consistente para Apache Spark en todos los sistemas compatibles. La eliminación de variaciones en el hardware y el sistema operativo asegura que el entorno de desarrollo y despliegue sea uniforme, reduciendo posibles conflictos y mejorando la reproducibilidad del análisis de datos.

- **Aislamiento de Recursos:**
  - La encapsulación de Apache Spark en contenedores Docker proporciona un aislamiento eficaz de recursos. Este aislamiento evita interferencias con otros servicios o aplicaciones en ejecución, garantizando un ambiente de ejecución predecible y optimizando la eficiencia en el uso de recursos, lo que se traduce en un rendimiento más consistente y confiable.

- **Control de Versiones:**
  - Docker facilita la gestión de versiones para Apache Spark de manera similar a como lo hace para Cassandra. Permite especificar la versión exacta de Spark a utilizar, contribuyendo a la consistencia y simplificación de la gestión de dependencias. Esto asegura que el proyecto funcione de manera coherente en diferentes entornos y facilita las actualizaciones controladas del framework de procesamiento de datos.

A pesar de la popularidad de **Google Colab *versión gratuita*** como plataforma de colaboración en línea para la ejecución de notebooks de Jupyter y desde 2017 convirtiéndose en una de las principales herramientas para tareas de análisis de datos y de machine learning, existen razones fundamentales que nos llevan a no optar por esta solución en nuestro proyecto. A continuación, se exponen las consideraciones clave que respaldan esta decisión:

- **Dependencia de la Conectividad a Internet:**
  - Google Colab requiere una conexión a Internet constante para acceder y ejecutar notebooks. Esta dependencia puede resultar limitante en entornos donde la conectividad no es constante o es crucial evitar la transmisión de datos sensibles a través de la red. Una mala conectividad de red es interpretada por Google Colab como un abandono de la sesión y por consiguiente un riesgo traducido en perdida de información.

- **Control sobre Versiones y Dependencias:**
  - Google Colab, aunque proporciona un entorno de ejecución flexible, puede presentar desafíos en términos de control de versiones y gestión de dependencias. La capacidad de especificar y mantener versiones específicas de bibliotecas y frameworks, como Apache Spark, es esencial para garantizar la coherencia y reproducibilidad del proyecto a lo largo del tiempo.

- **Limitaciones en Recursos de Hardware:**
  - Google Colab tiene limitaciones en términos de recursos de hardware, como la cantidad de memoria y el tiempo máximo de ejecución permitido a 10 horas por sesión. Para proyectos de análisis de datos de larga duración, estas restricciones pueden ser inconvenientes. La memoria disponible en la versión gratuita es estándar limitada a 12GB. En los términos y condiciones de uso de la plataforma, informan a los usuarios que Google se reserva el derecho de limitar los recursos ofrecidos sin aviso previo.

- **Coherencia con el Proyecto**
  - Para la ejecución de este proyecto, la capacidad computacional de un máquina es suficiente. Necesitamos un entorno virtual que nos permita ejecutar Apache Spark emulando un sistema distribuido, pero no necesitamos su capacidad de procesamiento para analizar un dataset que no alcanza los 2 MB. Además, es esencial destacar que este proyecto tiene como objetivo integrarse en el portfolio de su autor. En este sentido, se busca un entorno controlado que garantice la consistencia y reproducibilidad del análisis. Esto facilitará su alojamiento en diversas ubicaciones sin comprometer la integridad del proyecto. La elección de un entorno más contenido se alinea con la naturaleza y dimensiones específicas de la tarea, optimizando recursos y asegurando la coherencia con los objetivos y alcance del proyecto.

Es imperativo tener presente que, en la era del Big Data, la gestión y tratamiento de datos adquieren una relevancia crucial. No obstante, se debe destacar que este proceso debe ir de la mano con un uso sensato de la capacidad computacional disponible. Es fundamental reconocer la importancia de la coherencia al gestionar proyectos en Big Data, evitando el uso indiscriminado de recursos de sistemas distribuidos en casos donde los conjuntos de datos no requieren tal magnitud de capacidad. Este enfoque prudente asegura una asignación eficiente de recursos, optimizando el rendimiento del proyecto sin comprometer la eficacia del análisis de datos.

La capacidad de discernir y aplicar de manera proporcional los recursos disponibles, demuestra un nivel de madurez técnica y estratégica que eleva la calidad y sostenibilidad de nuestros esfuerzos en el ámbito del tratamiento de datos a gran escala.

No obstante, el usuario tiene la libertad de ejecutar los notebooks de este proyecto en el entorno que prefiera o con el cual esté más familiarizado. El código fuente se encuentra almacenado en archivos con extensión **.ipynb** y pueden ser ejecutados o visualizados de diversas formas.

Instalación de Apache Spark

Se prestan varias metodologías para la obtención de una imagen de Docker, seguida por la creación de su correspondiente contenedor. Es crucial destacar que el propósito de esta guía no consiste en instruir sobre el uso específico de Docker, sino en facilitar la generación del entorno y las dependencias esenciales para que el usuario pueda efectuar la ejecución de la base de datos en su máquina.

A continuación, abra el terminal o la consola PowerShell de Linux y ejecute la siguiente instrucción:

```
>> docker run -d \
   --name spark_tokio \
   -p 8888:8888 \
   -v indique_ruta_local:/home/jovyan/work \
   -e DOCKER_STACKS_JUPYTER_CMD=notebook
   jupyter/all-spark-notebook
```

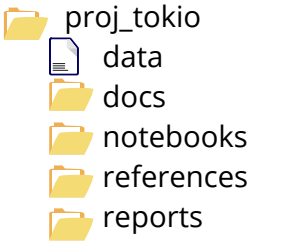
Explicación de las opciones utilizadas:

- **-d**: Ejecuta el contenedor en segundo plano (modo detach).
- **--name spark_tokio**: Asigna el nombre "spark_tokio" al contenedor.
- **-p 8888:8888**: Port Mapping al puerto 8888 del host hacia el puerto 8888 del contenedor, que es el puerto por defecto de esta imagen. Esto le permitirá acceder al contenedor desde su máquina.
- **-v indique_ruta_local:/home/jovyan/work**: Monta un volumen llamado "indique_ruta_local" en el directorio /home/jovyan/work del contenedor. Esto es necesario para persistir los datos de nuestro entorno y poder mover información de manera fácil y rápida.
- **-e DOCKER_STACKS_JUPYTER_CMD=notebook:** Configura el entorno de trabajo para ser usado por Jupyter Notebook *(Por defecto es Jupyter Labs)*.
- Finalmente, **jupyter/all-spark-notebook** es la imagen que vamos a utilizar.

Asegúrese de reemplazar "indique_ruta_local" con el nombre que deseé darle al volumen (recomendamos crear una carpeta en el escritorio e indicar esa ruta).

Estos comandos descargarán automáticamente la imagen y ejecutarán un contenedor para que pueda acceder a la base de datos.
Por último, eliminé las carpetas que Docker ha creado dentro de su “indique_ruta_local” y reemplácelo por el contenido existente en la carpeta Cassandra que encontrará en /proj_tokio.

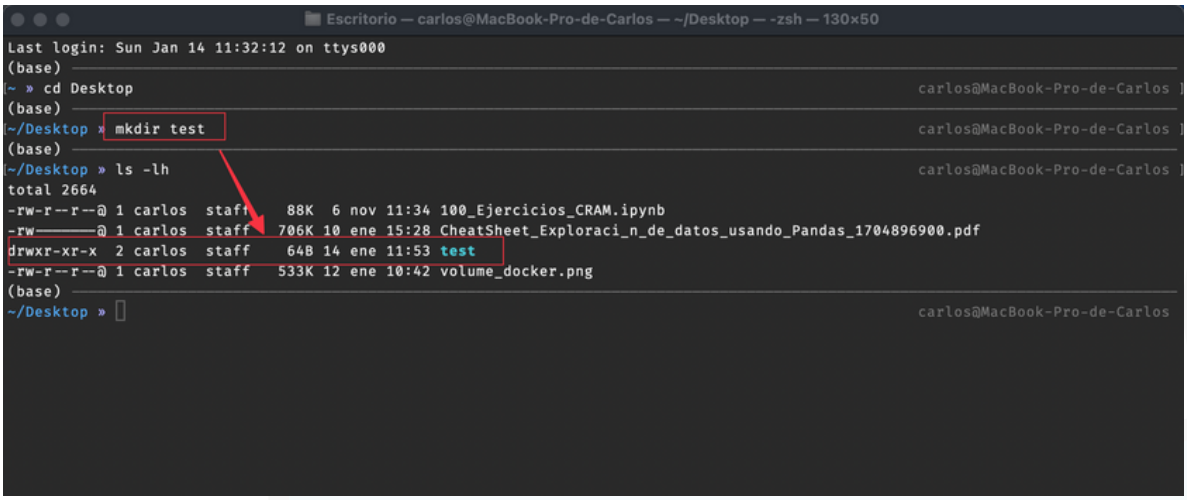
La estructura de la carpeta es la siguiente:



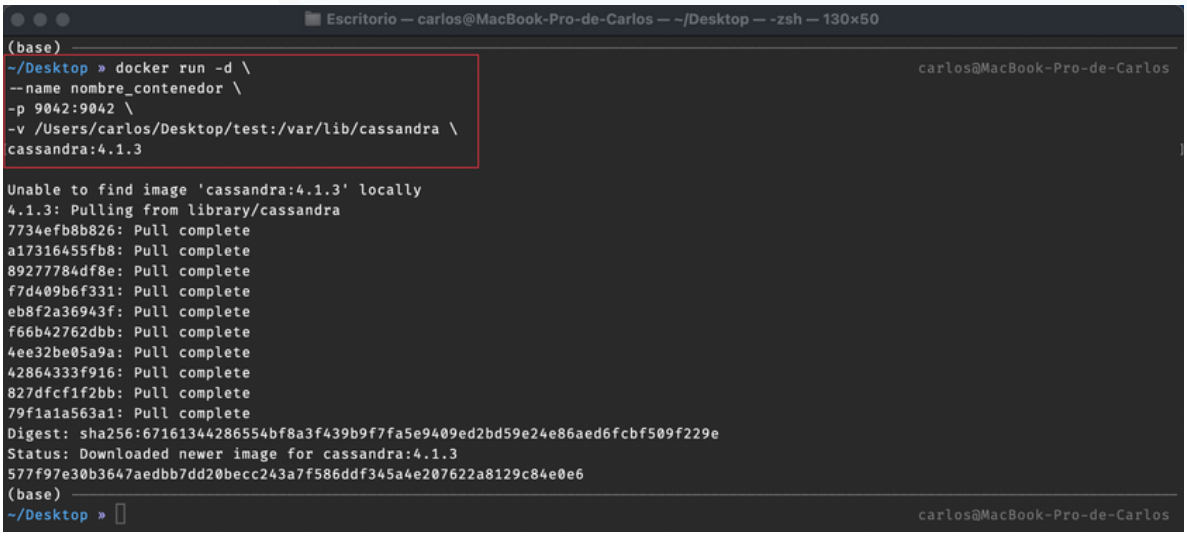
Entorno de Desarrollo con Imágenes

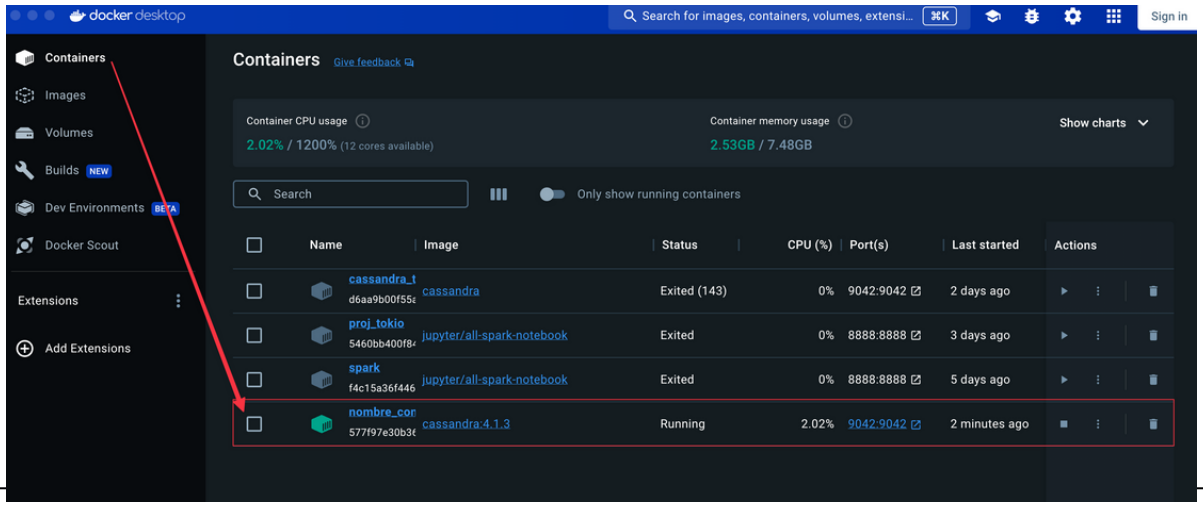
Instalación con Imágenes

Creamos una nueva carpeta, por ejemplo en el escritorio. Esta carpeta será la que vamos a vincular al contenedor



Ejecutamos la instrucción mencionada más arriba y observamos como tenemos creado un contenedor en ejecución

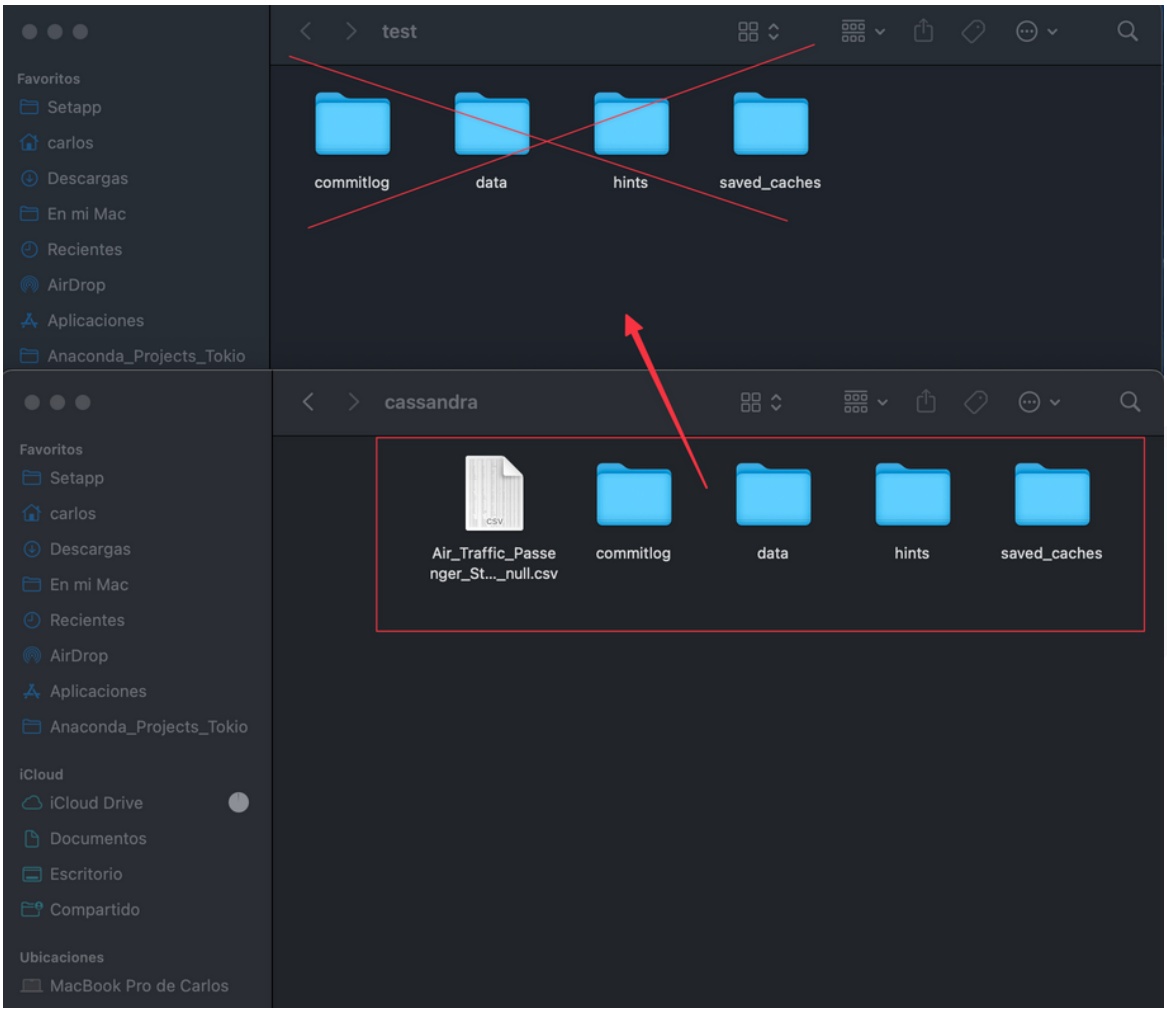




Entorno de Desarrollo con Imágenes

Instalación con Imágenes

Eliminamos el contenido de la carpeta que hemos creado, en este caso **test,** y lo remplazamos por el contenido de la carpeta de este proyecto **/proj_tokio/data/cassandra**



Ya podemos acceder a Cassandra