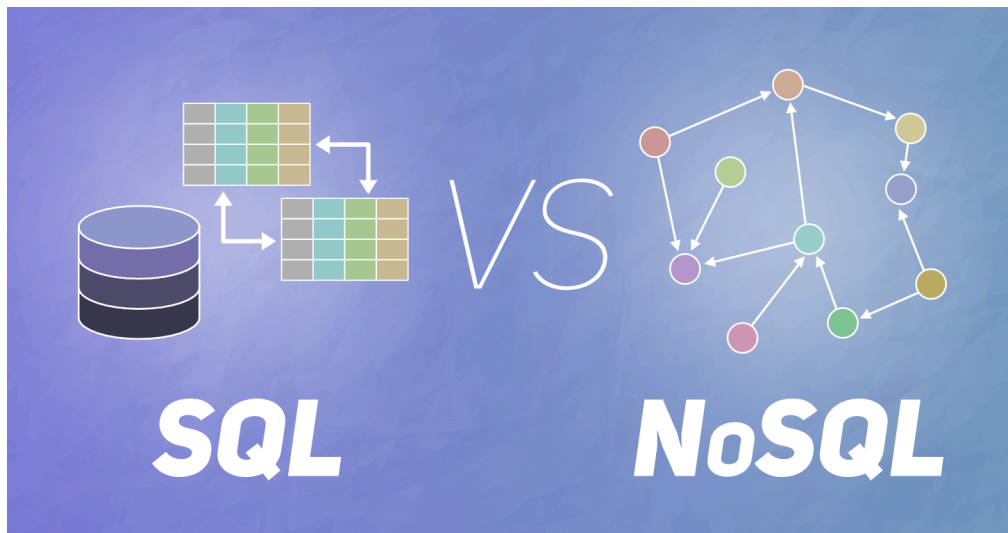


# Bases de Datos



## Estudio sobre bases de datos NoSQL.



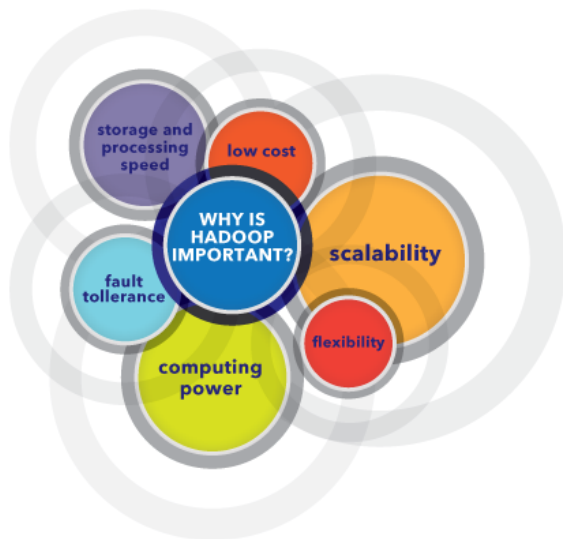
# Introducción

Antes de comenzar a hablar de HBase debemos explicar Hadoop para poder entender mejor HBase.

## ¿Qué es Hadoop?

Apache Hadoop es un marco de código abierto que se utiliza para almacenar y procesar de manera eficiente conjuntos de datos grandes cuyo tamaño varía entre los gigabytes y los petabytes de datos. En lugar de utilizar una sola computadora grande para procesar y almacenar los datos, Hadoop facilita la creación de clústeres de varias computadoras para analizar conjuntos de datos masivos en paralelo y con mayor rapidez.

## ¿Por qué es importante Hadoop?



Capacidad de almacenar y procesar enormes cantidades de cualquier tipo de datos, al instante. Con el incremento constante de los volúmenes y variedades de datos, en especial provenientes de medios sociales y la Internet de las Cosas (IoT), ésta es una consideración importante.

Poder de cómputo. El modelo de cómputo distribuido de Hadoop procesa big data a gran velocidad. Cuantos más nodos de cómputo utiliza usted, mayor poder de procesamiento tiene.

Tolerancia a fallos. El procesamiento de datos y aplicaciones está protegido contra fallos del hardware. Si falla un nodo, los trabajos son

redirigidos automáticamente a otros nodos para asegurarse de que no falle el procesamiento distribuido. Se almacenan múltiples copias de todos los datos de manera automática.

Flexibilidad. A diferencia de las bases de datos relacionales, no tiene que procesar previamente los datos antes de almacenarlos. Puede almacenar tantos datos como desee y decidir cómo utilizarlos más tarde. Eso incluye datos no estructurados como texto, imágenes y videos.

Bajo costo. La estructura de código abierto es gratuita y emplea hardware comercial para almacenar grandes cantidades de datos.

Escalabilidad. Puede hacer crecer fácilmente su sistema para que procese más datos con sólo agregar nodos. Se requiere poca administración.

## ¿Cuáles son los cuatro módulos principales de Hadoop?

Hadoop consta de cuatro módulos principales:

Sistema de archivos distribuido de Hadoop (HDFS): sistema de archivos distribuido que se ejecuta en hardware estándar o de gama baja. HDFS proporciona un mejor rendimiento de datos que los sistemas de archivos tradicionales, además de una alta tolerancia a errores y compatibilidad nativa con conjuntos de datos de gran tamaño.

Yet Another Resource Negotiator (YARN): administra y supervisa los nodos del clúster y el uso de recursos. Programa trabajos y tareas.

MapReduce: un marco que ayuda a los programas a realizar el cálculo paralelo de los datos. La tarea de Map toma los datos de entrada y los convierte en un conjunto de datos que se puede calcular en pares de valores clave. La salida de la tarea de Map la consumen las tareas de Reduce para agregar la salida y proporcionar el resultado deseado.

Hadoop Common: proporciona bibliotecas Java comunes que se pueden usar en todos los módulos.

### ¿Qué es HBase?

HBase es una base de datos NoSQL distribuida y escalable, desarrollada como parte del proyecto Hadoop Distributed File System (HDFS). Está diseñada para manejar grandes cantidades de datos y proporcionar un acceso rápido y eficiente a los mismos.

Apache HBase, maneja grandes cantidades de datos estructurados y no estructurados, lo que significa que puede almacenar una amplia variedad de datos. Es una base de datos de clave-valor, en la que los datos se almacenan en una tabla hash. La clave es un identificador único que se utiliza para acceder a los datos y el valor es la información que se quiere almacenar. HBase utiliza la tecnología de indexación de Apache Hadoop, lo que permite un acceso rápido y eficiente a los datos.

Una de las principales características de HBase es su capacidad de escalar horizontalmente. Esto significa que se pueden agregar nuevos nodos al cluster para aumentar la capacidad de almacenamiento y procesamiento de datos. HBase también proporciona una alta disponibilidad de datos, lo que significa que si un nodo falla, los datos se replican en otros nodos del cluster y se pueden acceder sin interrupción.

HBase utiliza el modelo de consistencia eventual, lo que significa que los cambios en la base de datos pueden tardar un tiempo en propagarse a través del cluster. Sin embargo, esto permite una alta disponibilidad y escalabilidad, lo que la convierte en una opción popular para aplicaciones en las que la disponibilidad es más importante que la consistencia inmediata de los datos.

Otra característica importante de HBase es su capacidad de realizar consultas en tiempo real. HBase utiliza la tecnología de indexación de Apache Hadoop, que permite consultas rápidas y eficientes en grandes cantidades de datos. Además, HBase es compatible con una amplia variedad de lenguajes de programación, como Java, Python, Ruby y PHP.

En resumen, HBase es una base de datos NoSQL distribuida y escalable, diseñada para manejar grandes cantidades de datos y proporcionar acceso rápido y eficiente a los mismos. Su capacidad de escalar horizontalmente, alta disponibilidad, capacidad de realizar consultas en tiempo real y compatibilidad con una amplia variedad de lenguajes de programación la convierten en una opción popular para aquellos que necesitan una base de datos NoSQL confiable y escalable para sus aplicaciones.

HBase se basa en ZooKeeper para una coordinación de alto rendimiento. ZooKeeper está integrado en HBase, pero si está ejecutando un clúster de producción, se sugiere que tenga un clúster ZooKeeper dedicado que esté integrado con su clúster HBase.

HBase funciona bien con Hive, un motor de consulta para el procesamiento por lotes de big data, para habilitar aplicaciones de big data tolerantes a fallos.

## Cómo se relacionan HBase y HDFS

### HDFS

HDFS es un sistema de ficheros distribuido adecuado para almacenar archivos de gran tamaño.

HDFS no admite búsquedas rápidas registro individual.

Proporciona una alta latencia de procesamiento, usa un concepto de procesamiento por lotes.

Sólo proporciona acceso secuencial de los datos.

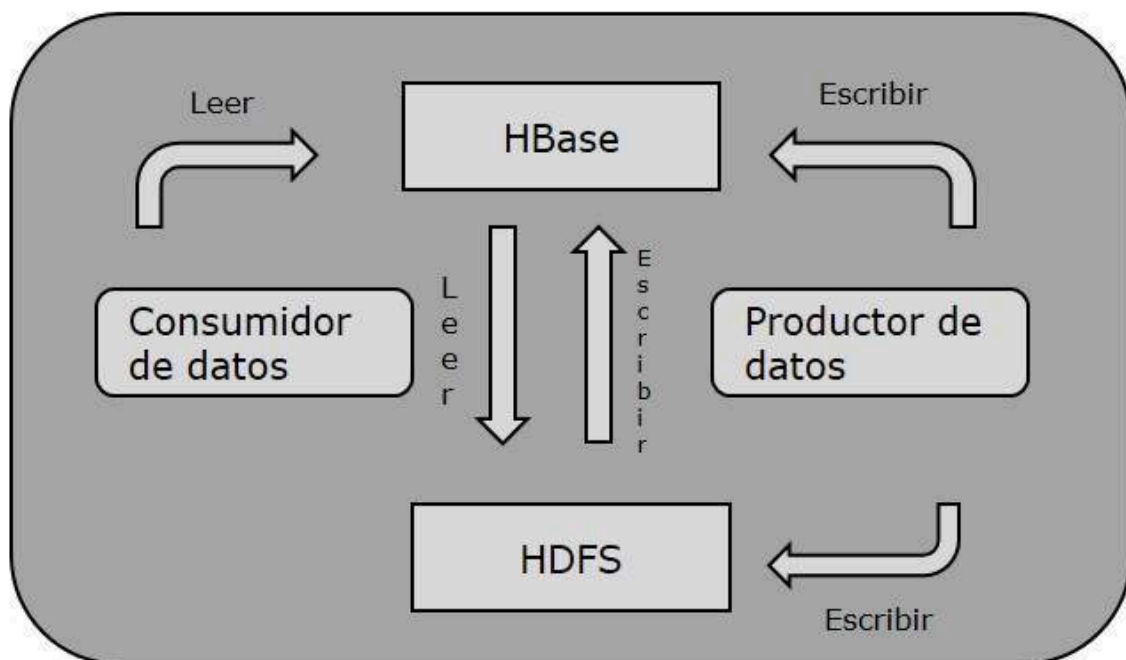
### HBase

HBase es una base de datos creada en la parte superior de la HDFS.

HBase proporciona búsquedas rápidas tablas más grandes

Proporciona acceso de baja latencia a filas de miles de millones de registros (acceso aleatorio).

HBase internamente usa tablas Hash y proporciona acceso aleatorio, y que almacena los datos en archivos indexados HDFS búsquedas más rápido.



## Apache Hbase: ventajas y desventajas.

Una vez que hemos visto todas las características de Apache HBase, también es importante ver cuáles son sus ventajas y desventajas en el trabajo dentro del análisis de datos. Empecemos con las ventajas de Hbase:

- HBase permite hacer búsquedas de versiones, así como de los históricos de los datos. De esta manera se puede encontrar fácilmente la versión de la información que necesitamos.
- Cuando existe una carga alta de información se puede escalar el sistema simplemente añadiendo nuevas máquinas al clúster de ordenadores.
- La integración con Hadoop garantiza la fiabilidad de sus datos para el análisis para Big Data.
- Los posibles fallos se pueden evitar ya que cada parte del clúster tiene copias de seguridad de todos los elementos que forman parte de la base de datos, por lo que, si uno falla, los otros pueden seguir trabajando.

No obstante, a pesar de todas sus ventajas, HBase también tiene algunas desventajas:

- Su implementación basada en Java y la arquitectura de Hadoop hace que su API sea más adecuada para proyectos desarrollados en este lenguaje de programación.
- El entorno de desarrollo de los nodos necesita distintas dependencias lo que hace que configurarlo sea problemático.
- Ocupa mucha memoria y al estar configurado y construido para HDFS para el análisis por lotes, el rendimiento de la lectura de datos no es especialmente alto.
- Es relativamente torpe en comparación con otras bases de datos no relacionales.

A pesar de sus desventajas, HBase forma parte de Hadoop y, con ello, resulta útil para empresas que buscan soluciones Big Data a costes reducidos para proyectos desarrollados con arquitectura Java.

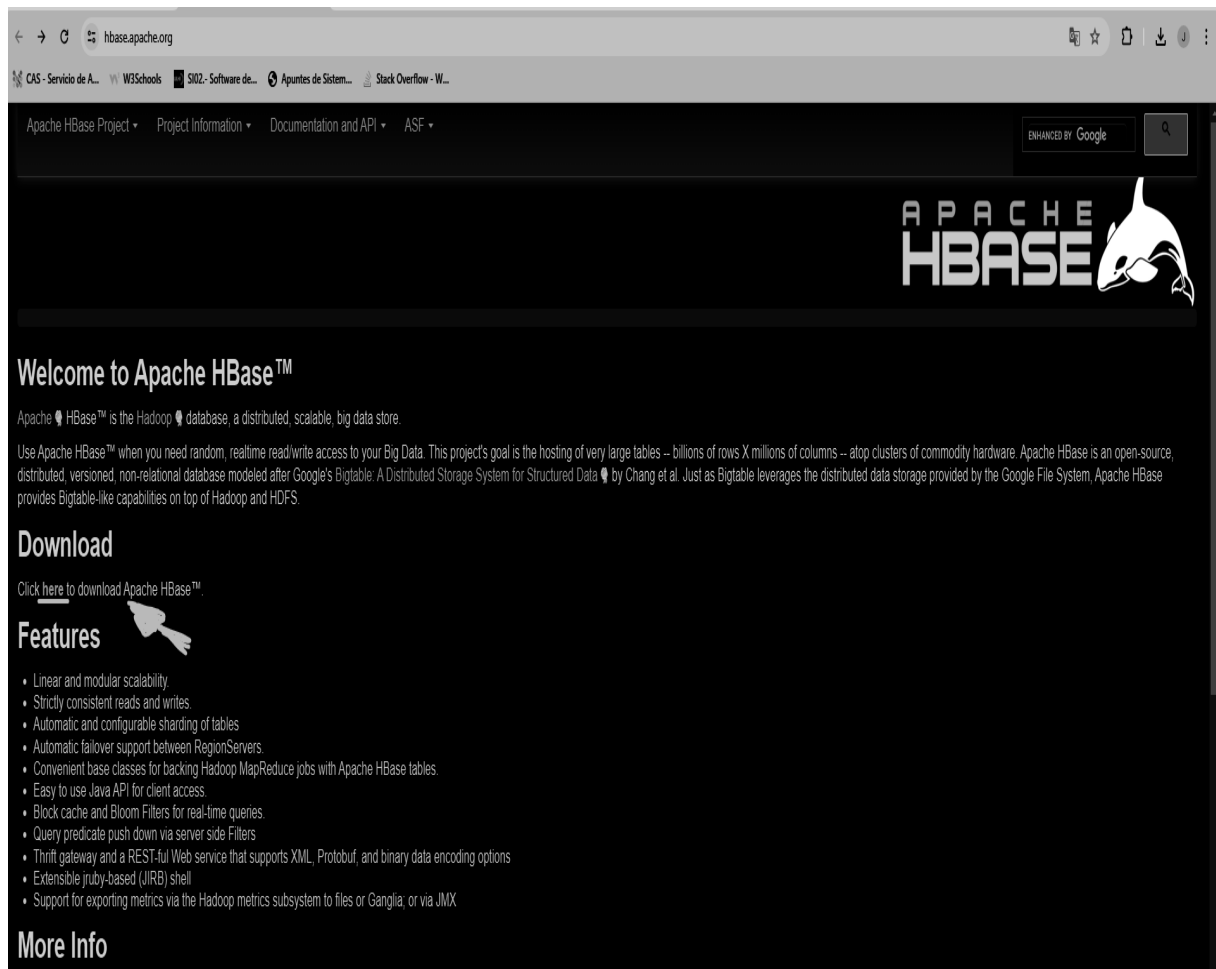
## Usos de HBase en empresas reales:

1. Facebook: Facebook utiliza HBase para almacenar su Graph API, que es una API que permite a los desarrolladores acceder a los datos sociales de Facebook. HBase también se utiliza para almacenar el chat de Facebook y la información del usuario.
2. Twitter: Twitter utiliza HBase para almacenar sus tweets, menciones, seguidores y otra información relacionada con la plataforma. HBase permite a Twitter escalar horizontalmente y manejar grandes cantidades de datos en tiempo real.
3. Yahoo!: Yahoo! utiliza HBase para almacenar y procesar grandes cantidades de datos. HBase se utiliza en Yahoo! Mail, Yahoo! Sports y en otros productos de Yahoo! que requieren una gran cantidad de datos.
4. Netflix: Netflix utiliza HBase para almacenar y procesar datos de sus usuarios, como las películas y programas de televisión que han visto y las calificaciones que han dado. HBase permite a Netflix escalar horizontalmente y proporcionar recomendaciones personalizadas a sus usuarios.
5. Adobe: Adobe utiliza HBase como base de datos escalable para almacenar y procesar grandes cantidades de datos para su plataforma de marketing en la nube. HBase se utiliza para almacenar información del cliente, como datos de transacciones y comportamiento en línea.
6. Alibaba: Alibaba, el gigante chino del comercio electrónico, utiliza HBase para almacenar y procesar los datos de sus aplicaciones en tiempo real, como la gestión de pedidos y el seguimiento de envíos.
7. Pinterest: Pinterest utiliza HBase para almacenar y procesar los datos de su plataforma, incluyendo la información de los usuarios, los pines y tableros.
8. Spotify: Spotify utiliza HBase para almacenar y procesar datos relacionados con la música, como las canciones que han sido reproducidas por los usuarios, las listas de reproducción y las recomendaciones personalizadas.
9. Microsoft: Microsoft utiliza HBase como base de datos para almacenar y procesar grandes cantidades de datos en su plataforma de nube, Azure. HBase se utiliza en aplicaciones como Bing, Office y Skype.
10. Airbnb: Airbnb utiliza HBase para almacenar y procesar datos relacionados con las reservas, las evaluaciones de los usuarios y la información de los anfitriones.

Entre otras muchas aplicaciones en grandes empresas.

# Tutorial de descarga e instalación:

Entrar en página principal de HBase y clicar en 'here' en la sección de download.



El paso anterior es si se quiere instalar en nuestro PC. En nuestro caso trabajaremos en un repositorio de Docker.

El primer paso es tener docker instalado en el ordenador descargando el ejecutable desde:

<https://docs.docker.com/desktop/release-notes/>



The screenshot shows the Docker Desktop release notes page. The left sidebar contains a navigation menu with categories like Overview, Get Docker, Docker Desktop, and various guides. The main content area is titled 'Docker Desktop release notes' and includes a 'Note' section stating that the information is applicable to all platforms. It also features a 'Download Docker Desktop' section with links for Windows, Mac, and Linux. The right sidebar contains a 'Table of contents' with links to various release notes and a 'Tags' section.

Se instala en el ordenador, se ejecuta el cmd como administrador y se usa el comando: `wsl --update` para tener actualizado el virtualizador de Linux sobre Windows, este paso es fundamental para el correcto funcionamiento.(En nuestro caso ya lo tenemos realizado y actualizado):

```
C:\Windows\system32>wsl --update
Comprobando actualizaciones.
La versión más reciente de Subsistema de Windows para Linux ya está instalada.
C:\Windows\system32>
```

Tras este paso se reinicia el ordenador y se ejecuta Docker.  
A partir de aquí seguiremos los pasos de la siguiente página para proceder a la descarga de la imagen de la base de datos para poder usarla en el repositorio.

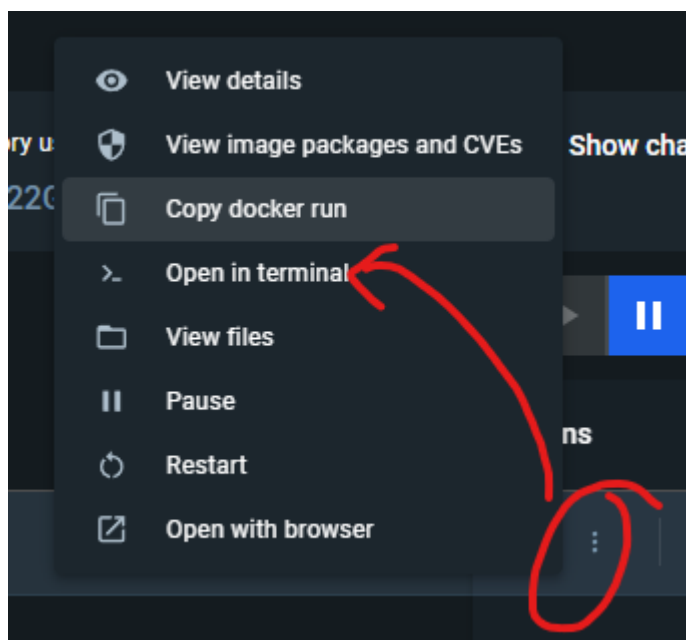
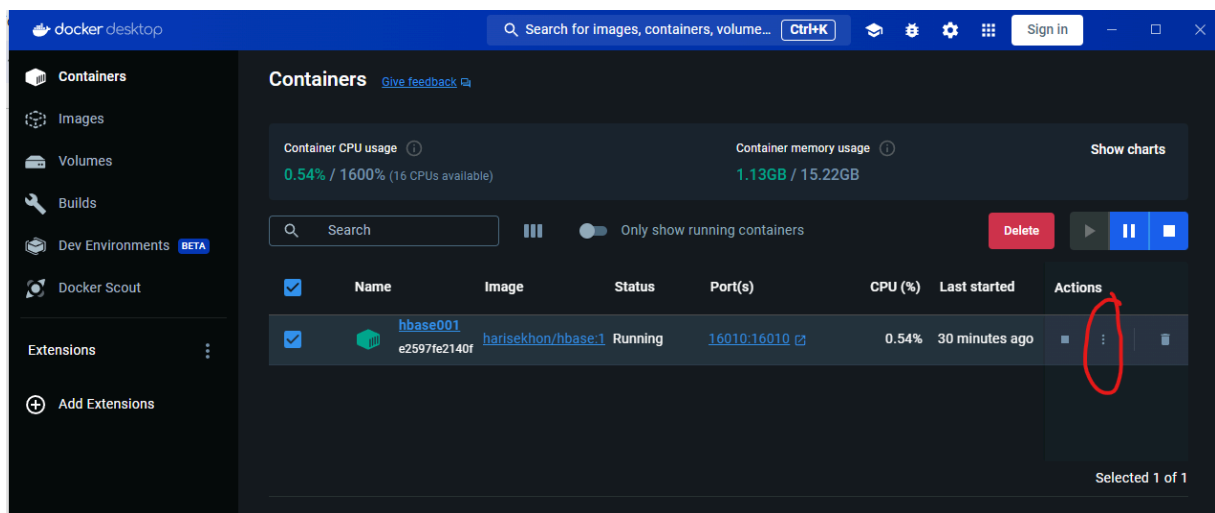
<https://www.programmersought.com/article/6344708049/>

Lanzando desde cmd estos 2 comandos ya inicia la descarga de la imagen y la ejecuta desde docker

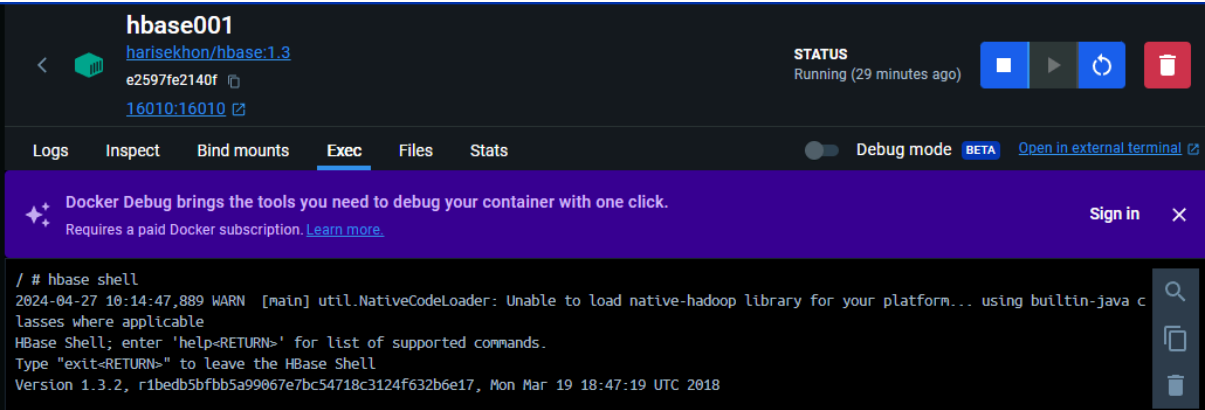
```
docker pull harisekhon/hbase:1.3
```

```
docker run -d --name hbase001 -P harisekhon/hbase:1.3
```

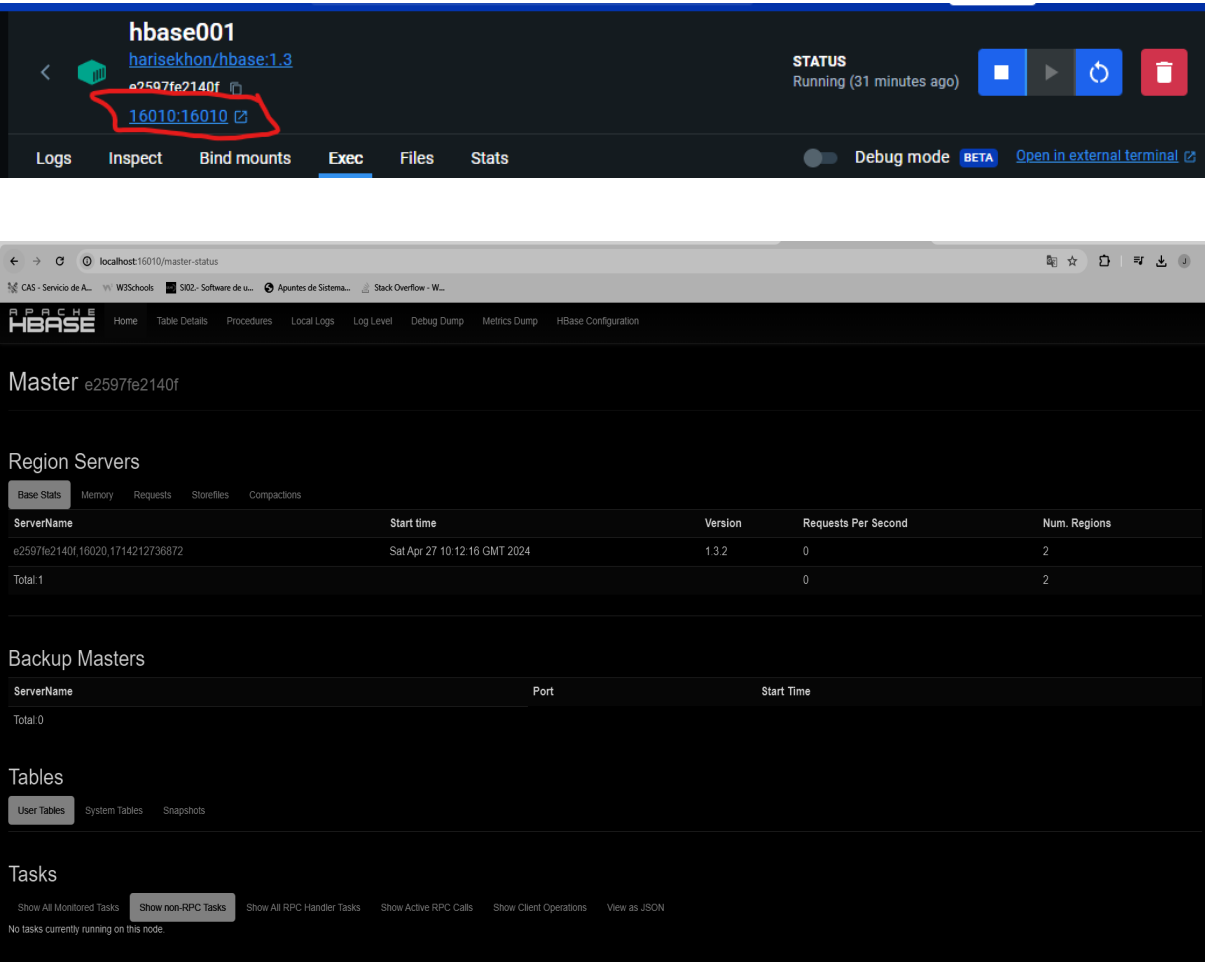
Tras esto ya tendremos instalada la imagen en el docker y podremos acceder a la base de datos desde el terminal de Docker para poder empezar a trabajar con ella.



En la terminal se lanza el comando `hbase shell` para entrar en la terminal de la base de datos y poder trabajar con ella:



Utilizando el host indicado en el docker se abre una interfaz gráfica que nos da información sobre la base de datos:



## Uso de comandos básicos de HBASE

1. Creamos una tabla con el comando create 'TablaPrimera', 'columna1', 'columna2'

Este comando crea una tabla con las columnas deseadas indicadas tras el nombre de la tabla.

```
hbase(main):001:0> create 'TablaPrimera', 'columna1', 'columna2'
0 row(s) in 1.3610 seconds
```

2. Lanzamos los comandos list y describe 'TablaPrimera'. El primero de estos comando sirve para mostrar una lista con todas las tablas que existen en la base de datos y el segundo comando sirve para recibir información detallada de las columnas creadas en la tabla indicada:

```
=> Hbase::Table - TablaPrimera
hbase(main):002:0> list
TABLE
TablaPrimera
1 row(s) in 0.0140 seconds

=> ["TablaPrimera"]
hbase(main):003:0> describe 'TablaPrimera'
Table TablaPrimera is ENABLED
TablaPrimera
COLUMN FAMILIES DESCRIPTION
{NAME => 'columna1', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'columna2', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.0570 seconds
```

3. A continuación se añadirán valores a los campos de la tabla. Usamos el comando put 'TablaPrimera', 'Id1', 'columna1:nombre', 'juan'. De esta forma estaremos añadiendo a la fila con el Id1 (Clave principal o identificador de línea) que añada en la primera columna (columna1) un campo llamado nombre con el valor juan. De esta forma estamos generando una rejilla dentro de columna1. Se repite el proceso para añadir otro campo más en columna1 y otros 2 campos más en columna2.

```

hbase(main):006:0> put 'TablaPrimera', 'Id1', 'columna1:nombre', 'juan'
0 row(s) in 0.0280 seconds

hbase(main):007:0> put 'TablaPrimera', 'Id1', 'columna1:apellidos', 'perez'
0 row(s) in 0.0030 seconds

hbase(main):008:0> put 'TablaPrimera', 'Id1', 'columna2:edad', '30'
0 row(s) in 0.0030 seconds

hbase(main):009:0> put 'TablaPrimera', 'Id1', 'columna2:direccion', 'calle desconocida'
0 row(s) in 0.0020 seconds

```

```

hbase> put 'ns1:t1', 'r1', 'c1', 'value'
hbase> put 't1', 'r1', 'c1', 'value'
hbase> put 't1', 'r1', 'c1', 'value', ts1
hbase> put 't1', 'r1', 'c1', 'value', {ATTRIBUTES=>{'mykey'=>'myvalue'}}
hbase> put 't1', 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>{'mykey'=>'myvalue'}}
hbase> put 't1', 'r1', 'c1', 'value', ts1, {VISIBILITY=>'PRIVATE|SECRET'}

```

4. Tras esto mostraremos los valores almacenados en cada uno de los campos de la fila con Id1. Usamos el comando get 'TablaPrimera', 'Id1'. De esta forma mostraremos todos los campos de las diferentes columnas de la fila. También se pueden añadir más argumentos enseñando solo la columna, campo o valor deseado:

```

hbase(main):013:0> get 'TablaPrimera', 'Id1'
COLUMN                                CELL
columna1:apellidos                    timestamp=1714215451791, value=perez
columna1:nombre                        timestamp=1714215431504, value=juan
columna2:direccion                    timestamp=1714215483802, value=calle desconocida
columna2:edad                          timestamp=1714215468793, value=30
1 row(s) in 0.0140 seconds

```

```

hbase> t.get 'r1'
hbase> t.get 'r1', {TIMERANGE => [ts1, ts2]}
hbase> t.get 'r1', {COLUMN => 'c1'}
hbase> t.get 'r1', {COLUMN => ['c1', 'c2', 'c3']}
hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
hbase> t.get 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2], VERSIONS => 4}
hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}
hbase> t.get 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}
hbase> t.get 'r1', 'c1'
hbase> t.get 'r1', 'c1', 'c2'
hbase> t.get 'r1', ['c1', 'c2']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}

```

5. Se puede realizar una búsqueda por valor para que nos devuelva el la fila columna y campo en el que se encuentra. Se usa el comando `scan 'TablaPrimera', { FILTER =>"ValueFilter(=, 'regexstring:juan')" }` para hacer la búsqueda del valor juan devolviendonos la fila columna y campo donde se encuentra:

```
hbase(main):014:0> scan 'TablaPrimera', { FILTER =>"ValueFilter(=, 'regexstring:juan')" }
ROW                                COLUMN+CELL
  Id1                                column=columna1:nombre, timestamp=1714215431504, value=juan
1 row(s) in 0.0290 seconds
```

6. Realizaremos también un borrado de una celda en concreto. Usamos el comando `delete 'TablaPrimera', 'Id1', 'columna1:apellidos'` para el borrado de esa celda y mostramos la información de la fila con el comando `get`:

```
hbase(main):003:0> delete 'TablaPrimera', 'Id1', 'columna1:apellidos'
0 row(s) in 0.0270 seconds

hbase(main):004:0> get 'TablaPrimera', 'Id1'
COLUMN                                CELL
  columna1:nombre                      timestamp=1714215431504, value=juan
  columna2:direccion                  timestamp=1714215483802, value=calle desconocida
  columna2:edad                       timestamp=1714215468793, value=30
1 row(s) in 0.0160 seconds
```

7. También se puede realizar un borrado de la fila completa. Usamos el comando `deleteall 'TablaPrimera', 'Id1'` eliminando así toda la fila y mostramos la información de la fila mostrando que está vacía.

```
hbase(main):001:0> deleteall 'TablaPrimera', 'Id1'
0 row(s) in 0.1300 seconds
```

```
hbase(main):003:0> get 'TablaPrimera', 'Id1'
COLUMN                                CELL
0 row(s) in 0.0120 seconds
```

8. HBase también da una opción bastante útil que es deshabilitar la tabla sin tener que borrarla con el comando `disable`:

```
hbase(main):004:0> disable 'TablaPrimera'
0 row(s) in 2.2840 seconds
```

9. Para volver a activar o habilitar la tabla se usa el comando enable:

```
hbase(main):006:0> enable 'TablaPrimera'  
0 row(s) in 1.2340 seconds
```

10. Y por último se puede borrar la tabla con el comando drop una vez deshabilitada la tabla, haciendo así más seguro el proceso de eliminado de la tabla evitando posibles eliminaciones por error

```
hbase(main):007:0> disable 'TablaPrimera'  
0 row(s) in 2.2550 seconds  
  
hbase(main):008:0> drop 'TablaPrimera'  
0 row(s) in 1.2570 seconds  
  
hbase(main):009:0> list  
TABLE  
0 row(s) in 0.0070 seconds  
  
=> []
```

## Bibliografía:

<https://www.ibm.com/es-es/topics/hbase>

<https://hbase.apache.org/>

[https://www.tutorialspoint.com/es/hbase/hbase\\_overview.htm](https://www.tutorialspoint.com/es/hbase/hbase_overview.htm)

<https://www.youtube.com/watch?v=9YkurBN5Pt0>

<https://www.tokioschool.com/noticias/hbase/>

<https://aws.amazon.com/es/what-is/hadoop/>

<https://www.josebernalte.com/tool/hbase/>