

Comparación de Despliegue de un Gestor de Base de Datos NoSQL Mediante Docker

MARKO ANTONIO RIVAS RIOS

JORGE LUIS MAMANI MAQUERA

ORLANDO ANTONIO ACOSTA ORTIZ

YOFER NAIN CATARI CABRERA

ORESTES RAMIREZ TICONA

ROBERTO ZEGARRA REYES

Universidad Privada de Tacna

Junio 22, 2019

Abstract

Resumen

Docker es un proyecto open source creado en 2013 y que ha supuesto una revolución para el desarrollo y despliegue de operaciones. Docker abstrae el hardware y el sistema operativo del host ejecutando las aplicaciones en contenedores, compartimentos aislados que contienen todos los recursos para una aplicación o servicio. En este trabajo veremos cómo usar Docker para el desarrollo de aplicaciones sencillas, aprendiendo a desplegar una base de datos NOSQL con Docker.

Abstract

Docker is an open source project created in 2013 and which has been a revolution for the development and deployment of operations. Docker abstracts the host's hardware and operating system by running the applications in containers, isolated compartments that contain all the resources for an application or service. In this work we will see how to use Docker for the development of simple applications, learning how to deploy a NOSQL database with Docker.

I. INTRODUCCIÓN

EL potente concepto de Microservicios está cambiando poco a poco la industria. Grandes servicios monolíticos están dando paso lentamente al enjambre microservicios pequeños y autónomos que trabajan en conjunto. El proceso va acompañado de otra tendencia del mercado: la contenerización. Juntos, ayudan a construir sistemas sin precedentes. La contenerización cambia no sólo la arquitectura de los servicios, sino también la estructura de ambientes utilizados para crearlos.

Ahora, cuando el software se distribuye en contenedores, los desarrolladores tienen plena libertad para decidir qué aplicaciones necesitan. Como resultado, incluso los entornos complejos, como los servidores de grandes bases de datos e infraestructura de análisis complejos pueden crear instancias en cuestión de segundos. El desarrollo de software se hace más fácil y más eficaz.

II. MATERIALES Y MÉTODOS

i. Materiales

- Virtualización activada en el BIOS
- Docker Desktop
- Windows 10 64bit: Pro, Enterprise o Education, con al menos 4GB de RAM.

ii. Métodos

- Se utilizó como material artículos y libros relacionados a la base de datos NoSQL y sus tipos, así como páginas web.

III. MARCO TEÓRICO

Docker es una tecnología de containers muy popular. ¿No has usado containers? Sirven para desplegar aplicaciones en un entorno virtual aislado, pero sin el overhead de tener un Sistema Operativo (SO) nuevo como se tiene en una Virtual Machine (VM).

La principal diferencia es que los containers virtualizan el SO en vez del hardware, como lo hace una VM. De esta forma un container usa menos espacio en disco (ya que no requiere una copia del SO), y es más ágil (ya que usa menos recursos del host para ejecutar).

i. Creación de una base de datos NoSQL en docker

Instala Docker Desktop para Windows Docker Desktop para Windows es la Community Edition (CE) de Docker para Microsoft Windows. Para descargar Docker Desktop para Windows, diríjase a Docker Hub.

Lo que hay que saber antes de instalar es:

README FIRST para los usuarios de Docker Toolbox y Docker Machine : Docker Desktop para Windows requiere Microsoft Hyper-V para ejecutarse. El instalador de Docker Desktop para Windows habilita Hyper-V para usted, si es necesario, y reinicia su máquina. Después de habilitar Hyper-V, VirtualBox ya no funciona, pero las imágenes de VirtualBox VM permanecen. Las máquinas virtuales de VirtualBox creadas con docker-machine(incluida la default que se crea normalmente durante la instalación de Toolbox) ya no se inician. Estas máquinas virtuales no se pueden usar en paralelo con Docker Desktop para Windows. Sin embargo, todavía puede utilizar docker-machine para administrar máquinas virtuales remotas.

Requisitos del sistema :

- Windows 10 64bit: Pro, Enterprise o Education (Build 15063 o posterior).
- La virtualización está habilitada en la BIOS. Normalmente, la virtualización está habilitada por defecto. Esto es diferente de tener Hyper-V habilitado. Para obtener más detalles, consulte La virtualización debe estar habilitada en Solución de problemas.
- Característica capaz de CPU SLAT.
- Al menos 4GB de RAM.

Nota : Si su sistema no cumple con los requisitos para ejecutar Docker Desktop para Windows, puede instalar Docker Toolbox , que utiliza Oracle Virtual Box en lugar de Hyper-V.

Lo que incluye la instalación de Docker Desktop para Windows : La instalación proporciona Docker Engine , Docker CLI client, Docker Compose , Docker Machine y Kitematic . Los contenedores y las imágenes creadas con Docker Desktop para Windows se comparten entre todas las cuentas de usuario en las máquinas donde está instalado. Esto se debe a que todas las cuentas de Windows usan la misma máquina virtual para crear y ejecutar contenedores. Los escenarios de virtualización anidados, como ejecutar Docker Desktop para Windows en una instancia de VMWare o Parallels, pueden funcionar, pero no hay garantías. Para obtener más información, consulte Ejecución de Docker Desktop para Windows en escenarios de virtualización anidados.

Instalar la aplicación de escritorio Docker Desktop para Windows

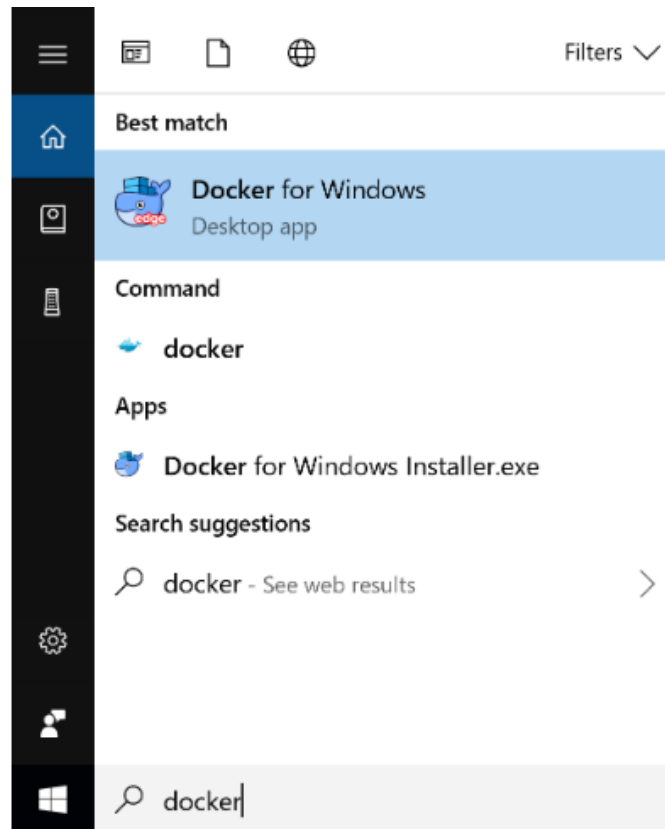
1.- Haga doble clic en Docker Desktop para Windows Installer.exe para ejecutar el instalador. Si aún no ha descargado el instalador (Docker Desktop Installer.exe), puede obtenerlo en download.docker.com . Normalmente se descarga a su Downloads carpeta, o puede ejecutarlo desde la barra de descargas recientes en la parte inferior de su navegador web.

2.-Siga el asistente de instalación para aceptar la licencia, autorice el instalador y continúe con la instalación. Se le solicitará que autorice Docker.app con su contraseña del sistema durante el proceso de instalación. Se necesita acceso privilegiado para instalar componentes de red, enlaces a las aplicaciones de Docker y administrar las máquinas virtuales de Hyper-V.

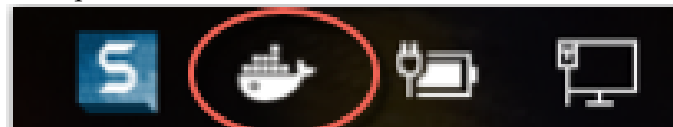
3.-Haga clic en Finalizar en el cuadro de diálogo de instalación completa para iniciar Docker.

Iniciar Docker Desktop para Windows

Docker no se inicia automáticamente después de la instalación. Para iniciarlo, busque Docker, seleccione Docker Desktop para Windows en los resultados de búsqueda y haga clic en él (o presione Entrar).



Cuando la ballena en la barra de estado se mantiene estable, Docker está en funcionamiento y accesible desde cualquier ventana de terminal.



Si la ballena está oculta en el área de Notificaciones, haga clic en la flecha hacia arriba en la barra de tareas para mostrarla. Para obtener más información, consulte Configuración de Docker . Si acaba de instalar la aplicación, también recibe un mensaje emergente de éxito con los siguientes pasos sugeridos y un enlace a esta documentación.



Cuando se complete la inicialización, seleccione Acerca de Docker en el ícono del área de Notificaciones para verificar que tiene la última versión. Está en funcionamiento con Docker Desktop para Windows.

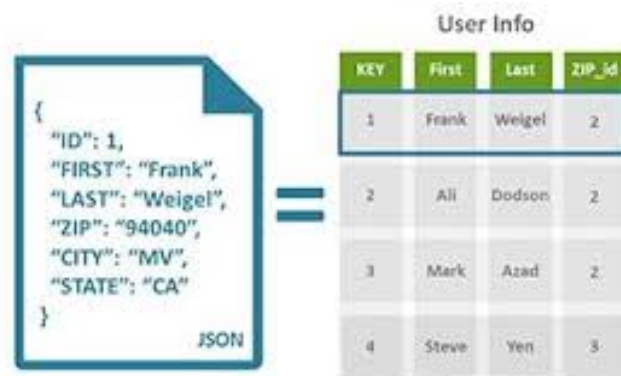
Creacion de base de datos NoSQL con MongoDB en Docker

Instalar MongoDB en Docker

- Ingresar sus credenciales creadas en Docker Hub para iniciar sesión en el aplicativo. Ubicar la aplicación PowerShell, ejecutarla como Administrador. En la ventana de comandos de PowerShell escribir lo siguiente.
- Para instalar MongoDB primero tenemos que ejecutar el siguiente código.

- Verificar que el contenedor se este ejecutando correctamente mediante el comando:
 - Proceder a verificar la imagen con el siguiente comando:
 - Seguidamente ejecutar el comando.Como respuesta se visualiza a un ID que corresponde al contenedor:
 - Para conectar a nuestro localhost
 - Para exponer ese puerto para nuestro que podamos acceder al contenedor
- MongoDB está preparado para escalar fácilmente de manera horizontal**
- En PowerShell ejecutar el siguiente comando ,Verificar la eliminación del contenedor con ejecutando
 - ejecutar el comando:Como respuesta se visualizará un ID que corresponde al contenedor
 - conectar a nuestro localhost
 - conexion de Mongo,instancia de Mongo Corriendo dentro de este contenedor docker

ii. Inserción de datos y Consulta de datos (en una base de datos NOSQL)



Existen varias diferencias con respecto a cómo los distintos tipos de bases de datos permiten a los usuarios / aplicaciones realizar consultas. Desde las consultas más básicas por clave primaria, como por ejemplo, los almacenes clave – valor, pasando por otros que ofrecen un acceso a la información algo más complejo. En este terreno se encontrarían las bases de datos documentales.

SQL	MongoDB
CREATE DATABASE empresa;	USE empresa
<pre>CREATE TABLE dpto (dep NUMBER(3) PRIMARY KEY, nom VARCHAR(20) NOT NULL); CREATE TABLE empleado (id NUMBER(4) PRIMARY KEY, nombre VARCHAR(20) NOT NULL, edad VARCHAR(20) NOT NULL, grado NUMBER(3) NOT NULL, dep NUMBER(3) REFERENCES dpto);</pre>	<pre>db.createCollection("dpto") db.createCollection("empleado")</pre>
CREATE INDEX indiceDep ON empleado (dep);	db.empleado.ensureIndex({dep: 1})
DROP TABLE empleado;	<i>/Suponiendo que empleado tiene una clave dep.</i> db.empleado.drop()

En general, la flexibilidad y la riqueza de las qrys no son demasiado elevadas, puesto que

lo que se prima por encima de las consultas es el rendimiento y la escalabilidad, por lo que es habitual que se delegue a la aplicación el implementar opciones más avanzadas en este terreno.

Un método de consulta llamado Companion SQL database consiste en tener una base de datos auxiliar (que puede ser una base de datos SQL o una TextDB) de forma que se utilice esta secundaria para almacenar ciertos metadatos importantes para realizar la búsqueda, y se empleen para facilitar la búsqueda posterior en el contenedor NoSQL.

SQL	MongoDB	Resultado en MongoDB
SELECT * FROM empleado WHERE edad < 30 AND grado = 8;	db.empleado.find((edad: {\$lt: 30}, grado: 8))	{ "_id": 1, "nombre": "Juan", "edad": 23, "grado": 8, "dep": 10 }
SELECT * FROM empleado WHERE nombre IN ('Pablo', 'Ana');	db.empleado.find({ nombre: { \$in: ["Pablo", "Ana"] } })	{ "_id": 2, "nombre": "Pablo", "edad": 25, "grado": 10, "dep": 10 } { "_id": 3, "nombre": "Ana", "edad": 11, "grado": 6, "dep": 20 }
SELECT * FROM empleado WHERE nombre LIKE "%blo";	db.empleado.find((nombre/./blo/))	{ "_id": 2, "nombre": "Pablo", "edad": 25, "grado": 10, "dep": 10 }
SELECT SUM(edad) AS total FROM empleado;	db.empleado.aggregate([{\$group: { _id: null, total: {\$sum: "\$edad"} }])	[{ "_id": null, "total": 69 }]
SELECT dep. MAX(grado) AS mg FROM empleado GROUP BY dep;	db.empleado.aggregate(\$group: { _id: "\$dep", mg: {\$max: "\$grado"} }])	[{ "_id": 10, "max": 10 }, { "_id": 20, "mg": 6 }]

Búsqueda local dispersa. Otra forma de realizar consultas consiste en, puesto que se tiene el conjunto de datos repartido entre los distintos servidores, repartir de igual modo la consulta, de forma que cada servidor ejecute localmente cada consulta y reenvíe los resultados a un nodo maestro, que sería el encargado de juntar todos los resultados y presentárselos a la aplicación.

Arboles B+ Distribuidos.

Una forma eficiente de acelerar las búsquedas consiste en mantener un árbol B+ que forme un índice de entradas a la base de datos NoSQL (Aquilera, Golab and Shah 2008).

El procedimiento consistiría en sacar los valores hash de los atributos que nos interese indexar, y construir con ellos el árbol B+. Cuando se quiera realizar una consulta, se comenzará desde la raíz y se irá descendiendo en orden hasta llegar a la hoja correspondiente, que nos dará la entrada concreta donde se encuentra el registro que se está buscando.

Operación	SQL	MongoDB
Creación base de datos	CREATE DATABASE nombreBD;	USE nombreBD
Creación tabla (en SQL) / Colección (en MongoDB)	CREATE TABLE nombreTabla (atributo1 tipo_de_dato restricciones <, atributo2...>);	db.createCollection("nombreColección")
Creación índice	CREATE INDEX nombreIndice ON nombreTabla (atributo1 <, atributo2... >);	db.nombreColeccion.ensureIndex({ "atributo": 1 })
Destrucción tabla (en SQL) / Colección (en MongoDB)	DROP TABLE nombreTabla;	db.nombreColeccion.drop()

Al tratarse de un árbol B+, se debe tener en cuenta las particularidades de este tipo de estructura de datos a la hora de realizar los mantenimientos necesarios, las inserciones y borrados que puedan hacer redimensiones en el árbol, etc.

iii. Comparacion de distintos tipos de base de datos NoSQL

Dependiendo de la forma en la que almacenen la información, nos podemos encontrar varios tipos distintos de bases de datos NoSQL. Veamos los tipos más utilizados.

- Bases de datos clave – valor
- Bases de datos documentales
 - Bases de datos en grafo
 - Bases de datos orientadas a objetos






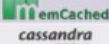

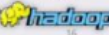
Documentales	Datos semi-estructurados en documentos (XML, YAML, JSON y BSON)	 elastic  MongoDB
Grafo	Datos estructurados como nodos relacionados entre si	 Neo4j  OPENLINK SOFTWARE <small>Managing Relationships. Revolutionizing Data.</small>
Clave / valor	Datos estructurados como clave / valor	 redis  MemCached
Familia de columnas	Datos estructurados en columnas donde cada fila puede tener una configuración diferente	 cassandra  hadoop

Tabla comparativa			
	MongoDB	Cassandra	DynamoDB
Descripción	Servicio de base de datos alojado y escalable de Amazon con los datos almacenados en la nube de Amazonas	Almacén de gran columna basado en ideas de BigTable y DynamoDB	Una de las tiendas de documentos más populares
Modelo de base de datos primaria	Document store: Las tiendas de documentos, también llamadas sistemas de bases de datos orientadas a documentos, se caracterizan por su organización de datos sin esquema. Key-value store: Las tiendas de valores clave son probablemente la forma más simple de sistemas de administración de bases de datos. Solo pueden almacenar pares de claves y valores, así como también recuperar valores cuando se conoce una clave.	Wide column store: Las tiendas de columnas anchas, también llamadas tiendas de discos extensibles, almacenan datos en registros con la capacidad de mantener un gran número de columnas dinámicas.	Document store: Las tiendas de documentos, también llamadas sistemas de bases de datos orientadas a documentos, se caracterizan por su organización de datos sin esquema.
Versión inicial	2012	2008	2009
Basado en la nube	sí	no	no
Lenguaje de implementación		Java	C ++
Esquema de datos	sin esquema	sin esquema	sin esquema
Índices secundarios	sí	restringido	sí
SQL	no	Sentencias DML y DDL similares a SQL (CQL)	no
API y otros métodos de acceso	API RESTful HTTP	Protocolo propietario Ahorro	protocolo propietario usando JSON
Lenguajes de programación admitidos	.Net Erlang Groovy Java JavaScript Perl PHP	C# C++ Clojure Erlang Go Haskell Java	C C# C++ Clojure info D info Erlang Go info

	Python Ruby	JavaScript info Perl PHP Python Ruby Scala	Groovy info Haskell Java JavaScript PHP PowerShell info Prolog info Python Ruby
Scripts del lado del servidor	no	no	JavaScript
Triggers	sí	sí	no
Métodos de particionamiento	Sharding	Sharding	Sharding
Métodos de replicación	sí	factor de replicación seleccionable	Replicación maestro-esclavo
Llaves Foraneas	no	no	no
Conceptos de transacción	no	no	no
Concurrencia	sí	sí	sí
Durabilidad	sí	sí	sí
Entornos de operación	hosting de amazon	BSD Linux OS X Windows	Linux OS X Solaris Windows
Ventajas	<p>Totalmente administrado</p> <p>Amazon DynamoDB es un servicio de base de datos NoSQL en la nube totalmente administrado; usted simplemente crea una tabla de base de datos, establece su rendimiento y deja que el servicio se encargue del resto.</p>	<p>Ningún punto único de falla garantiza el 100% de disponibilidad.</p> <p>Simplicidad operativa para el menor costo total de propiedad.</p> <p>La mejor escalabilidad de las plataformas NoSQL.</p>	<p>MongoDB mantiene las características más valiosas de las bases de datos relacionales: coherencia fuerte, lenguaje de consulta expresivo e índices secundarios. Como resultado, los desarrolladores pueden construir aplicaciones altamente funcionales más rápido que las bases de datos NoSQL.</p> <p>MongoDB proporciona flexibilidad de modelo de datos, escalabilidad elástica y alto rendimiento y disponibilidad de bases de datos</p>

			NoSQL. Como resultado, los ingenieros pueden mejorar continuamente las aplicaciones y entregarlas a escala casi ilimitada en hardware básico.
Desventajas	<p>- DynamoDB es una base de datos NoSQL. Eso significa que no puede hacer consultas complejas o de unión en dynamoDB. Además, no ACID, ya que no es un RDBMS</p> <p>- No se pueden escribir objetos grandes (BLOB) en dynamo DB. Definitivamente puede almacenar metadatos para BLOBS en dynamoDB, pero objetos reales que puede almacenar en S3. La recuperación del ajuste S3 es otra cosa que debe tener en cuenta para obtener un mejor rendimiento, como el uso de orden lexicográfico, el uso el uso de orden lexicográfico, el uso aleatorio y la paralelización de los GET.</p>	<p>La desventaja más grande para Cassandra es que no es compatible con los índices B-Tree y, por lo tanto, no admite consultas de rango.</p> <p>-Los CounterColumns no son un sustituto del autoincrement de las bases relacionales. Los contadores no son muy confiables. Si está haciendo decrementos, este puede no ser el resultado deseado.</p>	<p>No utiliza joins. Se diseña como si nunca hubieran sido una opción.</p> <p>Se escala bien en un rango estrecho, pero otras soluciones NoSQL son mejores para escalar.</p> <p>Sin transacciones, excepto en el nivel de registro. Si desea una transacción, debe ajustar toda la transacción en un solo registro.</p> <p>Difícil de asegurar correctamente sin contar con una licencia Enterprise.</p> <p>No hay parches, tiene que hacer actualizaciones completas y las actualizaciones completas y las actualizaciones completas se emiten varias veces al año. Entonces tienes que construir un programa de parches alrededor de ellos. No hay garantía de que su actualización funcione con su controlador dado, por lo que las pruebas deben programarse antes de la actualización.</p>

CouchDB:

CouchDB es catalogado muchas veces como una base de datos NoSQL, un término que se hizo cada vez más popular a finales de 2009 y principios de 2010. Si bien este término es una caracterización más bien genérica de una base de datos, o almacén de datos, sí define claramente un descanso de SQL tradicional bases de datos. Una base de datos CouchDB carece de un esquema o estructuras de datos pre-definidos rígidos tales como tablas.

Los datos almacenados en CouchDB es un documento (s) JSON. La estructura de los datos, o documento(s), puede cambiar dinámicamente para adaptarse a las necesidades cambiantes. CouchDB es una base de datos que abarca por completo la web. Almacene sus datos con documentos JSON. Tenga acceso a sus documentos y consultar sus índices con su navegador web, a través de HTTP. Índice, combinar y transformar sus documentos con JavaScript. CouchDB funciona bien con la web moderna y aplicaciones móviles. Usted puede incluso servir aplicaciones web directamente de CouchDB. Y usted puede distribuir sus datos o sus aplicaciones, de manera eficiente mediante la replicación incremental de los CouchDB. CouchDB soporta configuraciones maestro-maestro con detección automática de conflictos. CouchDB viene con una serie de características, como la transformación de documentos sobre la marcha y notificaciones de cambio en tiempo real, que hace que el desarrollo de aplicaciones web una brisa. Incluso viene con un fácil utilizar la consola de administración web.

Las principales características son las siguientes:

- **Almacenamiento de documentos:**

Almacena los datos como documentos esto es, uno o más pares campo/valor expresados en JSON. Los valores de los campos pueden ser datos simples como cadenas de caracteres, números o fechas. Pero también se pueden usar listas ordenadas y vectores asociativos. Todos los documentos en una base de datos CouchDB tienen un identificador único y no requieren un esquema determinado.

- **Vistas e índices Map/Reduce:**

Los datos almacenados se estructuran por medio de vistas. En CouchDB, cada vista se construye por medio de una función JavaScript que actúa como la mitad Map de una operación map/reduce. La función recibe un documento y lo transforma en un único valor, retornándolo. CouchDB puede indexar vistas y mantener actualizados esos índices a medida que se agregan, eliminan o actualizan documentos.

- **Arquitectura distribuida con replicación:**

Se diseñó con teniendo en mente la replicación bidireccional (o sincronización) y la operación off-line. Eso significa que múltiples réplicas pueden tener cada una sus propias copias de los mismos datos, modificarlas y luego sincronizar esos cambios en un momento posterior.

- **Interfaz REST:**

Todos los ítems tienen una URI única que queda expuesta vía HTTP. REST usa los métodos HTTP POST, GET, PUT y DELETE para las cuatro operaciones básicas CRUD (Create, Read, Update, Delete) con todos los recursos.

- **Consistencia Eventual:**

Garantiza consistencia eventual para poder ofrecer tanto disponibilidad como tolerancia a las particiones.

- **Hecha para operar offline:**

Puede replicar datos a dispositivos (como smartphones) que pueden quedar offline y manejar automáticamente la sincronización de los datos cuando el dispositivo vuelve a estar en línea.

Neo4j:

Es una base de datos orientada a grafos escrita en Java, es decir la información se almacena de forma relacionada formando un grafo dirigido entre los nodos y las relaciones entre ellos. Se integra perfectamente con múltiples lenguajes como Java, PHP, Ruby, .Net, Python, Node, Scala, etc. La base de datos está embebida en un servidor Jetty. Está especialmente indicada para modelar redes sociales y sistemas de recomendación.

Se distribuye en dos versiones: la community edition (open source) y la Enterprise edition. Para hacer pruebas de concepto nos basta con la community edition pero si quieres sacarle todo el partido a Neo4j la opción enterprise es la más recomendable ya que permite ponerla en cluster, monitorización, backups en caliente y un sistema de cache de alto rendimiento, además de soporte de sus creadores.

Otra de las ventajas que tiene Neo4j es que se pueden efectuar las consultas directamente a través de un API Rest lo que hace especialmente interesante su integración con aplicaciones web.

Principales características de neo4j:

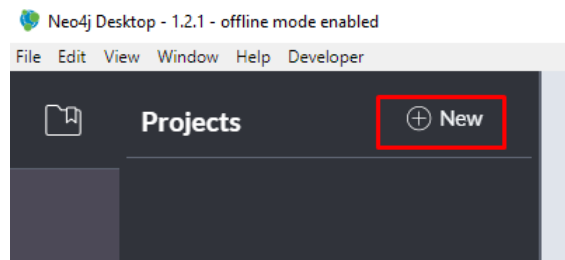
- Alto desempeño y alta disponibilidad (Escalamiento de lectura) Soporte sólido y real para transacciones ACID.
- Escalable: 32 miles de millones de Nodos, 32 miles de millones de Relaciones, 64 miles de millones de Propiedades.
- Servidor con una API REST o usable como una biblioteca Java.

IV. RESULTADOS

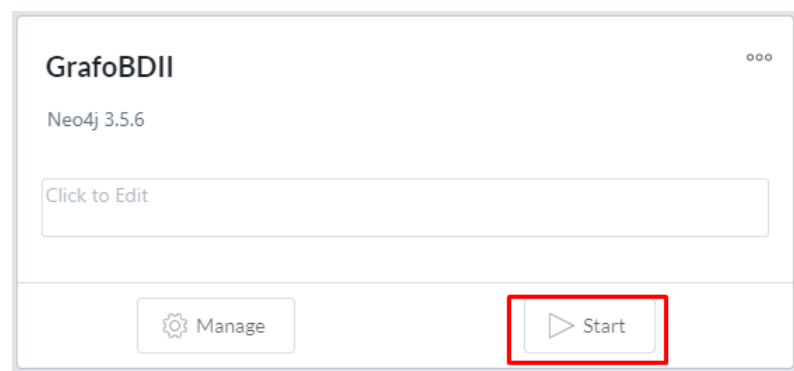
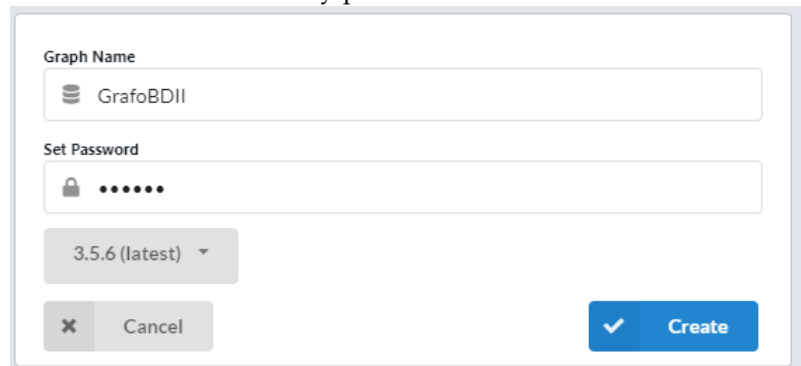
Comparaciones de 2 Bases de Datos NoSQL

i. Grafos

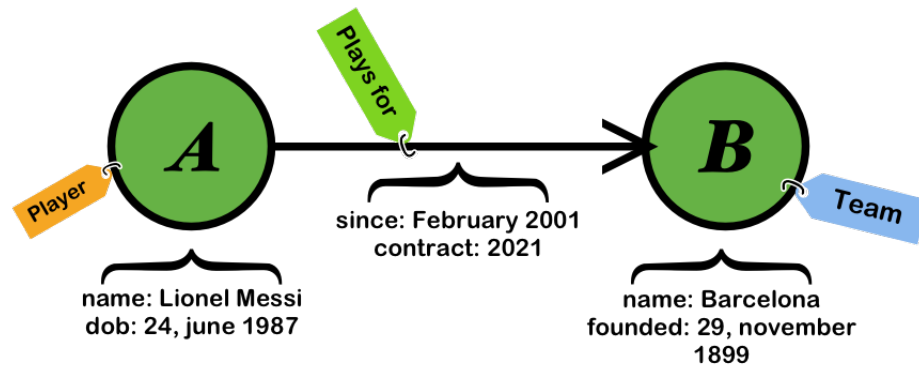
En este caso usaremos Neo4j la cual es un base de datos NoSql, procedemos a descargar y a crear un nuevo proyecto.



Le damos un nombre a nuestro nodo y procedemos a iniciarlo.

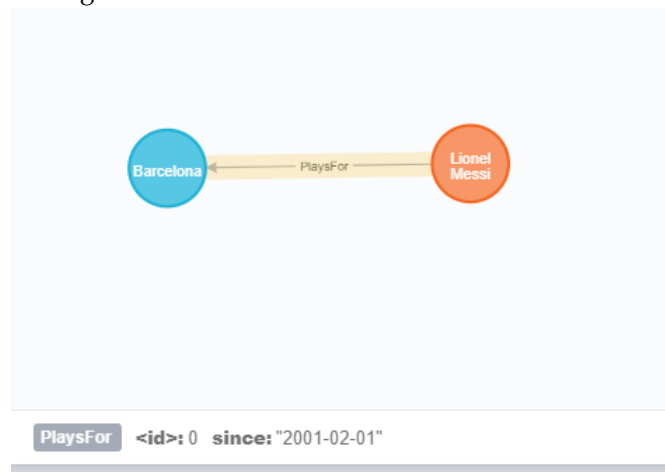


Crearemos el siguiente nodo:



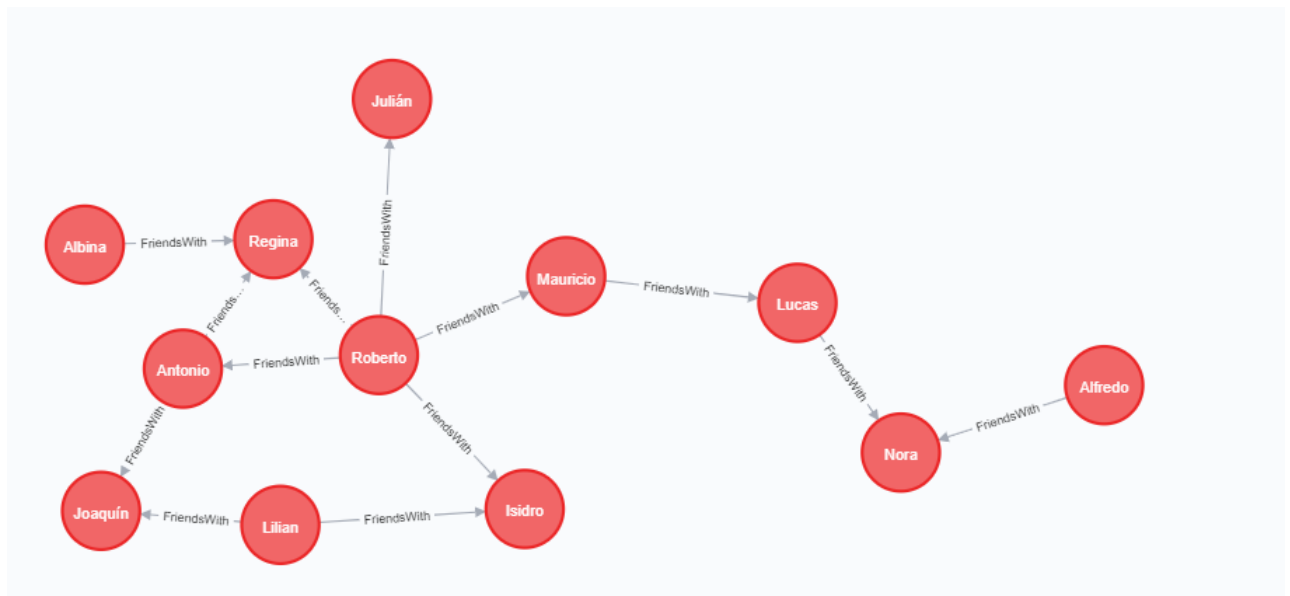
```
1 CREATE
2   (m:Player{name:"Lionel Messi"}),
3   (b:Team{name:"Barcelona"})
4 WITH m, b
5 CREATE (m)-[p:PlaysFor]->(b)
6 SET p.since=date("2001-02-01")
7 RETURN m, p, b
```

Y el resultado sera el siguiente:



Ahora haremos un ejemplo mas complejo, creamos una red de amigos:

```
1 CREATE (rob:Person{name:'Roberto'}), (isidro:Person{name:'Isidro'}),
2      (tony:Person{name:'Antonio'}), (nora:Person{name:'Nora'}),
3      (lily:Person{name:'Lilian'}), (freddy:Person{name:'Alfredo'}),
4      (lucas:Person{name:'Lucas'}), (mau:Person{name:'Mauricio'}),
5      (alb:Person{name:'Albina'}), (reg:Person{name:'Regina'}),
6      (j:Person{name:'Joaquín'}), (julian:Person{name:'Julián'})
7 // CREANDO RELACION
8 CREATE
9      (rob)-[:FriendsWith]->(isidro), (rob)-[:FriendsWith]->(tony), (rob)-[:FriendsWith]->(reg),
10     (rob)-[:FriendsWith]->(mau), (rob)-[:FriendsWith]->(julian),
11     (tony)-[:FriendsWith]->(reg), (tony)-[:FriendsWith]->(j),
12     (alb)-[:FriendsWith]->(reg), (lily)-[:FriendsWith]->(isidro), (lily)-[:FriendsWith]->(j),
13     (mau)-[:FriendsWith]->(lucas), (lucas)-[:FriendsWith]->(nora), (freddy)-[:FriendsWith]->(nora);
```



Ahora averigüemos quiénes son amigos de Lucas :

```
1 MATCH friends=(a:Person{name:'Lucas'})-[:FriendsWith]-(friend)
2 RETURN friends
```

Si queremos buscar los amigos de los amigos de Lucas sería con la siguiente consulta:

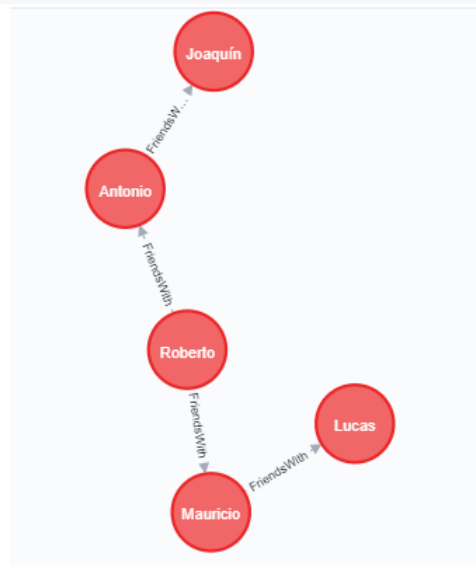

```
$ MATCH friends=(a:Person{name:'Lucas'})-[:FriendsWith*3]-(friend) RETURN friend.name
```

friend.name
"Isidro"
"Antonio"
"Regina"
"Julián"

Esto haciendolo desde SQL seria muy dificil y tedioso pero aca lo hacemos de una manera muy facil.

Finalmente podemos buscar el camino mas corto entre Joaquin y Lucas:

```
MATCH (lucas:Person{name:'Lucas'}), (joaquin:Person{name:'Joaquín'}),  
  p = shortestPath((lucas)-[*]-(joaquin))  
RETURN p
```



ii. Base de datos Documental

Una base de datos documental, también llamada una base de datos orientada a documentos u tienda de documentos, es un subconjunto de un tipo de base de datos NoSQL. Algunos almacenes de documentos también pueden ser bases de datos de valores clave. Una base de datos de documentos se utiliza para almacenar, recuperar y administrar datos semiestructurados. A diferencia de las bases de datos relacionales tradicionales, el modelo de datos en una base de datos de documentos no está estructurado en un formato de tabla de filas y columnas. El esquema puede variar, proporcionando mucha más flexibilidad para el modelado de datos que las bases de datos relacionales. Las bases de datos documental almacena cada registro y sus datos asociados en un solo documento. Cada documento contiene datos semiestructurados que pueden ser consultados con el uso de varias herramientas de consulta y análisis del DBMS.

Funcionamiento

Una base de datos de documentos utiliza documentos como la estructura para almacenamiento y consultas. En este caso, el término "documento" puede referirse a un documento

de texto, pero comúnmente es un archivo de XML o JSON. En lugar de columnas con nombres y tipos de datos que se utilizan en una base de datos relacional, un documento contiene una descripción del tipo de datos y el valor de esa descripción.

Ejemplo de Base de Datos Documental en XML

Ejemplo en XML:

```
<artist>
  <artistname>Iron Maiden</artistname>
  <albums>
    <album>
      <albumname>The Book of Souls</albumname>
      <datereleased>2015</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Killers</albumname>
      <datereleased>1981</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Powerslave</albumname>
      <datereleased>1984</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Somewhere in Time</albumname>
      <datereleased>1986</datereleased>
      <genre>Hard Rock</genre>
    </album>
  </albums>
</artist>
```

Ejemplo de Base de Datos Documental en Json

Ejemplo en JSON:

```
{
  '_id' : 1,
  'artistName' : { 'Iron Maiden' },
  'albums' : [
    {
      'albumname' : 'The Book of Souls',
      'datereleased' : 2015,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Killers',
      'datereleased' : 1981,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Powerslave',
      'datereleased' : 1984,
      'genre' : 'Hard Rock'
    }, {
      'albumname' : 'Somewhere in Time',
      'datereleased' : 1986,
      'genre' : 'Hard Rock'
    }
  ]
}
```

Beneficios

Las tiendas de documentos ofrecen importantes ventajas cuando se requieren características específicas, que incluyen: Modelado flexible de datos: a medida que las aplicaciones web, móviles, sociales e IoT cambian la naturaleza de los modelos de datos de aplicaciones, las bases de datos de documentos eliminan la necesidad de forzar modelos de datos relacionales para admitir nuevos tipos de modelos de datos de aplicaciones. Rendimiento de escritura

rápido: a diferencia de las bases de datos relacionales tradicionales, algunas bases de datos de documentos priorizan la disponibilidad de escritura sobre la estricta consistencia de los datos. Esto garantiza que las escrituras siempre serán rápidas, incluso si una falla en una parte del hardware o de la red da como resultado un pequeño retraso en la replicación de datos y la coherencia en todo el entorno. Rendimiento rápido de consultas: muchas bases de datos de documentos tienen potentes motores de búsqueda y funciones de indexación que proporcionan capacidades de consulta rápidas y eficientes.

iii. Base de Datos Valor - Clave

Una base de datos de valores-clave (también conocida como almacén de valores-clave y base de datos key-value) es un tipo de base de datos NoSQL que utiliza un método simple de clave / valor para almacenar datos. La parte clave-valor se refiere al hecho de que la base de datos almacena datos como una colección de pares clave / valor. Este es un método simple de almacenar datos, y se sabe que escala bien. El par clave-valor es un concepto bien establecido en muchos lenguajes de programación. Los lenguajes de programación normalmente se refieren a una clave-valor como una matriz asociativa o estructura de datos. Un valor-clave también se conoce comúnmente como diccionario o hash. Un store de valores-clave o una base de datos de valores-clave es una base de datos simple que usa un arreglo asociativo (piensa en un mapa o diccionario) como el modelo de datos fundamental donde cada clave está asociada con un solo valor en una colección. Esta relación se conoce como un par clave-valor.

Ejemplos de base de datos clave-valor

A continuación hay ejemplos de tiendas de valores clave. Estos son ejemplos simples, pero el objetivo es proporcionar una idea de cómo funciona una base de datos clave-valor.

Directorio

Clave	Valor
Juan	(123) 456-7890
Pedro	(234) 567-8901
Maria	(345) 678-9012
Francisca	(456) 789-0123

¿Qué tipo de datos se pueden almacenar en una base de datos clave-valor?

La clave

La clave en un par clave-valor debe (o al menos, debería) ser única. Este es el identificador único que le permite acceder al valor asociado con esa clave. En teoría, la clave podría ser cualquier cosa. Pero esto puede depender del DBMS. Un DBMS puede imponer limitaciones mientras que otro puede imponer ninguno. En Redis, por ejemplo, el tamaño de clave máximo permitido es 512 MB. Puede usar cualquier secuencia binaria como clave, desde una secuencia de texto corta hasta el contenido de un archivo de imagen. Incluso la cadena vacía es una clave válida. Sin embargo, por motivos de rendimiento, debe evitar tener una clave demasiado larga. Pero una clave demasiado corta también puede causar problemas de legibilidad. En cualquier caso, la clave debe seguir una convención convenida para mantener las cosas consistentes.

El valor

El valor en un almacén de clave-valor puede ser cualquier cosa, como texto (largo o corto), un número, código de marcado como HTML, código de programación como PHP, una imagen, etc. El valor también podría ser una lista, o incluso otro par clave-valor encapsulado en un objeto. Algunos DBMS de almacenamiento de claves le permiten especificar un tipo de datos

para el valor. Por ejemplo, puede especificar que el valor sea un entero. Otros DBMS no proporcionan esta funcionalidad y, por lo tanto, el valor podría ser de cualquier tipo.

¿Para qué se puede usar una base de datos de valores-clave?

Las bases de datos de valores clave se pueden aplicar a muchos escenarios. Por ejemplo, las tiendas de valores clave pueden ser útiles para almacenar cosas como las siguientes.

V. DISCUSIÓN

VI. CONCLUSIONES

REFERENCES

1. <http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/23/21>
2. <https://programarfacil.com/blog/que-es-un-orm/>
3. <https://www.beeva.com/beevea-view/tecnologia/mas-alla-de-la-virtualizacion-contenedores/>
4. <https://searchdatacenter.techtarget.com/es/definicion/virtualizacion-basada-en-contenedores-virtualizacion-a-nivel-de-sistema-operativo>
5. <https://www.incibe-cert.es/blog/asegurando-virtualizacion-tus-sistemas-control>
6. <http://www.datakeeper.es/?p=716>
7. <https://sigmodrecord.org/publications/sigmodRecord/1012/pdfs/04.surveys.cattell.pdf>
8. <https://ieeexplore.ieee.org/abstract/document/6625441>
9. <http://nosql-database.org/>
10. <https://www.oracle.com/technetwork/es/articles/datawarehouse/oracle12c-docker-win10-4485487-esa.html>