

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN



**Arquitectura de contenedores y ETL: Integración eficiente de bases de datos y
APIs en un entorno dockerizado**

Asignatura: Modelado Avanzado de Base de Datos

NRC: 10052

Integrantes: Jacome Ivonne

Rodriguez Danny

Salazar Daniel

Docente: Alexis Dario Estevez Salazar

2023-2024

Sangolqui-Quito

INTRODUCCION

Tenemos como objetivo describir la arquitectura de un sistema basado en contenedores y su integración con bases de datos, APIs y procesos de extracción, transformación y carga de datos (ETL).

En el contexto del proyecto, se han creado contenedores MASTER, CONFIG-SERVER, MONGO-ROUTER que alojan bases de datos dockerizadas y replicadas. Estas bases de datos están estrechamente vinculadas con una API y un sistema CRUD, permitiendo así la gestión eficiente de la información. Por un lado, tenemos otro contenedor MONGODB que cuenta con dos esclavos, en su arquitectura, se establece una conexión a través de un ETL, esta segunda base de datos también está dockerizada y posee una API que únicamente permite operaciones de lectura.

Esta estructura de contenedores y la configuración de la arquitectura permiten asegurar una mayor escalabilidad, modularidad y disponibilidad del sistema, así como una gestión eficiente de los datos.

El uso de contenedores dockerizados facilita la portabilidad y el despliegue de los componentes del sistema, mientras que la replicación de las bases de datos garantiza la redundancia y la tolerancia a fallos.

A lo largo de este informe, se detallarán las características y funcionalidades de cada componente, así como los beneficios que aporta esta arquitectura en términos de rendimiento, escalabilidad y flexibilidad. También se explorarán las consideraciones técnicas y los desafíos asociados a la implementación y mantenimiento de este sistema basado en contenedores y su integración con bases de datos y ETL.

DESARROLLO

Partes que contiene:

ETL (Extract, Transform, Load):

Es un proceso utilizado en la gestión y análisis de datos. Consiste en la extracción de datos de diversas fuentes, su transformación para adaptarlos a un formato y estructura común, y finalmente cargarlos en un destino, como una base de datos o un data warehouse.

Etapas principales del proceso ETL:

- **Extracción (Extract):** En esta etapa, los datos se extraen de múltiples fuentes, como bases de datos, archivos planos, aplicaciones en la nube, servicios web, entre otros. Se realiza una lectura de los datos y se recopilan en bruto para su posterior procesamiento.
- **Transformación (Transform):** Una vez que los datos se han extraído, se someten a diversas transformaciones y limpiezas para garantizar su calidad y coherencia. Durante esta etapa, se aplican reglas de negocio, se realizan cálculos, se normalizan los formatos y se filtran los datos no deseados. También se puede realizar la integración de datos provenientes de diferentes fuentes, para consolidarlos y tener una visión más completa.
- **Carga (Load):** En la etapa final, los datos transformados se cargan en un destino adecuado, como una base de datos relacional, un data warehouse o un sistema de almacenamiento en la nube. Se definen los esquemas y estructuras de la base de datos de destino y se insertan los datos transformados en ella.

DOCKERIZACION:

Implica dividir una aplicación en componentes más pequeños y empaquetarlos en contenedores independientes. Estos contenedores incluyen el código de la aplicación, las bibliotecas y dependencias necesarias, así como los archivos de configuración. Cada contenedor se ejecuta en un entorno aislado y se puede implementar y ejecutar en diferentes sistemas operativos y entornos de forma consistente. Además, permite una gestión más eficiente de los recursos, ya que los contenedores son ligeros y se pueden escalar fácilmente según la demanda de la aplicación. También simplifica la implementación y la administración de la aplicación, ya que los contenedores se pueden crear, actualizar y eliminar de forma rápida y sencilla.

SHARDING:

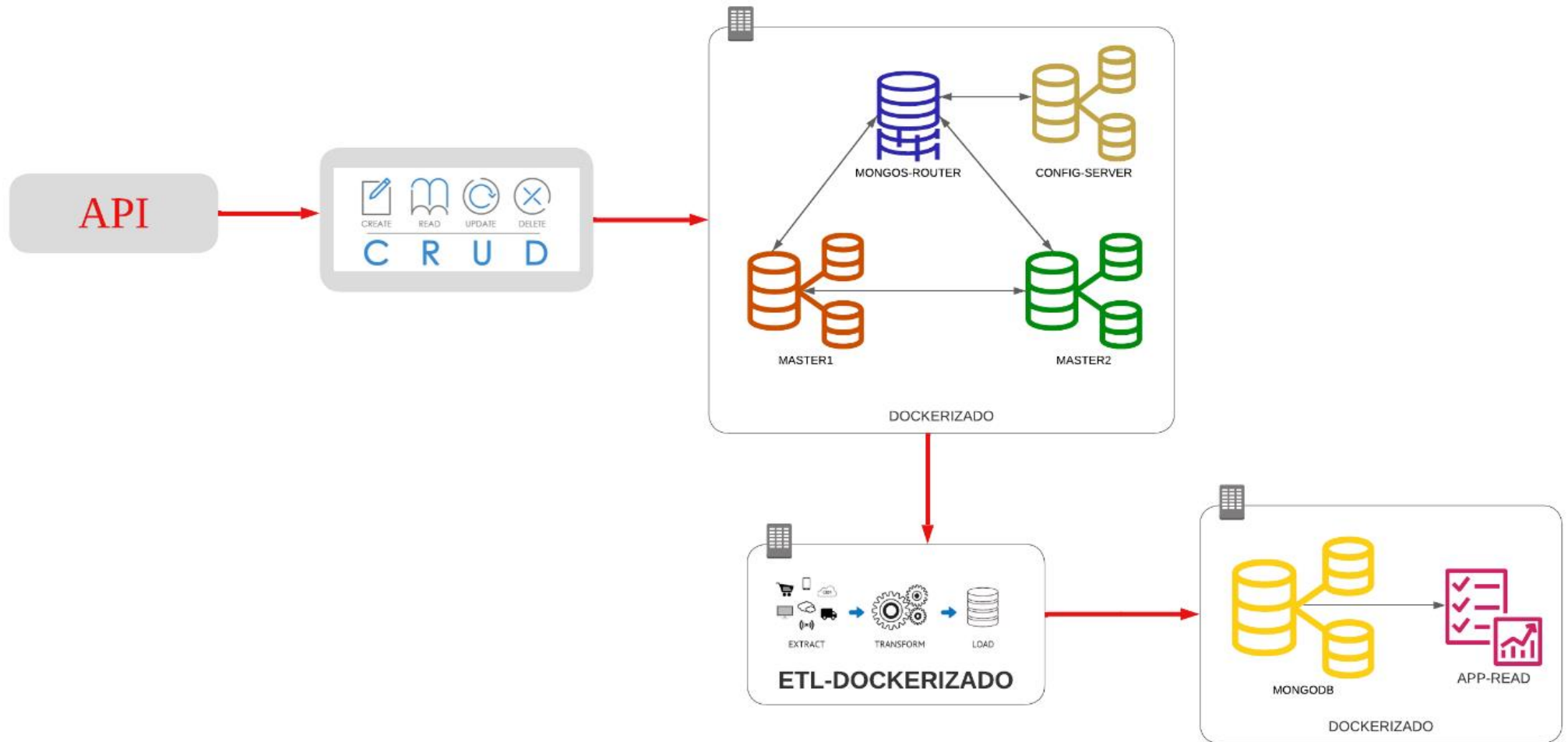
Consiste en dividir una base de datos grande en fragmentos más pequeños llamados "shards" o "fragmentos", y distribuir esos fragmentos en varios servidores. El objetivo principal del sharding es distribuir la carga de trabajo y los datos a través de múltiples servidores, lo que facilita la escalabilidad horizontal. A medida que crece la cantidad de datos o la demanda de acceso a la base de datos, se pueden agregar más servidores para manejar la carga adicional sin afectar significativamente el rendimiento.

API:

API (Application Programming Interface) es un conjunto de reglas y protocolos que permiten la comunicación entre diferentes componentes de software. Es una interfaz que define cómo los diferentes elementos de un sistema de software deben interactuar entre sí.

- **Node.js:** es un entorno de tiempo de ejecución de JavaScript que permite ejecutar código JavaScript en el lado del servidor. Es especialmente adecuado para aplicaciones web en tiempo real y escalables. Se puede utilizar Node.js para crear el servidor que alojará y ejecutar API.
- **Express:** es un marco de aplicación web para Node.js. Proporciona una capa de abstracción sobre Node.js que simplifica la creación de aplicaciones web y APIs. Express permite definir rutas, manejar solicitudes y respuestas HTTP, manejar middleware y mucho más. Utilizamos Express para crear las rutas y los controladores de la API.
- **MongoDB:** es una base de datos NoSQL orientada a documentos.

TOPOLOGIA LOGICA:



CONFIGURACION DEL SERVIDOR

- EJECUTAMOS EL CONTENEDOR DE

`docker-compose -f config-server/docker-compose.yaml up -d`

- ENTRAR AL SHELL DE MONGO DEL CONTENEDOR

`mongosh mongodb://192.168.100.85:40001`

- PARA INICIAR LA REPLICA

```
rs.initiate(  
  {  
    _id: "confServers",  
    configsvr: true,  
    members: [  
      { _id: 0, host: "192.168.100.85:40017" },  
      { _id: 1, host: "192.168.100.85:40018" },  
      { _id: 2, host: "192.168.100.85:40019" }  
    ]  
  }  
)
```

- CONFIGURACION DEL MASTER1

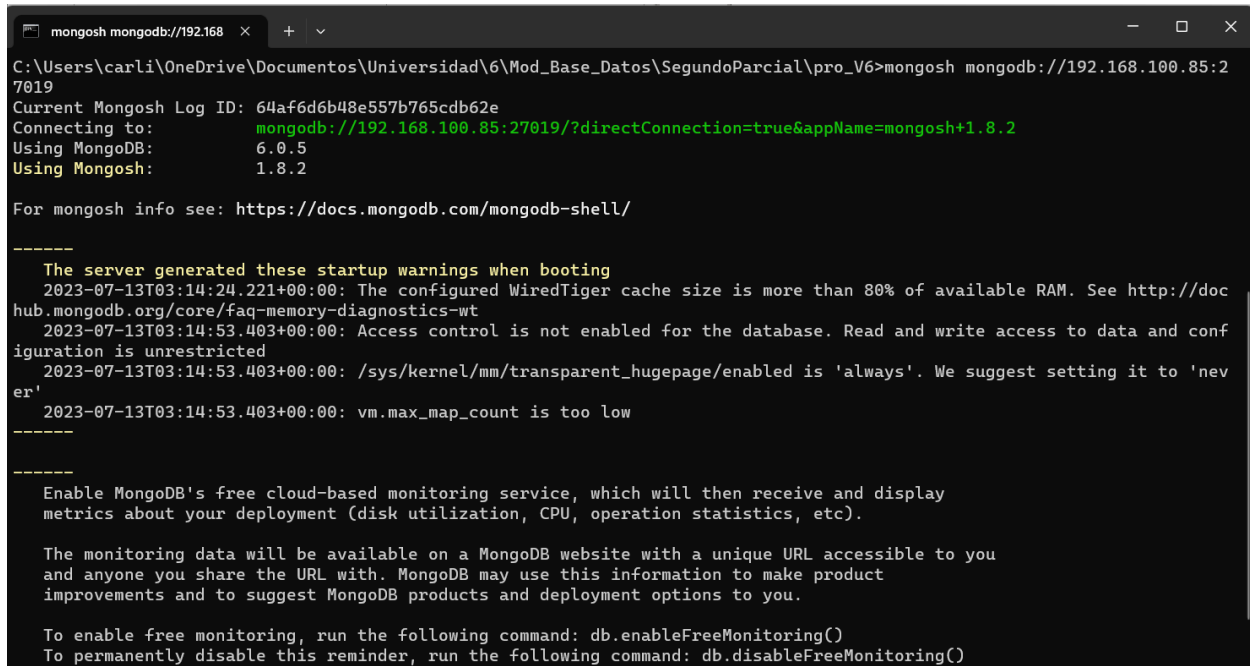
EJECUTAMOS EL CONTENEDOR

`docker-compose -f docker-compose.yaml up -d`

```
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>docker-compose -f docker-compose.y  
aml up -d  
time="2023-07-12T22:42:30-05:00" level=warning msg="Found orphan containers ([configsvr4 mongorouter02]) for this projec  
t. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag  
to clean it up."  
[+] Running 13/13  
✓ Container esc2a Running 0.0s  
✓ Container mongorouter Running 0.0s  
✓ Container esc2b Running 0.0s  
✓ Container esc1a Running 0.0s  
✓ Container master2 Running 0.0s  
✓ Container esc3a Running 0.0s  
✓ Container esc1b Running 0.0s  
✓ Container master1 Running 0.0s  
✓ Container configsvr1 Running 0.0s  
✓ Container esc3b Running 0.0s  
✓ Container configsvr2 Running 0.0s  
✓ Container configsvr3 Running 0.0s  
✓ Container master3 Running 0.0s
```

- ENTRAR AL SHELL DE MONGO DEL CONTENEDOR

mongosh mongodb://192.168.100.85:27019



```

mongosh mongodb://192.168
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>mongosh mongodb://192.168.100.85:27019
Current Mongosh Log ID: 64af6d6b48e557b765cdb62e
Connecting to:   mongodb://192.168.100.85:27019/?directConnection=true&appName=mongosh+1.8.2
Using MongoDB:   6.0.5
Using Mongosh:   1.8.2

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2023-07-13T03:14:24.221+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://doc
hub.mongodb.org/core/faq-memory-diagnostics-wt
2023-07-13T03:14:53.403+00:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
2023-07-13T03:14:53.403+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'nev
er'
2023-07-13T03:14:53.403+00:00: vm.max_map_count is too low
-----

-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()

```

- PARA INICIAR LA REPLICA DE CONFIGURACION DE LOS SERVIDORES

CONFIGSVR

```

rs.initiate(
{
  _id: "configserver",
  configsvr: true,
  members: [
    { _id : 0, host : "192.168.100.85:27019" },
    { _id : 1, host : "192.168.100.85:27020" },
    { _id : 2, host : "192.168.100.85:27021" }
  ]
}
)

```

```
test> rs.initiate(
...   {
...     _id: "configserver",
...     configsvr: true,
...     members: [
...       { _id : 0, host : "192.168.100.85:27019" },
...       { _id : 1, host : "192.168.100.85:27020" },
...       { _id : 2, host : "192.168.100.85:27021" }
...     ]
...   }
... )
{ ok: 1, lastCommittedOpTime: Timestamp({ t: 1689218564, i: 1 }) }
configserver [direct: other] test>

configserver [direct: primary] test> |
```

- ENTRAR AL SHELL DE MONGO DEL MASTER1 LADO IZQUIERDO

mongosh mongodb://192.168.100.85:27030

```
mongosh mongodb://192.168.100.85:27030
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>mongosh mongodb://192.168.100.85:27030
Current Mongosh Log ID: 64af7581c8eeb4e4093a9579
Connecting to:  mongodb://192.168.100.85:27030/?directConnection=true&appName=mongosh+1.8.2
Using MongoDB:  6.0.5
Using Mongosh:  1.8.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2023-07-13T03:14:20.915+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-07-13T03:14:49.502+00:00: Access control is not enabled for the database. Read and write access to data and configuration is un
restricted
2023-07-13T03:14:49.503+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
2023-07-13T03:14:49.503+00:00: vm.max_map_count is too low
-----

-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
```

- PARA INICIAR LA REPLICA DE CONFIGURACION DEL MASTER1RS LADO IZQUIERDO

```
rs.initiate(
{
  _id: "master1rs",
```



```

members: [
  { _id : 0, host : "192.168.100.85:27030" },
  { _id : 1, host : "192.168.100.85:27031" },
  { _id : 2, host : "192.168.100.85:27032" }
]
}
)

```

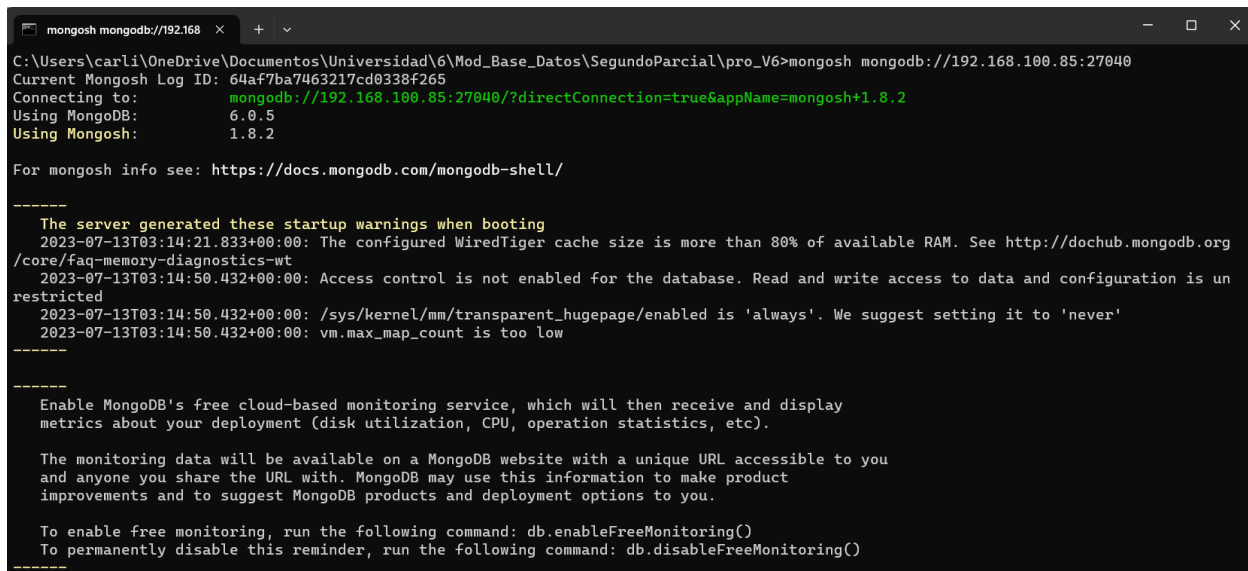
```

test> rs.initiate(
...  {
...    _id: "master1rs",
...    members: [
...      { _id : 0, host : "192.168.100.85:27030" },
...      { _id : 1, host : "192.168.100.85:27031" },
...      { _id : 2, host : "192.168.100.85:27032" }
...    ]
...  }
... )
{ ok: 1 }
master1rs [direct: other] test>

```

- ENTRAR AL SHELL DE MONGO DEL MASTER2 LADO IZQUIERDO

mongosh mongodb://192.168.100.85:27040



```

mongosh mongodb://192.168
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>mongosh mongodb://192.168.100.85:27040
Current Mongosh Log ID: 64af7ba7463217cd0338f265
Connecting to:  mongodb://192.168.100.85:27040/?directConnection=true&appName=mongosh+1.8.2
Using MongoDB:  6.0.5
Using Mongosh:  1.8.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2023-07-13T03:14:21.833+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-07-13T03:14:50.432+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-07-13T03:14:50.432+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
2023-07-13T03:14:50.432+00:00: vm.max_map_count is too low
-----

-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

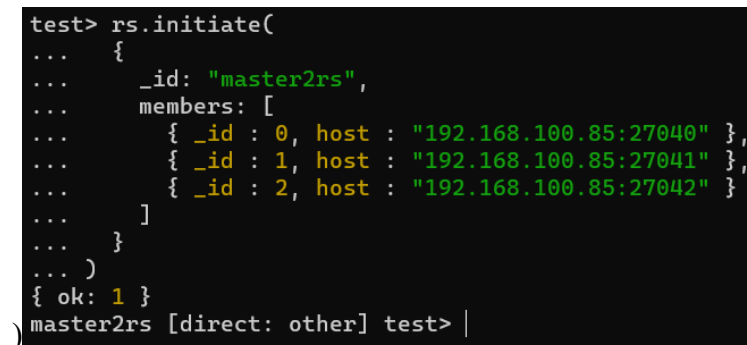
The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----

```

- PARA INICIAR LA REPLICA DE CONFIGURACION DEL MASTER2RS LADO IZQUIERDO

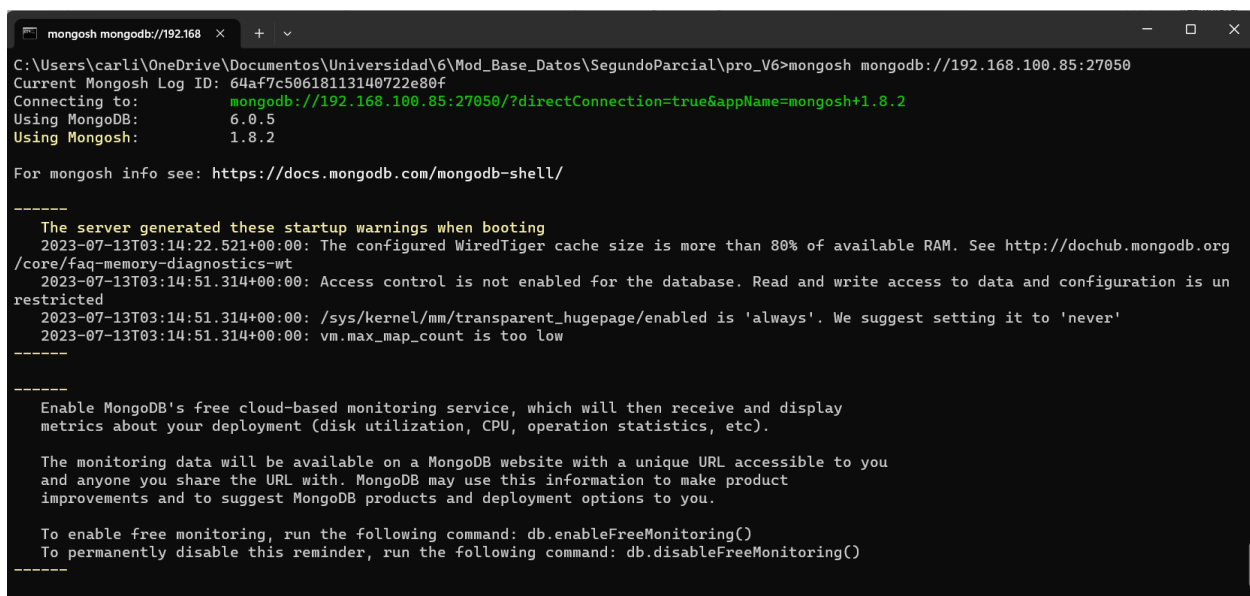
```
rs.initiate(
{
  _id: "master2rs",
  members: [
    { _id : 0, host : "192.168.100.85:27040" },
    { _id : 1, host : "192.168.100.85:27041" },
    { _id : 2, host : "192.168.100.85:27042" }
  ] })
```



```
test> rs.initiate(
... {
...   _id: "master2rs",
...   members: [
...     { _id : 0, host : "192.168.100.85:27040" },
...     { _id : 1, host : "192.168.100.85:27041" },
...     { _id : 2, host : "192.168.100.85:27042" }
...   ]
... }
... )
{ ok: 1 }
master2rs [direct: other] test> |
```

- ENTRAR AL SHELL DE MONGO DEL MASTER3 LADO IZQUIERDO

```
mongosh mongodb://192.168.100.85:27050
```



```
mongosh mongodb://192.168
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>mongosh mongodb://192.168.100.85:27050
Current Mongosh Log ID: 64af7c50618113140722e80f
Connecting to:  mongodb://192.168.100.85:27050/?directConnection=true&appName=mongosh+1.8.2
Using MongoDB:  6.0.5
Using Mongosh:  1.8.2

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2023-07-13T03:14:22.521+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-07-13T03:14:51.314+00:00: Access control is not enabled for the database. Read and write access to data and configuration is un
restricted
2023-07-13T03:14:51.314+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
2023-07-13T03:14:51.314+00:00: vm.max_map_count is too low
-----

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
```

- PARA INICIAR LA REPLICA DE CONFIGURACION DEL MASTER3RS LADO DERECHO

```
rs.initiate(
{
  _id: "master3rs",
  members: [
    { _id : 0, host : "192.168.100.85:27050" },
    { _id : 1, host : "192.168.100.85:27051" },
    { _id : 2, host : "192.168.100.85:27052" }
  ]
}
)
```

```
test> rs.initiate(
...  {
...    _id: "master3rs",
...    members: [
...      { _id : 0, host : "192.168.100.85:27050" },
...      { _id : 1, host : "192.168.100.85:27051" },
...      { _id : 2, host : "192.168.100.85:27052" }
...    ]
...  }
... )
{ ok: 1 }
master3rs [direct: other] test> |
```

- CONFIGURACION DEL MONGOS ROUTER

```
C:\Users\carli\OneDrive\Documentos\Universidad\6\Mod_Base_Datos\SegundoParcial\pro_V6>mongosh mongodb://192.168.100.85:27018
Current Mongosh Log ID: 64af7df75d4da36c3b3daa54
Connecting to:  mongodb://192.168.100.85:27018/?directConnection=true&appName=mongosh+1.8.2
Using MongoDB:  6.0.5
Using Mongosh:  1.8.2

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2023-07-13T03:14:16.315+00:00: Access control is not enabled for the database. Read and write access to data and configuration is un
restricted
-----
[direct: mongos] test> |
```

- AÑADIMOS LOS SHARDS DE CADA MASTER DEL LADO IZQUIERDO

```
sh.addShard("master1rs/192.168.100.85:27031,192.168.100.85:27032")
```

```
sh.addShard("master2rs/192.168.100.85:27041,192.168.100.85:27042")
```

```
[direct: mongos] test> sh.addShard("master1rs/192.168.100.85:27031,192.168.100.85:27032")
{
  shardAdded: 'master1rs',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1689222765, i: 7 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1689222765, i: 7 })
}
[direct: mongos] test> sh.addShard("master2rs/192.168.100.85:27041,192.168.100.85:27042")
{
  shardAdded: 'master2rs',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1689222777, i: 7 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1689222777, i: 7 })
}
```

- VEMOS QUE LOS SHARDS SON AÑADIDOS CORRECTAMENTE

```
mongosh mongodb://192.168
[direct: mongos] test> sh.status()
shardingVersion
{
  _id: 1,
  minCompatibleVersion: 5,
  currentVersion: 6,
  clusterId: ObjectId("64af6e0fe58aaa36ec3b2047")
}
---
shards
[
  {
    _id: 'master1rs',
    host: 'master1rs/192.168.100.85:27030,192.168.100.85:27031,192.168.100.85:27032',
    state: 1,
    topologyTime: Timestamp({ t: 1689222765, i: 4 })
  },
  {
    _id: 'master2rs',
    host: 'master2rs/192.168.100.85:27040,192.168.100.85:27041,192.168.100.85:27042',
    state: 1,
    topologyTime: Timestamp({ t: 1689222777, i: 5 })
  }
]
---
```

- CREAMOS LA BASE DE DATOS Y LA COLECCION

```
[direct: mongos] test> use jrs_proyecto
switched to db jrs_proyecto
[direct: mongos] jrs_proyecto>

[direct: mongos] jrs_proyecto> db
jrs_proyecto
[direct: mongos] jrs_proyecto> db.createCollection("anime")
{ ok: 1 }
```

Fragmentamos la colección en este caso es anime (min 5:13 fragmentar tiene que ser partes completas no solo el nombre de la colección así que las demostraciones de fragmentos son el punto de la base de datos y el nombre de la colección y luego una clave de fragmento)

sh.shardCollection("jrs_proyecto.anime", {"anime_id":"hashed"})

```
[direct: mongos] jrs_proyecto> sh.shardCollection("jrs_proyecto.anime", {"anime_id":"hashed"})
{
  collectionsharded: 'jrs_proyecto.anime',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1689223467, i: 36 }),
    signature: {
      hash: Binary(Buffer.from("00000000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1689223467, i: 32 })
}
[direct: mongos] jrs_proyecto> |
```

- HABILITAMOS LA FRAGMENTACION Y EL NOMBRE DE LA BASE DE DATOS

sh.enableSharding("jrs_proyecto")

```
[direct: mongos] jrs_proyecto> sh.enableSharding("jrs_proyecto")
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1689223737, i: 2 }),
    signature: {
      hash: Binary(Buffer.from("00000000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1689223737, i: 2 })
}
```

- VEMOS LA DISTRIBUCION DE LOS FRAGMENTOS DE CADA REPLICA

```
sh.addShard("master1rs/192.168.100.85:27030,192.168.100.85:27031,192.168.100.85:27032")
```

```
sh.addShard("master2rs/192.168.100.85:27040,192.168.100.85:27041,192.168.100.85:27042")
```

```
sh.addShard("master3rs/192.168.100.85:27050,192.168.100.85:27051,192.168.100.85:27052")
```

```
db.anime.getShardDistribution()
```

EJECUTAMOS EL CONTENEDOR

```
docker-compose -f mongos/docker-compose.yaml up -d
```

ENTRAR AL SHELL DE MONGO DEL CONTENEDOR

```
mongosh mongodb://192.168.100.85:60017
```

AÑADIR SHARDS

```
mongos>
```

```
sh.addShard("master1rs/192.168.100.85:27030,192.168.100.85:27031,192.168.100.85:27032")
```

```
mongos> sh.status()
```

ENTORNO DE API

Agregar Nuevo

Buscar por Anime ID

Buscar

ID	Anime ID	Nombre	Género	Tipo	Episodios	Rating	Members	Acciones
64b0573c2c07b8c13fe3b593	925329	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572	<div>Editar</div> <div>Eliminar</div>
64b0573c2c07b8c13fe3b594	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen	TV	51	9.16	151266	<div>Editar</div> <div>Eliminar</div>
64b0573c2c07b8c13fe3b595	32935	Haikyuu!! Karasuno Koukou VS Shiratorizawa Gakuen Koukou	Comedy, Drama, School, Shounen, Sports	TV	10	9.15	93351	<div>Editar</div> <div>Eliminar</div>
64b0573c2c07b8c13fe3b596	11061	Hunter x Hunter (2011)	Action, Adventure, Shounen, Super Power	TV	148	9.13	425855	<div>Editar</div> <div>Eliminar</div>
64b0573c2c07b8c13fe3b597	820	Ginga Eiyuu Densetsu	Drama, Military, Sci-Fi, Space	OVA	110	9.11	80679	<div>Editar</div> <div>Eliminar</div>

Siguiente

Ver transformación

EDITAR

Editar Anime

Anime ID:

9253293

Nombre:

Steins;Gate

Género:

Sci-Fi, Thriller

Tipo:

TV

Episodios:

24

Rating:

9.17

Members:

673572

Guardar

AGREGAR

Agregar Nuevo Anime

Anime ID:

1722365143

Nombre:

Daniel Salazar

Género:

masculino

Tipo:

TV

Episodios:

2

Rating:

6

Members:

1

Guardar

CONCLUSIONES

- En conclusión, la arquitectura de contenedores y ETL descrita ofrece una solución eficiente para la gestión de bases de datos y APIs en un entorno dockerizado. La estructura de contenedores permite una mayor escalabilidad, modularidad y disponibilidad del sistema, así como una gestión eficiente de los datos.
- Además, el uso de contenedores dockerizados ofrece ventajas en términos de portabilidad y despliegue de componentes del sistema. En resumen, esta arquitectura es una solución efectiva para la gestión de datos en entornos empresariales.