Importar datos en MongoDB con mongoimport

mongoimport es una herramienta para importar datos en MongoDB desde archivos en formato JSON, CSV o TSV. A continuación, desarrollaremos 5 ejercicios prácticos.

Requisitos:

1. Tener MongoDB instalado y configurado.
2. Disponer de archivos JSON o CSV para los diferentes ejemplos.
3. Asegurarse de que el servicio de MongoDB está corriendo.

1: Importar un archivo JSON simple en una base de datos

Paso 1: Crear un archivo JSON

Crea un archivo llamado *clientes.json* con el siguiente contenido:

```
[
  { "_id": 1, "nombre": "Juan", "edad": 28, "ciudad": "Madrid" },
  { "_id": 2, "nombre": "Ana", "edad": 34, "ciudad": "Barcelona" },
  { "_id": 3, "nombre": "Luis", "edad": 41, "ciudad": "Sevilla" }
]
```

Paso 2: Usar mongoimport para cargar los datos

Ejecuta el siguiente comando en tu terminal:

```
mongoimport --db taller_mongo --collection clientes --file clientes.json --jsonArray
```

Paso 3: Verificar la importación

Conéctate a la base de datos usando mongosh o cualquier cliente MongoDB y ejecuta:

```
use taller_mongo
db.clientes.find()
```

Resultado esperado: Los documentos del archivo JSON deben haberse importado en la colección clientes.

2: Importar un archivo CSV y especificar el delimitador

Paso 1: Crear un archivo CSV

Crea un archivo llamado *productos.csv* con el siguiente contenido:

```
_id,nombre,precio,categoria
1,Televisor,400,Electrónica
2,Microondas,150,Electrodomésticos
3,Cafetera,80,Cocina
```

Paso 2: Usar mongoimport para cargar los datos

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection productos --type csv --file productos.csv --headerline
```

- --type csv: Indica que el archivo es un CSV.
- --headerline: Usa la primera línea del archivo como nombres de campos.

Paso 3: Verificar la importación

En tu cliente MongoDB:

```
use taller_mongo
db.productos.find()
```

Los productos deben aparecer como documentos en la colección productos.

3: Importar un archivo JSON con una estructura anidada

Paso 1: Crear un archivo JSON

Crea un archivo llamado *pedidos.json* con el siguiente contenido:

```
[
  { "_id": 1, "cliente": "Juan", "productos": [{ "nombre": "Televisor", "cantidad": 1 }] },
  { "_id": 2, "cliente": "Ana", "productos": [{ "nombre": "Cafetera", "cantidad": 2 }] }
]
```

Paso 2: Usar mongoimport para cargar los datos

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection pedidos --file pedidos.json --jsonArray
```

Paso 3: Verificar la importación

En tu cliente MongoDB:

```
use taller_mongo
db.pedidos.find()
```

Los documentos deben reflejar los pedidos con los productos en formato anidado.

4: Sobrescribir documentos existentes durante la importación

Paso 1: Crear un archivo JSON actualizado

Crea un archivo llamado *clientes_actualizados.json*:

```
[
  { "_id": 1, "nombre": "Juan Pérez", "edad": 29, "ciudad": "Madrid" },
  { "_id": 2, "nombre": "Ana Gómez", "edad": 35, "ciudad": "Barcelona" }
]
```

Paso 2: Usar mongoimport con la opción --mode merge

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection clientes --file clientes_actualizados.json --jsonArray --mode merge
```

- --mode merge: Actualiza documentos existentes con el mismo _id.

Paso 3: Verificar los cambios

En tu cliente MongoDB:

```
use taller_mongo
db.clientes.find()
```

Los documentos deben haberse actualizado con los nuevos datos.

5: Importar un archivo CSV y asignar un índice único

Paso 1: Crear un archivo CSV

Crea un archivo llamado *usuarios.csv*:

```
_id,email,nombre,edad
1,juan@example.com,Juan,28
2,ana@example.com,Ana,34
3,luis@example.com,Luis,41
```

Paso 2: Usar mongoimport y crear un índice

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection usuarios --type csv --file usuarios.csv --headerline
```

Después, en tu cliente MongoDB, crea un índice único en el campo email:

```
use taller_mongo
db.usuarios.createIndex({ email: 1 }, { unique: true })
```

Paso 3: Verificar el índice

Consulta los índices de la colección:

```
db.usuarios.getIndexes()
```

El índice único debe aparecer en los resultados.

6: Importar datos desde una URL

Paso 1: Preparar una URL con datos JSON

Para este ejercicio, usaremos un archivo JSON disponible en la web. Puedes utilizar este ejemplo:

Descargar ejemplo JSON.

Paso 2: Usar mongoimport con --uri

Ejecuta el siguiente comando para importar los datos desde una URL:

✅ Opción 0: Usando Linux

```
mongoimport --uri "mongodb://localhost:27017/taller_mongo" --collection usuarios_remotos --file <(curl -s https://raw.githubusercontent.com/prust/wikipedia-movie-data/master/movies.json) --jsonArray
```

✅ Opción 1: Usando PowerShell

Descargar el archivo JSON:

```
Invoke-WebRequest -Uri "https://raw.githubusercontent.com/prust/wikipedia-movie-data/master/movies.json" -OutFile "movies.json"
```

Importar a MongoDB:

```
mongoimport --uri "mongodb://localhost:27017/taller_mongo" --collection usuarios_remotos --file "movies.json" --jsonArray
```

✅ Opción 2: Desde CMD (símbolo del sistema)

Abre CMD y ejecuta esto para descargar el archivo:

```
curl -o movies.json https://raw.githubusercontent.com/prust/wikipedia-movie-data/master/movies.json
```

Luego, importa con:

```
mongoimport --uri "mongodb://localhost:27017/taller_mongo" --collection usuarios_remotos --file movies.json --jsonArray
```

Paso 3: Verificar la importación

En tu cliente MongoDB:

```
use taller_mongo
db.usuarios_remotos.find()
```

Los datos de la URL se deben haber importado en la colección usuarios_remotos.

7: Ignorar documentos duplicados durante la importación

Paso 1: Crear un archivo JSON

Crea un archivo llamado clientes_duplicados.json con datos que incluyen duplicados en el campo _id:

```
[
  { "_id": 1, "nombre": "Carlos", "edad": 30, "ciudad": "Valencia" },
  { "_id": 2, "nombre": "Sofia", "edad": 25, "ciudad": "Madrid" },
  { "_id": 1, "nombre": "Carlos Duplicado", "edad": 31, "ciudad": "Barcelona"
}
]
```

Paso 2: Usar mongoimport con --stopOnError

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection clientes --file clientes_duplicados.json --jsonArray --stopOnError
```

- --stopOnError: Detiene la importación cuando encuentra documentos con claves duplicadas.

Paso 3: Verificar los datos

En tu cliente MongoDB:

```
db.clientes.find()
```

Solo se deben importar los documentos sin duplicados.

8: Importar datos desde un archivo comprimido

Paso 1: Comprimir un archivo JSON

Crea un archivo *productos.json* con el siguiente contenido:

```
[
  { "_id": 1, "nombre": "Laptop", "precio": 800 },
  { "_id": 2, "nombre": "Tablet", "precio": 300 }
]
```

Comprime el archivo usando gzip: `gzip productos.json`

Paso 2: Usar mongoimport con un archivo comprimido

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection productos_comprimidos --file productos.json.gz --jsonArray
```

Paso 3: Verificar los datos

En tu cliente MongoDB:

```
db.productos_comprimidos.find()
```

9: Importar un subconjunto de columnas desde un CSV

Paso 1: Crear un archivo CSV

Crea un archivo llamado *empleados.csv*:

```
_id,nombre,edad,departamento,salario
1,Pedro,30,IT,3000
2,Laura,28,RRHH,2500
3,Carlos,35,Marketing,4000
```

Paso 2: Usar mongoimport y seleccionar columnas

Ejecuta el siguiente comando para importar solo las columnas nombre y edad:

```
mongoimport --db taller_mongo --collection empleados --type csv --file empleados.csv --fields nombre,edad --headerline
```

- --fields: Especifica qué columnas incluir durante la importación.

Paso 3: Verificar los datos

En tu cliente MongoDB: `db.empleados.find()`

10: Usar un esquema de validación para la importación

Paso 1: Crear un archivo JSON con esquema válido e inválido

Crea un archivo llamado *vehiculos.json*:

```
[
  { "_id": 1, "marca": "Toyota", "modelo": "Corolla", "año": 2020 },
  { "_id": 2, "marca": "Honda", "modelo": "Civic", "año": "veinte veinte" }
]
```

Paso 2: Configurar validación a nivel de colección

Configura un validador en la colección vehículos:

```
db.createCollection("vehiculos", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["marca", "modelo", "año"],
      properties: {
        marca: { bsonType: "string" },
        modelo: { bsonType: "string" },
        año: { bsonType: "int", minimum: 1900, maximum: 2025 }
      }
    }
  }
})
```

Paso 3: Usar mongoimport y validar

Ejecuta el siguiente comando:

```
mongoimport --db taller_mongo --collection vehiculos --file vehiculos.json --
jsonArray
```

Paso 4: Verificar los datos

En tu cliente MongoDB:

```
db.vehiculos.find()
```