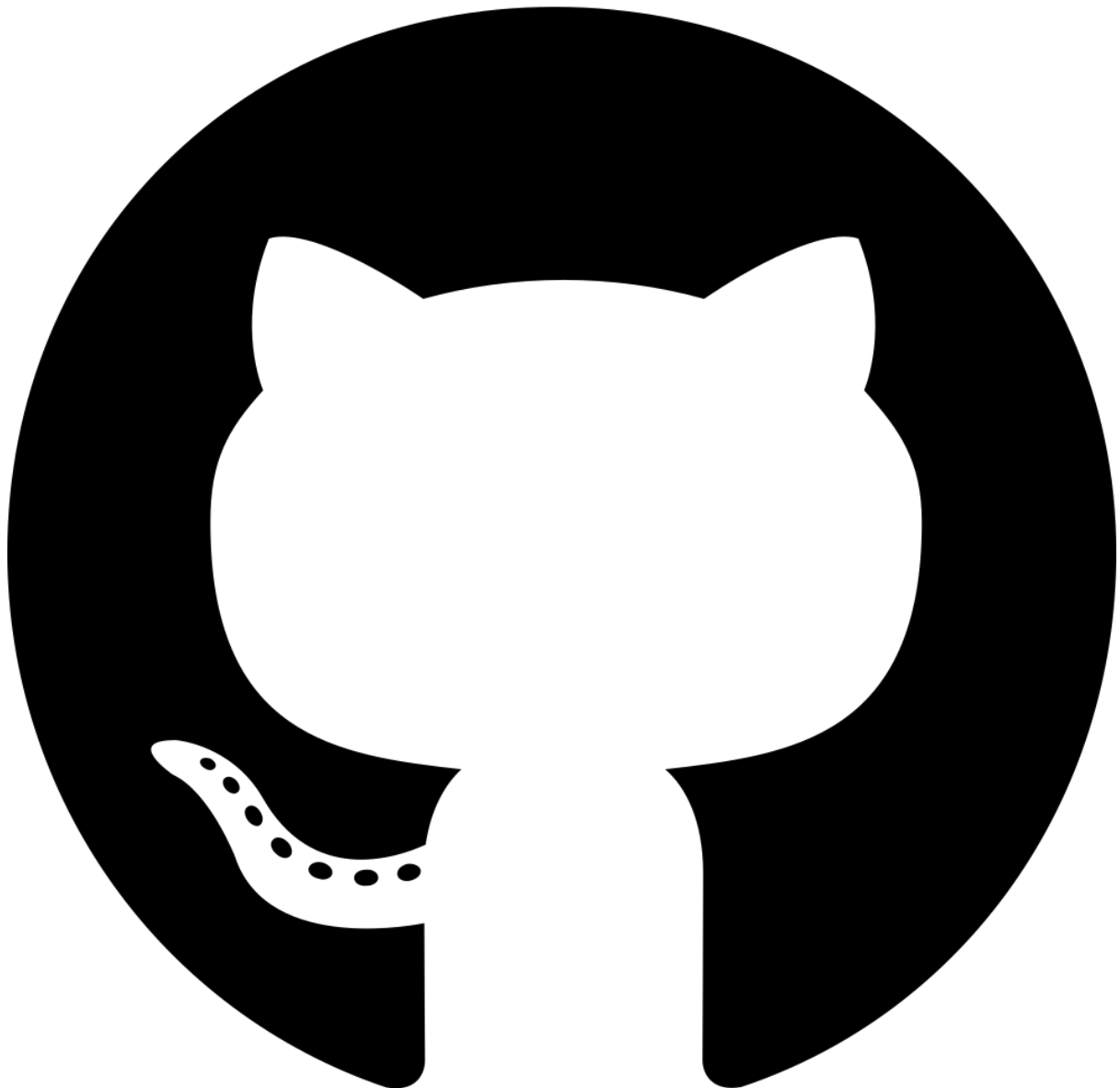


PROYECTO FINAL

2ºTRIMESTRE



-ÍNDICE

-Estructura de la base de datos

-Explicación de las consultas con los operadores vistos en clase.

-Explicación uso MongoAtlas

-Export e Import MongoAtlas y Local en Compass.

-Estructura de la base de datos.

```
db.ventas.insertMany([
  {
    id:1,
    articulo:[
      {modelo:{
        nombre:"Alpha 7 III",
        marca:"Sony",
        megapixeles:37.8,
        estabilizador:true
      }},
      objetivo:"Gran angular",
      conexion:{
        bluetooth:true,
        wifi:true
      },
      memoriascompatibles:["MS Duo","MS Pro-HG Duo","SD","SDHC","SDXC"],
      fpsgrabacion:[120,60,30],
      dimensiones:{
        altura:80.7,
        ancho:145.9,
        pesogr:885},
      precioventa: 1100,
      preciofabrica: 900,
      unidades: 18,
      garantia: 2
    },
    cliente: {
      nombre:"Juan Garcia",
      telefono:"630224575",
      direccion: {
        calle:"Guadalupe",
        codigopostal:"87827",
        numero: 9,
      }
    },
    vendedor: {
      nombre:"Paco Perez",
      dni:"30256946D",
      empresa:"Amazon",
      antiguedad: 10,
      salario: 2100.50
    },
    fecha: new Date("2022-02-19")
  },
])
```

En mi estructura podemos ver los diferentes campos que son los siguientes:

- id**: que es el identificador principal de la venta tipo numérico.
- artículo**: que es un tipo array de documentos con diferentes artículos.
- cliente**: que es un tipo documento que muestra información sobre los clientes.
- vendedor**: que es un tipo documento que muestra información sobre los vendedores
- fecha**: que es un tipo fecha que muestra la fecha de la venta.

Dentro de cada campo tenemos tipos string, numéricos, arrays, booleanos, documentos y arrays de tipo numérico y de tipo string.

Como podemos ver esta venta de ejemplo que he mostrado sólo tiene un artículo también encontraremos ventas con varios artículos.

-Explicación de las consultas con los operadores vistos en clase.

Una vez creado el Insert Principal con todos los datos he creado a partir de él diferentes colecciones usando la etapa \$project y \$out para que me mostrara los datos que les especificamos.

En esta primera colección creamos la colección vendedores mostrándonos solo los campos que especificamos con el operador \$project, y con el operador \$out lo lanzamos a la base de datos que queramos en este caso a la base "ProyectoAggregation" además de poner en el campo "coll" el nombre de la colección que queramos en este caso "vendedores" como hemos mencionado anteriormente.

```
db.ventas.aggregate([
  {
    $project: {
      _id: 0,
      id: 1,
      nombre: "$vendedor.nombre",
      dni: "$vendedor.dni",
      empresa: "$vendedor.empresa",
      antigüedad: "$vendedor.antigüedad",
      salario: "$vendedor.salario"
    }
  },
  {
    $out: {
      db: "ProyectoAggregation",
      coll: "vendedores"
    }
  }
])
```

En esta segunda colección repetimos el mismo procedimiento anterior pero ahora solo queremos los campos del campo “cliente” y le llamamos “clientes”.

```
db.ventas.aggregate([
  {
    $project: {
      _id: 0,
      id: 1,
      nombre: "$cliente.nombre",
      telefono: "$cliente.telefono",
      direccion: "$cliente.direccion"
    }
  },
  {
    $out: {
      db: "ProyectoAggregation",
      coll: "clientes"
    }
  }
])
```

En la tercera colección que creamos además del **\$project** con los campos especificados hacemos un **\$unwind** ya que el campo artículo es un array y queremos todos los datos de ese array separado por documentos para crear esta colección.

```
db.ventas.aggregate([
  {
    $unwind: {
      path: "$articulo"
    }
  },
  {
    $project: {
      _id: 0,
      id: 1,
      modelo: "$articulo.modelo",
      objetivo: "$articulo.objetivo",
      conexion: "$articulo.conexion",
      memoriascompatibles: "$articulo.memoriascompatibles",
      fpsgrabacion: "$articulo.fpsgrabacion",
      dimensiones: "$articulo.dimensiones",
      precioventa: "$articulo.precioventa",
      preciofabrica: "$articulo.preciofabrica",
      unidades: "$articulo.unidades",
      garantia: "$articulo.garantia"
    }
  },
  {
    $out: {
      db: "ProyectoAggregation",
      coll: "articulos"
    }
  }
])
```

Y por último creamos una colección con todos los datos de la venta excepto los artículos para ello volvemos a hacer un **\$project** especificando los campos que queremos y usamos el **\$out** como siempre para sacar la colección enviándola a la base de datos.

```
db.ventas.aggregate([
  {
    $project: {
      _id: 0,
      id: 1,
      cliente: 1,
      vendedor: 1,
      fecha: 1
    }
  },
  {
    $out: {
      db: "ProyectoAggregation",
      coll: "venta"
    }
  }
])
```

-Explicación de las consultas.

```
// Media de memorias aceptadas por cada cámara redondeada.
```

```
db.ventas.aggregate([
  {
    $unwind: {
      path: "$articulo"
    }
  },
  {
    $project: {
      _id: 0,
      numeromemorias: { $size: "$articulo.memoriascompatibles" }
    }
  },
  {
    $group: {
      _id: null,
      mediamemorias: { $avg: "$numeromemorias" }
    }
  },
  {
    $project: {
      _id: 1,
      mediamemorias: { $round: ["$mediamemorias", 0] }
    }
  }
])
```

En esta primera consulta con la colección completa de ventas usamos el operador **\$unwind** para sacar todos los artículos de el array, seguidamente usamos una primera etapa **\$project** para que no nos muestre el **_id** y usando el operador **\$size** que nos cuente el número de elementos que hay en el array “memoriascompatibles”, a continuación agrupamos los documentos con **\$group** y hacemos la media del número de “memoriascompatibles” usando el operador **\$avg** y terminamos haciendo otro **\$project** para mostrar solo el **_id** nulo con la media de las memorias redondeada con 0 decimales utilizando **\$round**.

```
// Dinero total generado por cada vendedor ordenado de mayor a menor.
db.articulos.aggregate([
  {
    $lookup: {
      from: "vendedores",
      localField: "id",
      foreignField: "id",
      as: "vendedores"
    }
  },
  {
    $project: {
      _id: 0,
      vendedores: { $arrayElemAt: [ "$vendedores", 0 ] },
      dinerototal: { $multiply: [ "$unidades", "$precioventa" ] },
    }
  },
  {
    $project: {
      dinerototal: 1,
      vendedor: "$vendedores.nombre"
    }
  },
  {
    $group: {
      _id: "$vendedor",
      dinerototal: { $sum: "$dinerototal" }
    }
  },
  {
    $sort: {
      dinerototal: -1
    }
  }
])
```

En la segunda consulta usamos la colección artículos y comenzamos haciendo uso del operador **\$lookup** para tener la información de otra colección en la que estamos trabajando y poder usar sus datos para nuestra consulta creando un campo nuevo tipo array con todos los campos de la colección, a continuación usamos una etapa **\$project** para mostrar solo

vendedores desglosando el array que creamos anteriormente con el **\$lookup** y además creamos un campo con el nombre “dinerototal” en el que usamos el operador **\$multiply** para multiplicar los valores de los campos “unidades” y “precioventa”, seguidamente con el **\$project** proyectamos la información que queremos y ahora podemos usar la información de la colección que añadimos con el lookup, luego agrupamos por vendedor con **\$group** y sumamos el dinerototal haciendo uso del operador **\$sum** y ordenamos en otra última etapa con **\$sort** en este caso de mayor a menor.

```
// Beneficios obtenidos por cada vendedor
```

```
db.articulos.aggregate([
  {
    $lookup: {
      from: "vendedores",
      localField: "id",
      foreignField: "id",
      as: "vendedores"
    }
  },
  {
    $project: {
      _id: 0,
      beneficios: {
        $subtract: [
          {
            $multiply: [
              "$unidades",
              "$precioventa"
            ]
          },
          {
            $multiply: [
              "$unidades",
              "$preciofabrica"
            ]
          }
        ]
      },
      vendedores: { $arrayElemAt: ["$vendedores", 0] },
    }
  },
  {
    $project: {
      beneficios: 1,
      vendedor: "$vendedores.nombre"
    }
  },
  {
    $group: {
      _id: "$vendedor",
      totalbeneficios: {
        $sum: "$beneficios"
      }
    }
  }
])
```


Esta tercera consulta con la colección artículos volvemos a usar información de otra colección, por tanto volvemos a usar **\$lookup** para poder usar la colección vendedores, seguimos con el operador **\$project** para mostrar los beneficios, haciendo una resta con **\$subtract** de la multiplicación con **\$multiply** que nos da el valor del dinero generado por precio de venta y fábrica, a continuación sacamos del array de vendedores la información que proyectaremos en la siguiente etapa con otro **\$project** y por último agrupamos con **\$group** por vendedores y sumamos los beneficios de cada documento en la agrupación con **\$sum**.

```
// Buscar cámaras que hayan sido vendidas en 2022 y mostrar su máxima capacidad de grabación
```

```
db.venta.aggregate([
  {
    $lookup: {
      from: "articulos",
      localField: "id",
      foreignField: "id",
      as: "articulos"
    }
  },
  {
    $match: {
      fecha: { $gt: new Date("2021-12-31") }
    }
  },
  {
    $unwind: {
      path: "$articulos"
    }
  },
  {
    $project: {
      _id: 0,
      nombre: "$articulos.modelo.nombre",
      maxfps: {
        $max: "$articulos.fpsgrabacion"
      },
      fecha: "$fecha"
    }
  }
])
```

En la cuarta consulta usamos la colección venta y usamos información de la colección artículos usando **\$lookup**, seguidamente usando el operador **\$match** filtramos el resultado por el campo fecha mayor a un valor con el operador **\$gt**, luego realizamos un **\$unwind** para sacar todos los elementos del array artículos y poder usarlos en el project ya que provienen de otra colección como hemos dicho anteriormente. Y para finalizar **\$project** proyectando los campos especificados, donde también en uno de ellos usamos el operador **\$max** para que nos muestre el máximo valor de un array.

```
// Número de unidades compradas por cada Cliente y crear un campo país con el valor España a todos los clientes.
```

```
db.clientes.aggregate([
  {
    $lookup: {
      from: "articulos",
      localField: "id",
      foreignField: "id",
      as: "articulos"
    }
  },
  {
    $unwind: {
      path: "$articulos"
    }
  },
  {
    $addFields: {
      "direccion.pais": "España"
    }
  },
  {
    $group: {
      _id: { nombre: "$nombre", pais: "$direccion.pais" },
      totalunidades: { $sum: "$articulos.unidades" },
    }
  },
  {
    $project: {
      _id: 0,
      nombre: "$_id.nombre",
      totalunidades: "$totalunidades",
      pais: "$_id.pais"
    }
  }
])
```

Quinta consulta donde usaremos la colección “clientes” y volvemos a usar información de otra colección por lo tanto operador **\$lookup**, en la siguiente etapa sacamos todos los elementos del array como en la anterior consulta con **\$unwind** y añadimos un nuevo campo a la colección con la que estamos trabajando con el operador **\$addFields** este nuevo campo se meterá a un campo tipo documento, por último agrupamos por lo especificado con **\$group** y creamos un nuevo campo, que nos suma los artículos ordenados, por el campo especificado con **\$sum**, y pasamos a finalizar con el operador **\$project** para mostrar los campos que necesitamos en esta consulta.

```
// Nombre de los vendedores,dni,salario (si tienen más de 5 años de
antigüedad bonus *1.5 donde x son
// los años de antigüedad)
```

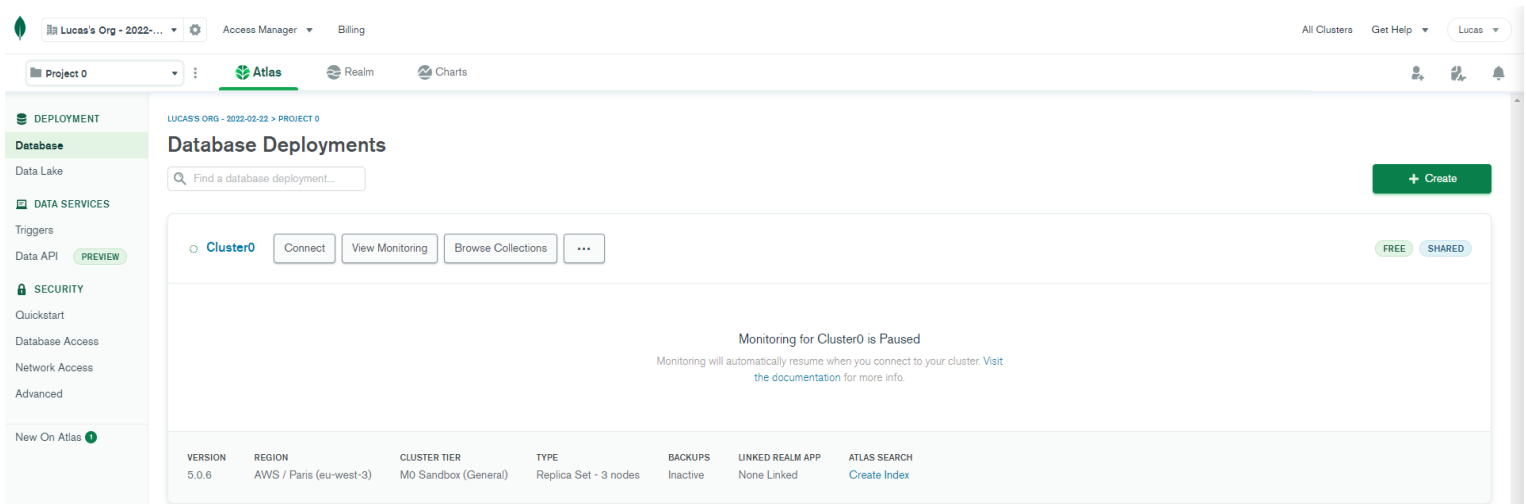
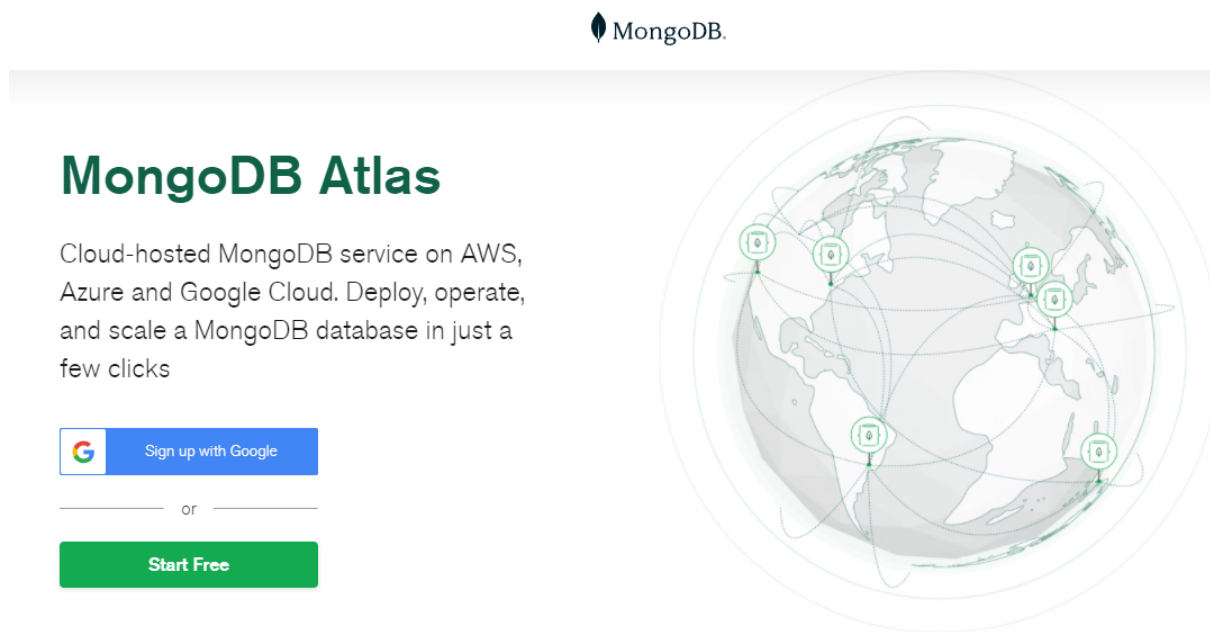
```
db.vendedores.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "id",
      foreignField: "id",
      as: "clientes"
    }
  },
  {
    $project: {
      nombre:"$nombre",
      dni:"$dni",
      salario:"$salario",
      bonus:{
        $cond:{
          if:{ $gt:['$antigüedad',5] },
          then: 1.5,
          else: 1
        }
      }
    }
  },
  {
    $project: {
      nombre:"$nombre",
      dni:"$dni",
      salario:{ $round:[ {$multiply:["$salario","$bonus"] },2] }
    }
  },
  {
    $group: {
      _id:{nombre:"$nombre",dni:"$dni",salario:"$salario"}
    }
  }
])
```

En esta última consulta usamos la colección vendedores y de nuevo usamos información de otra colección, repetimos una vez más el operador **\$lookup** usando la local y foreign field que tengan en común las colecciones y vamos a la siguiente etapa **\$project** para proyectar la información que queremos en este caso nombre,dni y salario de nuestra colección con la que trabajamos y además un campo bonus en el que metemos un nuevo operador **\$cond** con el que fijamos que si la antigüedad es mayor a 5 recibe un 1.5 si no se quedará igual, a continuación una etapa **\$project** para mostrar los campos especificados nombre dni y salario usando el operador **\$round** para redondear la multiplicación del bonus con **\$multiply**. Para finalizar la etapa \$group con la que agrupamos por los 3 campos anteriores para tener el nombre sueldo y dni sin repetir.

-Explicación uso MongoDB Atlas

https://www.mongodb.com/cloud/atlas/lp/try2?utm_source=google&utm_campaign=gs_eme_a_spain_search_core_brand_atlas_desktop&utm_term=mongoatlas&utm_medium=cpc_paid_search&utm_ad=e&utm_ad_campaign_id=12212624563&adgroup=115749706943&gclid=CjwKCAiA1JGRBhBSEiwAxXblwURVu3u3Wx0A-93uFthN7ilrZ2PcxSBI6NbAJlcmxz0Z2wuNYnVHRBoCgilQAvD_BwE

Primeramente accedemos a la web oficial de MongoDB Atlas y nos registramos.



En la siguiente interfaz creamos un nuevo proyecto arriba a la izquierda de la siguiente manera

Project 0

Search for a Project...

Project 0 (current)

View All Projects

+ New Project

Name Your Project

Project names have to be unique within the organization (and other restrictions).

ProjectoFinal2ºTrimestre

Cancel

Next

LUCAS'S ORG - 2022-02-22 > PROJECTS

Create a Project

✓ Name Your Project

Add Members

Add Members and Set Permissions

Lucas

Give your members access permissions below.

lhiljim1212@g.educaand.es
(you)

Project Owner

Cancel

← Go Back

Create Project



Create a database

Choose your cloud provider, region, and specs.

[Build a Database](#)

Once your database is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).

Seleccionamos la opción gratuita

FREE

 **Shared**

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

[Create](#)

Starting at

FREE

Cloud Provider & Region

AWS, Paris (eu-west-3) ▾



★ Recommended region ⓘ Paid tier region ⓘ

NORTH AMERICA	EUROPE	AUSTRALIA
🇺🇸 Oregon (us-west-2) ★	🇩🇪 Frankfurt (eu-central-1) ★	🇦🇺 Sydney (ap-southeast-2) ★
🇺🇸 N. Virginia (us-east-1) ★	🇸🇪 Stockholm (eu-north-1) ★	
🇺🇸 Ohio (us-east-2) ★ ⓘ	🇮🇪 Ireland (eu-west-1) ★	ASIA
🇺🇸 N. California (us-west-1) ⓘ	🇫🇷 Paris (eu-west-3) ★	🇯🇵 Tokyo (ap-northeast-1) ★
🇨🇦 Montreal (ca-central-1) ⓘ	🇬🇧 London (eu-west-2) ★ ⓘ	🇮🇳 Mumbai (ap-south-1)
		🇰🇷 Seoul (ap-northeast-2)
SOUTH AMERICA	🇮🇹 Milan (eu-south-1) ★ ⓘ	🇸🇬 Singapore (ap-southeast-1) ★
🇧🇷 Sao Paulo (sa-east-1)		🇯🇵 Osaka (ap-northeast-3) ★ ⓘ
	MIDDLE EAST	
	🇧🇭 Bahrain (me-south-1) ★	
	AFRICA	
	🇿🇦 Cape Town (af-south-1) ★	

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) ^{Encrypted} ^

Additional Settings

MongoDB 5.0, No Backup ^

Cluster Name

Cluster0 ^

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

[Back](#)

Create Cluster

Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username

usuario1

Password 

.....

 Autogenerate Secure Password

 Copy

Create User

2 Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

My Local Environment

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

ADVANCED

Cloud Environment

Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters.

IP Address

Description

Enter IP Address

Enter description

Add Entry

Add My Current IP Address

Finish and Close

Congratulations on setting up access rules!

You will now be able to connect to your deployments. You can continue to add and update access rules in [Database Access](#) and [Network Access](#).

☒ Hide Quickstart guide in the navigation. You can visit [Project Settings](#) to access it in the future.

[Go to Databases](#)

LUCASS ORG - 2022-02-22 > PROYECTO FINAL 2° TRIMESTRE

Database Deployments

Find a database deployment...

Cluster0

Connect

View Monitoring

Browse Collections

...

R

W

Connections 0

Last 44 seconds

100.0%

In

Out

Data Size 0.0 B

Last 44 seconds

100.0 B/s

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED REALM APP	ATLAS SEARCH
5.0.6	AWS / Paris (eu-west-3)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

Use this connection string in your application:

```
mongosh "mongodb+srv://cluster0.y4io3.mongodb.net/myFirstDatabase" --apiVersion 1
--username usuario1
```

Replace **myFirstDatabase** with the name of the database that connections will use by default. You will be prompted for the password for the Database User, **usuario1**. When entering your password, make sure all special characters are [URL encoded](#).

Nos conectamos a MongoAtlas desde la Powershell copiando el enlace

```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6
PS C:\Users\Lucas> mongosh "mongodb+srv://cluster0.y4io3.mongodb.net/myFirstDatabase" --apiVersion 1 --username usuario1
Enter password: *****
Current Mongosh Log ID: 62256a4f979123a880a300af
Connecting to: mongodb+srv://cluster0.y4io3.mongodb.net/myFirstDatabase?appName=mongosh+1.2.2
Using MongoDB: 5.0.6 (API Version 1)
Using Mongosh: 1.2.2
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
Warning: Found ~/.mongorc.js, but not ~/.mongoshrc.js. ~/.mongorc.js will not be loaded.
You may want to copy or rename ~/.mongorc.js to ~/.mongoshrc.js.
Atlas atlas-zuxu0w-shard-0 [primary] myFirstDatabase>
```

```
Atlas atlas-zuxu0w-shard-0 [primary] myFirstDatabase> show collections
articulos
clientes
vendedores
venta
ventas
```

Metemos las colecciones como hicimos de manera local.

-Export e Import MongoAtlas y Local en Compass.

Available Downloads

Version


100.5.2

Platform

Windows x86_64

Package

msi

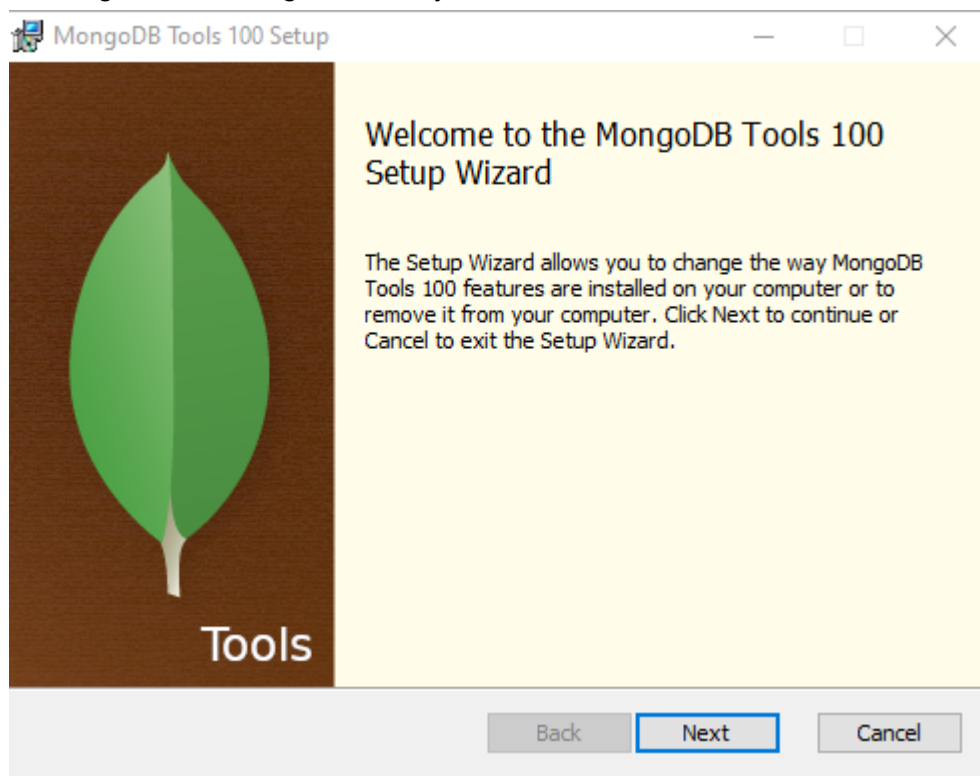
 **Download**

Copy Link

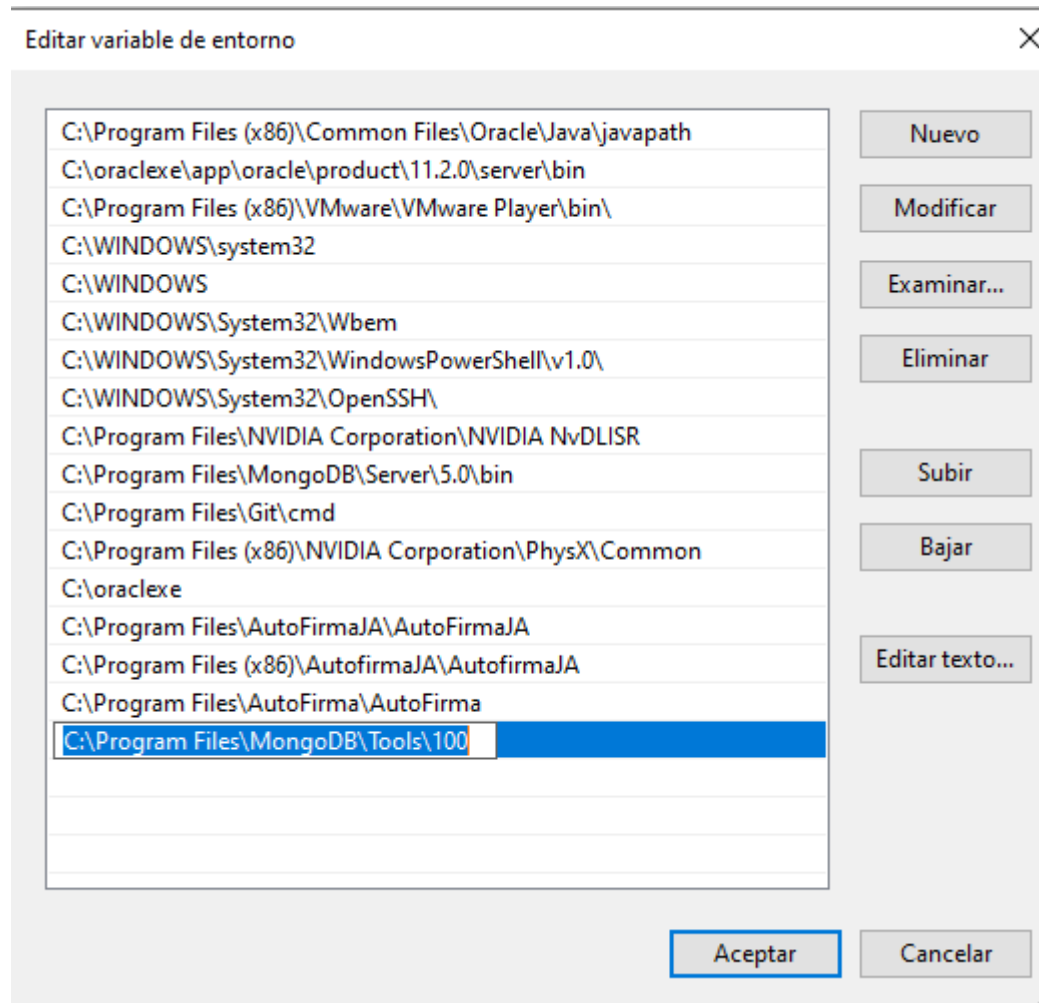
Documentation

Archived releases

Descargamos el MongoDB tools y lo instalamos



Lo metemos en las variables de entorno editando el Path



-Exportar e Importar Local

Export

```
mongoexport -d ProyectoAggregation -c ventas --out  
C:\Users\Lucas\Documents\ventas.json
```

```
PS C:\Users\Lucas> mongoexport -d ProyectoAggregation -c ventas --out C:\Users\Lucas\Documents\ventas.json  
2022-03-07T04:17:56.891+0100 connected to: mongod://localhost/  
2022-03-07T04:17:56.983+0100 exported 10 records
```

```
mongoexport -d ProyectoAggregation -c ventas --type csv --fields  
_id,articulo,cliente,vendedor,fecha --out  
C:\Users\Lucas\Documents\ventas.csv
```

```
PS C:\Users\Lucas> mongoexport -d ProyectoAggregation -c ventas --type csv --fields _id,articulo,cliente,vendedor,fecha  
--out C:\Users\Lucas\Documents\ventas.csv  
2022-03-07T04:21:35.535+0100 connected to: mongod://localhost/  
2022-03-07T04:21:35.588+0100 exported 10 records
```

ventas.json	07/03/2022 4:17	Archivo de origen ...	12 KB
ventas.csv	07/03/2022 4:21	Hoja de cálculo d...	13 KB

Import

```
mongoimport -d ProyectoAggregation -c ventas --file  
C:\Users\Lucas\Documents\ventas.json
```

```
PS C:\Users\Lucas> mongoimport -d ProyectoAggregation -c ventas --file C:\Users\Lucas\Documents\ventas.json  
2022-03-07T04:28:08.416+0100 connected to: mongodb://localhost/  
2022-03-07T04:28:08.517+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f532') }  
2022-03-07T04:28:08.517+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f531') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f533') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f530') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f536') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f534') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f535') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f537') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f52f') }  
2022-03-07T04:28:08.518+0100 continuing through error: E11000 duplicate key error collection: ProyectoAggregation.ven  
tas index: _id_dup key: { _id: ObjectId('62250d3b5349dd1eaa34f538') }  
2022-03-07T04:28:08.518+0100 0 document(s) imported successfully. 10 document(s) failed to import.
```

Me sale error porque ya la he metido anteriormente para realizar mis consultas.

```
mongoimport -d ProyectoAggregation -c ventas --type csv --headerline  
--file C:\Users\Lucas\Documents\ventas.csv
```

```
PS C:\Users\Lucas> mongoimport -d ProyectoAggregation -c ventas --type csv --headerline --file C:\Users\Lucas\Documents\  
ventas.csv  
2022-03-07T04:30:15.556+0100 connected to: mongodb://localhost/  
2022-03-07T04:30:15.632+0100 10 document(s) imported successfully. 0 document(s) failed to import.
```

-Exportar e Importar MongoAtlas

Export

```
mongoexport --uri  
mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggr  
egation --collection ventas --type json --out  
C:\Users\Lucas\Documents\ventas.json
```

```
PS C:\Users\Lucas> mongoexport --uri mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggregation --co  
llection ventas --type json --out C:\Users\Lucas\Documents\ventas.json  
2022-03-07T04:43:01.581+0100 connected to: mongodb+srv://[**REDACTED**]@cluster0.y4io3.mongodb.net/ProyectoAggregatio  
n  
2022-03-07T04:43:01.688+0100 exported 0 records
```

```
mongoexport --uri  
mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggr  
egation --collection ventas --type csv --fields  
_id,articulo,cliente,vendedor,fecha --out  
C:\Users\Lucas\Documents\ventas.csv
```

```
PS C:\Users\Lucas> mongoexport --uri mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggregation --co  
llection ventas --type csv --fields _id,articulo,cliente,vendedor,fecha --out C:\Users\Lucas\Documents\ventas.csv  
2022-03-07T04:46:16.936+0100 connected to: mongodb+srv://[**REDACTED**]@cluster0.y4io3.mongodb.net/ProyectoAggregatio  
n  
2022-03-07T04:46:17.055+0100 exported 0 records
```

Me está dando algún tipo de error que no consigo averiguar pero entiendo el funcionamiento del comando.

Import

```
mongoimport --uri  
mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggr  
egation --collection ventas --type json --file  
C:\Users\Lucas\Documents\ventas.json
```

```
PS C:\Users\Lucas> mongoimport --uri mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggregation --co  
llection ventas --type json --file C:\Users\Lucas\Documents\ventas.json  
2022-03-07T04:47:50.502+0100    connected to: mongodb+srv://[**REDACTED**]@cluster0.y4io3.mongodb.net/ProyectoAggregation  
2022-03-07T04:47:50.582+0100    0 document(s) imported successfully. 0 document(s) failed to import.
```

```
mongoimport --uri  
mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggr  
egation --collection ventas --type csv --headerline --file  
C:\Users\Lucas\Documents\ventas.csv
```

```
2022-03-07T04:47:50.582+0100    0 document(s) imported successfully. 0 document(s) failed to import.  
PS C:\Users\Lucas> mongoimport --uri mongodb+srv://usuario1:usuario1@cluster0.y4io3.mongodb.net/ProyectoAggregation --co  
llection ventas --type csv --headerline --file C:\Users\Lucas\Documents\ventas.csv  
2022-03-07T04:49:46.926+0100    Failed: EOF  
2022-03-07T04:49:46.973+0100    0 document(s) imported successfully. 0 document(s) failed to import.
```