



# Curso de **Mongo DB**

---

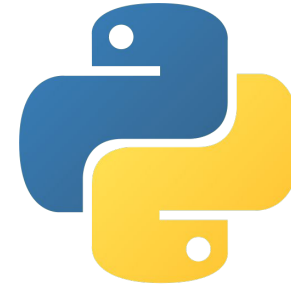
Albert Ramírez

---

# Proyecto



API REST



MongoDB

---

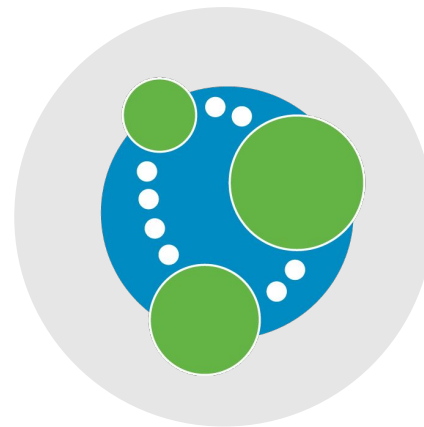
# Bases de datos NoSQL

---

# Según el modelo de datos



Key-value stores



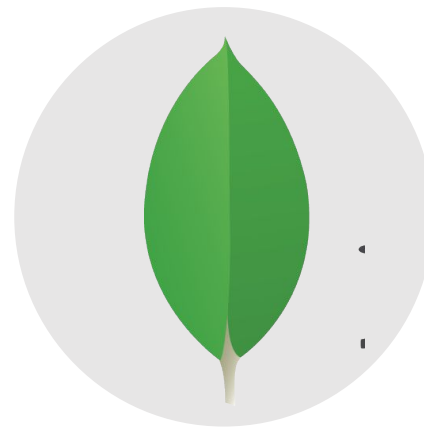
Graph  
databases

---

# Según el modelo de datos



Wide-column  
stores



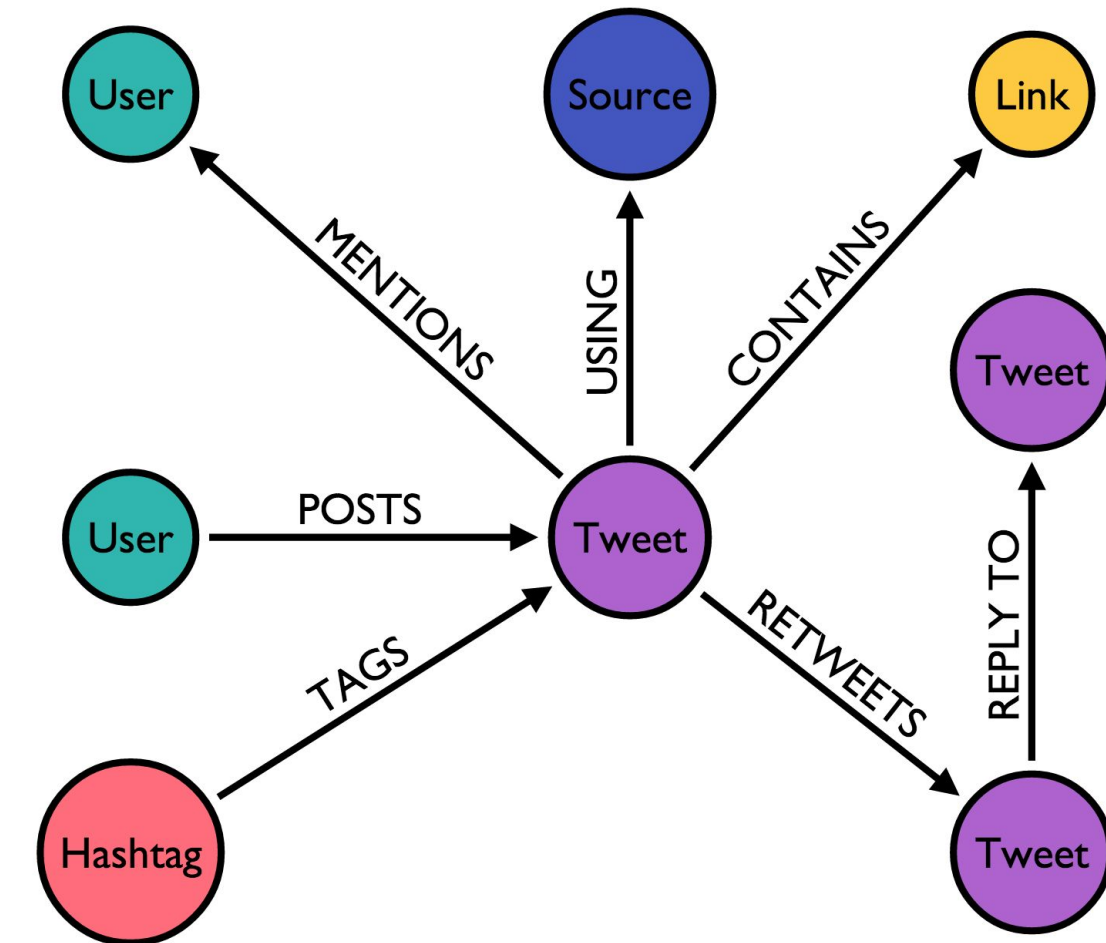
Document  
databases

---

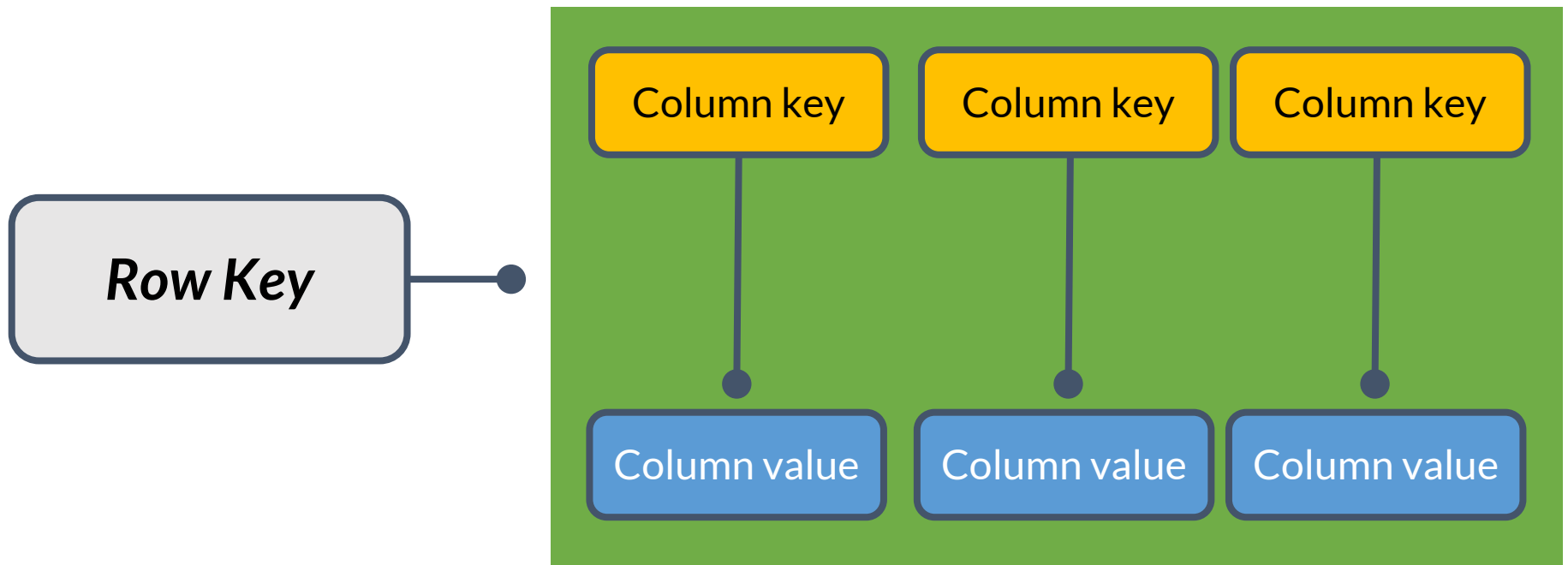
# Key-Value stores



# Graph databases



# Wide-column stores





# Document databases

```
{
  "name": "Albert Ramirez",
  "age" : 25,
  "city": "Bogotá",
  "cars": [
    {
      "model" : "Renault",
      "year"  : 2011,
      "value" : 20000
    },
    {
      "model" : "Chevrolet",
      "year"  : 2009,
      "value" : 10000
    }
  ]
}
```

---

# En resumen

	<i>Caso de uso</i>	<i>Ejemplos</i>
<i>Key-value store</i>	Ideal para almacenar información de sesión, recomendaciones.	Redis, Memcached
<i>Graph database</i>	Es buena para establecer relaciones, mejor rendimiento que bases de datos relacionales.	Neo4j, JanusGraph
<i>Wide-column store</i>	Alto rendimiento y arquitectura escalable.	Cassandra, HBase
<i>Document database</i>	Propósito general.	MongoDB, Couchbase

MongoDB

---

# ¿Qué es MongoDB?

---

# Características de MongoDB

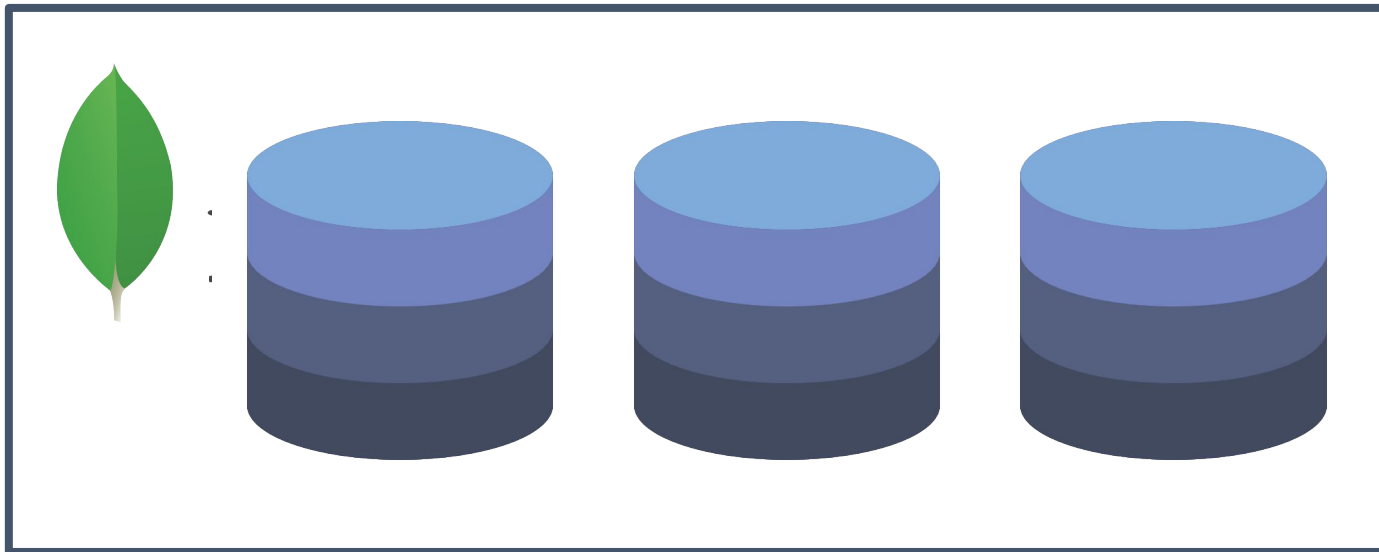
---

Los documentos son almacenados en una estructura parecida a un JSON (BSON)

```
{
  "name": "Albert Ramirez",
  "age" : 25,
  "city": "Bogotá",
  "cars": [
    {
      "model" : "Renault",
      "year"  : 2011,
      "value" : 20000
    },
    {
      "model" : "Chevrolet",
      "year"  : 2009,
      "value" : 10000
    }
  ]
}
```

---

# Es una base de datos distribuida



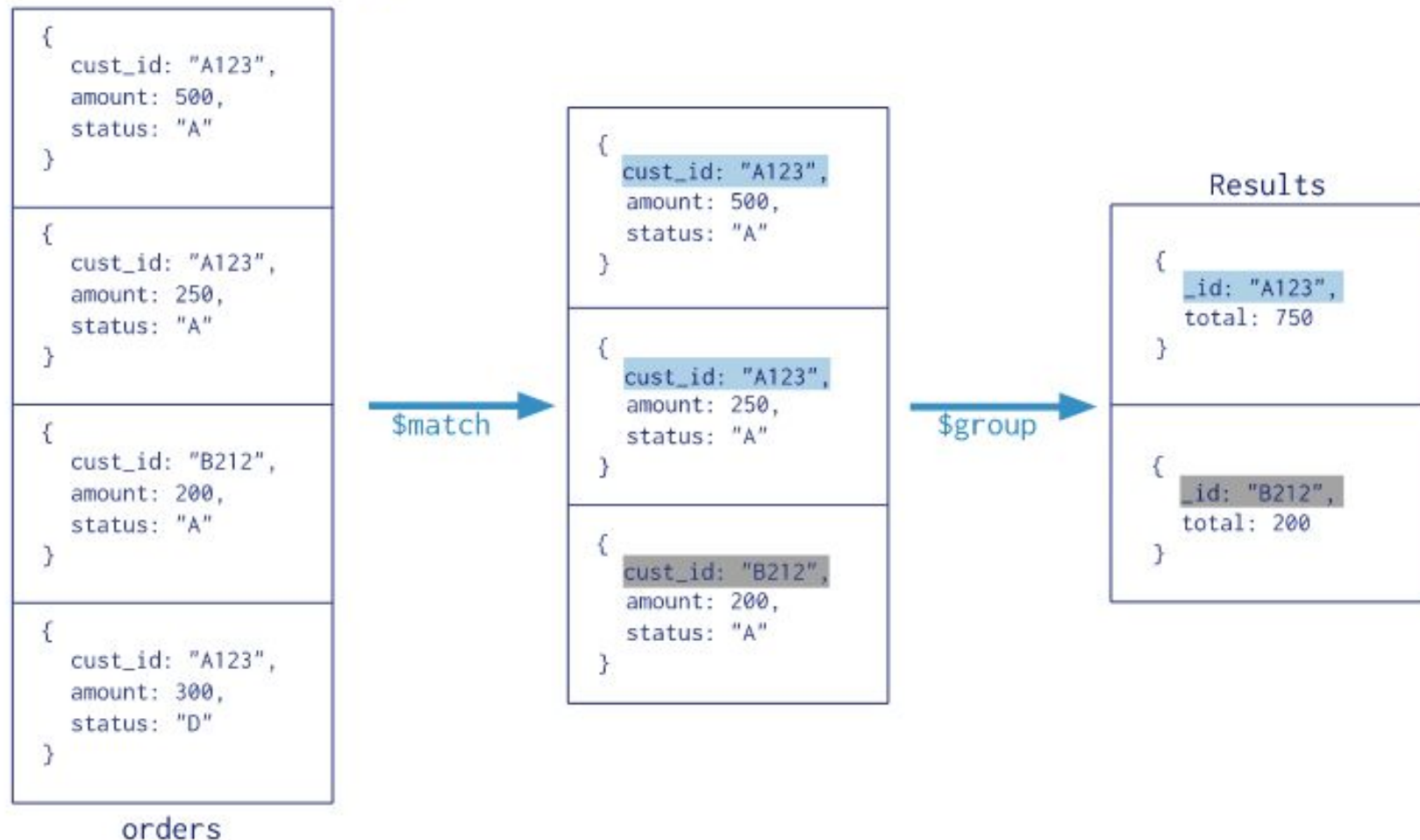
## Es schema less

```
{  
  "name": "Albert Ramirez",  
  "age"  : 25,  
  "city": "Bogotá",  
  "cars": [  
    {  
      "model" : "Renault",  
      "year"  : 2011,  
      "value" : 20000  
    },  
    {  
      "model" : "Chevrolet",  
      "year"  : 2009,  
      "value" : 10000  
    }  
  ]  
}
```

```
{  
  "name": "Pedro Gómez",  
  "city": "Bogotá",  
  "nickname": "pgomez"  
}
```

# Queries, índices y agregaciones

Collection  
↓  
db.orders.aggregate( [  
 \$match stage → { \$match: { status: "A" } },  
 \$group stage → { \$group: { \_id: "\$cust\_id", total: { \$sum: "\$amount" } } }  
] )





---

# Es gratis y de código abierto



MongoDB

---

# Ecosistema de MongoDB

---

*MongoDB*

MongoDB Server

MongoDB Mobile

Stitch

Community

Enterprise

Atlas (Cloud)

MongoDB Shell

MongoDB Compass

Conectores

MongoDB Charts

MongoDB

---

# MongoDB Atlas

---

# MongoDB como servicio



---

# MongoDB Atlas

- Aprovisionamiento automático de clusters con MongoDB
- Alta disponibilidad
- Altamente escalable
- Seguro
- Disponible en AWS, GCP y Azure
- Fácil monitoreo y optimización

---

# Creación de cuenta en MongoDB Atlas

---

MongoDB

---

# MongoDB + Drivers



---

**¿Qué son los  
drivers?**

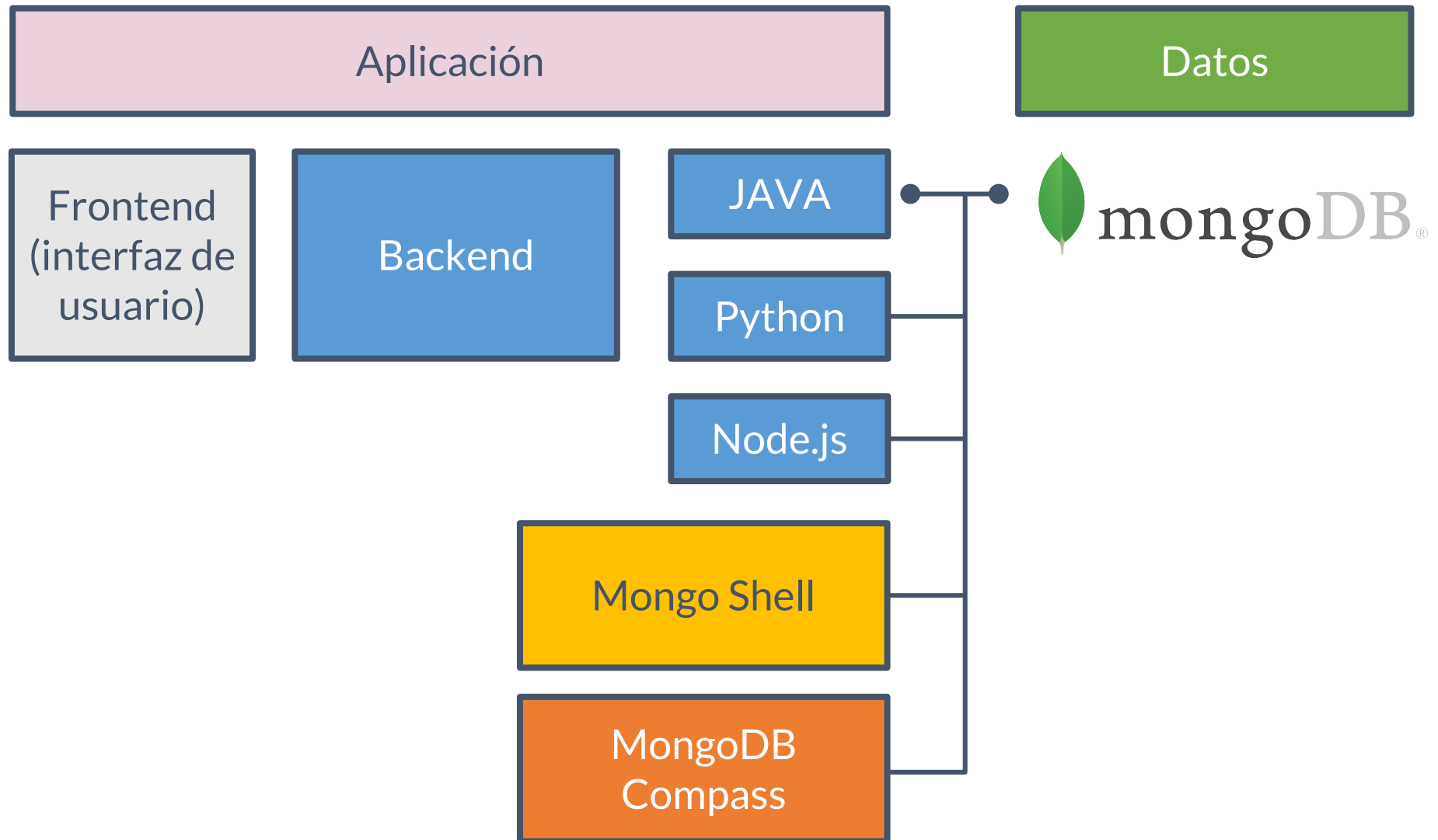
---

---

# Drivers oficiales

C	PHP
C++	Python
C#	Motor (Python Async)
Go (beta)	Ruby
Java	Mongoid(Ruby ODM)
Node.js	Scala
Perl	

# Arquitectura



---

# Agregar el driver a los proyectos


**DOWNLOAD**

mongodb-driver-sync ▼

3.10.1 ▼

Gradle

```
dependencies {  
  compile 'org.mongodb:mongodb-driver-sync:3.10.1'  
}
```

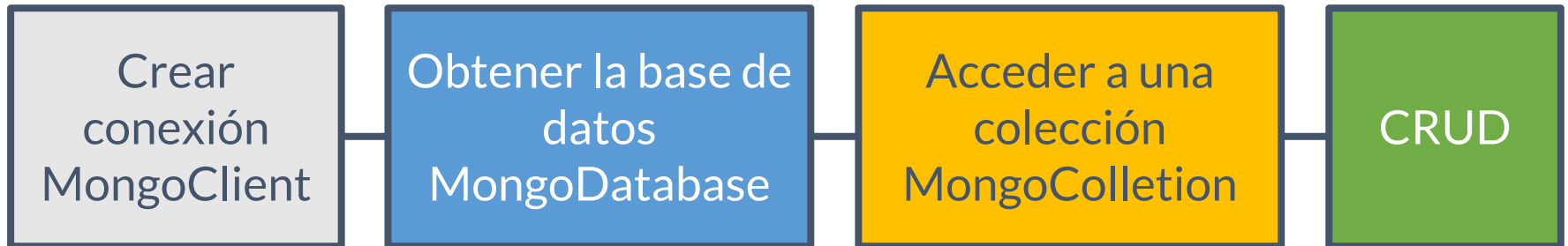


`python -m pip install pymongo`

`npm install mongodb --save`

---

# Inicio rápido transversal a la mayoría de lenguajes



MongoDB

---

# Bases de datos, colecciones y documentos

# Base de datos

- Contenedor físico de colecciones
- Cada base de datos tiene su archivo propio en el sistema de archivos
- Un cluster puede tener múltiples bases de datos

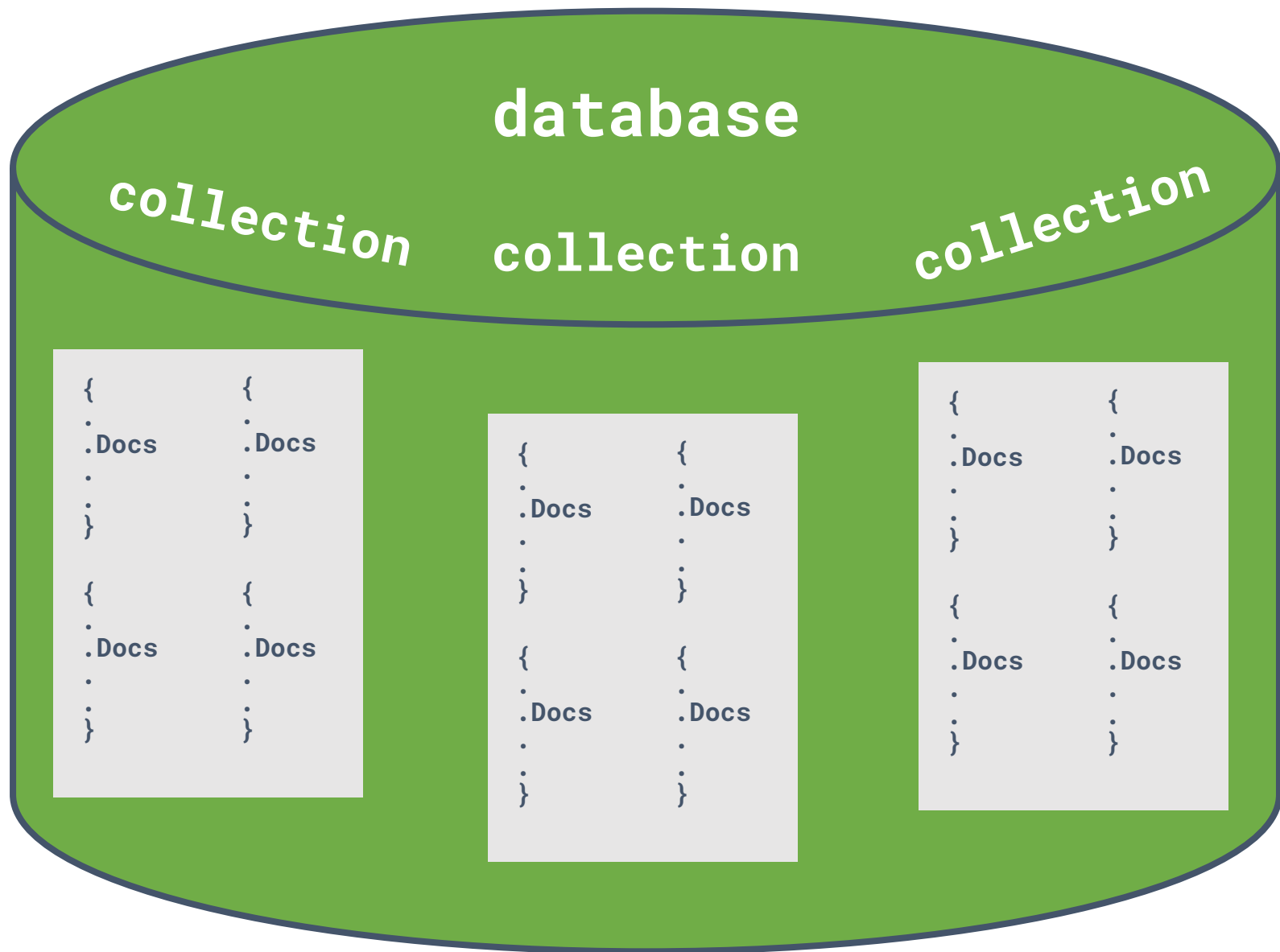
# Colecciones

- Agrupación de documentos
- Equivalente a una tabla en las bases de datos relacionales
- No impone un esquema



# Documentos

- Un registro dentro de una colección
- Es análogo a un objeto JSON (BSON)
- La unidad básica dentro de MongoDB



# MongoDB Drivers

## JSON

```
{  
  "name": "Some name",  
  "age": 30,  
  "nickName": "nickname"  
}
```

## BSON

Codificación  
binaria de  
documentos JSON

MongoDB

---

# Tipos de datos en MongoDB

---

Strings

“Algún texto”

Boolean

True/False

ObjectId

ObjectId("5c68...")

Date

ISODate("2019-02-18T...")

---

*Number*

```
graph TD; Number[Number] --- Double[Double]; Number --- Integer32[Integer 32 bits]; Number --- Integer64[Integer 64 bits]; Number --- Decimal[Decimal];
```

Double

Integer 32 bits

Integer 64 bits

Decimal

---

# Documentos embebidos

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
    phone: "123-456-7890",
    email: "xyz@example.com"
  },
  access: {
    level: 5,
    group: "dev"
  }
}
```

Embedded sub-document

Embedded sub-document

# Arreglos con documentos embebidos

```
{
  "_id": ObjectId(<ObjectId1>).
  "userName": "123user",
  "age": 35,
  "devices": [
    {
      "type": "computer",
      "value": 1200,
      "model": 2015
    },
    {
      "type": "phone",
      "value": 600,
      "model": 2017
    }
  ]
}
```



MongoDB

---

**¿Qué son los  
esquemas y las  
relaciones?**

MongoDB no impone un esquema dentro de las colecciones

```
{
  "_id":      ObjectId(<ObjectId1>).
  "userName": "123user",
  "age": 35,
},
{
  "_id":      ObjectId(<ObjectId2>).
  "name": "Name",
  "lastName": "Last Name",
  "city": "Bogota"
}
```

# MongoDB



# Mundo SQL

```
{
  "_id": ObjectId(<ObjectId1>),
  "userName": "123username",
  "age": 30,
},
{
  "_id": ObjectId(<ObjectId2>).
  "name": "User Name",
  "lastName": "Last Name",
  "city": "Bogota"
}
```

# MongoDB



# Mundo SQL

```
{
  "_id":      ObjectId(<ObjectId1>).
  "userName": "123username",
  "age":      30
},
{
  "_id":      ObjectId(<ObjectId2>).
  "userName": "456username",
  "age":      36
  "city":     "Bogota"
}
```

# MongoDB



# Mundo SQL

```
{
  "_id":      ObjectId(<ObjectId1>
  "userName": "123username",
  "age": 30,
  "city": null
},
{
  "_id":      ObjectId(<ObjectId2>
  "userName": "456username",
  "age": 36,
  "city": "Bogota"
}
```

---

# ¿Qué son las relaciones?

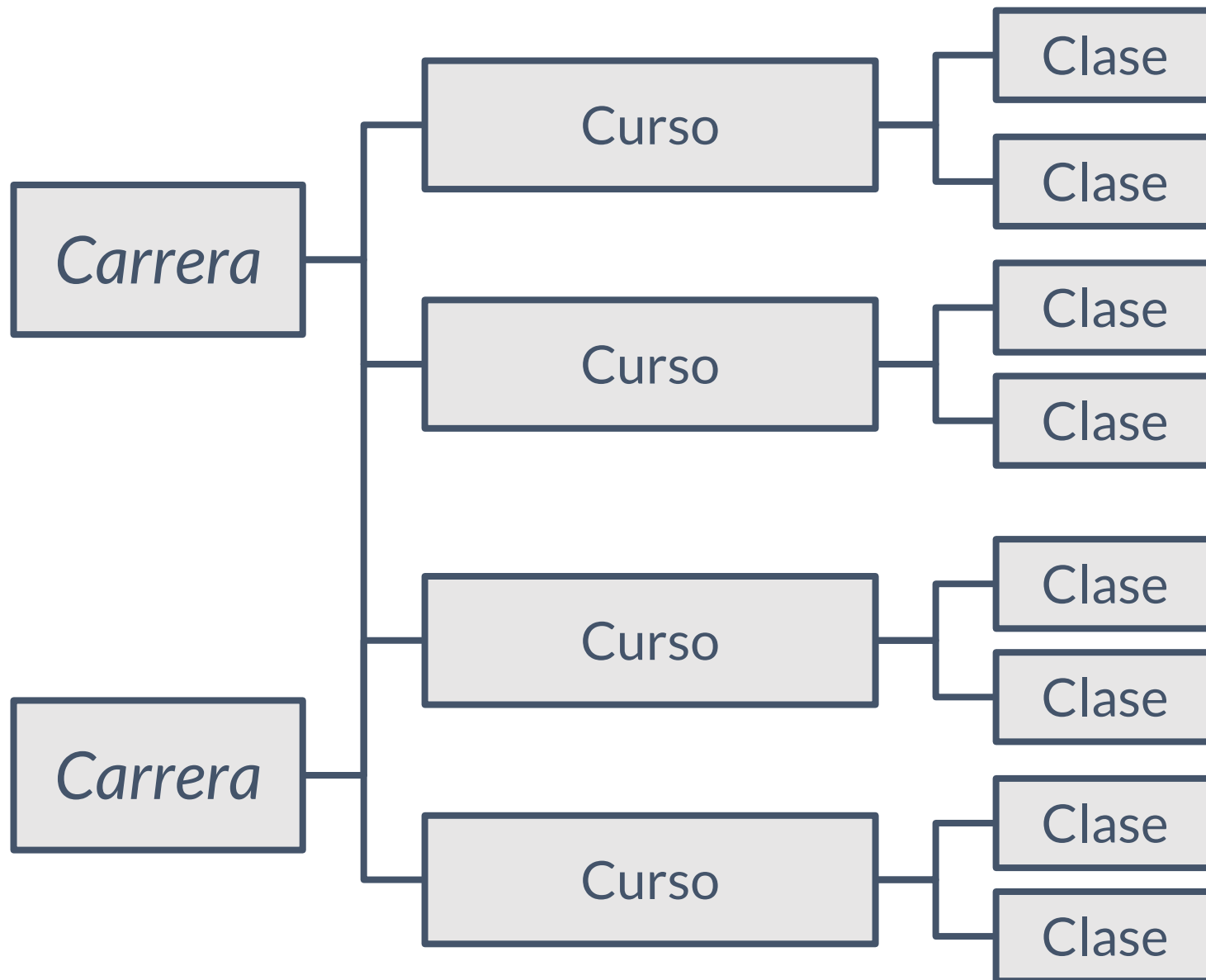
---

---

¿Por qué hablamos de  
relaciones dentro de  
MongoDB?

---

## Relaciones dentro del proyecto Platzi Mongo





MongoDB

---

# Relaciones entre documentos

## Relaciones uno a uno usando referencias

```
{  
  _id: "joe",  
  name: "Joe Bookreader"  
}  
  
{  
  patron_id: "joe",  
  street: "123 Fake Street",  
  city: "Faketon",  
  state: "MA",  
  zip: "12345"  
}
```

## Relaciones uno a uno usando documentos embebidos

```
{  
  _id: "joe",  
  name: "Joe Bookreader",  
  address: {  
    street: "123 Fake Street",  
    city: "Faketon",  
    state: "MA",  
    zip: "12345"  
  }  
}
```

---

# Relaciones uno a muchos

---

# Relaciones uno a muchos usando documentos embebidos

```
{
  _id: "joe",
  name: "Joe Bookreader"
}

{
  patron_id: "joe",
  street: "Street 1",
  city: "Faketon",
  state: "MA",
  zip: "12345"
}

{
  patron_id: "joe",
  street: "Street 2",
  city: "Boston",
  state: "MA",
  zip: "12345"
}
```

```
{
  _id: "joe",
  name: "Joe Bookreader",
  addresses: [
    {
      street: "Street 1",
      city: "Faketon",
      state: "MA",
      zip: "12345"
    },
    {
      street: "Street 2",
      city: "Boston",
      state: "MA",
      zip: "12345"
    }
  ]
}
```

```
{  
  title: "MongoDB: The Definitive Guide",  
  author: [ "Kristina Chodorow", "Mike Dirolf" ],  
  published_date: ISODate("2010-09-24"),  
  pages: 216,  
  language: "English",  
  publisher: {  
    name: "O'Reilly Media",  
    founded: 1980,  
    location: "CA"  
  }  
}
```

```
{  
  title: "50 Tips and Tricks for MongoDB Developer",  
  author: "Kristina Chodorow",  
  published_date: ISODate("2011-05-06"),  
  pages: 68,  
  language: "English",  
  publisher: {  
    name: "O'Reilly Media",  
    founded: 1980,  
    location: "CA"  
  }  
}
```

---

# Relaciones uno a muchos usando referencias

---



```
{  
  _id: 234567890,  
  title: "50 Tips and Tricks for MongoDB Developer",  
  author: "Kristina Chodorow",  
  published_date: ISODate("2011-05-06"),  
  pages: 68,  
  language: "English"  
}
```

```
{  
  _id: 123456789,  
  title: "MongoDB: The Definitive Guide",  
  author: [ "Kristina Chodorow", "Mike Dirolf" ],  
  published_date: ISODate("2010-09-24"),  
  pages: 216,  
  language: "English"  
}
```

```
{  
  name: "O'Reilly Media",  
  founded: 1980,  
  location: "CA",  
  books: [123456789, 234567890, ...]  
}
```

```
{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English",
  publisher_id: "oreilly"
}
{
  _id: 123456789,
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English",
  publisher_id: "oreilly"
}
```

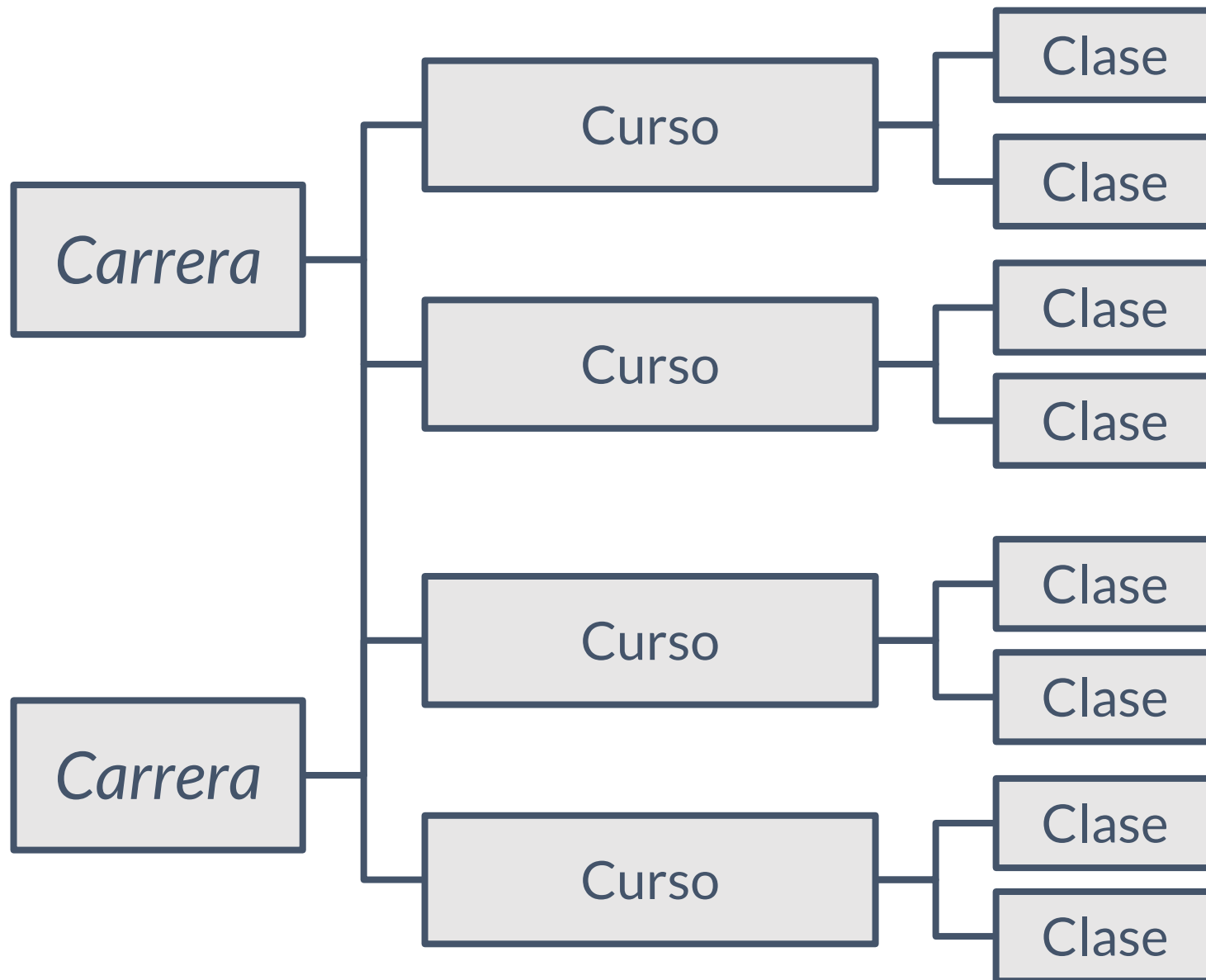
```
{  
  _id: "oreilly",  
  name: "O'Reilly Media",  
  founded: 1980,  
  location: "CA"  
}
```

MongoDB

---

# Esquema de platzi-mongo

## Relaciones dentro del proyecto Platzi Mongo



# Carreras

```
{
  "_id" : ObjectId("5c76..."),
  "nombre" : "Especialidad AWS",
  "descripcion" : "Desarrollar .....",
  "cursos" : [
    {
      "_id" : ObjectId("35f..."),
      "nombre" : "Fundamentos de AWS 2019"
    }
  ]
}
```

# Cursos y clases

```
{
  "_id" : ObjectId("052..."),
  "nombre" : "Fundamentos de AWS 2019",
  "descripcion" : "Aprenderás a desplegar ...",
  "clases" : [
    {
      "orden" : 1,
      "nombre" : "Clase 1",
      "descripcion" : "Descripción de la ...",
      "video" : "https://video.video",
      "archivos" : [
        "https://url1.com",
        "https://url2.com"
      ]
    }
  ]
}
```



MongoDB

---

# Operadores para realizar queries y proyecciones

# Filtros

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

```
{ <field1>: { <operator1>: <value1> }, ... }
```

# Proyecciones

```
db.inventory.findOne({status: "A"})

{
  "_id" : ObjectId("5c6c603aeafeac611c2d37a96"),
  "item" : "journal",
  "status" : "A",
  "size" : {
    "h" : 14,
    "w" : 21,
    "uom" : "cm"
  },
  "instock" : [
    {
      "warehouse" : "A",
      "qty" : 5
    }
  ]
}
```

# Proyecciones

```
db.inventory.findOne({status: "A"}, {item: 1, status: 1})

{
  "_id" : ObjectId("5c6c603aefead611c2d37a96"),
  "item" : "journal",
  "status" : "A"
}
```

# Operadores de comparación

<i>Nombre</i>	<i>Descripción</i>
\$eq	=
\$gt	>
\$gte	>=
\$lt	<
\$lte	<=
\$ne	!=
\$in	Valores dentro de un arreglo
\$nin	Valores que no están dentro de un arreglo

---

# Operadores lógicos

<i>Nombre</i>	<i>Descripción</i>
\$and	Une queries con un and lógico
\$not	Invierte el efecto de una query
\$nor	Une queries con un nor lógico
\$or	Une queries con un or lógico

---

# Operadores por elemento

<i>Nombre</i>	<i>Descripción</i>
\$exist	Documentos que cuentan con un campo específico
\$type	Documentos que cuentan con un campo de un tipo específico

---

# Operadores para arreglos

<i>Nombre</i>	<i>Descripción</i>
\$all	Arreglos que contengan todos los elementos de la query
\$elemMatch	Documentos que cumplen la condición del \$elemMatch en uno de sus elementos
\$size	Documentos que contienen un campo tipo arreglo de un tamaño específico



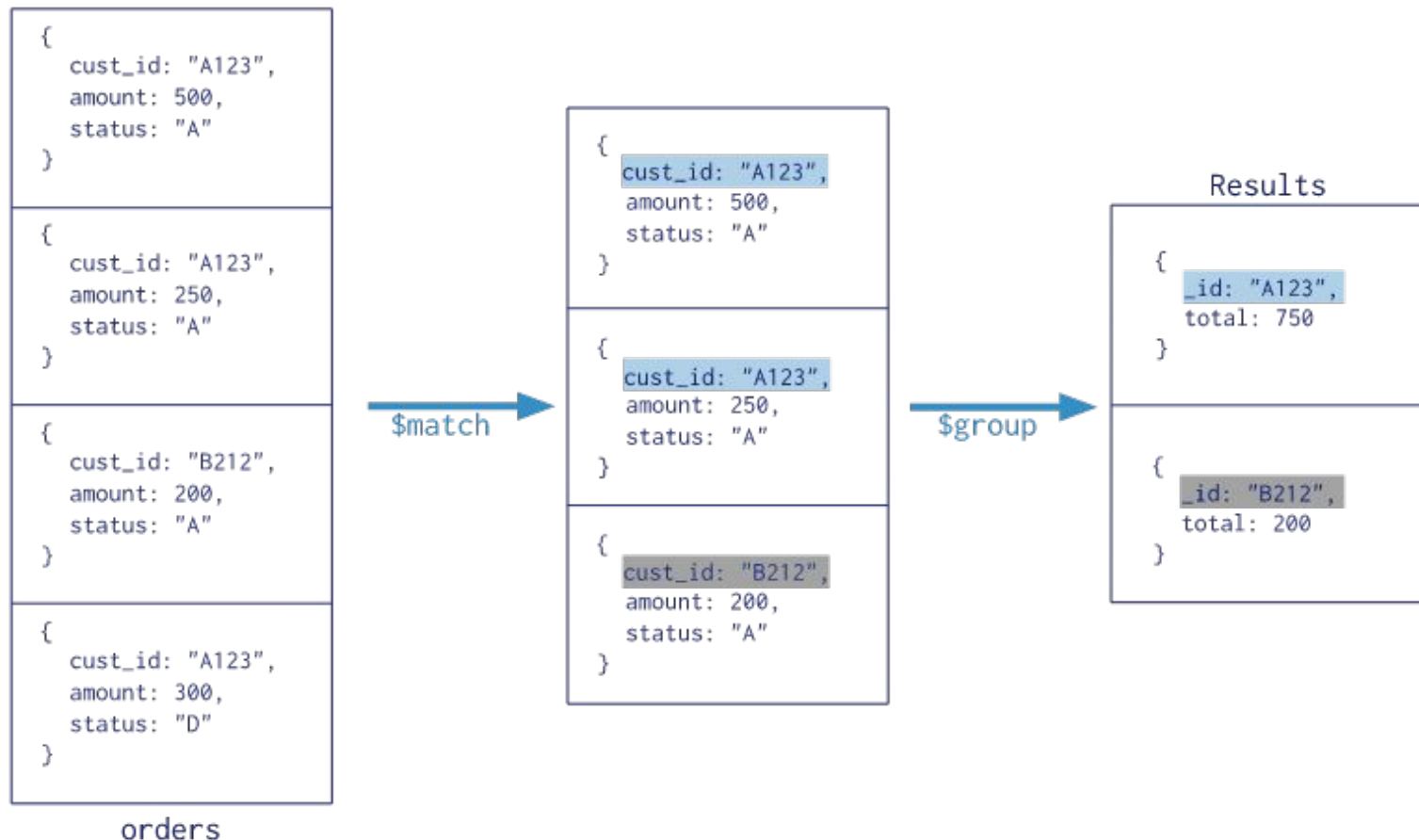
MongoDB

---

# Agregaciones

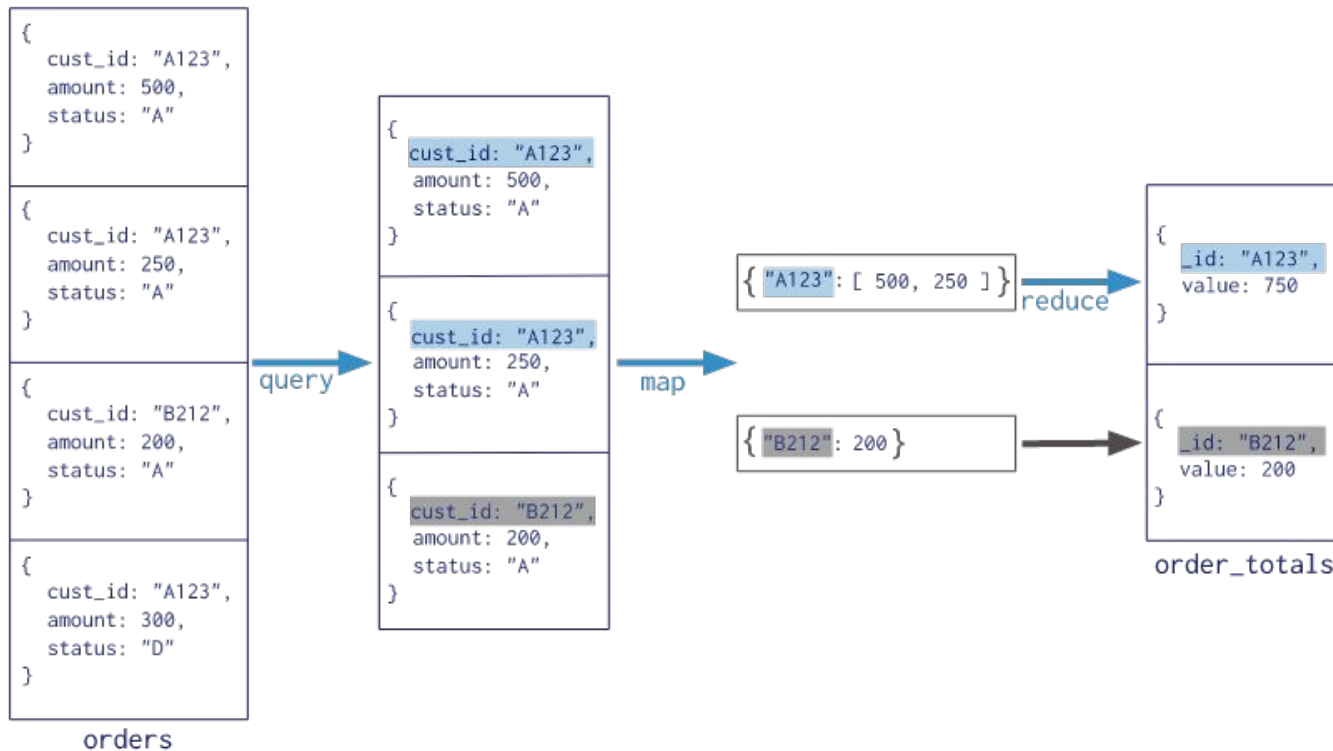
# Pipeline de agregaciones

Collection  
↓  
db.orders.aggregate( [  
 \$match stage → { \$match: { status: "A" } },  
 \$group stage → { \$group: { \_id: "\$cust\_id", total: { \$sum: "\$amount" } } }  
] )



# Map-Reduce

Collection  
↓  
db.orders.mapReduce(  
  map     → function() { emit( this.cust\_id, this.amount ); },  
  reduce  → function(key, values) { return Array.sum( values ) },  
  {  
    query: { status: "A" },  
    out: "order\_totals"  
  }  
)



# Agregaciones de único propósito

```
db.collection.estimatedDocumentCount()
```

```
db.collection.count()
```

```
db.collection.distinct()
```

---


Collection



```
db.orders.distinct( "cust_id" )
```

<pre>{   cust_id: "A123",   amount: 500,   status: "A" }</pre>
<pre>{   cust_id: "A123",   amount: 250,   status: "A" }</pre>
<pre>{   cust_id: "B212",   amount: 200,   status: "A" }</pre>
<pre>{   cust_id: "A123",   amount: 300,   status: "D" }</pre>

orders

 `distinct` [ "A123", "B212" ]

MongoDB

---

# Índices

---

# Definición

En MongoDB los índices ayudan a que las consultas sean más rápidas. Sin índices, MongoDB debe hacer un escaneo de toda la colección.



# **Tipos de índices**



- De un solo campo
- Compuestos
- Multi llave
- Geoespaciales
- De texto
- Hashed

MongoDB

---

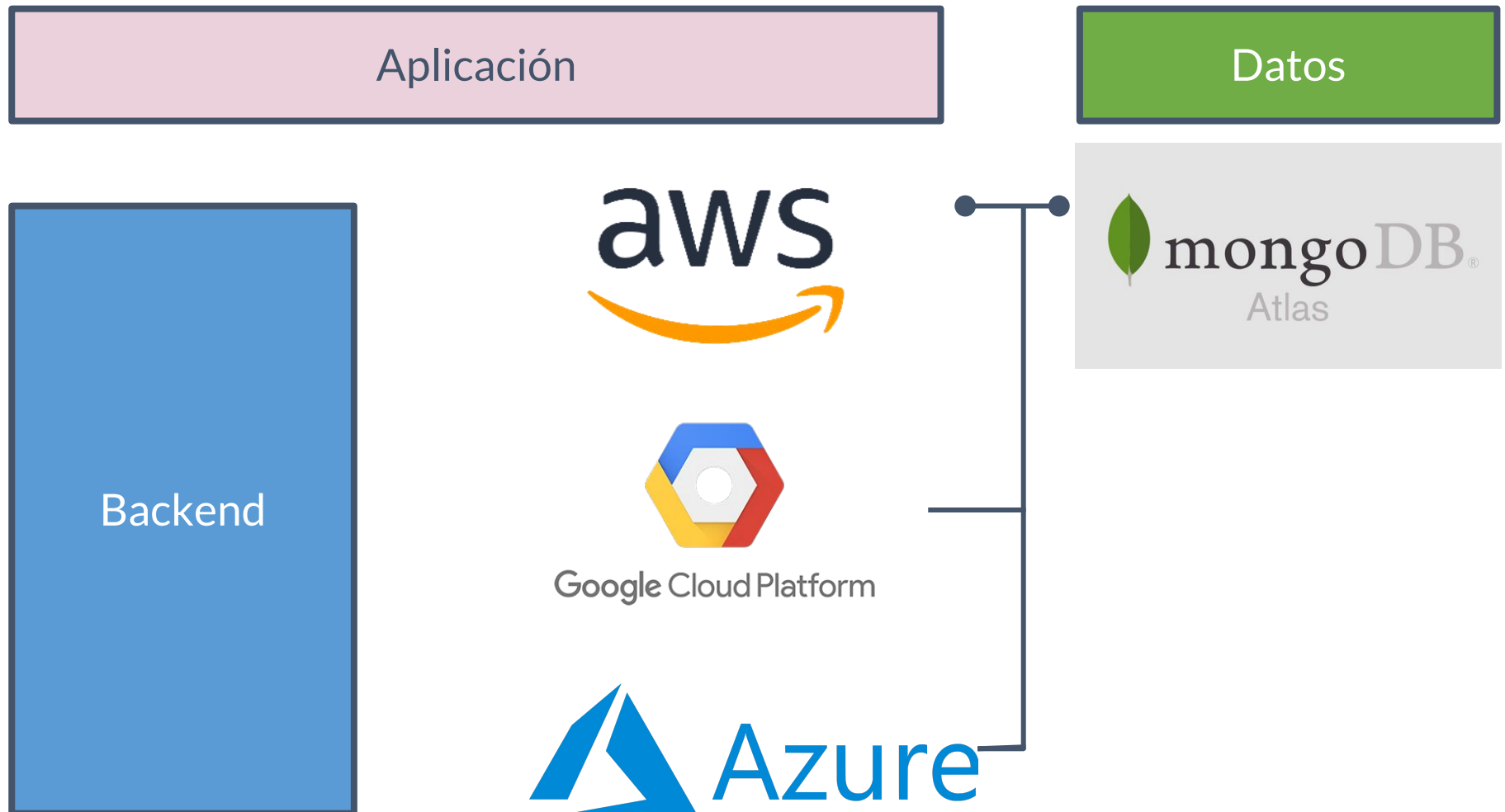
# Recomendaciones paso a producción

---

# Arquitectura recomendada

---

# Arquitectura



---

# Recomendaciones

---

- Guardar las credenciales en variables de entorno o archivos de configuración fuera del proyecto.
- Asegura que tu cluster se encuentre en la misma región del proveedor que tu aplicación.
- Has VPC peering entre la VPC de tu aplicación y la VCP de tu cluster.

- Cuida tu lista de IP's blancas
- Habilitar autenticación de dos pasos
- Actualiza constantemente tu versión de MongoDB

- Ten separados tus ambientes de dev/test/prod
- Habilita la opción de storage encriptado