MongoDB 101

Ivan Groenewold, Senior Architect

Buenos Aires, Argentina 2023

PERCONA UNIVERSITY

PERCONA

Arquitecto Senior en Percona

Co-founder CanalDBA

- Comunidad DBAs de habla hispana
- Unite a nuestro Slack! https://bit.ly/canaldba

X: @igroenew

Open Source

Simple

Flexible

Escalable

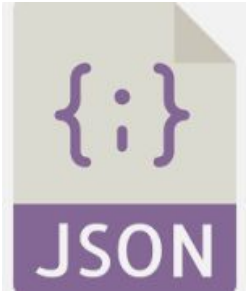
Performante

Creada en 2008 por 10gen

Para uso interno inicialmente

Éxito dentro de la compañía

Eventualmente lanzada como producto



JavaScript Object Notation (JSON)

- { nombre: “Juan”, “edad”: 30 }

El esquema cambia frecuentemente

Escalar fácilmente lecturas/escrituras

Cantidad significativa de datos

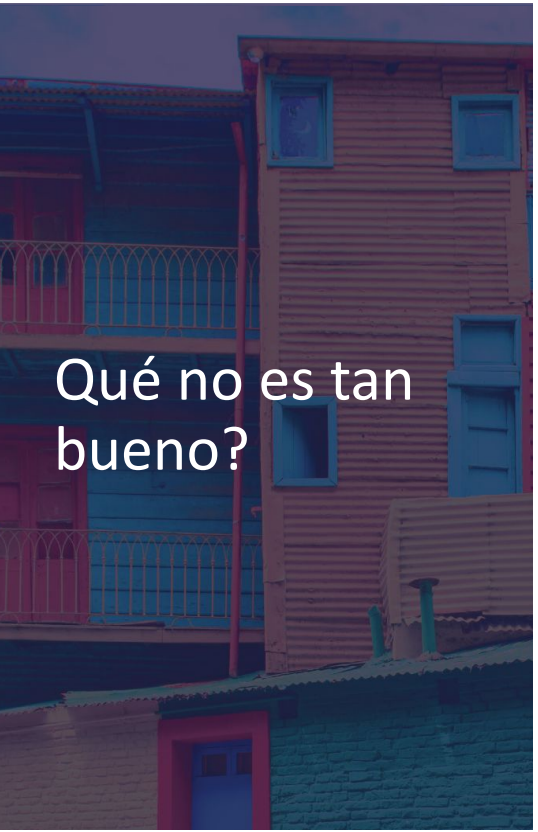
Cada documento es atómico

No mas JOINS*

Alta disponibilidad y sharding incluidos

Control del ciclo de vida de los datos

- Colecciones capped
- Indices TTL



No hay schema*

Soporte de transacciones primitivo

No hay integridad referencial (FK)

Lenguaje propietario (no SQL)

Objetivos de MongoDB

IoT, Cloud, base de datos distribuida

Fácil escalado horizontal y vertical

Desarrollo y release ágiles

Tipos de datos complejos

MongoDB Community Edition

MongoDB Enterprise Edition

- +LDAP, encriptación y auditoría

Percona Server for MongoDB

- Basada en MongoDB Community
- + features de Mongo Enterprise
- +features de Percona
- Gratis!

MongoDB Atlas

- Features adicionales

AWS DocumentDB (*)

Azure Cosmos DB Emulation API (*)

(*) Compatibles con el **protocolo** MongoDB

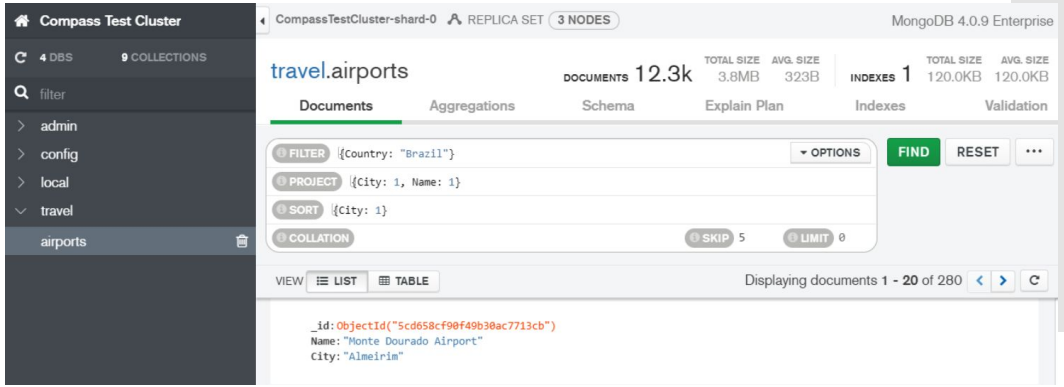
Mongo Shell

MongoDB Compass

MongoDB Atlas

Studio 3T

...



Conceptos de MongoDB

Instancia

Base de datos (BD)

Colección

Indice

Documento

Proceso *mongod* en memoria

Manejo de clientes, alocación de memoria, etc.

Operar con los archivos en disco

Mongo Shell nos da acceso a la instancia

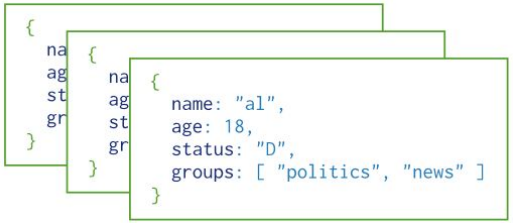
```
[vagrant@mongo0 ~]$ ps -ef | grep mongo | grep -v grep
mongod     986     1  1 12:59 ?        00:02:36 /usr/bin/mongod -f /etc/mongod.conf
```

Conjunto de archivos en disco

- /var/lib/mongo/

Colecciones e índices

Múltiples bases de datos por instancia

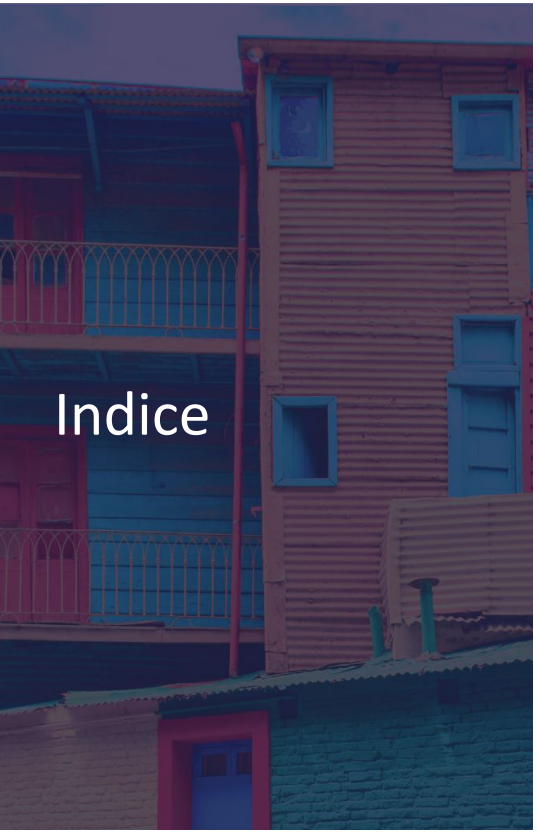


Conjunto de documentos

Una colección pertenece a una única BD

Límite opcional (cap) por colección

- Tamaño
- Numero de documentos

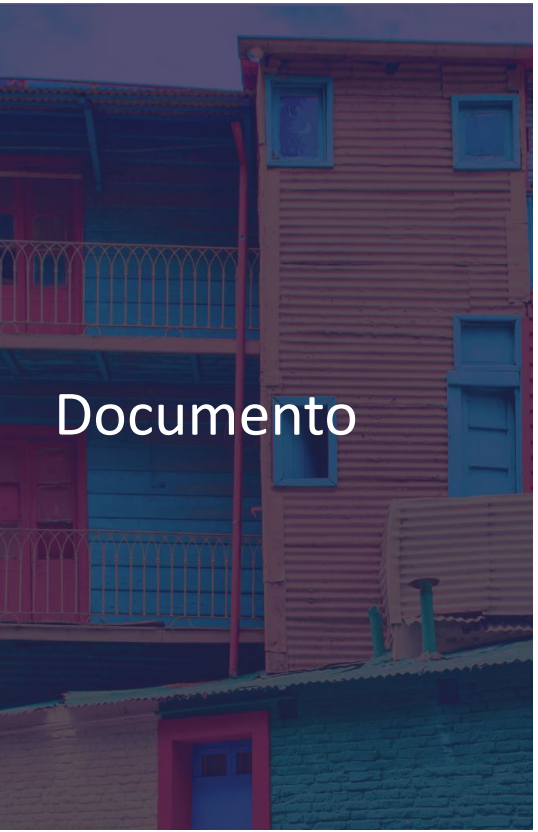


Los indices son BTree

Similares a otras bases de datos

Aceleran las lecturas

Penalizan las escrituras




```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

Consiste en pares de clave/valor

Cada campo tiene un tipo asociado (BSON)

Máximo 16 MB por documento

Opcional: Reglas de validación (Schema)

El campo _*id*

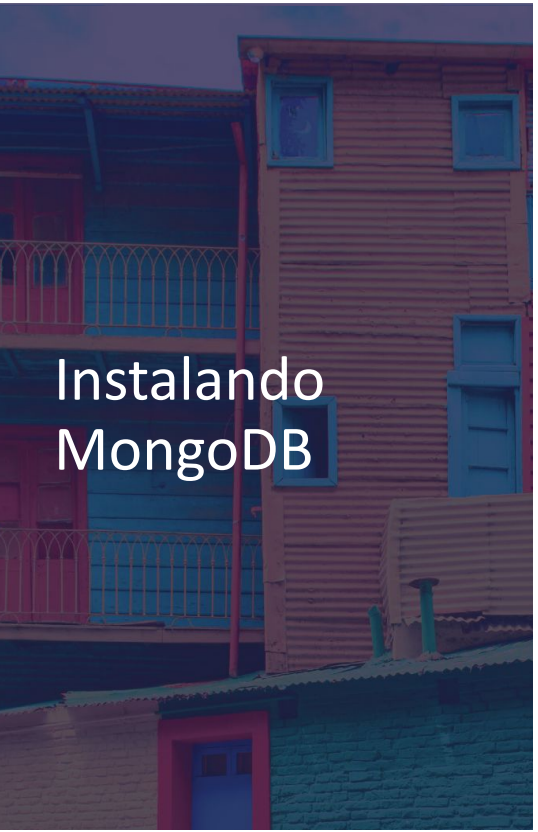
Cada documento requiere un _*id*

Se auto-genera si no lo especificamos

Puede usar cualquier tipo BSON menos Array

Default: tipo ObjectID

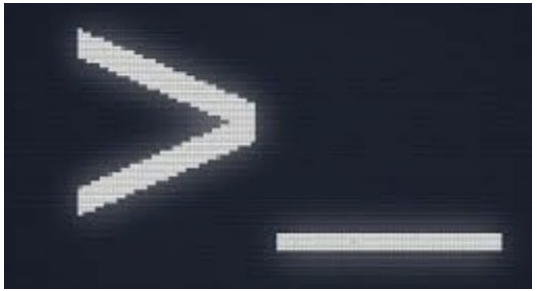
MongoDB	*BD Relacional*
Instancia	Instancia
Instancia	Instancia
Coleccion	**Tabla**
Indice	Indice
Documento	**Fila/Registro**
Campo	**Columna**



Repositorios de Percona o MongoDB Inc

Paths importantes

- /var/lib/mongo
- /etc/mongod.conf
- /var/log/mongo/mongod.log



Interfaz JavaScript

<Tab> para autocompletar

Queries

Administración

Autenticación

Por default MongoDB no requiere usuario/password

Localhost exception para crear el primer usuario

Los usuarios se crean en la colección system.users

- Por convención en la DB admin

Conectarse localmente

- mongosh
- mongosh admin -u dba -p pwd

Conectarse a un servidor remoto

- mongosh "mongodb://dba:pwd@127.0.0.1:27017/admin"

Insertar un documento

- use sample
- db.movies.insert({ “title”: “Home Alone”, “year”: 1990, “rating”: 6 })

Buscar documentos

- db.movies.find({ “year”: 1990 })
- db.movies.find({ “year”: 1990 }).pretty()

Insertar un documento más complejo

- use sample
- db.movies.insert([
  {
  "title" : "Avatar",
  "box_office" : {
  "gross" : 760,
  "budget" : 237,
  "opening_weekend" : 77 }
  }
  ])

Utilizar el punto (.) para buscar subdocumentos

- use sample
- db.movies.find({ "box_office.gross" : 760 })

$or: alguno de los valores

$not: negación

$nor: ninguno de los valores

$and: todos los valores

Comparadores

$lt: existe y menor que

$lte: existe y menor o igual que

$gt: existe y mayor que

$gte: existe y mayor o igual que

Películas de sci-fi o buen rating

- db.movies.find({ $or : [
  { "category" : "sci-fi" },
  { "imdb_rating" : { $gte : 7 } }
  ] })

Peliculas malas

- db.movies.find({ "imdb_rating" : { $not : { $gt : 7 } } })
- db.movies.find({ "imdb_rating" : { $lte : 7 } })

Aplicar sort() al cursor

- db.movies.find().sort({ "rating": 1 })

Orden descendente

- db.movies.find().sort({ "rating": -1 })

Múltiples campos

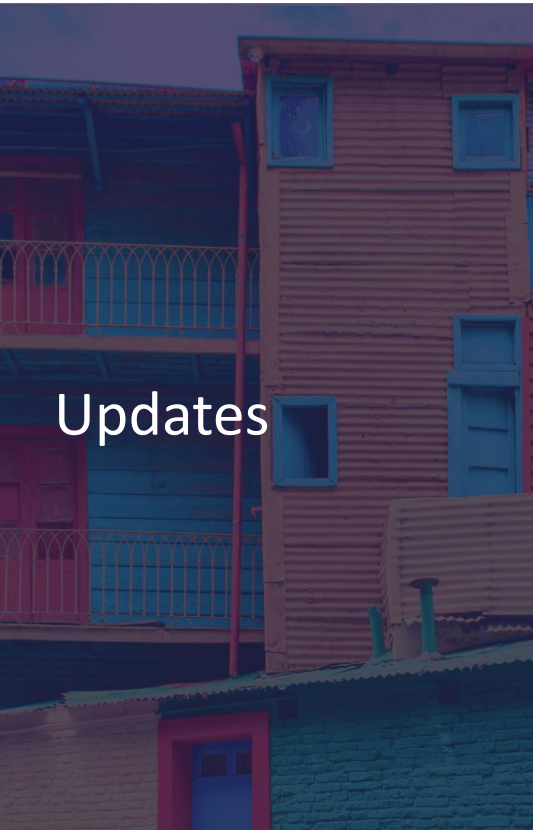
- db.movies.find().sort({ "rating": 1, “category”: 1 })

Aplicar limit() al cursor

Se puede combinar con sort()

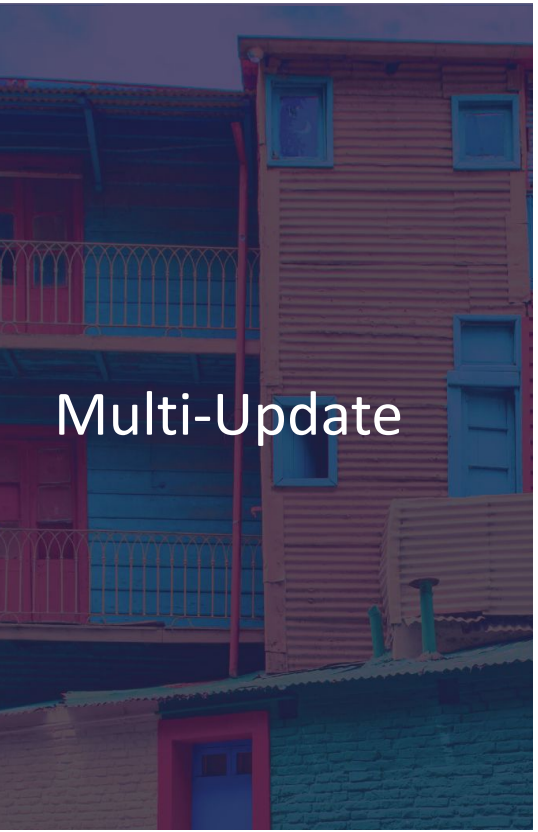
Top 5:

- db.movies.find().sort({ "rating": -1 }).limit(5)



db.movies.update({ query }, { update })

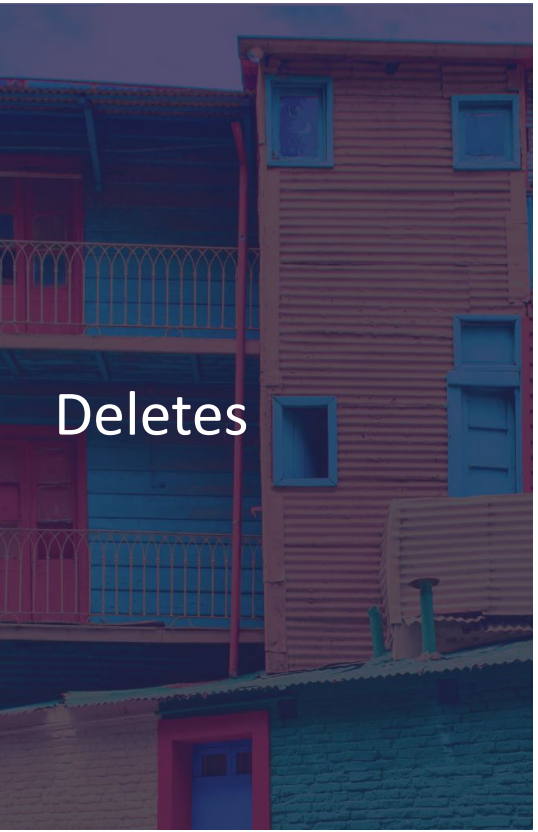
- db.movies.update(
  { 'category': 'action' },
  { $inc: { 'rating': 1 } }
)
  WriteResult({ "nMatched" : 2, "nUpserted" : 0,
  "nModified" : 1 })



db.movies.update({ query }, { update }, { opciones })

Modificar todos los documentos:

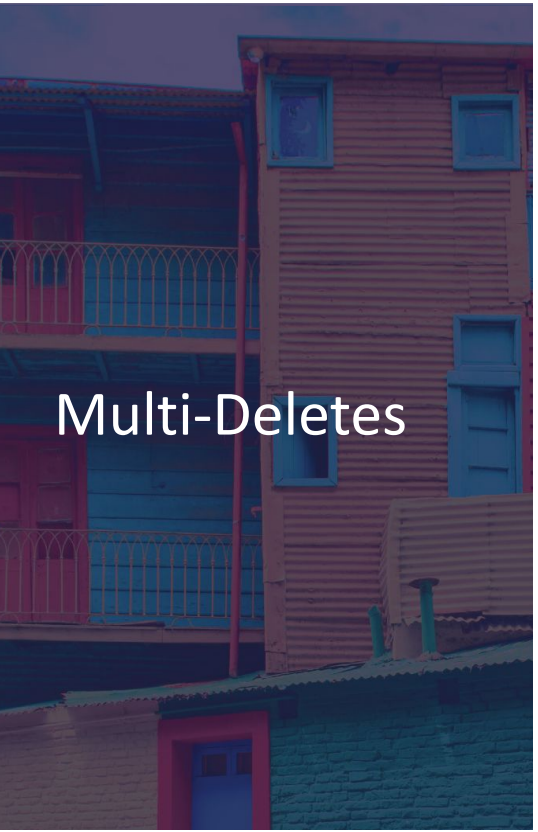
- db.movies.update(
  { 'category': 'action' },
  { $inc: { 'rating': 1 } },
  { multi: true }
)
  …
  WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })



db.movies.deleteOne()

Borra un único documento si cumple la condición

- db.movies.deleteOne({ "_id" : "Godzilla" })



db.movies.deleteMany()

Borra todos los documentos que cumplan la condición

- db.movies.deleteMany({ "category" : "adventure" })
  { "acknowledged" : true, "deletedCount" : 2 }

Proyección

Devuelve solo algunos campos

- Similar a “SELECT col1, col2, … FROM table…”

_*id*_ siempre se incluye

- Salvo que lo quitemos explícitamente

Devolver solo campos item y status

- db.inventory.find({ status: "A" }, { item: 1, status: 1 })

Mostrar todo menos el *_id*

- db.inventory.find({ status: "A" }, { _id: 0})

No mostrar un campo de un subdocumento

- db.inventory.find({ status: "A" }, { "size.uom": 0 })

Las consultas a la base son más rápidas y eficientes

Evitar barrer toda la colección (COLLSCAN)

Prevenir duplicados (Unique)

Devolver resultados pre-ordenados

Creando Indices

db.movies.createIndex({ title: 1 })

db.movies.createIndex({ title: -1 })

- Orden inverso

db.movies.createIndex({ location.city: 1 })

db.movies.createIndex({ title: 1 , category: 1 })

Text index

- db.stores.createIndex({ name: "text", description: "text" }
- db.stores.find({ $text: { $search: "java coffee shop" } }) // cualquier palabra

TTL index

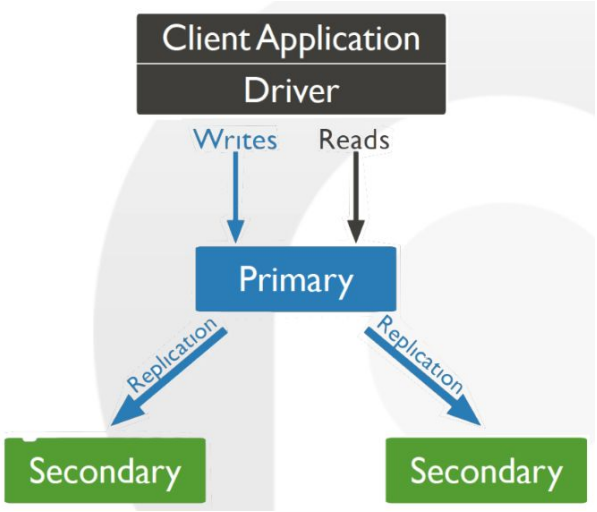
- db.eventlog.createIndex(
  { "lastModifiedDate": 1 },
  { expireAfterSeconds: 3600 }
)

**db.movies.find({category: "sci-fi" }
).explain().queryPlanner.winningPlan**

```
{
"stage" : "COLLSCAN",
"filter" : {
        "category" : {
                "$eq" : "sci-fi"
        }
},
"direction" : "forward"
}
```

Replica Set (RS)

- 1 o mas servers que tienen los mismos datos
- Auto provisionado
- Alta disponibilidad y redundancia
- Mínimo 3 nodos para producción



Cómo funciona la Replicación

Coleccion *oplog.rs* dentro de la BD local

Capped collection (default 5% del disco)

Un documento en el oplog por cada escritura en el primary

Cada operación es idempotente

Cómo funciona la Replicación (2)

Los secondary copian el oplog a través de la red

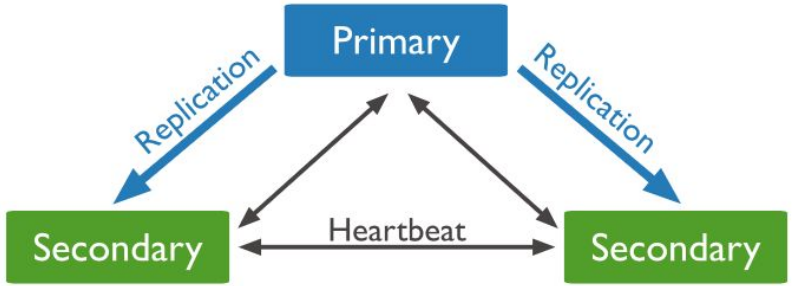
Aplican las transacciones en paralelo

Heartbeat entre nodos (cada 2 seg)

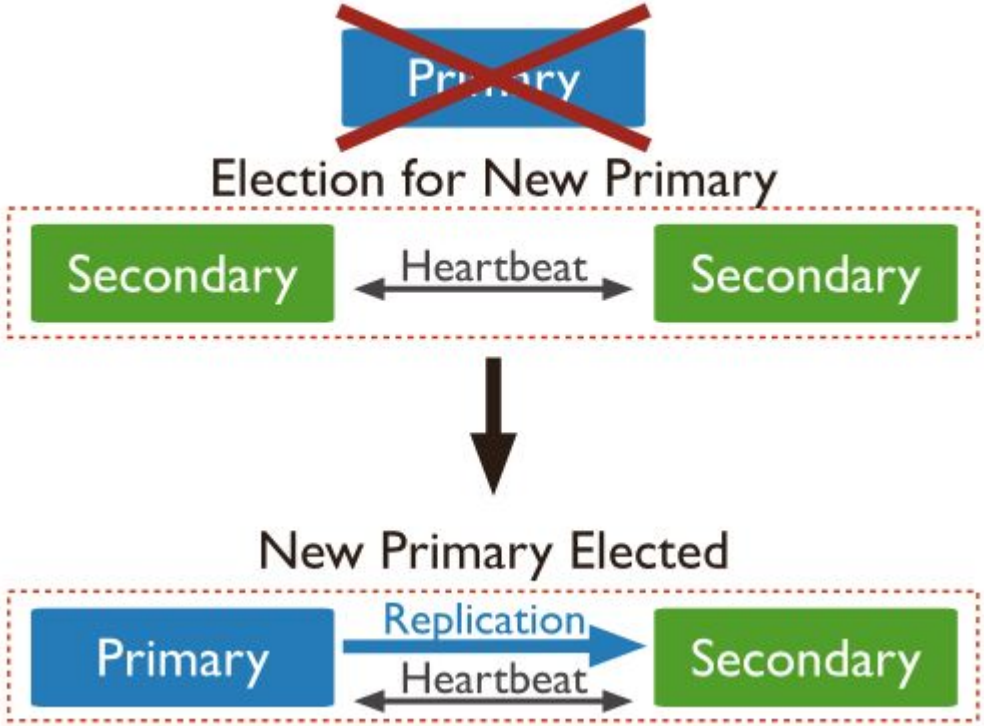
Si el Primary no responde -> elección

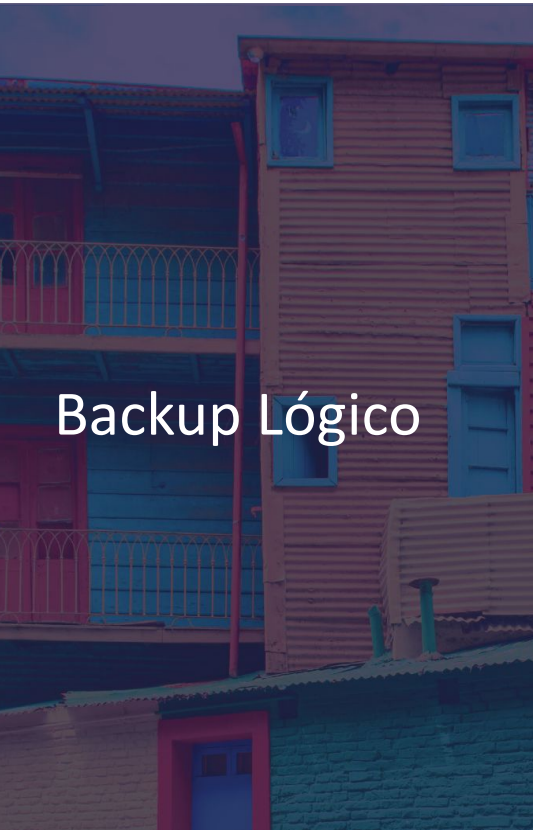
1 voto por nodo (default)

El número de nodos debe ser impar



Elección





Mongoexport

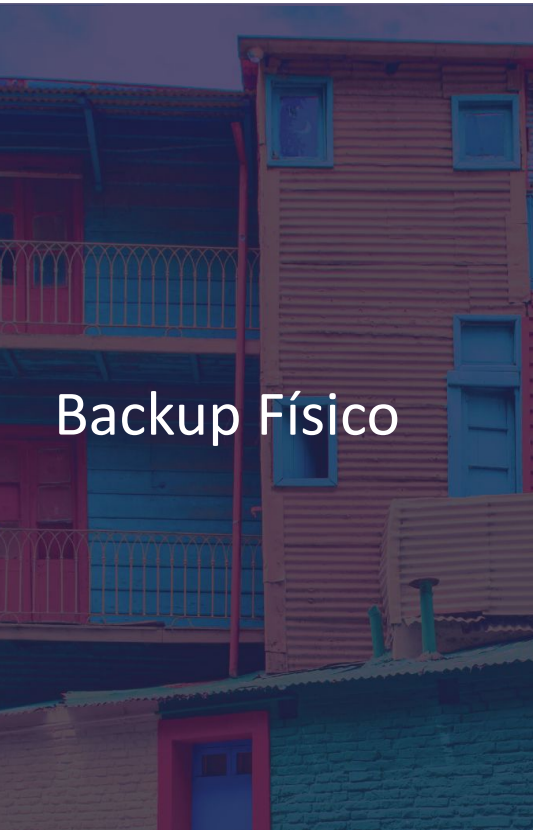
- JSON/CSV

Mongodump

- BSON

Percona Backup for MongoDB

- Soporta sharding y point-in-time recovery
- NFS/S3-compatible storage



MongoDB Atlas

- Snapshots

Cloud Manager/Ops Manager

- Streaming backup

Tu propios snapshots

- db.fsyncLock()

Percona Backup for MongoDB

- Hotbackup

MongoDB Ops/Cloud Manager ($$$)

Percona Monitoring & Management (PMM)

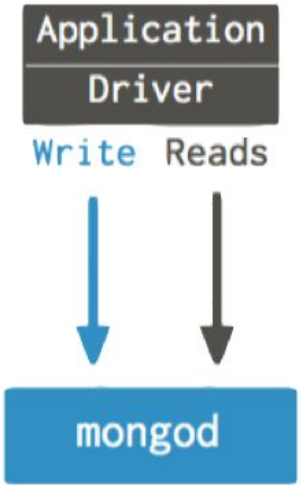
- Free Open Source Software

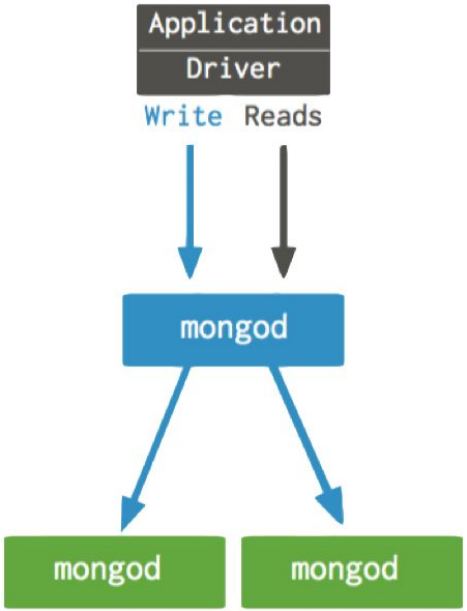
Grafana

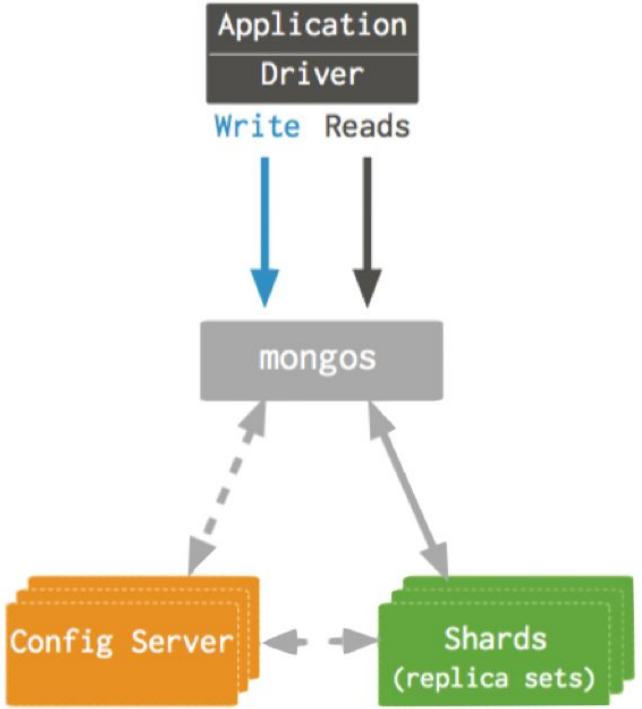
Datadog

NewRelic

Como Escalar





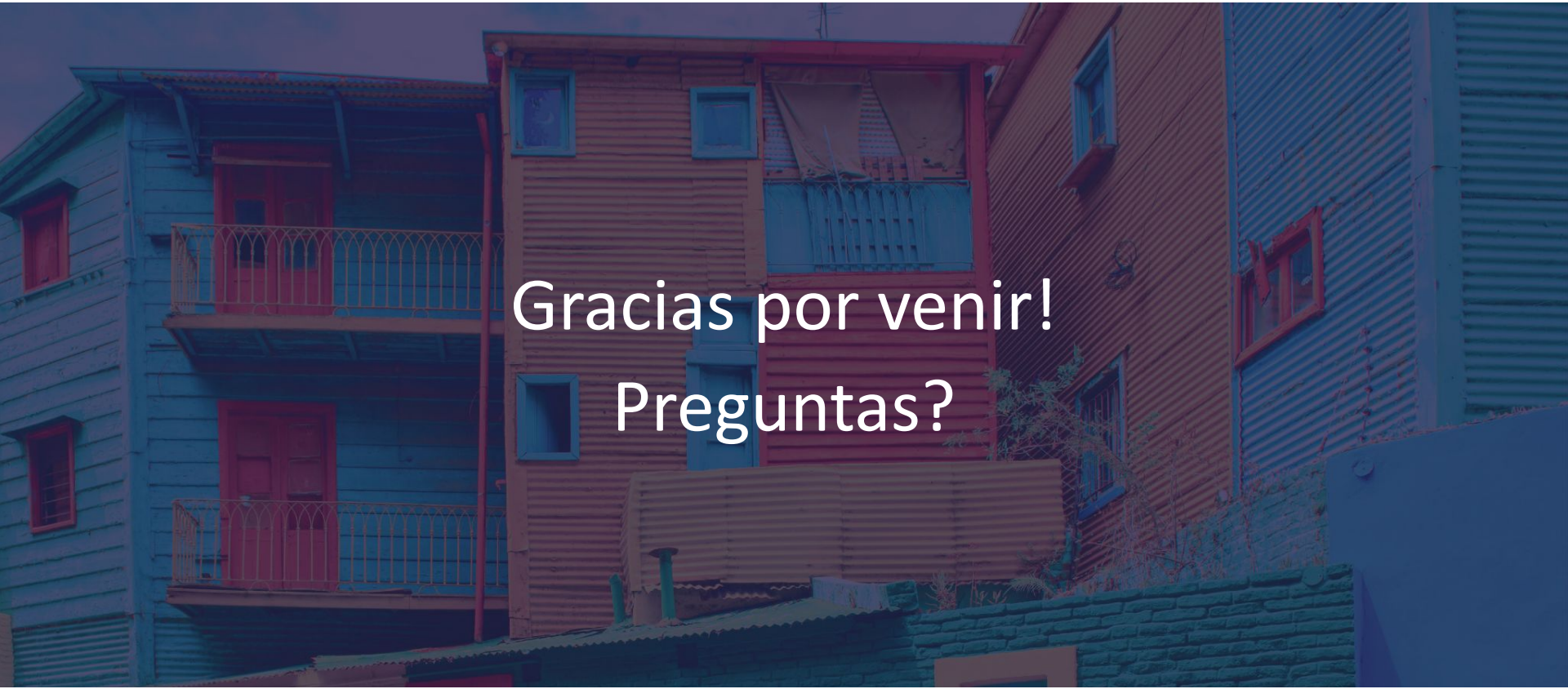


Facil de empezar

Integrado con JSON

Modelo de documentos flexible

Herramientas para escalar incluidas



PERCONA