mongoDB

Comandos y Herramientas adicionales

Enrique Barra

Comandos

EXPLAIN

- Para ver qué está haciendo la BBDD con la query. Si está usando un índice o recorriendo la colección completa
- No devuelve los datos sólo explica qué haría

- db.records.explain().find({ student_id: 134}).sort({class: -1})

- Verbosidad. 3 niveles.
  - **queryplanner**: es el por defecto. Muestra el plan para ejecutar la query
  - **executionStats**: muestra el plan para ejecutar la query y las estadísticas de ejecución de ese plan
  - **allPlansExecution**: muestra el plan ganador y todos los planes de ejecución que se han rechazado (con indice, sin indice, …)

- db.records.explain("executionStats").find({ student_id: 134})
- db.records.explain("allPlansExecution").find({ student_id: 134})

```
db.zips.find({pop: {$gt:10000}}).explain()

{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "zips2.zips",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "pop" : {
                                "$gt" : 10000
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "pop" : {
                                        "$gt" : 10000
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "pizarro",
                "port" : 27017,
                "version" : "3.2.3",
                "gitVersion" : "b326ba837cf6f49d65c2f85e1b70f6f31ece7937"
        },
        "ok" : 1
}
```

- db.zips.find({pop: {$gt:10000}}).explain("executionStats")


```
{
    "queryPlanner" : {...}
     "executionStats" : {
            "executionSuccess" : true,
            "nReturned" : 7645,
            "executionTimeMillis" : 28,
            "totalKeysExamined" : 0,
            "totalDocsExamined" : 29467,
            "executionStages" : {
                  "stage" : "COLLSCAN",
                  "filter" : {
                        "pop" : {
                              "$gt" : 10000
                        }
                  },
                  "nReturned" : 7645,
                  "executionTimeMillisEstimate" : 20,
                  "works" : 29469,
                  "advanced" : 7645,
                  "needTime" : 21823,
                  "needYield" : 0,
                  "saveState" : 230,
                  "restoreState" : 230,
                  "isEOF" : 1,
                  "invalidates" : 0,
                  "direction" : "forward",
                  "docsExamined" : 29467
            }
    },
```

db.restaurants.find({nombre: "Morris Park Bake Shop"}).explain("executionStats")


```
"queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "zips.restaurants",
        "indexFilterSet" : false,
        "parsedQuery" : {
                "nombre" : {
                        "$eq" : "Morris Park Bake Shop"
                }
        },
        "winningPlan" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "nombre" : {
                                "$eq" : "Morris Park Bake Shop"
                        }
                },
                "direction" : "forward"
        },
        "rejectedPlans" : [ ]
},
```


CONTINUACION:


```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 18,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 25360,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "nombre" : {
                                "$eq" : "Morris Park Bake Shop"
                        }
                },
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 10,
                "works" : 25362,
                "advanced" : 1,
                "needTime" : 25360,
                "needYield" : 0,
                "saveState" : 198,
                "restoreState" : 198,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 25360
        }
},
```

Agregamos índices

```
> db.zips.createIndex({pop:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}


> db.restaurants.createIndex({nombre:1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

```
> db.zips.find({pop: {$gt:10000}}).explain()
{
	"queryPlanner" : {
		"plannerVersion" : 1,
		"namespace" : "zips2.zips",
		"indexFilterSet" : false,
		"parsedQuery" : {
			"pop" : {
				"$gt" : 10000
			}
		},
		"winningPlan" : {
			"stage" : "FETCH",
			"inputStage" : {
				"stage" : "IXSCAN",
				"keyPattern" : {
					"pop" : 1
				},
				"indexName" : "pop_1",
				"isMultiKey" : false,
				"isUnique" : false,
				"isSparse" : false,
				"isPartial" : false,
				"indexVersion" : 1,
				"direction" : "forward",
				"indexBounds" : {
					"pop" : [
						"(10000.0, 1.#INF]"
					]
				}
			}
		},
		"rejectedPlans" : [ ]
	},
	"serverInfo" : {
		"host" : "pizarro",
		"port" : 27017,
		"version" : "3.2.3",
		"gitVersion" : "b326ba837cf6f49d65c2f85e1b70f6f31ece7937"
	},
	"ok" : 1
}
```

- db.zips.find({pop: {$gt:10000}}).explain("executionStats")


```
{
 "queryPlanner" : {…}
  "executionStats" : {
       "executionSuccess" : true,
       "nReturned" : 7645,
       "executionTimeMillis" : 12,
       "totalKeysExamined" : 7645,
       "totalDocsExamined" : 7645,
       "executionStages" : {
             "stage" : "FETCH",
             "nReturned" : 7645,
             "executionTimeMillisEstimate": 10,
             "works" : 7646,
             "advanced" : 7645,
             "needTime" : 0,
             "needYield" : 0,
             "saveState" : 59,
             "restoreState" : 59,
             "isEOF" : 1,
             "invalidates" : 0,
             "docsExamined" : 7645,
             "alreadyHasObj" : 0,
```


CONTINUACION:


```
"inputStage" : {
             "stage" : "IXSCAN",
             "nReturned" : 7645,
             "executionTimeMillisEstimate" : 10,
             "works" : 7646,
             "advanced" : 7645,
             "needTime" : 0,
             "needYield" : 0,
             "saveState" : 59,
             "restoreState" : 59,
             "isEOF" : 1,
             "invalidates" : 0,
             "keyPattern" : {"pop" : 1  },
             "indexName" : "pop_1",
             "isMultiKey" : false,
             "isUnique" : false,
             "isSparse" : false,
             "isPartial" : false,
             "indexVersion" : 1,
             "direction" : "forward",
             "indexBounds" : {
                   "pop" : [
                         "(10000.0, 1.#INF]"
                   ]
             },
             "keysExamined" : 7645,
             "dupsTested" : 0,
             "dupsDropped" : 0,
             "seenInvalidated" : 0
       }
}
```

- db.restaurants.find({nombre: "Morris Park Bake Shop"}).explain("executionStats")


```
{
    "queryPlanner" : {
        "plannerVersion" : 1,
        "namespace" : "zips.restaurants",
        "indexFilterSet" : false,
        "parsedQuery" : {
            "nombre" : {
                "$eq" : "Morris Park Bake Shop"
            }
        },
        "winningPlan" : {
            "stage" : "FETCH",
            "inputStage" : {
                "stage" : "IXSCAN",
                "keyPattern" : {
                    "nombre" : 1
                },
                "indexName" : "nombre_1",
                "isMultiKey" : false,
                "multiKeyPaths" : {
                    "nombre" : [ ]
                },
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 2,
                "direction" : "forward",
                "indexBounds" : {
                    "nombre" : [
                        "[\"Morris Park Bake Shop\", \"Morris Park Bake Shop\"]"
                    ]
                }
            }
        },
        "rejectedPlans" : [ ]
    },
    "executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 1,
        "totalDocsExamined" : 1,
        "executionStages" : {
            "stage" : "FETCH",
            "nReturned" : 1,
            "executionTimeMillisEstimate" : 0,
            "works" : 2,
            "advanced" : 1,
            "needTime" : 0,
            "needYield" : 0,
            "saveState" : 0,
            "restoreState" : 0,
            "isEOF" : 1,
            "invalidates" : 0,
            "docsExamined" : 1,
            "alreadyHasObj" : 0,
            "inputStage" : {
                "stage" : "IXSCAN",
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 0,
                "works" : 2,
                "advanced" : 1,
                "needTime" : 0,
                "needYield" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
                "invalidates" : 0
```

Otros comandos y utilidades

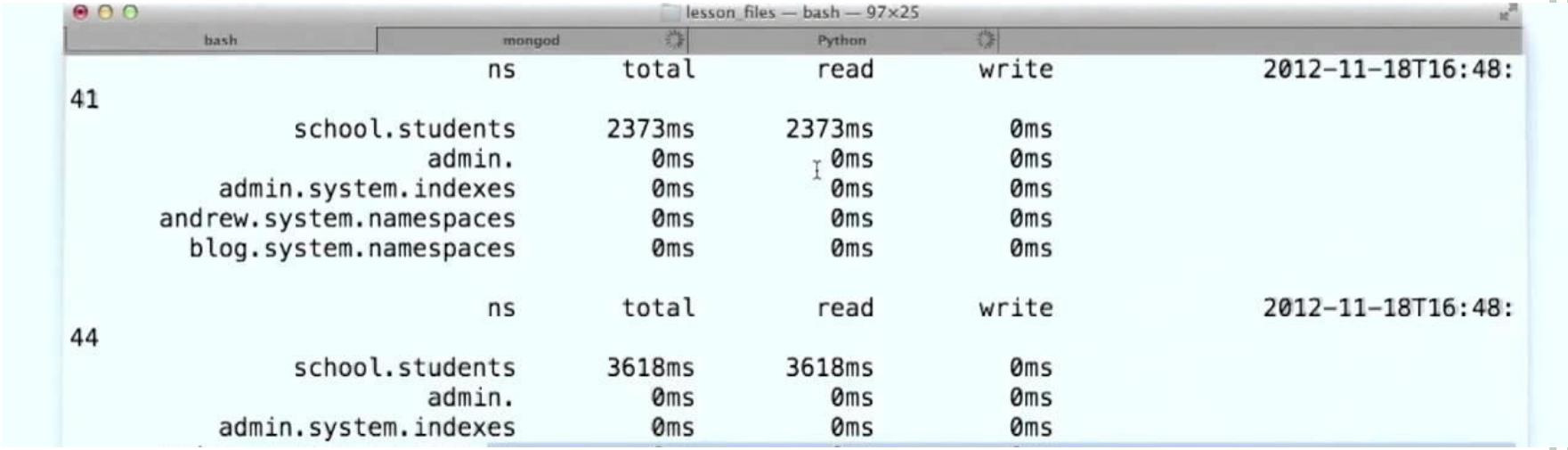
- **Profiling**.
  - https://docs.mongodb.org/manual/tutorial/manage-the-database-profiler/
  - Para analizar el rendimiento de las queries y logear queries lentas.
  - Mongo escribirá la salida del profiler a la colección system.profile

- Ver el estado del profiling:
  - db.getProfilingStatus()
- Fijar el profiling:
  - db.setProfilingLevel(**level, slowms**)
- **Level (integer)**: 0 para Off, 1 para log de slow queries, 2 para log de todas las queries
- **Slowms (integer)**: milisegundos que debe tardar una query al menos para ser considerada lenta

- Para hacer análisis, se puede llamar al profiler directamente:
  - db.system.profile.find({ millis : { $gt : 100 } })
  - Devuelve las operaciones que tardaron más de 100ms

Otros comandos y utilidades

- **Mongotop**:
  - Método para imprimir la cantidad de tiempo que MongoDB pasa leyendo y escribiendo datos. Estadísticas son por colección.
  - mongotop 3 #cada 3 segs
  - https://docs.mongodb.org/manual/reference/program/mongotop/



Otros comandos y utilidades

- **Mongostat**
  - Overview del estado de una instancia mongos o mongod. Similar al comando de Unix vmstat.
  - https://docs.mongodb.org/manual/reference/program/mongostat/

```
connected to: localhost:10001
insert  query update delete getmore command flushes mapped  vsize    res faults locked % idx miss %     qr|qw   ar|aw   netIn netOut  conn       set repl       time
    0      1      0      0       0     100       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     6k    25k   101 orchid_1    M   21:30:43
    0      0      0      0       1     109       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     6k    29k   101 orchid_1    M   21:30:44
    0      0      1      0       0      31       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     1k     9k   104 orchid_1    M   21:30:45
    0      0      0      0       0      23       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     1k     6k   104 orchid_1    M   21:30:46
    0      0      0      0       0      20       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     1k     6k   104 orchid_1    M   21:30:47
    0      1      0      0       1      32       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     2k     8k   103 orchid_1    M   21:30:48
    0      0      1      0       0      32       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     1k     9k   102 orchid_1    M   21:30:49
    0      0      0      0       0      44       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     2k    12k   103 orchid_1    M   21:30:50
    0      0      0      0       0      16       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0   932b     5k   104 orchid_1    M   21:30:51
    0      0      0      0       1      45       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     2k    12k   102 orchid_1    M   21:30:52
insert  query update delete getmore command flushes mapped  vsize    res faults locked % idx miss %     qr|qw   ar|aw   netIn netOut  conn       set repl       time
    0      1      1      0       0      17       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     1k     5k   104 orchid_1    M   21:30:53
    0      0      0      0       0       7       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0   585b     2k   104 orchid_1    M   21:30:54
    0      0      0      0       0       5       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0   294b     2k   102 orchid_1    M   21:30:55
    0      0      0      0       1       9       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0   748b     3k   103 orchid_1    M   21:30:56
    0      0      1      0       0      28       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     2k   188k   104 orchid_1    M   21:30:57
    0      1      0      0       0      31       0  6.23g   7.6g  2.13g      0        0          0       0|0     2|0     2k     7k   102 orchid_1    M   21:30:58
```

Imagen: https://jira.mongodb.org/secure/attachment/13567/mongostat.png

Backup y restore de los datos

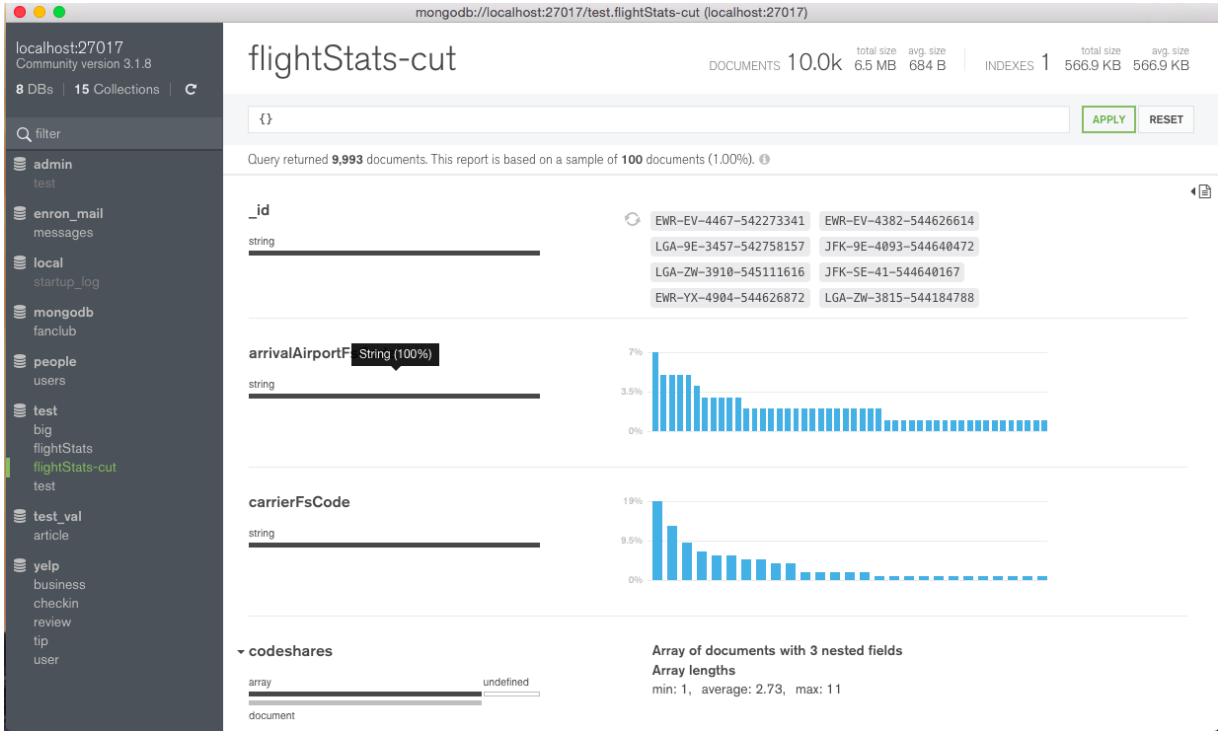
- Comandos mongodump y mongorestore usan BSON
- Comando **mongodump.** Sintaxis:
  - > mongodump --host HOST_NAME --port PORT_NUMBER
  - > mongodump --dbpath DB_PATH --out BACKUP_DIRECTORY
  - > mongodump --collection COLLECTION --db DB_NAME
- Comando **mongorestore**. Sintaxis:
  - mongorestore --collection COLLECTION --db DB_NAME file.bson
- Comandos mongoexport y mongoimport usan JSON (hay tipos de datos de BSON que no se soportan)
- Comando **mongoexport**:
  - mongoexport --db test --collection traffic --out traffic.json
- Comando **mongoimport**:
  - mongoimport --db users --collection contacts --file contacts.json

Herramientas adicionales

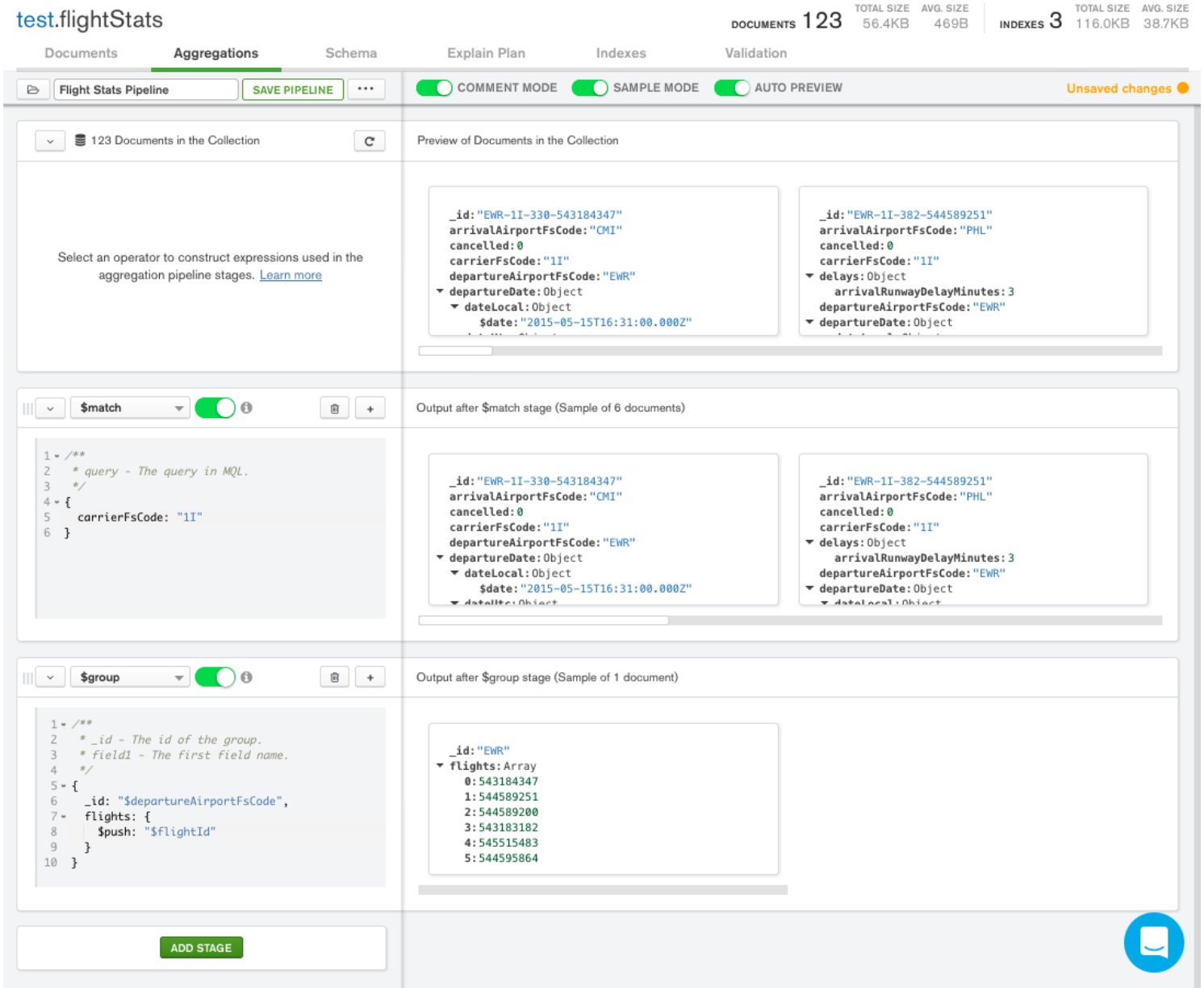
MongoDB Compass

MongoDB Compass

- https://www.mongodb.com/products/compass
- Es una GUI para MongoDB
- Funcionalidades:
  - explorar los datos visualmente,
  - ejecutar queries,
  - toda la funcionalidad CRUD



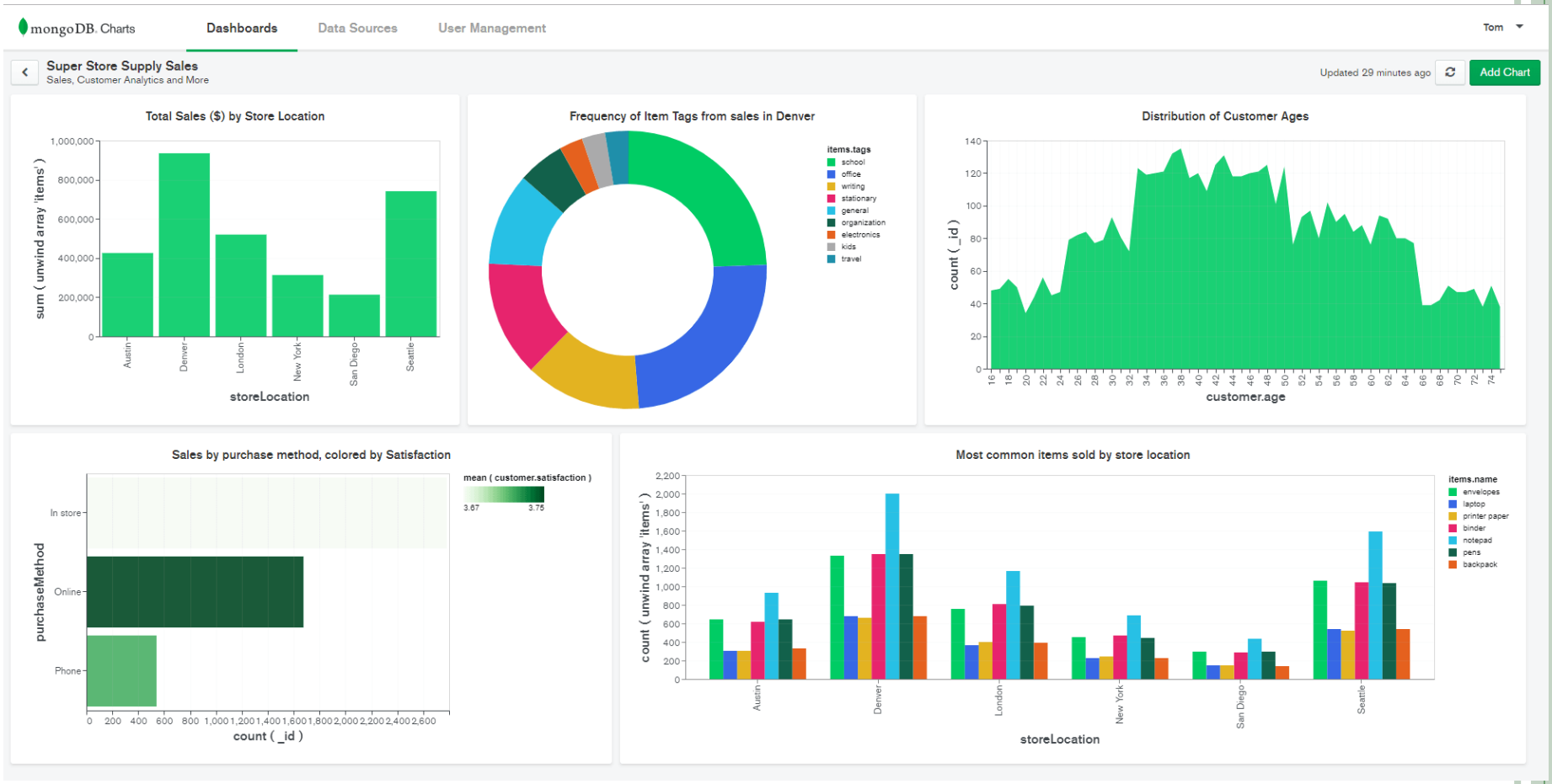
NOVEDAD EN COMPASS – CREADOR DE PIPELINES



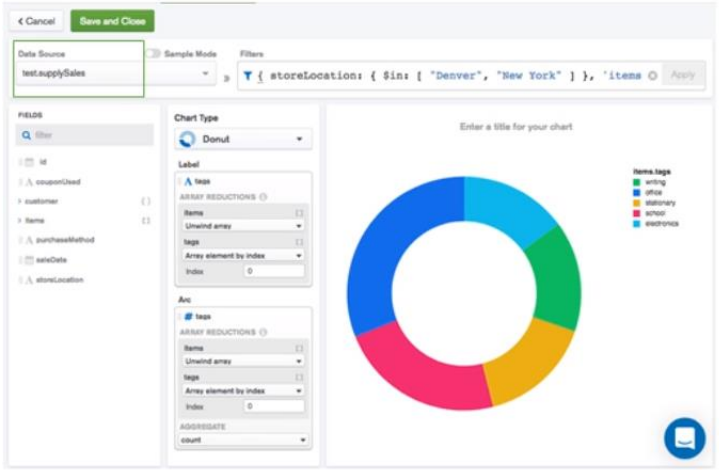
MongoDB Charts

MongoDB Charts

- https://docs.mongodb.com/charts/master/
- Visualización de datos (también en tiempo real):



MongoDB Charts: crear, visualizar, compartir



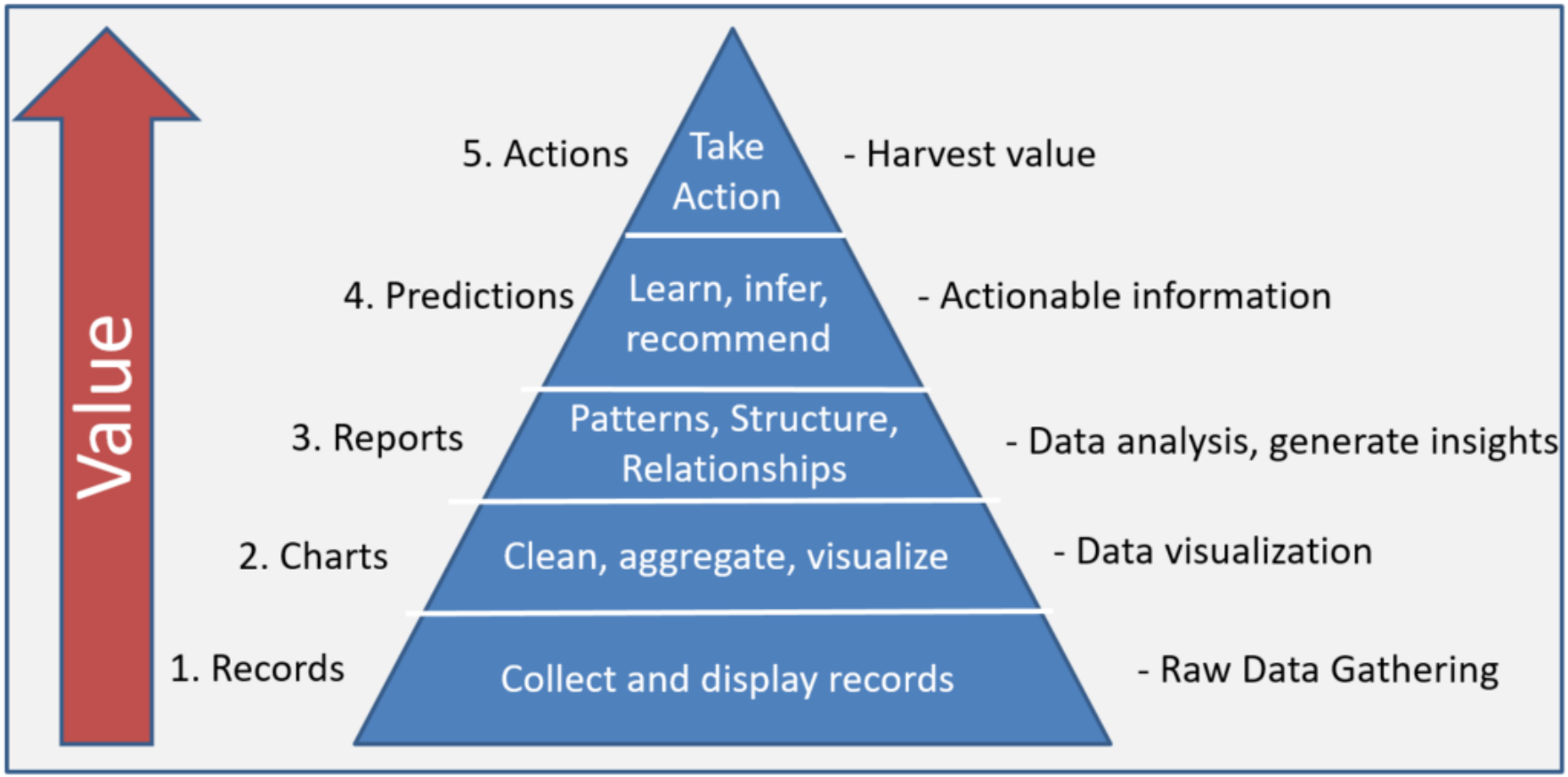
Trabajar con datos complejos.

Conectarse a fuentes de datos con seguridad. Filtros, muestras, visualizaciones.

Compartir paneles de control y colaborar.

mongoDB

Data Value Pyramid



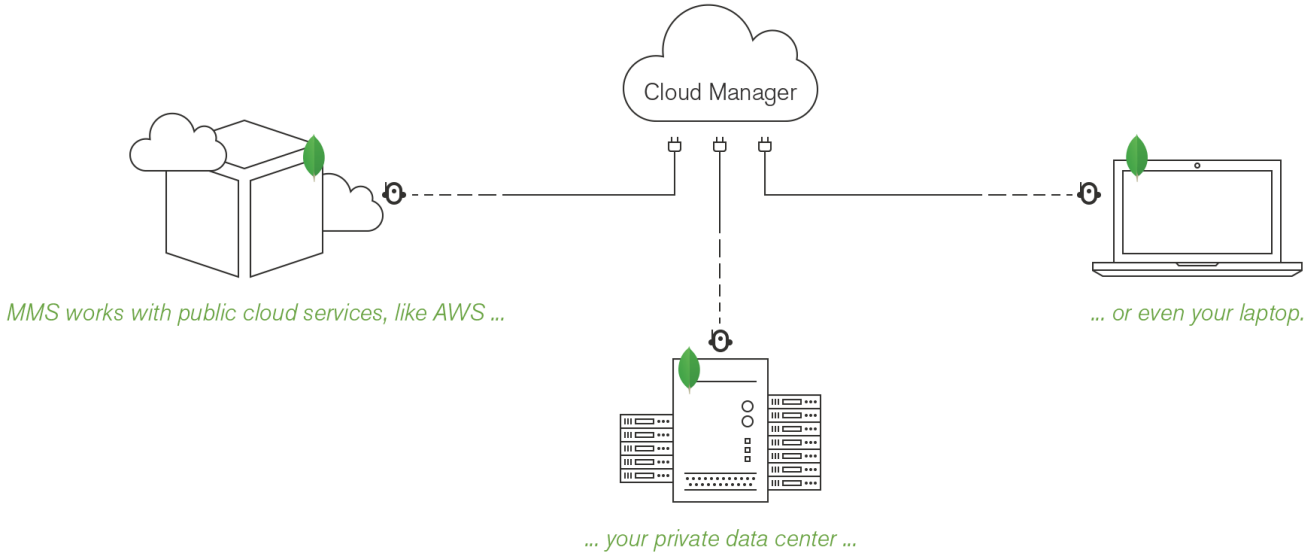
- Data Value Pyramid (Libro agile data code 2. p. 86 Figure II.1.)

MongoDB Cloud Manager

MongoDB Cloud Manager

- Servicio online que permite gestionar un despliegue de Mongodb
- Es de pago. Te da 30 días de trial gratuitos

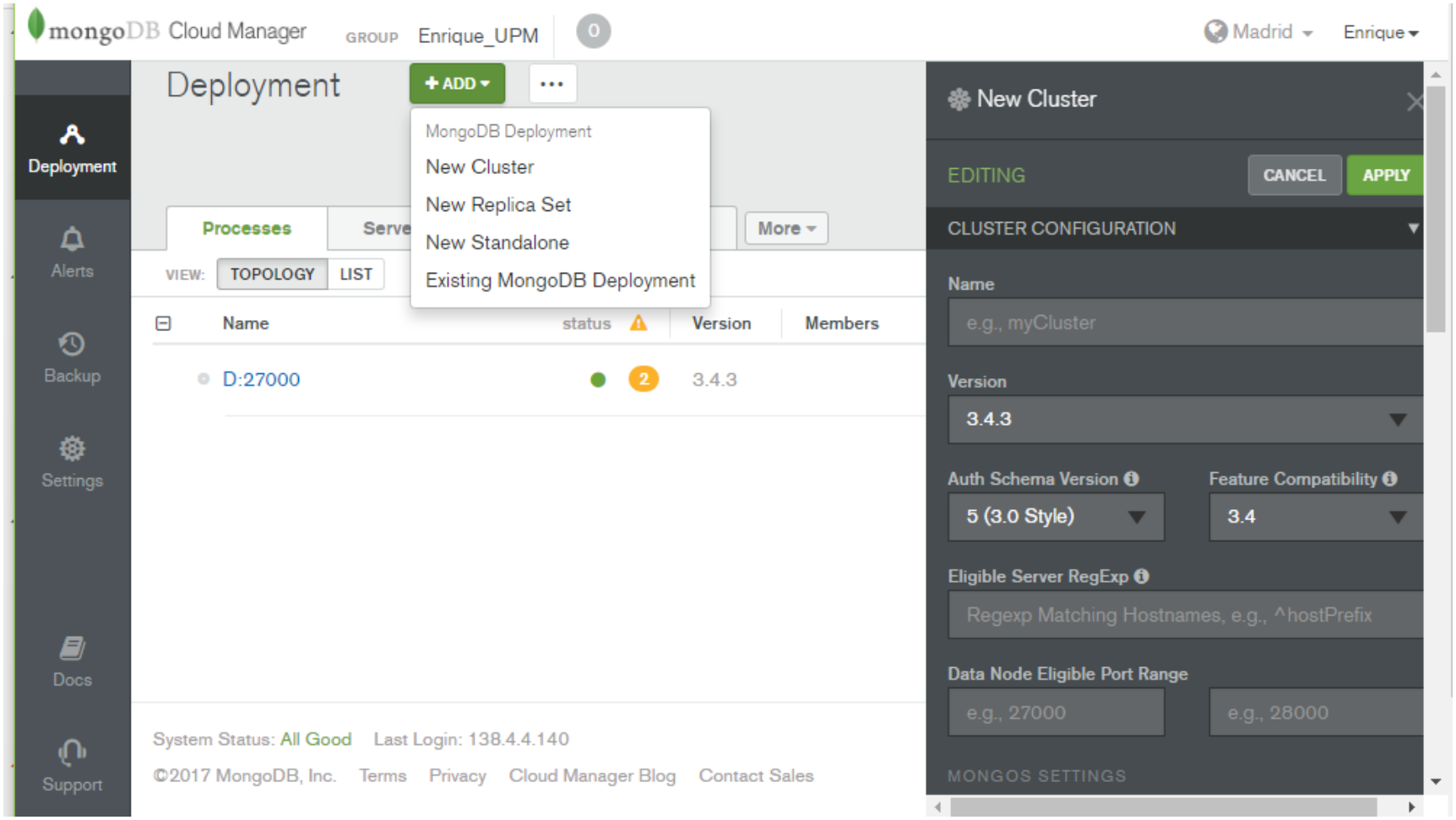
- Se pueden hacer despliegues locales, remotos o en el cloud de Amazon o Microsoft (Azure)
- Se pueden gestionar despliegues existentes

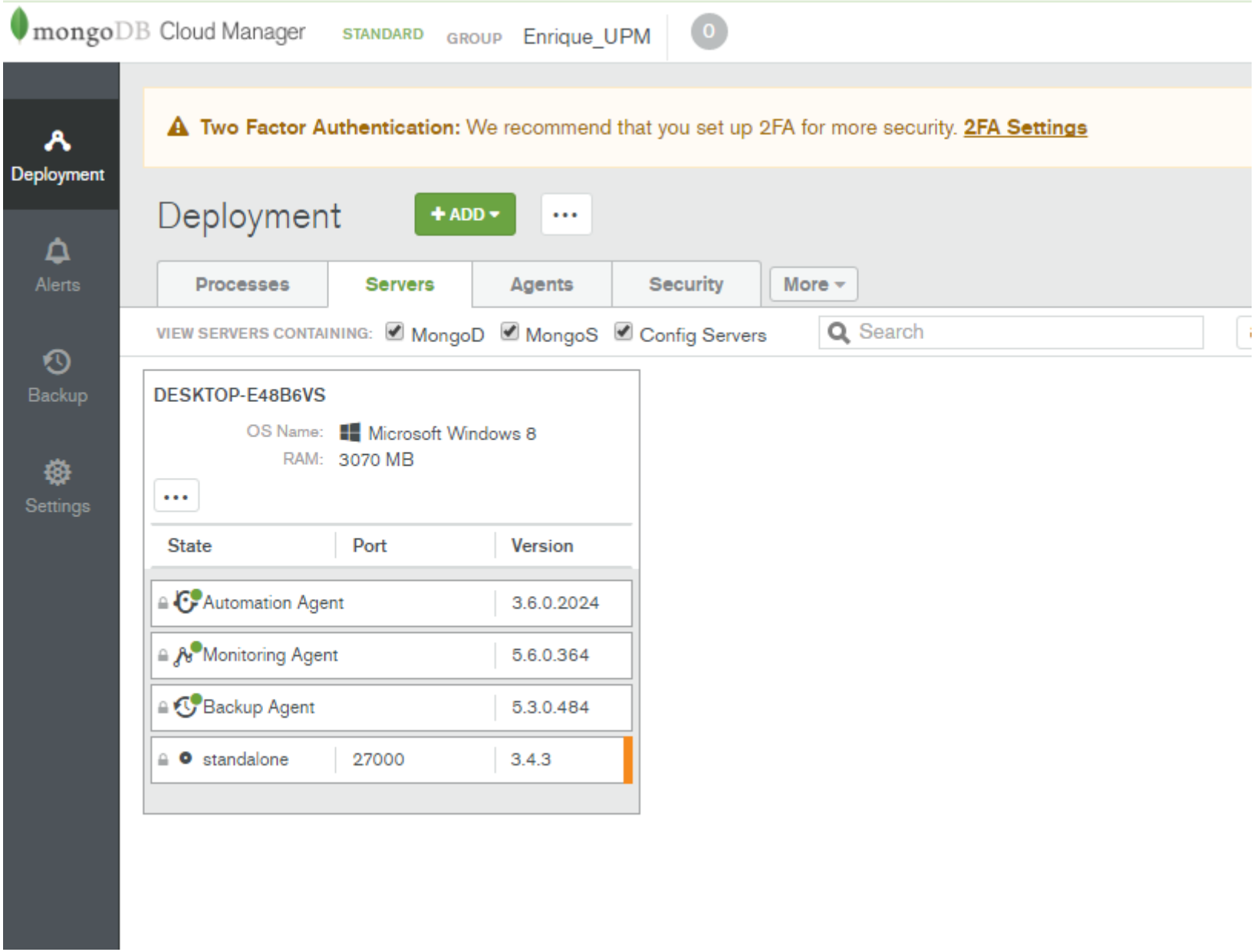


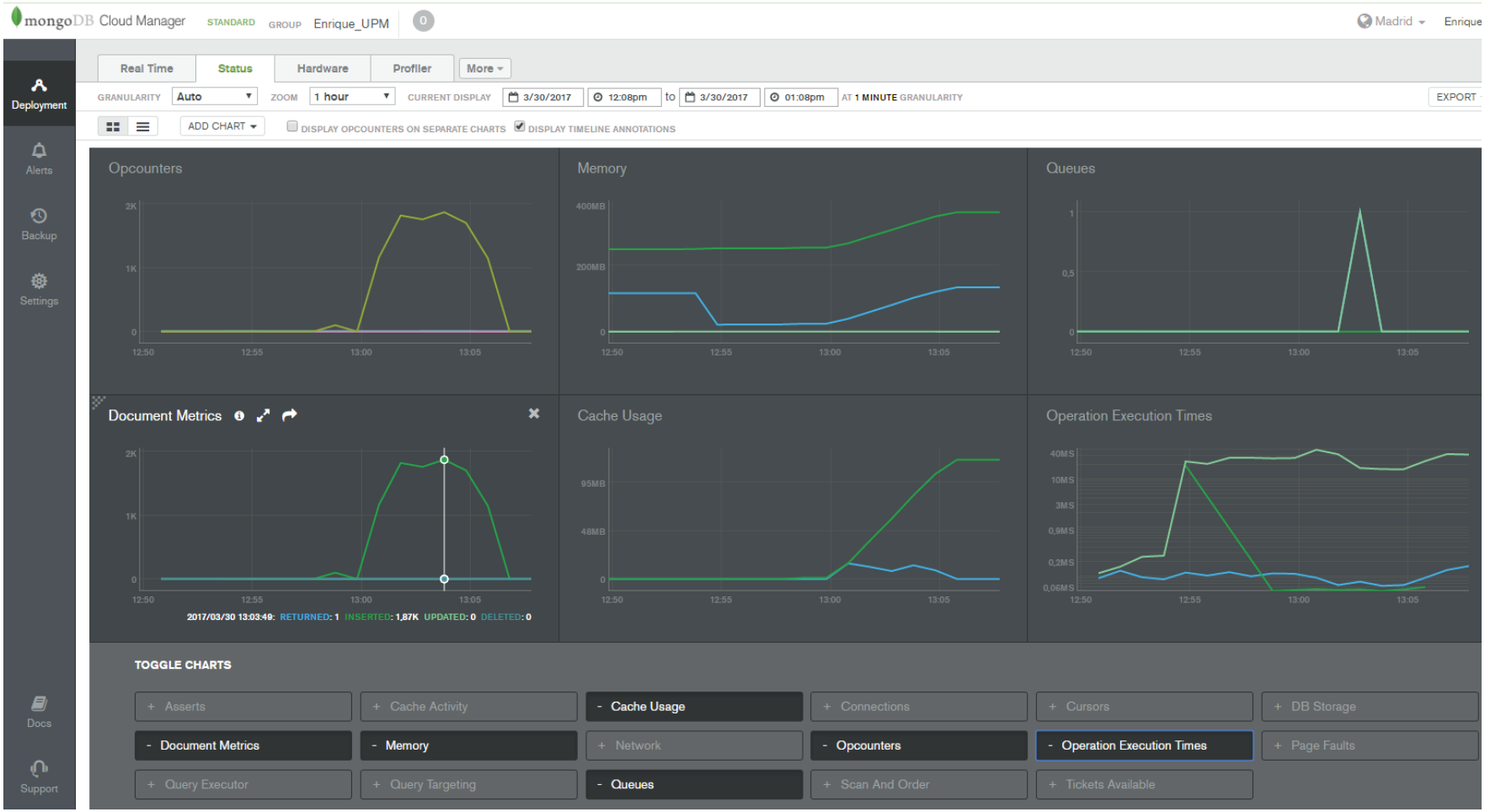
Fuente: https://docs.cloud.mongodb.com/application/

Funcionalidades

- Monitoriza todos los nodos (mongod, mongos, …)
- Reportes en tiempo real del estado de la BBDD y el hardware de los nodos
- Alertas
- Backups y restore


mongoDB Cloud Manager
GROUP Enrique_UPM
0
Madrid
Enrique
Deployment
+ ADD
MongoDB Deployment
New Cluster
New Replica Set
New Standalone
Existing MongoDB Deployment
Processes
More
VIEW:
TOPOLOGY
LIST
Name
status
Version
Members
D:27000
2
3.4.3
Deployment
Alerts
Backup
Settings
Docs
Support
System Status: All Good
Last Login: 138.4.4.140
©2017 MongoDB, Inc.
Terms
Privacy
Cloud Manager Blog
Contact Sales
New Cluster
EDITING
CANCEL
APPLY
CLUSTER CONFIGURATION
Name
e.g., myCluster
Version
3.4.3
Auth Schema Version
5 (3.0 Style)
Feature Compatibility
3.4
Eligible Server RegExp
Regexp Matching Hostnames, e.g., ^hostPrefix
Data Node Eligible Port Range
e.g., 27000
e.g., 28000
MONGOS SETTINGS

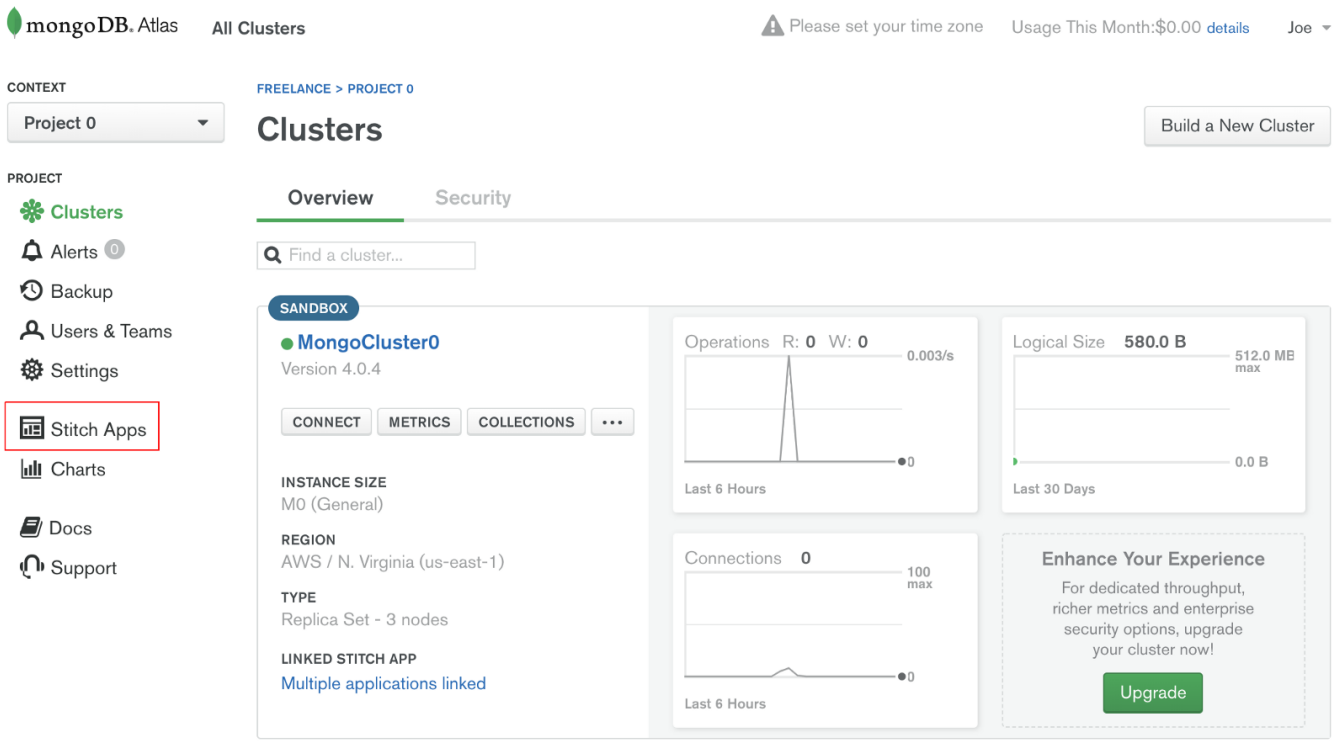

mongoDB Cloud Manager
STANDARD
GROUP
Enrique_UPM
0
Deployment
Alerts
Backup
Settings
Two Factor Authentication: We recommend that you set up 2FA for more security. 2FA Settings
Deployment
+ ADD
Processes
Servers
Agents
Security
More
VIEW SERVERS CONTAINING:
MongoD
MongoS
Config Servers
Search
DESKTOP-E48B6VS
OS Name: Microsoft Windows 8
RAM: 3070 MB
State
Port
Version
Automation Agent
3.6.0.2024
Monitoring Agent
5.6.0.364
Backup Agent
5.3.0.484
standalone
27000
3.4.3


mongoDB Cloud Manager
STANDARD
GROUP
Enrique_UPM
Madrid
Enrique
Deployment
Alerts
Backup
Settings
Docs
Support
Real Time
Status
Hardware
Profiler
More
GRANULARITY
Auto
ZOOM
1 hour
CURRENT DISPLAY
3/30/2017
12:08pm
to
3/30/2017
01:08pm
AT 1 MINUTE GRANULARITY
EXPORT
ADD CHART
DISPLAY OPCOUNTERS ON SEPARATE CHARTS
DISPLAY TIMELINE ANNOTATIONS
Opcounters
Memory
Queues
Document Metrics
2017/03/30 13:03:49: RETURNED: 1 INSERTED: 1,87K UPDATED: 0 DELETED: 0
Cache Usage
Operation Execution Times
TOGGLE CHARTS
+ Asserts
+ Cache Activity
- Cache Usage
+ Connections
+ Cursors
+ DB Storage
- Document Metrics
- Memory
+ Network
- Opcounters
- Operation Execution Times
+ Page Faults
+ Query Executor
+ Query Targeting
- Queues
+ Scan And Order
+ Tickets Available

MongoDB Atlas

MongoDB Atlas

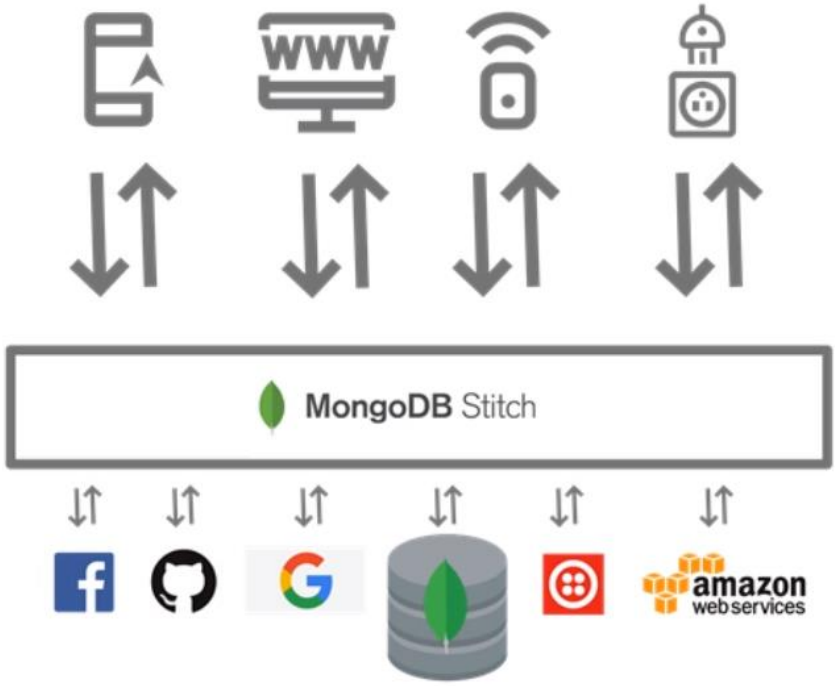
- https://www.mongodb.com/cloud/atlas
- Gestión del cloud de MongoDB con mucha atomatización y alertas
- Se diferencia del cloud manager en que nos da ya todo automatizado en el cloud. Es decir no es nuestro cloud sino que está alojado en el de MongoDB INC
- **De pago**



MongoDB Stitch

MongoDB Stitch

- https://www.mongodb.com/cloud/stitch
- The serverless platform from MongoDB (on top of MongoDB Atlas)



MongoDB Stitch

Plataforma sin servidor MongoDB Stitch: servicios

Stitch QueryAnywhere

Permite utilizar el lenguaje de consultas expresivo de MongoDB fuera del entorno local y con seguridad

Entornos iOS, Android, web e IdC

Funciones de Stitch

Integración de microservicios + lógica del servidor + servicios en la nube

Creación de aplicaciones completas o datos como servicio a través de API personalizadas

Stitch Triggers

Con las notificaciones en tiempo real, las funciones de las aplicaciones pueden reaccionar en cuanto se producen cambios en las bases de datos.

Stitch Mobile Sync

Sincronización automática de datos entre documentos almacenados de forma local en MongoDB Mobile y su base datos del backend

Optimiza el desarrollo de aplicaciones facilitando el acceso seguro a los datos y servicios del cliente, lo que evita la escritura de miles de líneas de código y la gestión de infraestructuras. Acelera el lanzamiento de aplicaciones al mercado y reduce los costes operativos.

MongoDB Stitch - ahorros

Sin Stitch	Con Stitch
Aprovisionar el servidor de backend	✓ No es necesario
Instalar el entorno de ejecución del backend	✓ No es necesario
Escribir código para la autenticación de usuarios	{Configuración JSON simple}
Escribir código para los controles de acceso de datos	{Configuración JSON simple}
Escribir código sobre cada API de servicio externo	Escribir código sobre un único SDK/API
Realizar un sondeo de la base de datos para detectar cambios	✓ No es necesario
Escribir código para la API REST para que el frontend pueda usar el backend	✓ No es necesario
Añadir código para que el backend tenga alta disponibilidad	✓ No es necesario
Añadir código para escalar el backend	✓ No es necesario
Escribir código para la lógica de aplicaciones del backend	Proporcionar código para las funciones de Stitch
Escribir código para el frontend de aplicaciones	Escribir código para el frontend de aplicaciones mediante un único SDK
Supervisar y gestionar la infraestructura del backend	✓ No es necesario

MongoDB Stitch

- NPM package
- Functionalities:
  - **Stitch QueryAnywhere**: directly from your web and mobile application frontend code
  - **Stitch Functions**: Allows developers to run simple JavaScript functions
  - **Stitch Triggers**: Real-time notifications that launch functions in response to changes in the database
- Examples:
  - Submit an expense claim to the database, and send a text notification to the approver – all with a single method call from the client application
  - Send a text message or email to notify a customer that their balance has fallen below their threshold

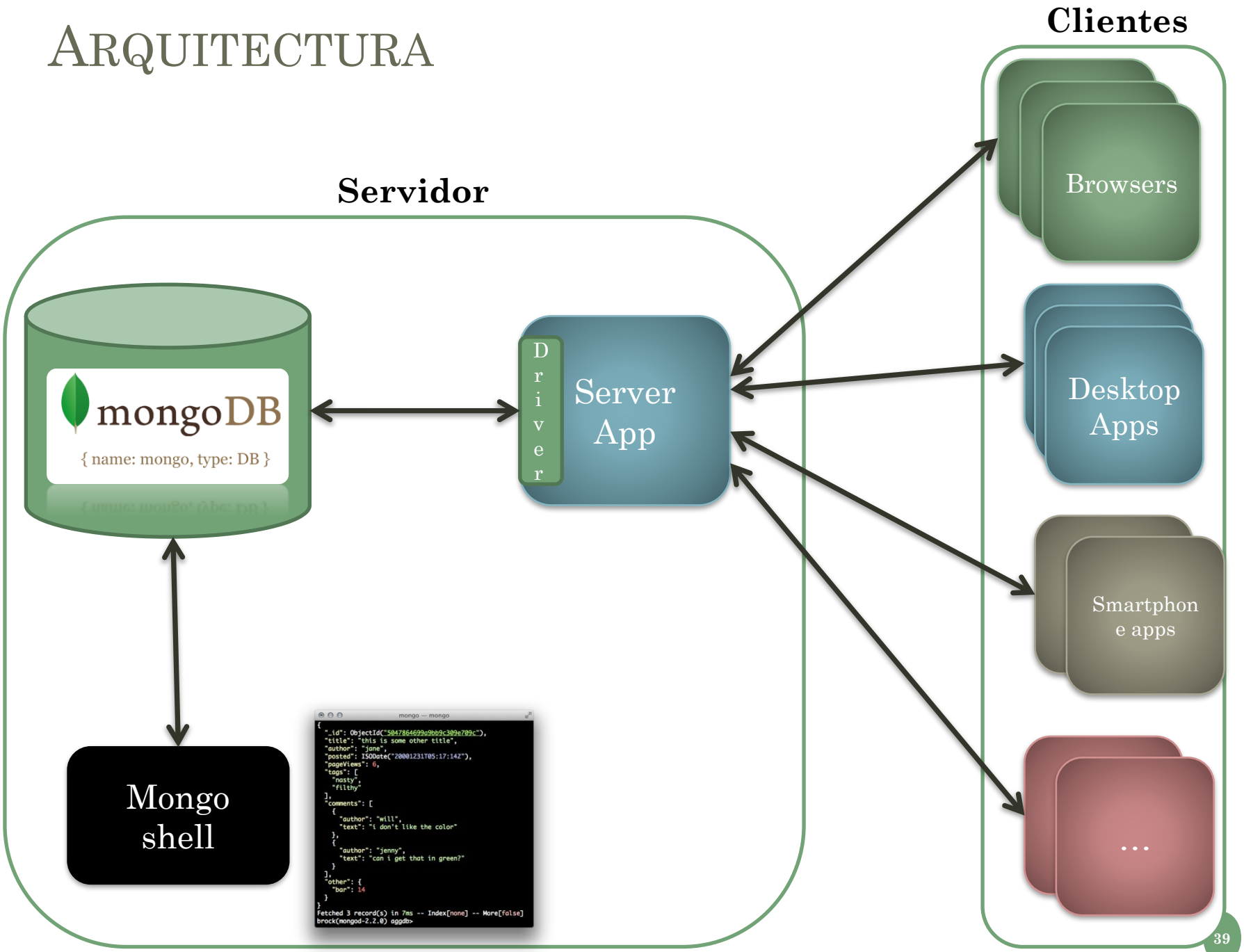
MongoDB Realm

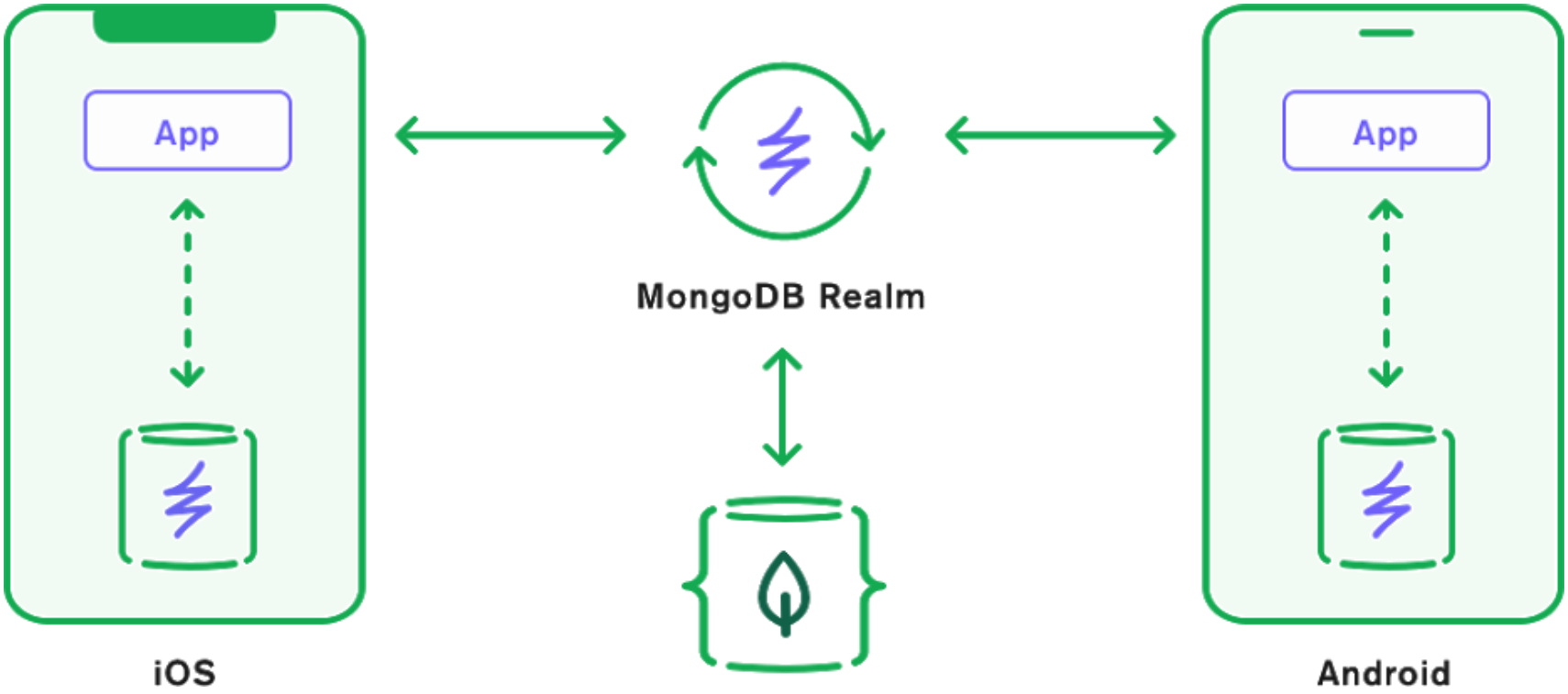
MongoDB Realm

- MongoDB compró una base de datos open source llamada **Realm (https://realm.io/)** orientada a BBDD en móviles
- No es una base de datos documental, sino basada en objetos que soporte multiples tipos de esquemas (entre ellos documentos)
- Noticia:
  - https://www.mongodb.com/press/mongodb-strengthens-mobile-offerings-with-acquisition-of-realm
- Presentación:
  - https://www.youtube.com/watch?v=WEL28rrG3DQ

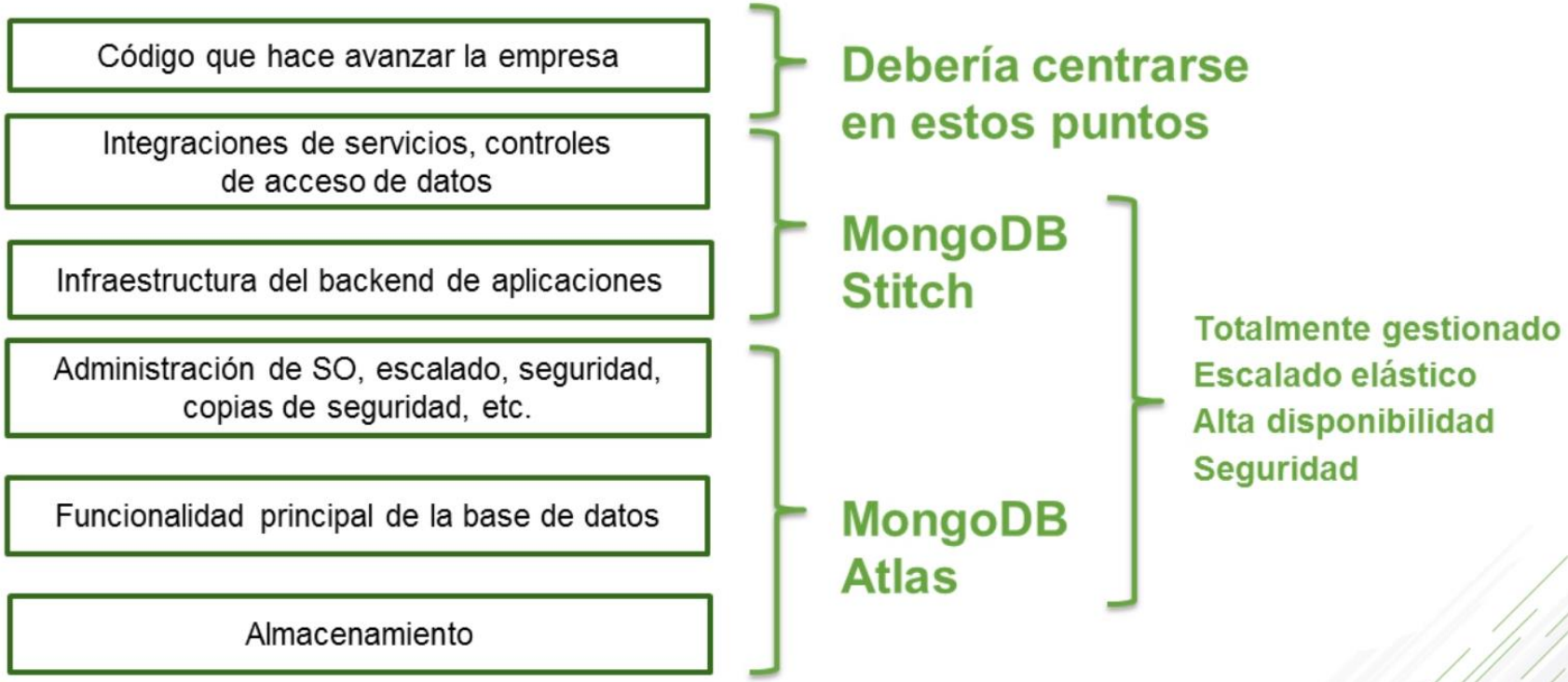
MongoDB Atlas + Stitch + Realm

Arquitectura




App
App
MongoDB Realm
iOS
Android

Céntrese en aquellos aspectos que le ayuden a marcar la diferencia



mongoDB