



{codemotion}
MADRID · NOV 27-28 · 2015



MongoDB Avanzado

Víctor Cuervo



\$db.getuser("victorcuervo")



```
{  
  name: "V́ctor Cuervo",  
  profession: "IT Architect. Santander Group",  
  socialinfo: {  
    twitter: "@victor_cuervo",  
    website: "http://lineadecodigo.com",  
    linkedin: "https://es.linkedin.com/in/victorcuervo",  
    email: "vcuervo@gmail.com"  
  },  
  hobbies: ["winetasting", "programming"],  
  programming_languages: ["java", "mongodb", "javascript", "python"]  
}
```

Recursos de la presentación



http://twitter.com/victor_cuervo



<https://github.com/victorcuervo/mongodb-codemotion>



<http://www.slideshare.net/victorcuervo/>

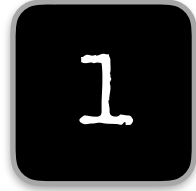


Menos “likes”. Más cervezas.

Índice



- MongoDB
- Nuevo en MongoDB 3.2
- Consultas Avanzadas
- Índices Avanzados
- Modelado en MongoDB
- DBaaS y PaaS
- Tooling con MongoDB



Una base de datos NOSQL llamada MongoDB

- ¿Todavía no sabes qué es MongoDB?
- 10 comandos sobre MongoDB
- Gartner

`{twitter:"@victor_cuervo"}`

¿Todavía no sabes qué es MongoDB?



- **Base de datos NOSQL** orientada a documentos (BSON-JSON).
- Trabajo con **modelos de datos flexibles**
- Proporciona **alto rendimiento y alta escalabilidad**.
- Ofrece múltiples APIs sobre **diferentes lenguajes de programación**: java, python, shell, nodejs, go,...
- Maneja bases de datos, colecciones, campos, índices,...**y no permite hacer un xxxxxx join.**



10 comandos MongoDB

```
db.ciudades.find({ciudad:"Madrid"})
db.ciudades.find({habitantes:{$lt:60000}})
db.ciudades.find().sort({ciudad:1}).limit(10)
db.ciudades.find({ciudad: {$in:["Avila","Madrid"]}})
db.ciudades.insert({"ciudad":"Avila","habitantes":58915})
db.ciudades.update({"ciudad":"Avila"},{$set:{habitantes:58915}})
db.ciudades.update({"ciudad":"Avila"},{$set:{habitantes:58915}},
{upsert:true})
db.ciudades.delete({ciudad:"Zamora"})
db.ciudades.update({ciudad:"Avila", {$push : {monumentos: {$each:
["Murallas", "San Vicente", "La Santa"]}}})
db.ciudades.createIndex({ciudad:"1"})
```


Cuadrante Mágico Gartner



Cuadrante sobre **bases de datos operacionales**.
MongoDB dentro del cuadrante de **challengers**.

(*) <https://www.mongodb.com/blog/post/gartner-positions-mongodb-challenger-magic-quadrant-operational-database-management>



Si te has quedado con dudas...

- [Manual MongoDB](https://docs.mongodb.org/manual/)

<https://docs.mongodb.org/manual/>

- [NOSQL: Primeros pasos en MongoDB](http://www.slideshare.net/victorcuervo/nosql-primeros-pasos-en-mongodb-codemotion)

<http://www.slideshare.net/victorcuervo/nosql-primeros-pasos-en-mongodb-codemotion>

- [Ejemplos de código en MongoDB](http://lineadecodigo.com/categoria/mongodb/)

<http://lineadecodigo.com/categoria/mongodb/>

Antes de empezar... Colección Ejemplo



```
{
  "_id" : ObjectId("565914732c23d80f730a1f49"),
  "gender" : "M",
  "name" : "Dennis Rogers",
  "username" : "drogers1",
  "birthday" : "06/11/1957",
  "email" : "drogers1@issuu.com",
  "social" : {
    "facebook" : "drogers1",
    "twitter" : "drogers1",
    "linkedin" : "drogers1"
  },
  "description" : "Proin interdum mauris non ligula....",
  "hobbies" : [
    "tv",
    "cine",
    "viajes"
  ]
}
```



Novedades sobre MongoDB 3.2

- Nuevos engines
- Validación de documentos
- MongoDB y RealTime
- MongoDB Compass
- API de Métricas

Todo el detalle de MongoDB 3.2 en <https://www.mongodb.com/mongodb-3.2?jmp=docs>

{twitter:"@victor_cuervo"}



Nuevos Engines

Se añaden nuevos engines de almacenamiento.

Las capacidades de la plataforma se pueden extender atendiendo al tipo de carga que tengamos para cada escenario:

- Almacenamiento en memoria para real-time
- Almacenamiento de datos securizados

Uso de WiredTiger como engine por defecto.

Nuevos Engines: Escenarios



Validación de documentos

En MongoDB 3.2 se puede personalizar si queremos forzar la validación del documento: estructura, tipos de datos, rangos,...

Ayuda al gobierno de los schemas y datos almacenados.





Validación de documentos

```
db.runCommand({  
  collMod: "contacts",  
  validator: {  
    $and: [  
      {year_of_birth: {$lte: 1994}},  
      {$or: [  
        {phone: { $type: "string" }},  
        {email: { $type: "string" }}  
      ]}}  
    ]}}  
  })
```

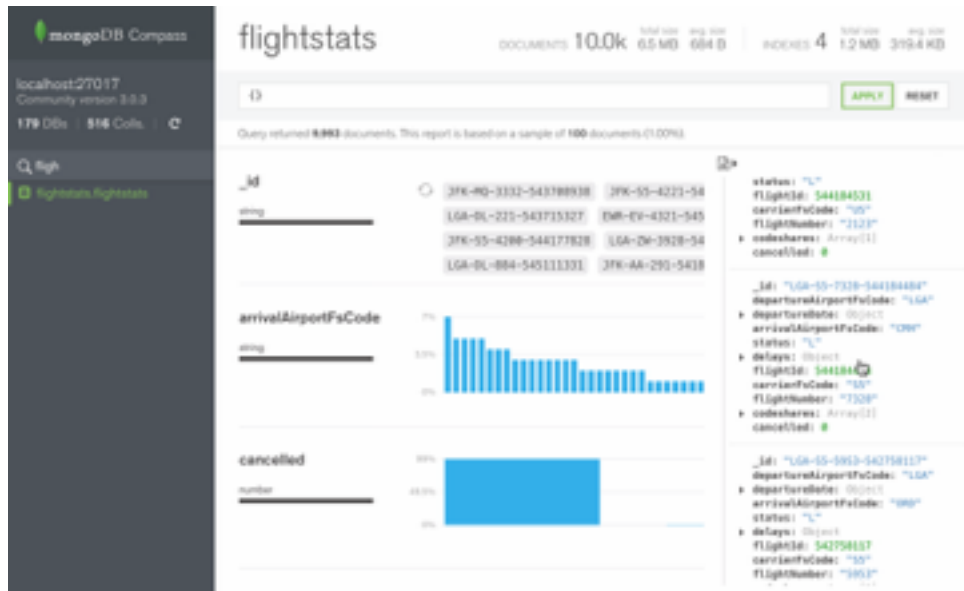

MongoDB y Real Time



“El 90% de los datos nunca son analizados y el 60% de los datos pierde el valor milisegundos después de ser generado”

El BI Connector permite integrar MongoDB con herramientas de visualización como Tableau, QlikView,...

MongoDB Compass



MongoDB Compass
es un GUI que permite
*explorar los datos de
forma visual* sin tirar
una consulta.



API de Métricas

MongoDB 3.2 ofrece un API que expone las métricas de rendimiento de las queries para que puedan ser consumidas por herramientas APM (Application Performance Monitoring).



Consultas Avanzadas en MongoDB

- Covered Queries
- Text Search
- Expresiones Regulares
- Bulking
- Capped Collection
- Two Phase Commit

`{twitter:"@victor_cuervo"}`

Covered Queries



Las covered queries son aquellas que:

- todos los campos de la query son parte del índice y
- todos los campos devueltos por una query están en el mismo índice

La búsqueda de MongoDB solo se realizará por el índice y no a través de los documentos. Como los índices están en memoria, la recuperación de información es muy rápida.



Ejemplo de Covered Query

```
{  
  "_id": ObjectId("53402597d852426020000002"),  
  "birthday": "11/18/1977",  
  "gender": "M",  
  "name": "V́ctor Cuervo",  
  "username": "victorcuervo"  
}
```

Índice

```
db.users.createIndex({gender:1,username:1})
```

#Consulta

```
db.users.find({gender:"M"},{username:1,_id:0})
```

Text Search



MongoDB soporta índices sobre elementos de texto. De esa manera podemos realizar búsquedas de cadenas de texto.

Para poder utilizar Text Search en MongoDB deberemos de:

- Activar las búsquedas de texto mediante el parámetro **searchTextEnabled**.
- Crear un índice de texto
- Realizar la consulta.



Ejemplo de Text Search

```
db.adminCommand({setParameter:true,textSearchEnabled:true})

{
  "_id": ObjectId("53402597d852426020000002"),
  "username": "raulanton",
  "description": "Apasionado de la lectura de novelas históricas...",
  "hobbies": ["lectura","novelas","cine","naturaleza"]
}

db.posts.createIndex({description:"text"})
db.posts.createIndex({description:"text"}, {default_language:spanish})

db.posts.find({$text:{$search:"novelas"}})
```




Ejemplo de Text Search

#Consulta de texto

```
db.posts.find({$text:{$search:"novelas"}})
```

Más nos vale tener el índice

```
Error: error: {  
  "waitedMS" : NumberLong(0),  
  "ok" : 0,  
  "errmsg" : "text index required for $text query",  
  "code" : 27  
}
```



Expresiones Regulares

Podemos realizar búsquedas por patrones mediante expresiones regulares.

MongoDB nos proporciona el operador **\$regex**

Se pueden indicar parámetro de las búsquedas, por ejemplo si queremos que sean “case sensitive”.



Ejemplo de Expresiones Regulares

```
{  
  "_id": ObjectId("53402597d852426020000002"),  
  "username": "raulanton",  
  "name": "Raúl Antón",  
  "description": "Apasionado de la lectura de novelas históricas...",  
  "hobbies": ["lectura", "novelas", "cine", "naturaleza"]  
}
```

```
db.posts.find({name: {$regex: "Ra"}})  
db.posts.find({name: /^R/})  
db.posts.find({contenido: {$regex: "ra", $options: "$i"}})
```



Bulking

MongoDB permite **ejecutar operaciones en batch**. De esta manera nos devuelve un documento resultado de la ejecución de todas las operaciones.

De forma ordenada o desordenada:

- **initializeOrderedBulkOp()**

- **initializeUnOrderedBulkOp()**



Ejemplo de bulking

```
bulk = db.products.initializeOrderedBulkOp()  
bulk.insert({name:"A",items:10})  
bulk.insert({name:"B",items:15})  
bulk.find({name:"A"}).update({ $inc: { items : 1 } })  
bulk.find({name:"B"}).removeOne()  
bulk.execute()
```



Capped Collection

Son colecciones de tamaño fijo que soportan un alto throughput de operaciones. Mantienen el orden de inserción, por lo cual no necesitan un índice.

Funcionan como un buffer circular, de tal manera que cuando se llena se eliminan los documentos más antiguos.



Ejemplo de Capped Collection

Crear una Capped Collection

```
db.createCollection( "log", { capped: true, size: 100000 } )
```

Validar si es una Capped Collection

```
db.log.isCapped()
```

#Recuperar los documento en orden inverso

```
db.log.find().sort( { $natural: -1 } )
```



Two Phase Commit

Las operaciones en MongoDB son atómicas sobre un solo documento. La capacidad de tener documentos anidados junto con la atomicidad sería un enfoque para los antiguos modelos “two phase commit”.

En el caso de que haya que tratar varios documentos de forma transaccional hay que implementar dicha transaccionalidad en la aplicación.



Two Phase Commit Pattern

Consideramos el escenario en el que vamos a **transferir desde la cuenta A a la cuenta B una cantidad de dinero.**

Partimos de dos colecciones:

- cuentas - accounts
- transacciones - transactions



Two Phase Commit Pattern

1. Inicializamos las cuentas

```
db.accounts.insert(  
  [  
    { _id: "A", balance: 1000, pendingTransactions: [] },  
    { _id: "B", balance: 1000, pendingTransactions: [] }  
  ]  
)
```



Two Phase Commit Pattern

2. Creamos una transacción

```
db.transactions.insert(  
  { _id: 1,  
    source: "A",  
    destination: "B",  
    value: 100,  
    state: "initial",  
    lastModified: new Date()  
  })
```



Two Phase Commit Pattern

3. Buscamos una transacción para ejecutar

```
var t = db.transactions.findOne( { state: "initial" } )
```

```
db.transactions.update(  
  { _id: t._id, state: "initial" },  
  {  
    $set: { state: "pending" },  
    $currentDate: { lastModified: true }  
  }  
)
```

Two Phase Commit Pattern



4. Movimiento en cuenta y transacciones Pendientes

```
db.accounts.update(  
  { _id: t.source, pendingTransactions: { $ne: t._id } },  
  { $inc: { balance: -t.value }, $push: { pendingTransactions: t._id  
} }  
)
```

```
db.accounts.update(  
  { _id: t.destination, pendingTransactions: { $ne: t._id } },  
  { $inc: { balance: t.value }, $push: { pendingTransactions:  
t._id } }  
)
```



Two Phase Commit Pattern

5. Transacciones Aplicadas

```
db.transactions.update(  
  { _id: t._id, state: "pending" },  
  {  
    $set: { state: "applied" },  
    $currentDate: { lastModified: true }  
  }  
)
```



Two Phase Commit Pattern

6. Eliminamos las pendientes de las cuentas

```
db.accounts.update(  
  { _id: t.source, pendingTransactions: t._id },  
  { $pull: { pendingTransactions: t._id } }  
)
```

```
db.accounts.update(  
  { _id: t.destination, pendingTransactions: t._id },  
  { $pull: { pendingTransactions: t._id } }  
)
```



Two Phase Commit Pattern

7. Transacción ejecutada

```
db.transactions.update(  
  { _id: t._id, state: "applied" },  
  {  
    $set: { state: "done" },  
    $currentDate: { lastModified: true }  
  }  
)
```




Y luego me dices que el Cobol no mola...



4 Índices Avanzados en MongoDB

- Índices multiclave
- Índices sobre textos
- Índices TTL

`{twitter:"@victor_cuervo"}`

Índices Multiclave



Son los índices que se aplican cuando el campo es un array.
Se crea una clave de índice por cada uno de los elementos que componen el array.

No se pueden crear índices multiclave compuestos y tampoco se puede especificar un índice multiclave como sharing key.

```
db.coll.createIndex( { <field>: < 1 or -1 > } )
```

Índices sobre textos



MongoDB cuenta con soporte para índices de texto para facilitar búsquedas de contenido. Soporta “stopwords” para ciertos idiomas.

```
db.users.createIndex( { description: "text" } )
```

```
# Para buscar por todos los campos
```

```
db.users.createIndex( { "$**": "text" } )
```

Índices TTL



MongoDB permite que haya índices que tengan un tiempo de vida. Pasado ese tiempo de vida, expiran.

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



5

Modelado en MongoDB

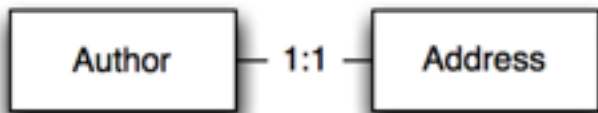
- One-To-One (1:1)
- One-To-Many (1:N)
- Many-To-Many (M:N)

```
{twitter:"@victor_cuervo"}
```



One-To-One (1:1)

Las relaciones 1:1 pueden ser modeladas de dos formas en MongoDB: insertar la relación en un único documento o bien tener un link al otro documento.



```
{  
  name: "V́ctor Cuervo",  
  age: 38  
}  
  
{  
  street: "Alcala 45",  
  city: "Madrid"  
}
```

One-To-One (1:1)



Embedding

```
{  
  name: "V́ctor Cuervo",  
  age: 38,  
  address: {  
    street: "Alcala, 45",  
    city: "Madrid"  
  }  
}
```

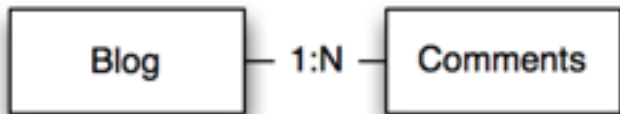
Linking

```
{  
  _id: 1,  
  name: "V́ctor Cuervo",  
  age: 38  
}  
  
{  
  user_id: 1,  
  street: "Alcala, 45",  
  city: "Madrid"  
}
```




One-To-Many (1:N)

Las relaciones 1:N pueden ser modeladas de tres formas en MongoDB: insertar la relación en un único documento, link al otro documento o bucketing.



One-To-Many (1:N)

Blog

```
{  
  title: "Línea de Código",  
  url: "http://lineadecodigo.com",  
  text: "Aprende a Programar"  
}
```

Comentarios

```
{  
  blog_entry_id: 1,  
  name: "Daniel Hernandez",  
  created_on: ISODate("2014-01-01T10:01:22Z"),  
  comment: "Me gusta tu blog"  
}  
  
{  
  blog_entry_id: 1,  
  name: "Fran Honrrubia",  
  created_on: ISODate("2014-01-01T11:01:22Z"),  
  comment: "Gran trabajo"  
}
```

One-To-Many (1:N). Embedding



```
{
  title: "Línea de Código",
  url: "http://lineadecodigo.com",
  text: "Aprende a Programar",
  comments: [{
    name: "Daniel Hernandez",
    created_on: ISODate("2014-01-01T10:01:22Z"),
    comment: "Me gusta tu blog"
  }, {
    name: "Fran Honrubia",
    created_on: ISODate("2014-01-01T11:01:22Z"),
    comment: "Gran trabajo"
  }]
}
```

El contenido insertado se crea dentro de un array.

Hay que tener cuidado con el tamaño del array y no sobrepasar los 16Mb.

A MongoDB le cuesta calcular el padding. Problemas de performance.

One-To-Many (1:N). Linking



```
{
  blog_entry_id: 1,
  name: "Daniel Hernandez",
  created_on: ISODate("2014-01-01T10:01:22Z"),
  comment: "Me gusta tu blog"
}

{
  blog_entry_id: 1,
  name: "Fran Honrubia",
  created_on: ISODate("2014-01-01T11:01:22Z"),
  comment: "Gran trabajo"
}
```

Hay que hacer tantas lecturas a la base de datos como documentos tengamos enlazados.

One-To-Many (1:N). Bucketing



```
{
  blog_entry_id: 1,
  page: 1,
  count: 50,
  comments: [{
    name: "Daniel Hernandez",
    created_on: ISODate("2014-01-01T10:01:22Z"),
    comment: "Me gusta tu blog"
  }, ...]
}
{
  blog_entry_id: 1,
  page: 2,
  count: 1,
  comments: [{
    name: "Fran Honrubia",
    created_on: ISODate("2014-01-01T11:01:22Z"),
    comment: "Gran trabajo"
  }]
}
{codeemotion}
```

Es una mezcla entre embedding y linking. Se divide en contenedores con un tamaño y se inserta el comentario en el contenedor que toca.

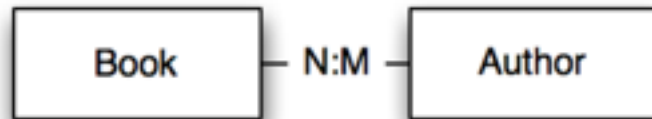
Tiene que tener algún concepto de división como fechas,....



Many-To-Many (N:N)

Las relaciones N:N se modelan mediante técnicas de embedding.

En el caso de que haya una de las partes predominantes se modelaría hacía un solo sentido.





Many-To-Many (N:N). Two Ways

```
{
  _id: 1,
  name: "Peter Standford",
  books: [1, 2]
}

{
  _id: 2,
  name: "Georg Peterson",
  books: [2]
}
```

```
{
  _id: 1,
  title: "A tale of two people",
  categories: ["drama"],
  authors: [1, 2]
}

{
  _id: 2,
  title: "A tale of two space ships",
  categories: ["scifi"],
  authors: [1]
}
```

Many-To-Many (N:N). One Way



```
{  
  _id: 1,  
  name: "drama"  
}
```

```
{  
  _id: 1,  
  title: "A tale of two people",  
  categories: [1],  
  authors: [1, 2]  
}
```




DBaaS y PaaS: MongoDB en Cloud



- MongoLab
- MongoDB y Docker
- MongoDB en PaaS

`{twitter:"@victor_cuervo"}`

MongoLab



MongoLab es un BDaaS que nos permite aprovisionarnos bases de datos de MongoDB de forma dinámica en cloud de una forma sencilla.

Funciona **en formato PaaS** que funciona sobre **Google, AWS y Azure.**

```
$ mongo ds047037.mongolab.com:47037/<db> -u <dbuser> -p <dbpassword>  
$ mongod --> mongod://<dbuser>:<dbpassword>@ds047037.mongolab.com:47037/<db>
```

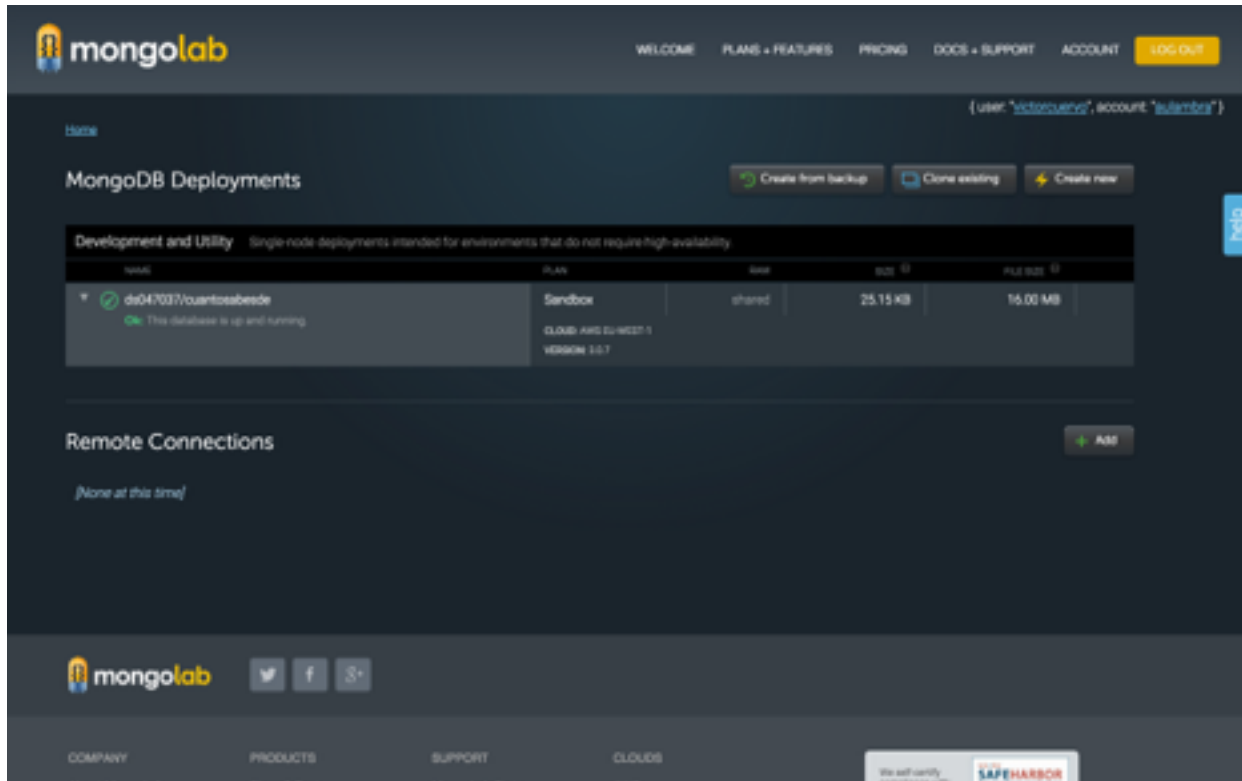
Características MongoLab



MongoLab nos permite de forma sencilla:

- Crear bases de datos dinámicamente.
- Monitorización y alertas 24x7.
- GUI para operaciones básicas.
- Análisis de índices y performance.
- Establecer seguridad en la conexión.
- Soporte para APIs de diferentes lenguajes.

MongoLab



MongoDB y Docker



MongoDB ya está “dockerizado”.

Disponible en el Docker Hub Registry (https://hub.docker.com/_/mongo/)

```
$ docker run --name some-mongo -d mongo
$ docker run -it --link some-mongo:mongo --rm mongo sh -c 'exec mongo
"$MONGO_PORT_27017_TCP_ADDR:$MONGO_PORT_27017_TCP_PORT/test"'
```

Crear una imagen Docker con MongoDB

<https://docs.docker.com/engine/examples/mongodb/>

MongoDB en PaaS



MongoDB está disponible en diferentes entornos
PaaS: BlueMix, Heroku, OpenShift,...





Herramientas de Tooling para MongoDB

- Mockaroo
- Mongo Hacker

`{twitter:"@victor_cuervo"}`

Herramientas de tooling para MongoDB



Existen múltiples herramientas de tooling alrededor de **MongoDB**: GUI, Conectores, Monitorización, Performance Tuning, Mejoras del Shell, Manejo para JSON,...

Hay un listado extenso en <http://mongodb-tools.com/>

Mockaroo




Herramienta online que nos permite crear **colecciones JSON**.

- Tipos de datos estándar: datos sobre personas, salud, monetarios, colores,
- Permite crear subdocumentos y arrays
- Trabajar con expresiones regulares
- Dispone de un API para invocarlo de forma externa.

<https://www.mockaroo.com>

Mockaroo

 realistic data generator

APIHELPCONTACTPRICINGSIGN IN

Need some mock data to test your app?

Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

[Need more data? Plans start at just \\$50/year.](#)

Field Name	Type	Options
<input type="text" value="id"/>	Row Number	to: blank: 0 N x
<input type="text" value="first_name"/>	First Name	to: blank: 0 N x
<input type="text" value="last_name"/>	Last Name	to: blank: 0 N x
<input type="text" value="email"/>	Email Address	to: blank: 0 N x
<input type="text" value="country"/>	Country	to: blank: 0 N x
<input type="text" value="ip_address"/>	IP Address v4	to: blank: 0 N x

Add another field

Rows: Format: ☒ include header

Want to save this for later? [Sign up for free.](#)

Mongo Hacker



```
mongo — mongo
{
  "_id": ObjectId("5047864699a9bb9c309e709c"),
  "title": "this is some other title",
  "author": "jane",
  "posted": ISODate("20001231T05:17:14Z"),
  "pageViews": 6,
  "tags": [
    "nasty",
    "filthy"
  ],
  "comments": [
    {
      "author": "will",
      "text": "i don't like the color"
    },
    {
      "author": "jenny",
      "text": "can i get that in green?"
    }
  ],
  "other": {
    "bar": 14
  }
}

```

Fetched 3 record(s) in 7ms -- Index[none] -- More[false]
 brock(mongod-2.2.0) aggdb>

Permite **mejorar el Shell de MongoDB**:

- Añadir colores
- Extensiones al API Shell
- Ayudas Framework Agregación

<https://tylerbrock.github.io/mongo-hacker/>



Recursos

- [Nuevo en MongoDB 3.2](#)
- [Capped collection](#)
- [Nuevo API de Bulking](#)
- [Two Phase Commit](#)
- [Modeling MongoDB Schema](#)
- [MongoLab](#)
- [Mockaroo](#)

Recursos de la presentación



http://twitter.com/victor_cuervo



<https://github.com/victorcuervo/mongodb-codemotion>



<http://www.slideshare.net/victorcuervo/>

Preguntas



