

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

NoSQL & MongoDB

Agenda

- NoSQL
 - Introduction
 - CAP Theorem, BASE
 - Types of NoSQL
- MongoDB
 - Getting started
 - Shell
 - Storage
 - Structure
 - Manipulating data
 - Querying data
 - Replication
 - Client application

NoSQL Introduction

- Concept is quite old (since 60')
- Term NoSQL used for the first time in 1998
 - By Carlo Strozzi
- The actual movement started in 2009
 - The name proposed by Eric Evans
- NoSQL stands for „Not only SQL“
- Good intros
 - <https://www.mongodb.com/nosql-explained>
 - <https://www.mongodb.com/nosql-explained/nosql-vs-sql>
 - <https://hostingdata.co.uk/nosql-database/>
 - <https://mansfeld.pl/bazy-danych/bazy-danych-nosql-zalety-wady/>

NoSQL Introduction

- Different mindset and pros
 - No (mandatory) schemas
 - Simpler and faster queries
 - Data that is accessed together should be stored together
 - Easily horizontally scalable
 - Maintainability much simpler (e.g. key-value pairs)
 - Easy replication and failover support
 - ACID in a node, BASE outside the node

NoSQL Introduction

- Cons, or where SQL is the choice
 - No or limited support for multi-record ACID trans.
 - But more DBs are adding the feature, e.g. MongoDB
 - No normalization and data redundancy
 - Sometimes very targeted use cases, e.g. Neo4j
 - SQL is multi-purpose, it is easier to cover very different use cases

```
const session = client.startSession()
await session.withTransaction(async () => {
  await collection.insertOne(doc1, { session })
  await collection.insertOne(doc2, { session })
})
session.commitTransaction()
session.endSession()
```

NoSQL Introduction

■ Comparison

	SQL Databases	NoSQL Databases
Data Storage Model	Tables with fixed rows and columns	Document: JSON documents, Key-value: key-value pairs, Wide-column: tables with rows and dynamic columns, Graph: nodes and edges
Development History	Developed in the 1970s with a focus on reducing data duplication	Developed in the late 2000s with a focus on scaling and allowing for rapid application change driven by agile and DevOps practices.
Examples	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	Document: MongoDB and CouchDB, Key-value: Redis and DynamoDB, Wide-column: Cassandra and HBase, Graph: Neo4j and Amazon Neptune
Primary Purpose	General purpose	Document: general purpose, Key-value: large amounts of data with simple lookup queries, Wide-column: large amounts of data with predictable query patterns, Graph: analyzing and traversing relationships between connected data
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Multi-Record ACID Transactions	Supported	Most do not support multi-record ACID transactions. However, some—like MongoDB—do.
Joins	Typically required	Typically not required
Data to Object Mapping	Requires ORM (object-relational mapping)	Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages.

NoSQL Introduction

- Example modeling (MongoDB)

Users

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working



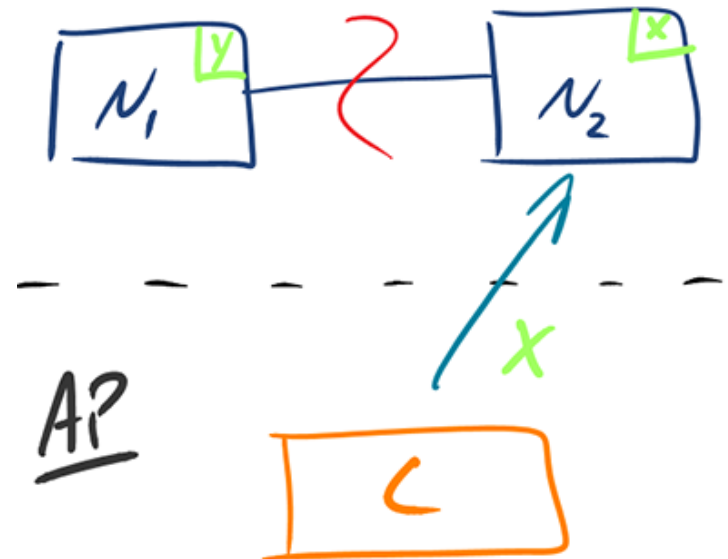
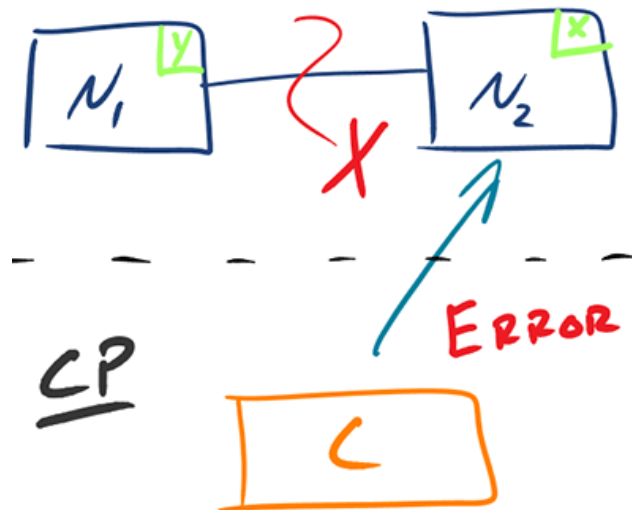
```
{
  "_id": 1,
  "first_name": "Leslie",
  "last_name": "Yepp",
  "cell": "8125552344",
  "city": "Pawnee",
  "hobbies": ["scrapbooking", "eating waffles", "working"]
}
```

Eric Brewer's CAP Theorem

- CAP stands for
 - Consistency
 - Each node offers the same, fresh data
 - Each client can see exactly the same data
 - Availability
 - Each active node is able to respond all requests within the reasonable time
 - Partition Tolerance
 - The system will continue to work even when network partition occurs
- CAP is defined for distributed systems

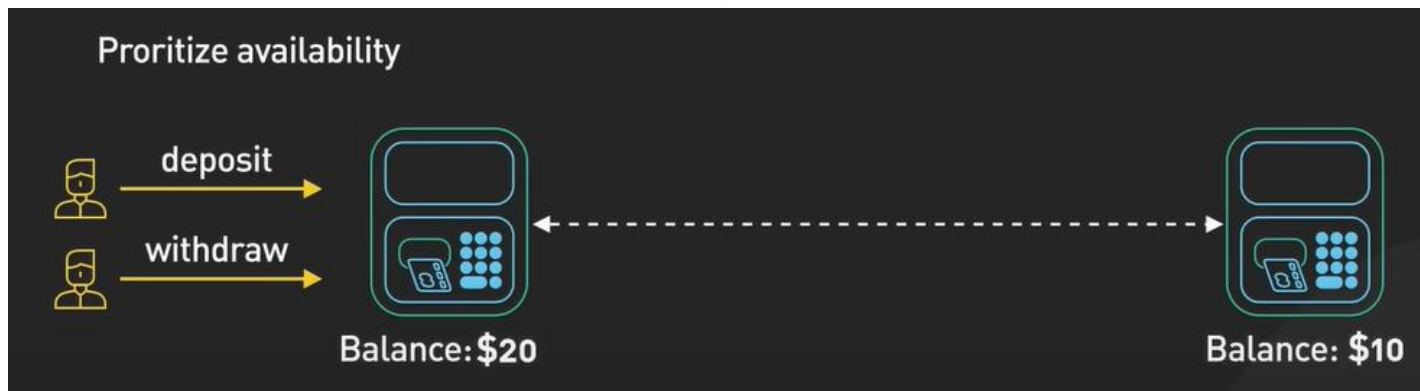
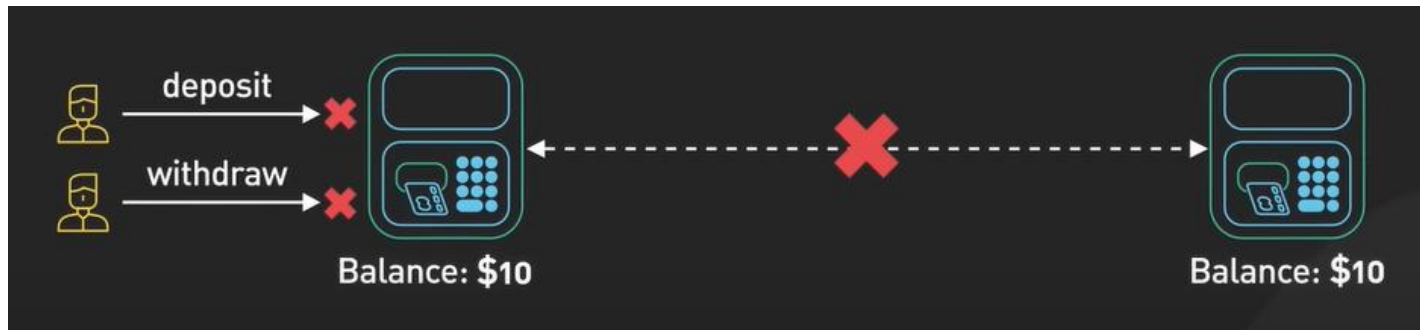
Eric Brewer's CAP Theorem

- CAP theorem states that you can't have all 3, only 2 are possible: CA, CP, or AP
- As distributed system requires P, we have



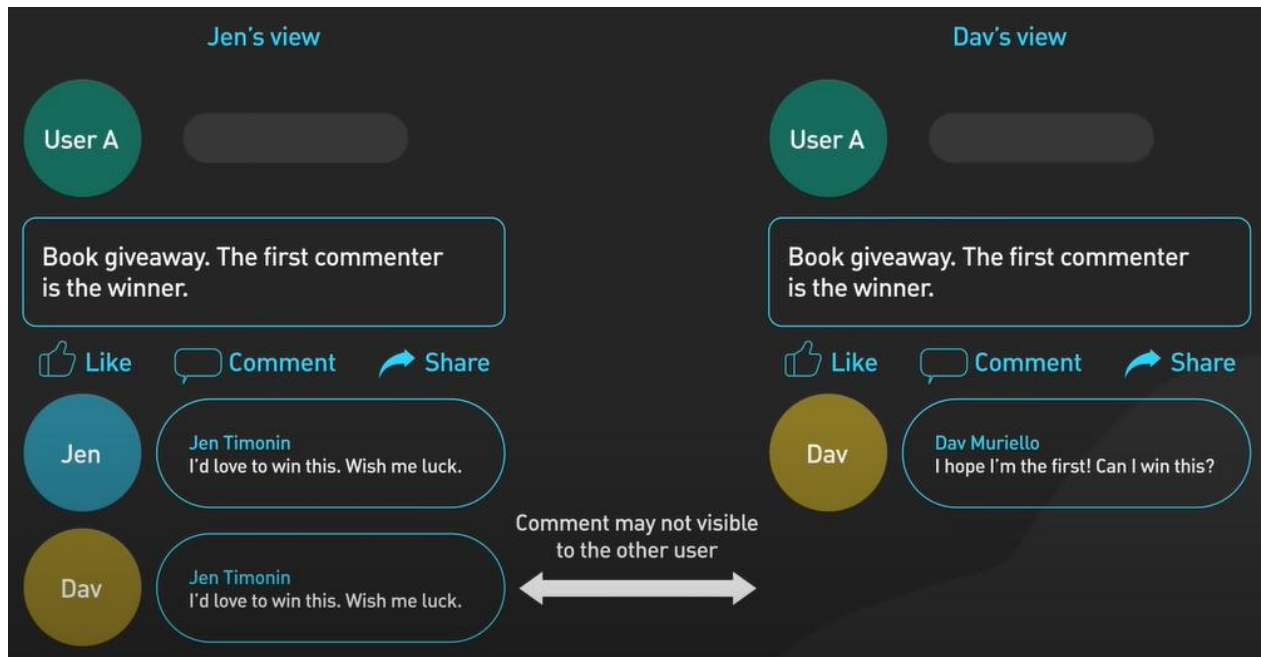
Eric Brewer's CAP Theorem

■ Example 1



Eric Brewer's CAP Theorem

- Example 2:



Eric Brewer's CAP Theorem

- In practice it is not black & white, e.g.



- Great introduction:
 - <https://www.youtube.com/watch?v=BHqjEjzAicA>
- Good further reading:
 - <https://dzone.com/articles/understanding-the-cap-theorem>
 - <https://robertgreiner.com/cap-theorem-revisited/>

Positioning 2PC transaction

- Where do we position 2 phase commit?
 - CA, CP or AP?
- It guarantees consistency, but...
 - 2PC coordinator – single point of failure
 - Locks are limiting availability
 - Any error network stops the transaction
 - Not supported by all systems

ACID vs. BASE

- Let revisit our requirements...
- ACID → BASE
 - Basic Availability
 - indicates that availability is valued more than consistency, so if only possible, the system will be available
 - Soft-state
 - indicates that the state of the system may change over time, even without input. This is because of the eventual consistency model.
 - Eventual consistency
 - indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

Types of NoSQL

- Document databases
 - MongoDB
- Key-value databases
 - Redis
- Wide-column stores
 - Cassandra
- Graph databases
 - Neo4J

More: <https://pl.wikipedia.org/wiki/NoSQL>

Types of NoSQL

- Document databases
 - Store XML or JSON documents (usually)
 - Documents can be nested
 - Usually no need for mapper
 - Nested structure is easier to map to code structure
 - Getting popular within developers
 - Use cases:
 - ecommerce platforms, trading platforms, and mobile app development across industries.
 - generally quite wide range of applications

Types of NoSQL

- Key-value databases
 - The simplest NoSQL database type
 - Structure based on key-value pairs
 - value can be from a string to a complex object
 - Use cases:
 - shopping carts, user preferences, user profiles
 - caching mechanisms,
 - configuration based on keys (e.g. windows registry)
 - More:
 - <https://www.mongodb.com/databases/key-value-database>

Types of NoSQL

- Column-oriented databases
 - Data organized in columns and rows, but
 - ... data is physically stored in column-oriented way
 - Many products offer big data processing possibility
 - Use cases
 - Efficient for analytical purposes
 - More:
 - Good Apache Cassandra introduction:
<https://www.youtube.com/watch?v=5qEoEAfAer8>
 - Good tutorial:
<https://www.youtube.com/watch?v=s1xc1HV5Rko&list=PLalrWAGybpB-L1PGA-NfFu2uiWHEsdscD&index=1>

Types of NoSQL

- Graph databases
 - Focuses on the relationship between data elements
 - May have attributes and can be more meaningful than in SQL
 - Elements stored as nodes
 - Optimized for searching for connections between data elements
 - Overcoming the overhead associated with JOINing multiple tables in SQL.
 - Usually combined with other DB types as e.g. SQL
 - It is rare that there is a single case for graph DB

Types of databases

- Comparison

Data model ⇄	Performance ⇄	Scalability ⇄	Flexibility ⇄	Complexity ⇄	Functionality ⇄
Key-value store	high	high	high	none	variable (none)
Column-oriented store	high	high	moderate	low	minimal
Document-oriented store	high	variable (high)	high	low	variable (low)
Graph database	variable	variable	high	high	graph theory
Relational database	variable	variable	low	moderate	relational algebra

Types of NoSQL

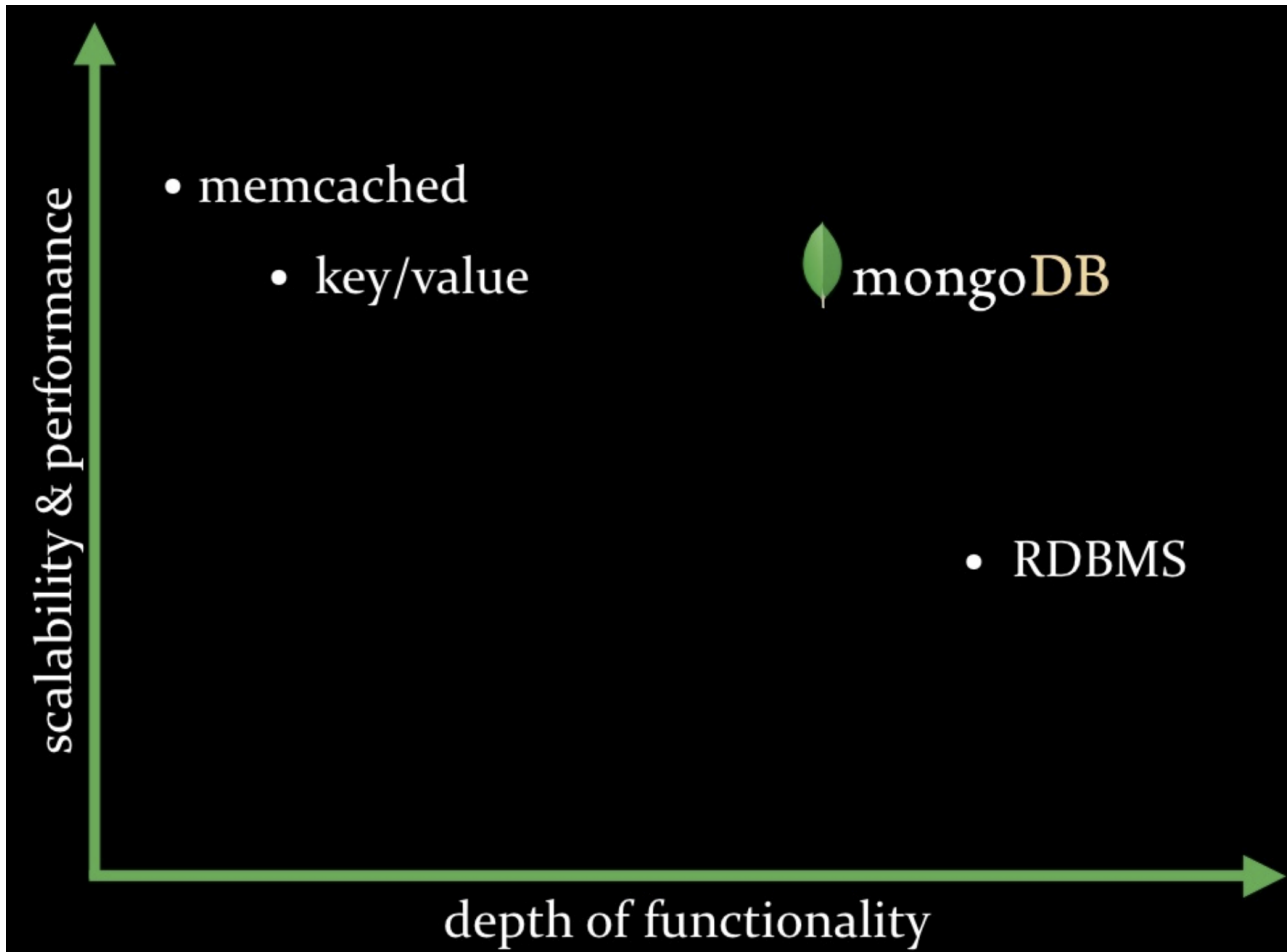
■ Microsoft Azure review

	Azure SQL Database	Azure SQL Managed Instance	Program SQL Server na maszynach wirtualnych	Azure Database for PostgreSQL	Azure Database for MySQL	Azure Database for MariaDB	Azure Cosmos DB	Azure Cache for Redis
Relacyjna baza danych	✓	✓	✓	✓	✓	✓		
Nierelacyjna baza danych (NoSQL)							✓	
Baza danych w pamięci								✓
Modele danych	Relacyjne	Relacyjne	Relacyjne	Relacyjne	Relacyjne	Relacyjne	Wiele modeli: Dokument, dane szerokokolumnowe, klucz-wartość, graf	Klucz-wartość
Hybrydowe	✓	✓	✓	✓ <i>(Hiperskalowanie)</i>				
Bezszybowe usługi obliczeniowe	✓						✓	
Skalowanie w poziomie magazynu	✓ <i>(Hiperskalowanie)</i>			✓ <i>(Hiperskalowanie)</i>			✓	✓
Skalowanie w poziomie środowiska obliczeniowego	✓ <i>(Hiperskalowanie — tylko do odczytu)</i>			✓ <i>(Hiperskalowanie)</i>			✓	✓
Rozproszone zapisy z wielowzorcowością (Zapis danych w różnych regionach)							✓	✓ <i>(Juz wkrótce)</i>
Usługa oparta na oprogramowaniu open-source (Edycja Community i obsługa otwartych rozszerzeń)				✓	✓	✓		✓
HTAP (Dostępne w usłudze Azure Synapse Link)	✓ <i>(Juz wkrótce)</i>			✓ <i>(Juz wkrótce)</i>			✓	

MongoDB

- NoSQL DB
- Open source
- Document DB
- No schema mandatory
- Indexing
- Highly scalable
- Easy Replication and sharding

MongoDB position

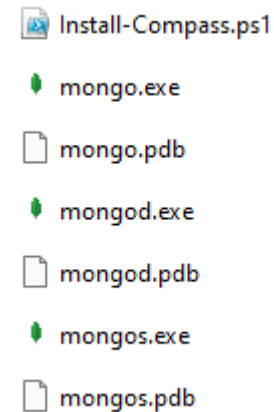


Establishing server (local)

- Web site:
 - <https://www.mongodb.com/>
- Download:
 - <https://www.mongodb.com/try/download/community>
- We can download latest stable version:
 - <https://www.mongodb.com/download-center/community/releases>

Establishing server (local)

- Files we get in archive
- Default
 - data folder: \Data\Db
 - port: 27017
- Options:
 - mongod --help
- Simplest usage:
 - mongod
 - mongo



Establishing server (local)

- Better usage:

- `mongod --dbpath c:\Data\Databases\MongoDB\db`
- `mongo`

- Best usage:

- `mongod -f c:\Data\Databases\MongoDB\mongod.conf`

- Config file:

```
1 dbpath=C:\Data\Databases\MongoDB\db
2 logpath=C:\Data\Databases\MongoDB\mongo-server.log
3 verbose=vvvv
```

Establishing server (Azure)

Home > Create a resource >

Marketplace

Get Started

Service Providers

Management

Private Marketplace

Private Offer Management

My Marketplace

Favorites

Recently created


Private products

Categories

Analytics (18)

Search: Azure Cosmos DB


Showing 1 to 20 of 36 results for 'Azure Cosmos DB'. [Clear search](#)




Azure Cosmos DB

Microsoft

Azure Service
Globally-distributed, multi-model database service.


Create ▼ 



Azure Cosmos DB Reserved Capacity

Microsoft

Azure Service
Reserved instances (RIs) for Cosmos DB significantly reduce your Cosmos DB costs compared to pay-as-you-go prices.

Create ▼ 

Create an Azure Cosmos DB account

Which API best suits your workload?

Azure Cosmos DB is a fully managed NoSQL and relational database service for building scalable, high performance applications. [Learn more](#)

To start, select the API to create a new account. The API selection cannot be changed after account creation.

Azure Cosmos DB for NoSQL

Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.

Create [Learn more](#)

Azure Cosmos DB for PostgreSQL

Fully-managed relational database service for PostgreSQL with distributed query execution, powered by the Citus open source extension. Build new apps on single or multi-node clusters—with support for JSONB, geospatial, rich indexing, and high-performance scale-out.

Create [Learn more](#)

Azure Cosmos DB for MongoDB

Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.

Create [Learn more](#)

Azure Cosmos DB for Apache Cassandra

Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.

Create [Learn more](#)

Azure Cosmos DB for Table

Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB.

Create [Learn more](#)

Azure Cosmos DB for Apache Gremlin

Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.

Create [Learn more](#)

Home > Create a resource >

Marketplace

Get Started

Service Providers

Management

Private Marketplace

Private Offer Management

My Marketplace

Favorites

Recently created

Private products


Categories

Databases (87)

Compute (53)

Search: MongoDB

Showing 1 to 20 of 131 results for 'MongoDB'. [Clear search](#)





MongoDB Atlas on Azure

MongoDB, Inc.

SaaS
The best way to run MongoDB on Azure

Starts at **Free**

Subscribe ▼ 





MongoDB Atlas on Azure for IoT

MongoDB, Inc.

SaaS
MongoDB developer data platform for IoT applications

Starts at **Free**

Subscribe ▼ 





MongoDB Atlas (pay-as-you-go)

MongoDB, Inc.

SaaS
Get started for free with Atlas, the best way to run MongoDB on Azure

Starts at **Free**

Subscribe ▼ 





MongoDB Server

Cloud Infrastructure Services

Virtual Machine
MongoDB Server 4.4 Community on Ubuntu Server 20.04. A Powerful Distributed Document Database.

Starts at **€0.025/hour**


Create ▼ 




Azure Cosmos DB API for MongoDB

Microsoft

Azure Service
Quickly setup a MongoDB compatible API database with built-in security and scalability.

Create ▼ 





MongoDB Container Virtual machine Image

Kockpit Analytics Pvt. Ltd

Virtual Machine
MongoDB Container Virtual machine Image

Starts at **€0.047/hour**

Create ▼ 




MongoDB VMI for Ubuntu 20.04.02

Kockpit Analytics Pvt. Ltd

Virtual Machine
MongoDB on Ubuntu 20.04 LTS

Starts at **€0.047/user/3 years**

Create ▼ 

Establishing server (Azure)

The screenshot displays the MongoDB Data Explorer interface. The top header shows 'mongodatabase - Data Explorer' and 'Azure Cosmos DB account'. The left sidebar contains navigation options: Overview, Dziennik aktywności, Kontrola dostępu (IAM), Tagi, Diagnostowanie i rozwiązywanie..., Quick start, and Data Explorer (highlighted). The main area is divided into three sections: a left sidebar for the MONGODB API (library, books, Documents, Scale & Settings, Stored Procedures, User Defined Functions, Triggers), a central document viewer showing a list of document IDs with one selected, and a right pane displaying the document's JSON structure.

MONGODB API

- library
 - books
 - Documents
 - Scale & Settings
 - Stored Procedures
 - User Defined Functions
 - Triggers

Mongo Docu... x

New Document Update Discard Delete

Type a query predicate (e.g. [a:"foo"]), or choose one from the drop down list, or leave emp. Apply Filter x

_id

- 5b1803cefa26ae03c0b9e36c
- 5b180419b4dfa7129c86f068

Load more

```
1 {
2   "_id" : ObjectId("5b1803cefa26ae03c0b9e36c"),
3   "title" : "Mistrz i Małgorzata",
4   "author" : "Bułhakow"
5 }
```

Establishing server (ATLAS)



MONGODB ATLAS

Choose a path. Adjust anytime.

Available as a fully managed service across 60+ regions on AWS, Azure, and Google Cloud

Dedicated Multi-Cloud & Multi-Region Clusters

For teams developing world-class applications that require multi-region resiliency or ultra-low latency.

- ✓ Includes all features from Shared and Dedicated Clusters
- ✓ Replicate data across clouds and regions
- ✓ Globally distributed read and write operations
- ✓ Control data residency at the document level

Create a cluster

Starting at
\$0.13/hr*
*estimated cost \$98.55/month

Dedicated Clusters

For teams building applications that need advanced development and production-ready environments.

- ✓ Includes all features from Shared Clusters
- ✓ Auto-scaling
- ✓ Network isolation
- ✓ Realtime performance metrics

Create a cluster

Starting at
\$0.08/hr*
*estimated cost \$56.94/month

Shared Clusters

For teams learning MongoDB or developing small applications.

- ✓ Highly available auto-healing cluster
- ✓ End-to-end encryption
- ✓ Role-based access control

Create a cluster

Starting at
FREE

[Dismiss](#)

[Advanced Configuration Options](#)

Establishing server (ATLAS)

[CLUSTERS](#) > CREATE A SHARED CLUSTER

Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Cloud Provider & Region

Azure, Netherlands (westeurope) ▾



★ Recommended region ⓘ

NORTH AMERICA

🇺🇸 Virginia-East2 (eastus2) ★

🇨🇦 Toronto (canadacentral) ★

🇺🇸 California (westus) ★

EUROPE

🇮🇪 Ireland (northeurope) ★

🇳🇱 Netherlands (westeurope) ★

ASIA

🇭🇰 Hong Kong (eastasia)

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) >
Encrypted

Additional Settings

MongoDB 4.4, No Backup >

Cluster Name

Cluster0 >

Establishing server (ATLAS)

The screenshot displays the Atlas web interface with a modal dialog box open for adding an IP access list entry. The background shows a sidebar with navigation options like 'DATA STORAGE', 'SECURITY', and 'Network Access', and a main content area with a heading 'Add an IP address' and a 'Learn more' link.

Add IP Access List Entry

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more](#).

Access List Entry:

Comment:

This entry is temporary and will be deleted in

Add an IP address
Configure which IP addresses can access your cluster.

[Learn more](#)

Feature Requests

Establishing server (ATLAS)

Add New Database User

Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding [Access Manager](#)

Authentication Method

Password Certificate AWS IAM (MongoDB 4.4 and up)

MongoDB uses [SCRAM](#) as its default authentication method.

Password Authentication

[SHOW](#)

Database User Privileges

Select a [built-in role](#) or [privileges](#) for this user.

Restrict Access to Specific Clusters/Data Lakes

Enable to specify the resources this user can access. By default, all resources in this project are accessible. OFF

Temporary User

This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week. OFF

Establishing server

- Recommendation for classes
 - ATLAS free tier, or
 - Local installation

Establishing client (shell)

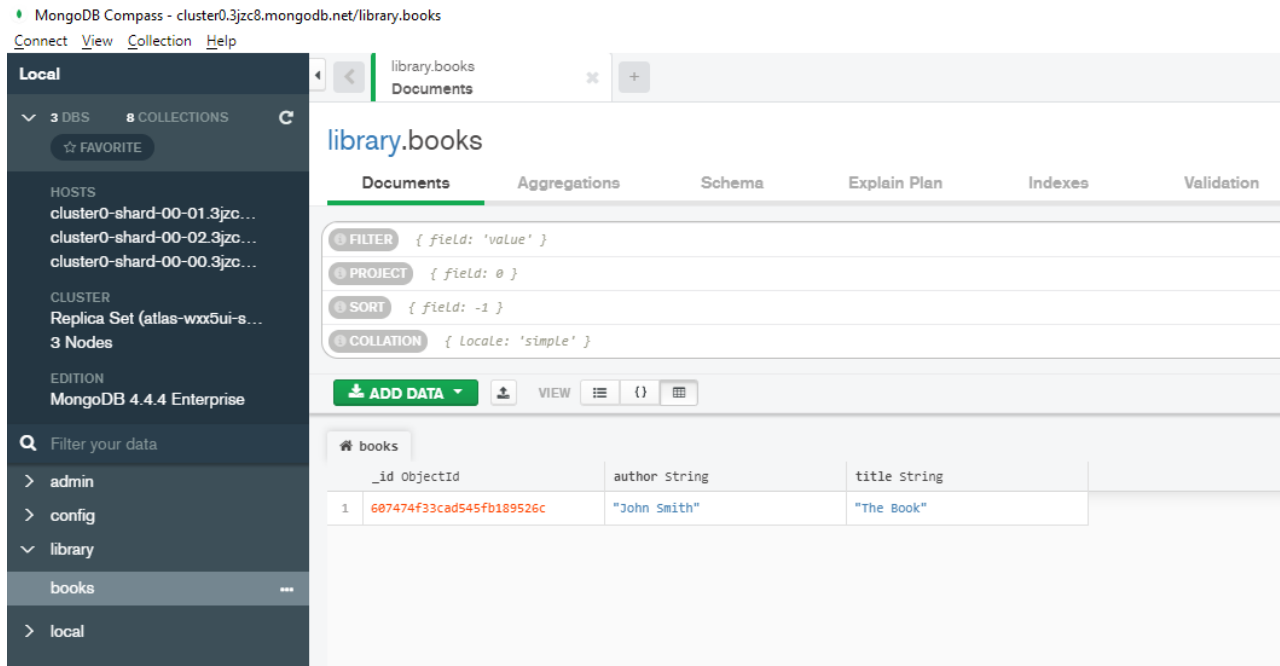
- mongo.exe
- JavaScript interpreter
- Multiline input is ok

```
> var hello = function() {  
... print("Hello World!");  
... }  
> hello();  
Hello World!
```

- Run external script
 - mongo C:\Data\Databases\MongoDB\booksCount.js
 - > load('C:/Data/Databases/MongoDB/booksCount.js')
- Non-interactive mode
 - mongo localhost/admin --eval
"printjson(db.runCommand({logRotate:1}))"

Establishing client (COMPASS)

- Download:
 - <https://www.mongodb.com/try/download/compass>



The screenshot displays the MongoDB Compass web interface. The title bar indicates the connection to 'cluster0.3jzc8.mongodb.net/library.books'. The left sidebar shows the 'Local' environment with 3 databases and 8 collections. The 'library' database is expanded, showing the 'books' collection. The main panel shows the 'library.books' collection with a 'Documents' tab selected. Below the tabs, there are configuration options for FILTER, PROJECT, SORT, and COLLATION. A table below shows a single document with the following fields:

	_id ObjectId	author String	title String
1	607474f33cad545fb189526c	"John Smith"	"The Book"

Structure

- Structure hierarchy
 - Instance → Databases → Collections → Documents
- Collection optionally may have a schema
 - <https://docs.mongodb.com/manual/core/schema-validation/>
- Rule: every document must have a key
 - `_id`
 - Present in all documents
 - Unique across collection
 - Any type (except array)

Storage

- Mongo talk with JSONs
 - JSONs can be easily imported and queried
- Documents stored in BSON
 - <http://bsonspec.org/>

BSON { 01010100
11101011
10101110
01010101 }

BSON [*bee · sahn*], short for Binary JSON, is a binary-encoded serialization of JSON-like documents. Like JSON, BSON supports the embedding of documents and arrays within other documents and arrays. BSON also contains extensions that allow representation of data types that are not part of the JSON spec. For example, BSON has a Date type and a BinData type.

BSON can be compared to binary interchange formats, like Protocol Buffers. BSON is more "schema-less" than Protocol Buffers, which can give it an advantage in flexibility but also a slight disadvantage in space efficiency (BSON has overhead for field names within the serialized data).

BSON was designed to have the following three characteristics:

1 Lightweight

Keeping spatial overhead to a minimum is important for any data representation format, especially when used over the network.

2 Traversable

BSON is designed to be traversed easily. This is a vital property in its role as the primary data representation for MongoDB.

3 Efficient

Encoding data to BSON and decoding from BSON can be performed very quickly in most languages due to the use of C data types.

specification

implementations

FAQ

discussion

Manipulating data

- Basic
 - show dbs
 - use library
 - db
 - db.books.save({_id:1, author:"Bułhakow", title:"Mistrz i Małgorzata"})
 - db.books.save({_id:2, author:"Golden", title:"Wyznania gejszy"})
 - db.books.save({_id:3, author:"Golding", title:"Władca much"})
 - db.books.find()
- ObjectId
 - db.books.save({author:"Rowling", title:"Harry Potter"})
 - ObjectId()
 - ObjectId().getTimestamp()
- Insert command
 - db.books.save({_id:3, author:"Orwell", title:"Folwark zwierzęcy"})
 - db.books.save({_id:3, author:"Golding", title:"Władca much"})
 - db.books.insert({_id:3, author:"Orwell", title:"Folwark zwierzęcy"})
 - db.books.insert({_id:4, author:"Orwell", title:"Folwark zwierzęcy", rating:8})
- More
 - <https://docs.mongodb.com/manual/tutorial/insert-documents/>

Manipulating data

- Save vs. Update vs. Insert
 - Insert
 - Add a new document
 - Update
 - Update an existing document
 - Save
 - Insert or Update depending if document already exists

Manipulating data

- Update problem with save
 - > var b = db.books.findOne({_id:4})
 - > b.rating = b.rating+1;
 - here someone else gets the book and modify rating
 - > db.books.save(b);

 - > db.books.save({_id:4, author:"Orwell", title:"Folwark zwierzęcy", rokWydania:1945})
 - -- and then
 - > db.books.save(b);

Manipulating data

- Update command
 - `db.col.update(query, update, options)`
- Examples
 - `db.books.update({_id:4}, {$inc:{rating:1}});`
- Operators
 - `$inc{rating:1}`
 - `$set:{y:3}`
 - `$unset:{y:0}`
 - `$rename:{'rko': 'rok'}`
- More
 - <https://docs.mongodb.com/manual/tutorial/update-documents/>

Manipulating data

- Delete

- `db.books.deleteMany({})`
- `db.inventory.deleteOne({ _id: 3 })`

- More

- <https://docs.mongodb.com/manual/tutorial/remove-documents/>

Querying data

- Query

- `db.col.find(query, projection)`
 - Projection: `{field:0|1, field:0|1, ...}` (all 0 or all 1)

- More

- <https://docs.mongodb.com/manual/tutorial/query-documents/>
- <https://docs.mongodb.com/manual/reference/method/db.collection.find/>

- Extend our data

```
db.books.save({_id:4, author:"Orwell", title:"Folwark zwierzęcy", year:1945, rating:8, location:{room:4,segment:2}, catalogue:
```

```
[  
  {number:"A01", available:true},  
  {number:"A02", available:false, rentDate:'2018-01-01'},  
  {number:"A03", available:true},  
]  
})
```

```
db.books.save({_id:5, author:"Steinbeck", title:"Grona gniewu", rating:7, location:{room:4,segment:3}, catalogue:
```

```
[  
  {number:"A04", available:false, rentDate:'2018-01-02'},  
  {number:"A05", available:false, rentDate:'2018-01-03'},  
]  
})
```

Querying data

- Basic queries
 - `> db.books.find({_id:4});`
 - `> db.books.find({_id:4}, {_id:1});`
 - `> db.books.find({_id:4}, {_id:0});`
 - `> db.books.find({_id: {$gt:2}})`
 - `> db.books.find({_id: {$not:{$gt:2}}})`
 - `> db.books.find({_id: {$in:[1,2]}})`
 - `> db.books.find({_id: {$nin:[1,2]}})`
 - `> db.books.find({author:/^Gold/});`
 - `> db.books.find().count()`

Querying data

■ Nested documents

- `> db.books.find({"location.room":4});`
- `> db.books.find({"catalogue.available":true}, {_id:1});`
- `> db.books.find({"catalogue.available":false}, {_id:1});`

■ Where

- `> db.books.find({$where: "this.author=='Golden' || this.title=='Władca much'"});`

■ Sorting

- `> db.books.find({}, {title:1}).sort({'catalogue.available':-1,title:1});`

■ Paging

- `> db.books.find({}, {_id:1}).sort({_id:1}).skip(2).limit(2);`

Querying data

- Iterating cursor

- > var c = db.books.find({}, {title:1});

- > c.size()

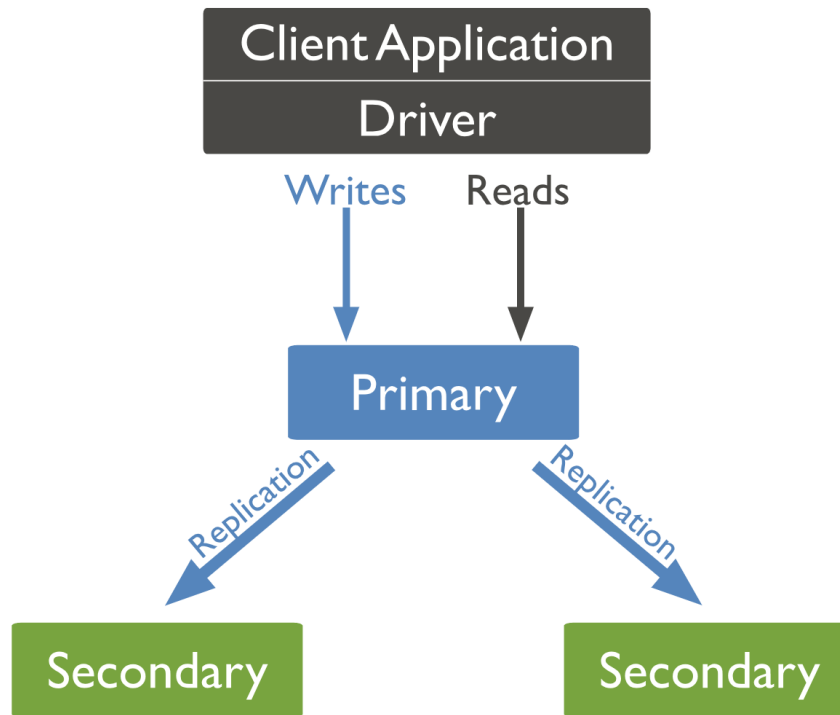
- > c.hasNext()

- > c.forEach(function(d){ print(d.title); })

- > c.hasNext()

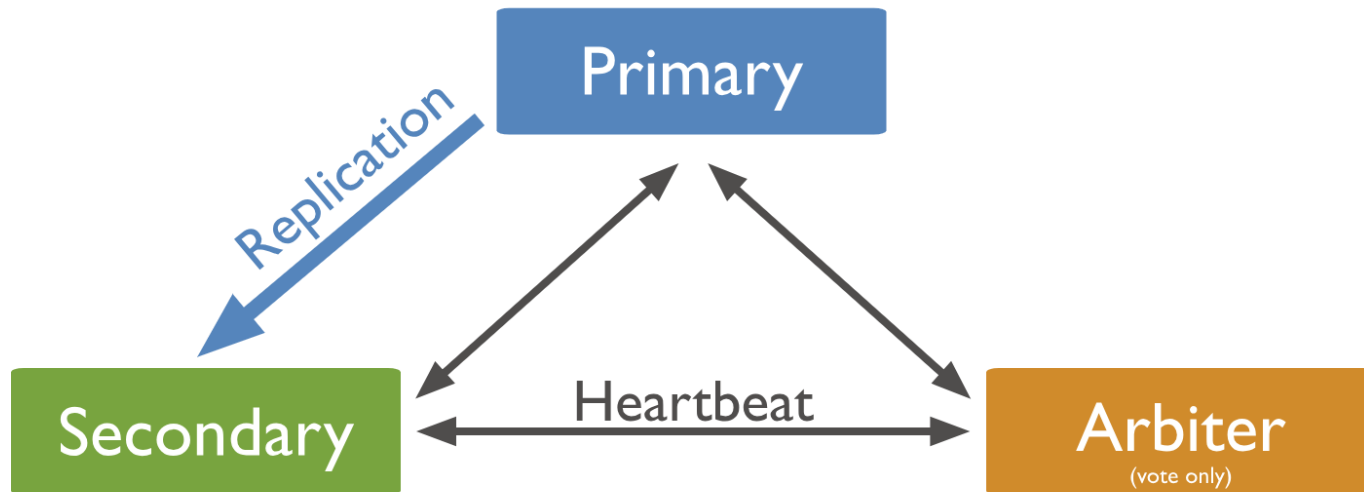
Replication

- Replica Set concept



Replication

- Role of arbiter
 - Doesn't have data, can be weak node
 - Support voting for primary in case there are even number of nodes



Replication

■ DEMO

■ Create folders

- c:\Data\Databases\MongoDB\db1
- c:\Data\Databases\MongoDB\db2
- c:\Data\Databases\MongoDB\db3

■ Run 3 instances

- start "A" mongod --dbpath c:\Data\Databases\MongoDB\db1 --port 10000 --replSet "demo"
- start "B" mongod --dbpath c:\Data\Databases\MongoDB\db2 --port 20000 --replSet "demo"
- start "C" mongod --dbpath c:\Data\Databases\MongoDB\db3 --port 30000 --replSet "demo"

■ Run shell

- mongo --port 10000

Replication

■ DEMO

■ Create a configuration object

(more <https://docs.mongodb.com/manual/reference/replica-configuration/>)

- `var rsConfig={ _id: "demo", members: [{_id: 0, host: 'localhost:10000', priority: 10}, {_id: 1, host: 'localhost:20000'}, {_id: 2, host: 'localhost:30000', arbiterOnly: true}]};`

■ Let's take a look

- `rsConfig`

■ Initiate a cluster

- `rs.initiate(rsConfig)`

Replication

- DEMO
 - Save something
 - use test;
 - `db.books.save({_id:1, title:"Mistrz i Małgorzata"})`
 - `db.books.find()`
 - Let's check second server
 - `mongo --port=20000`
 - `db.books.save({_id:2, title:"Wyznania gejszy"})`
 - `db.books.find();`
 - `rs.secondaryOk();`
 - `db.books.find();`

Replication

- DEMO
 - Let's check replication
 - Kill PRIMARY
 - Check SECONDARY
 - Resurrect PRIMARY
 - `start "A" mongod --dbpath c:\Data\Databases\MongoDB\db1 --port 10000 --replSet "demo"`
 - Check again

Client application

The screenshot displays the Visual Studio NuGet Package Manager interface. The main window shows a search for 'mongo' with several packages listed, including MongoDB.Driver, DCouple.Mongo, Bsynchro.DataAccess.Mongo.Core, Bsynchro.DataAccess.Mongo.Abstract, Graphene.Mongo, Repository.Mongo.Cqrs, Cqrs.Mongo, and Cqrs.Ninject.Mongo. A 'Preview Changes' dialog box is open in the foreground, showing the list of packages to be installed for the 'MongoDBClient' project. The dialog box contains the following text:

Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.

MongoDBClient

Installing:

- DnsClient.1.0.7
- Microsoft.NETCore.Targets.1.1.0
- Microsoft.Win32.Primitives.4.3.0
- Microsoft.Win32.Registry.4.0.0
- MongoDB.Bson.2.6.1
- MongoDB.Driver.2.6.1
- MongoDB.Driver.Core.2.6.1
- runtime.native.System.4.3.0
- runtime.native.System.Net.Http.4.0.1
- runtime.native.System.Security.Cryptography.4.0.1
- System.Buffers.4.3.0
- System.Collections.4.3.0
- System.Collections.Concurrent.4.3.0
- System.Collections.NonGeneric.4.0.1
- System.Collections.Specialized.4.0.1
- System.ComponentModel.4.0.1
- System.ComponentModel.Primitives.4.1.0
- System.ComponentModel.TypeConverter.4.1.0
- System.Diagnostics.Debug.4.3.0

Do not show this again

The background window shows the NuGet Package Manager for 'MongoDB.Driver'. The version selected is 'Latest stable 2.6.1'. The description is 'Official .NET driver for MongoDB'. The license is 'http://www.apache.org/licenses/LICENSE-2.0'. The date published is 'Thursday, May 17, 2018 (5/17/2018)'. The project URL is 'http://www.mongodb.org/display/DOCS/CSharp+Language+Center'. The report abuse URL is 'https://www.nuget.org/packages/MongoDB.Driver/2.6.1/ReportAbuse'. The tags are 'mongo, mongodb, nosql'. The dependencies are listed as follows:

Dependencies

- .NETFramework.Version=v4.5
- MongoDB.Bson (>= 2.6.1)
- MongoDB.Driver.Core (>= 2.6.1)
- .NETStandard.Version=v1.5
- MongoDB.Bson (>= 2.6.1)
- NETStandard.Library (>= 1.6.1)
- System.ComponentModel.TypeConverter (>= 4.1.0)
- MongoDB.Driver.Core (>= 2.6.1)
- System.Linq.Queryable (>= 4.0.1)

Other interesting stuff

- Indexing
- Aggregation
 - <https://docs.mongodb.com/manual/aggregation/>
- Views
- MapReduce
- Capped Collections
- Geo

References

- Introductions
 - <https://www.mongodb.com/nosql-explained>
 - <https://www.slideshare.net/Leesy/an-introduction-to-nosql-mongodb/>
 - <https://www.slideshare.net/mdirolf/introduction-to-mongodb>
 - <https://www.slideshare.net/mongodb>
 - <https://www.slideshare.net/drumwurzels/intro-to-mongodb/>
 - <https://www.toptal.com/database/the-definitive-guide-to-nosql-databases>
- Documentation
 - <https://docs.mongodb.com/manual/crud/>
 - <https://www.tutorialspoint.com/mongodb/index.htm>
- Client C# application
 - <https://docs.mongodb.com/ecosystem/drivers/csharp/>
 - <http://mongodb.github.io/mongo-csharp-driver/2.2/reference/driver/>
 - <https://blog.oz-code.com/how-to-mongodb-in-c-part-1/>
 - <https://code.visualstudio.com/docs/azure/mongodb>
- Cloud Hosting
 - <https://mlab.com/>