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
  - Quering 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
  - Maintanability much simpler (e.g. key-value pairs)
  - Easy replication and failover suport
  - 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.

https://www.mongodb.com/nosql-explained/nosql-vs-sql#differences-between-sql-and-nosql

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"]
}
```

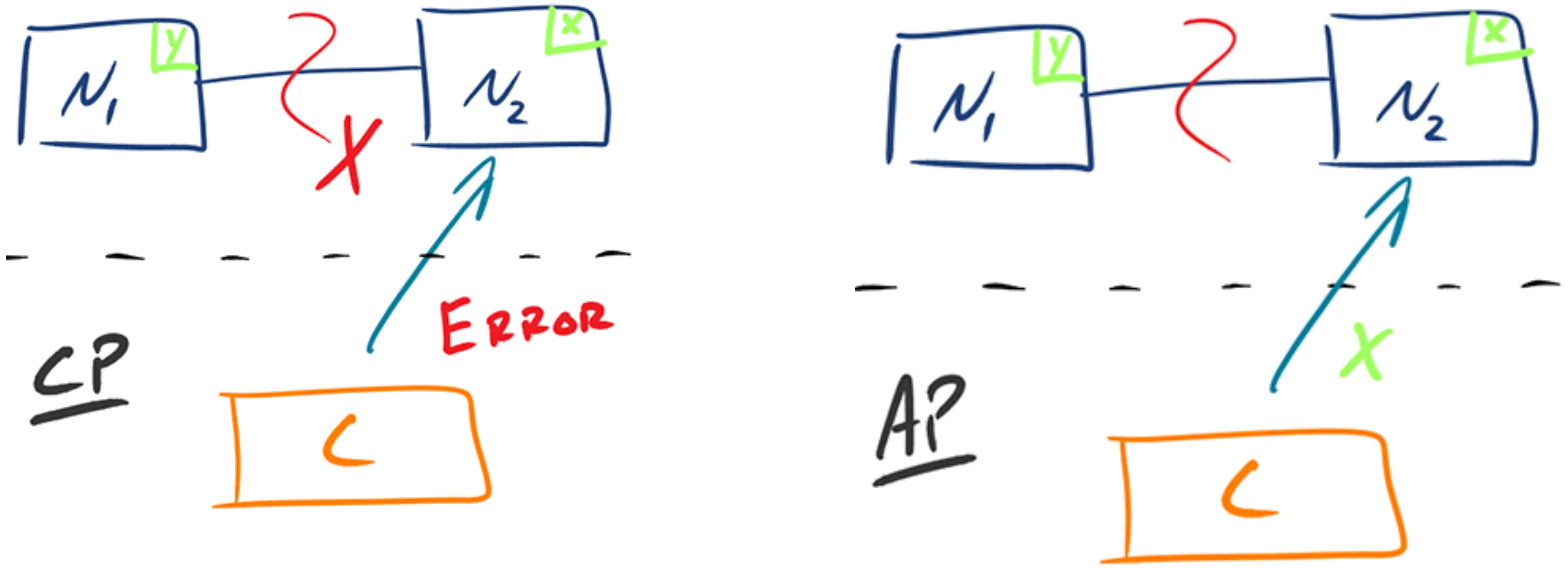
Eric Brewer's CAP Theorem

- 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

https://dzone.com/articles/understanding-the-cap-theorem

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



http://robertgreiner.com/2014/08/cap-theorem-revisited/

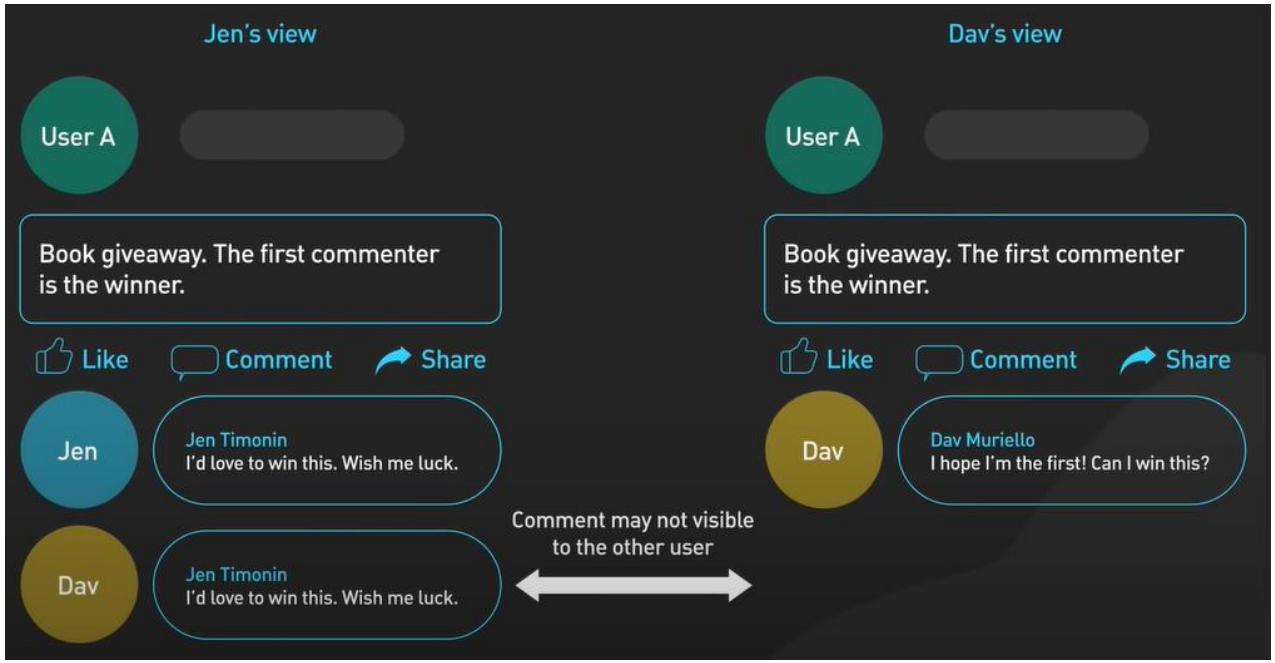
Eric Brewer's CAP Theorem

- Example 1

https://www.youtube.com/watch?v=BHqjEjzAicA

Eric Brewer's CAP Theorem

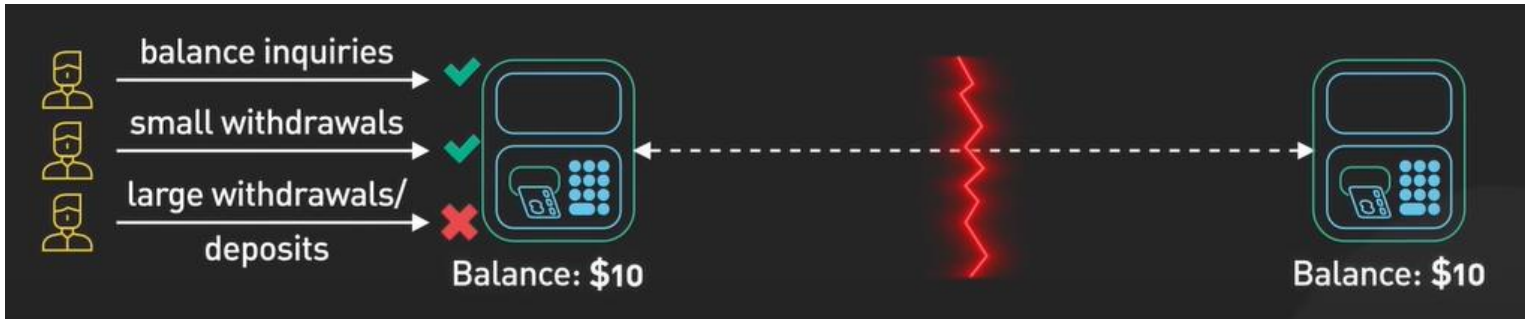
- Example 2:



https://www.youtube.com/watch?v=BHqjEjzAicA

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 transation

- 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.

https://neo4j.com/blog/acid-vs-base-consistency-models-explained/
https://stackoverflow.com/questions/3342497/explanation-of-base-terminology

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
  - Strukture 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

Types of NoSQL

- Graph databases
  - Focuses on the relationship between data elements
    - May have attributes and can be more meaningul 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

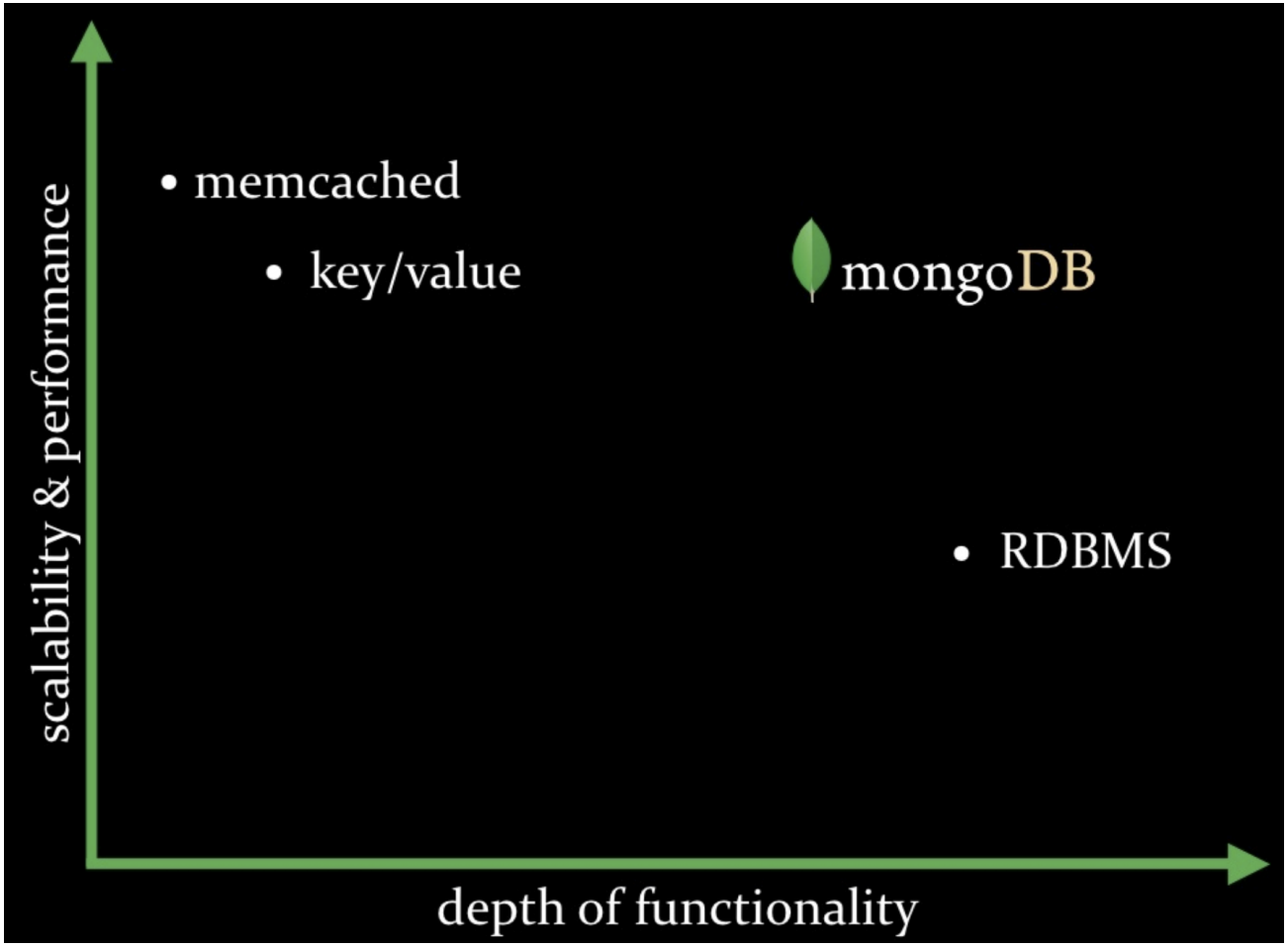
	Azure SQL Database	Azure SQL Managed Instance	Program SQL Server na maszynach wirtualnych	Azure Database for PostgreSQL	Azure Database for MySQL	Azure Database for MariaDB	Azure Cosmos DB	Azure Cache for Redis
Relacyjna baza danych	✓	✓	✓	✓	✓	✓		
Nierelacyjna baza danych (NoSQL)							✓	
Baza danych w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

https://azure.microsoft.com/pl-pl/product-categories/databases/

MongoDB

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

MongoDB position

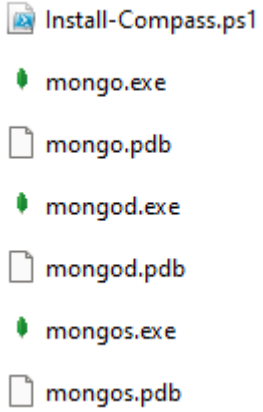


Establishing server (local)

- 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

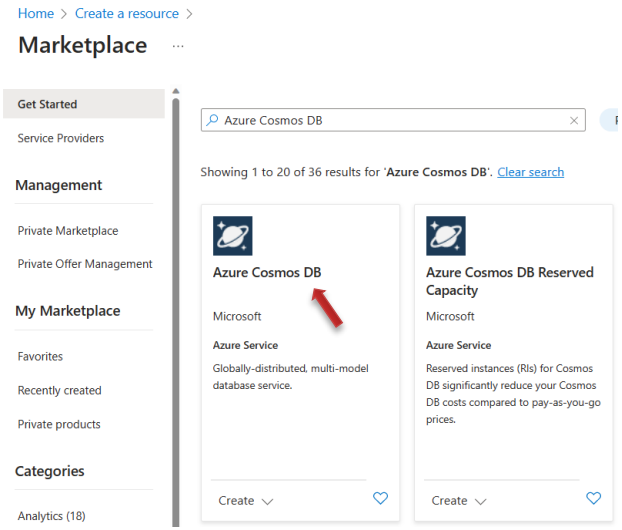


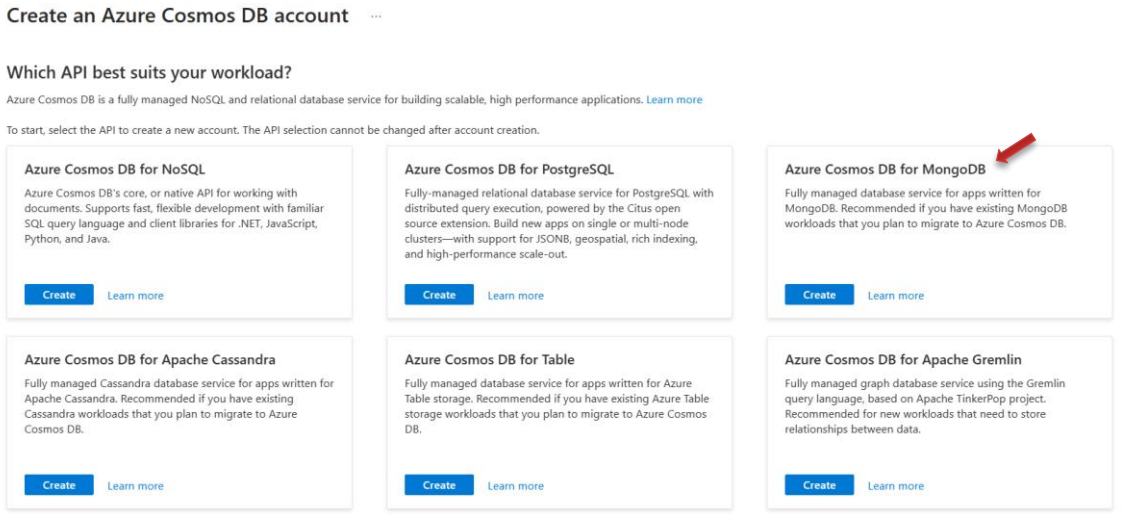
Establishing server (local)

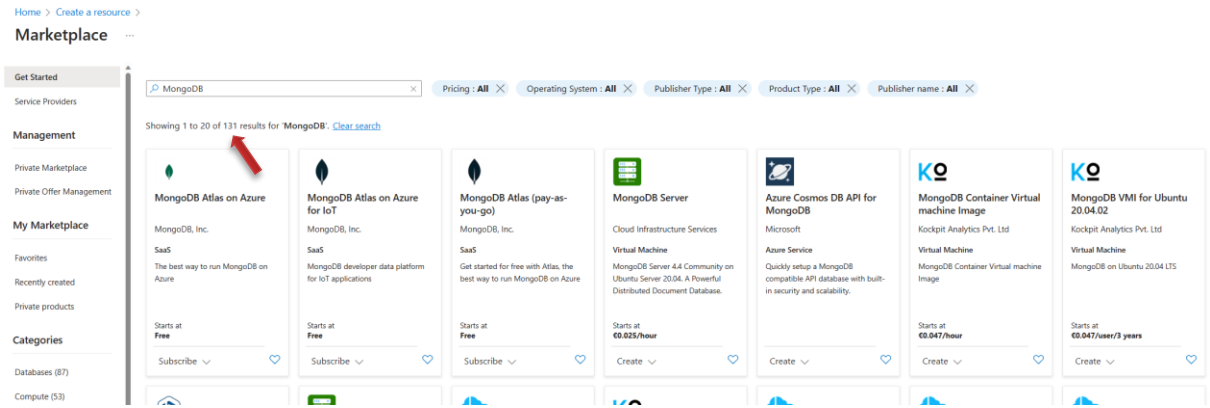
- Better usage:
  - mongod --dbpath c:\Data\Databases\MongoDB\db
  - mongo
- Best usage:
  - mongod -f c:\Data\Databases\MongoDB\mongod.conf
- Config file:

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

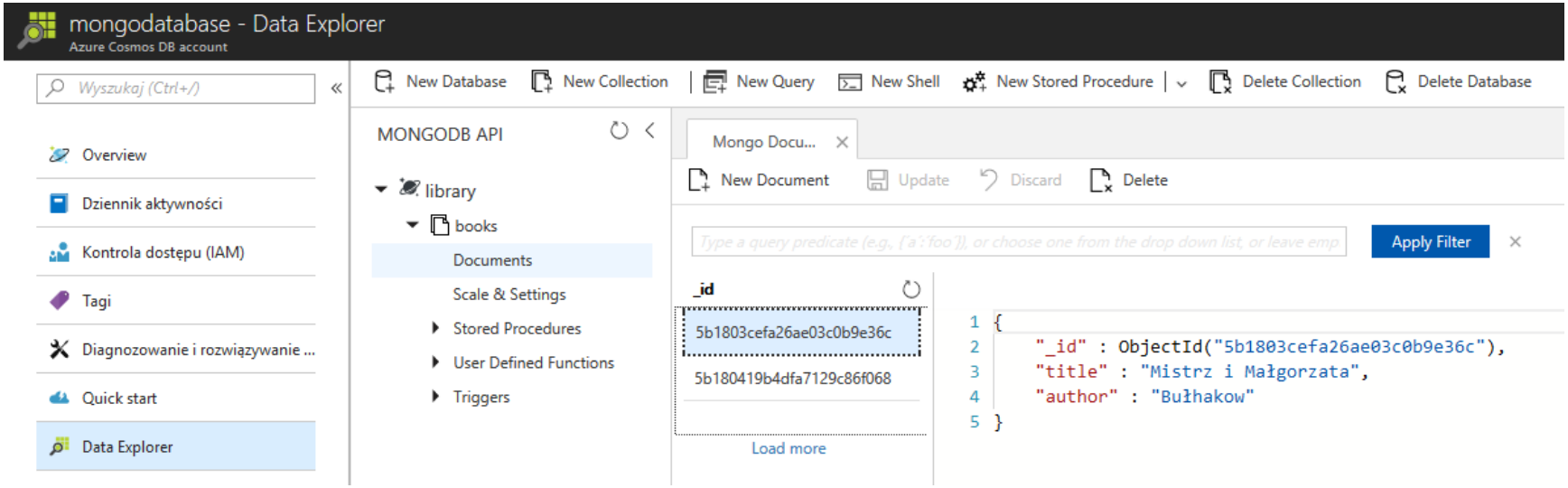
Establishing server (Azure)



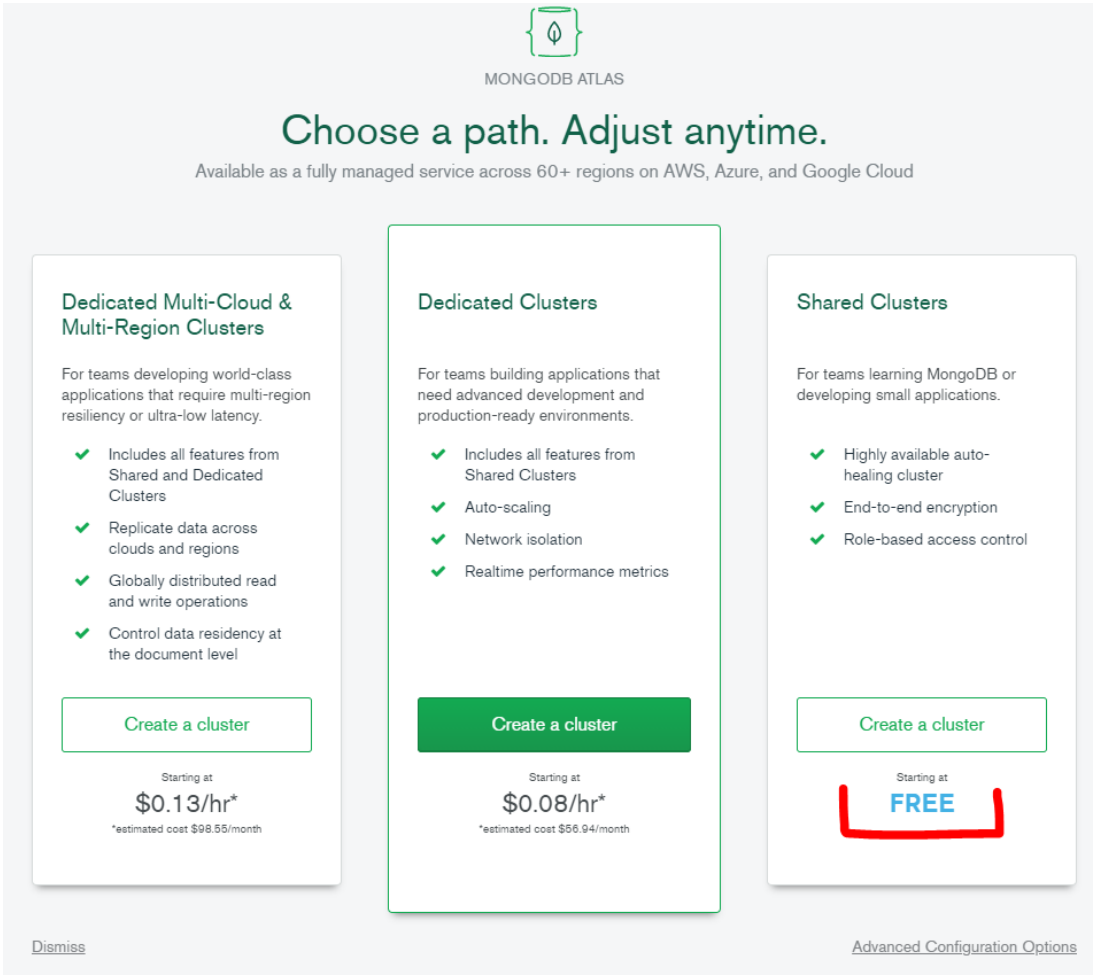




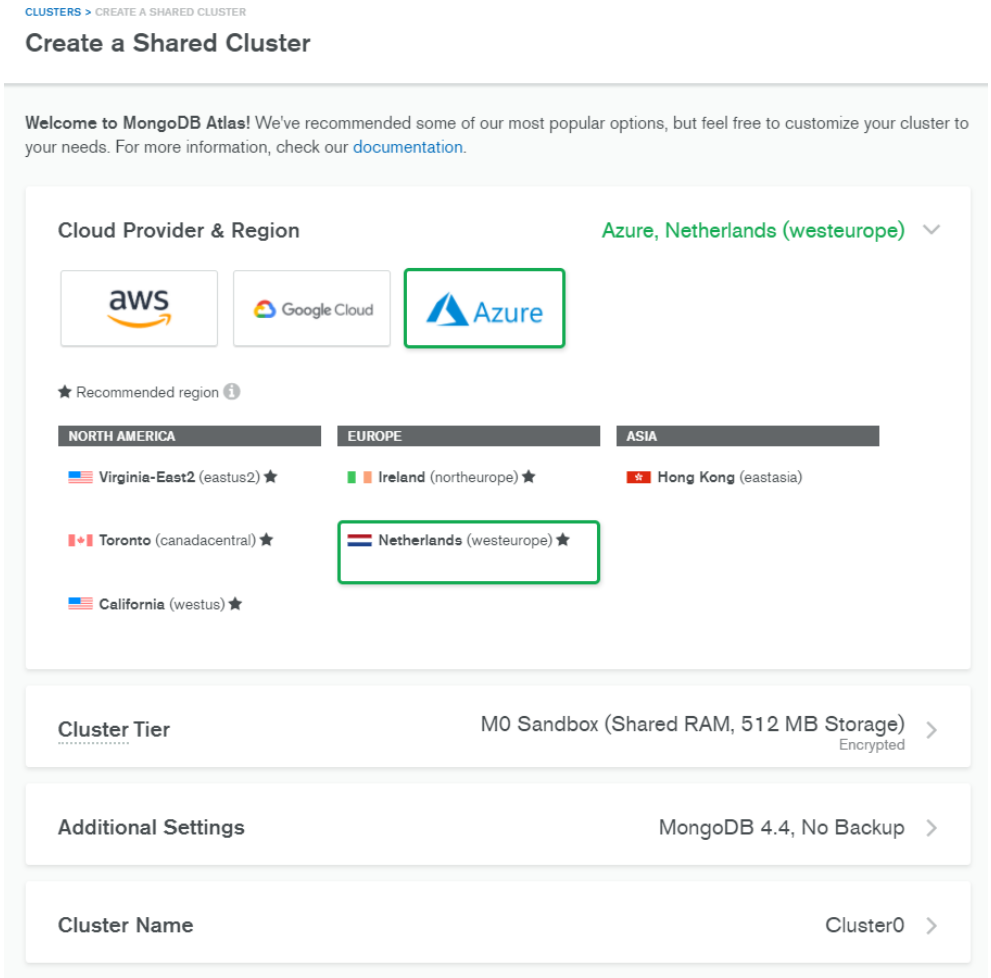
Establishing server (Azure)



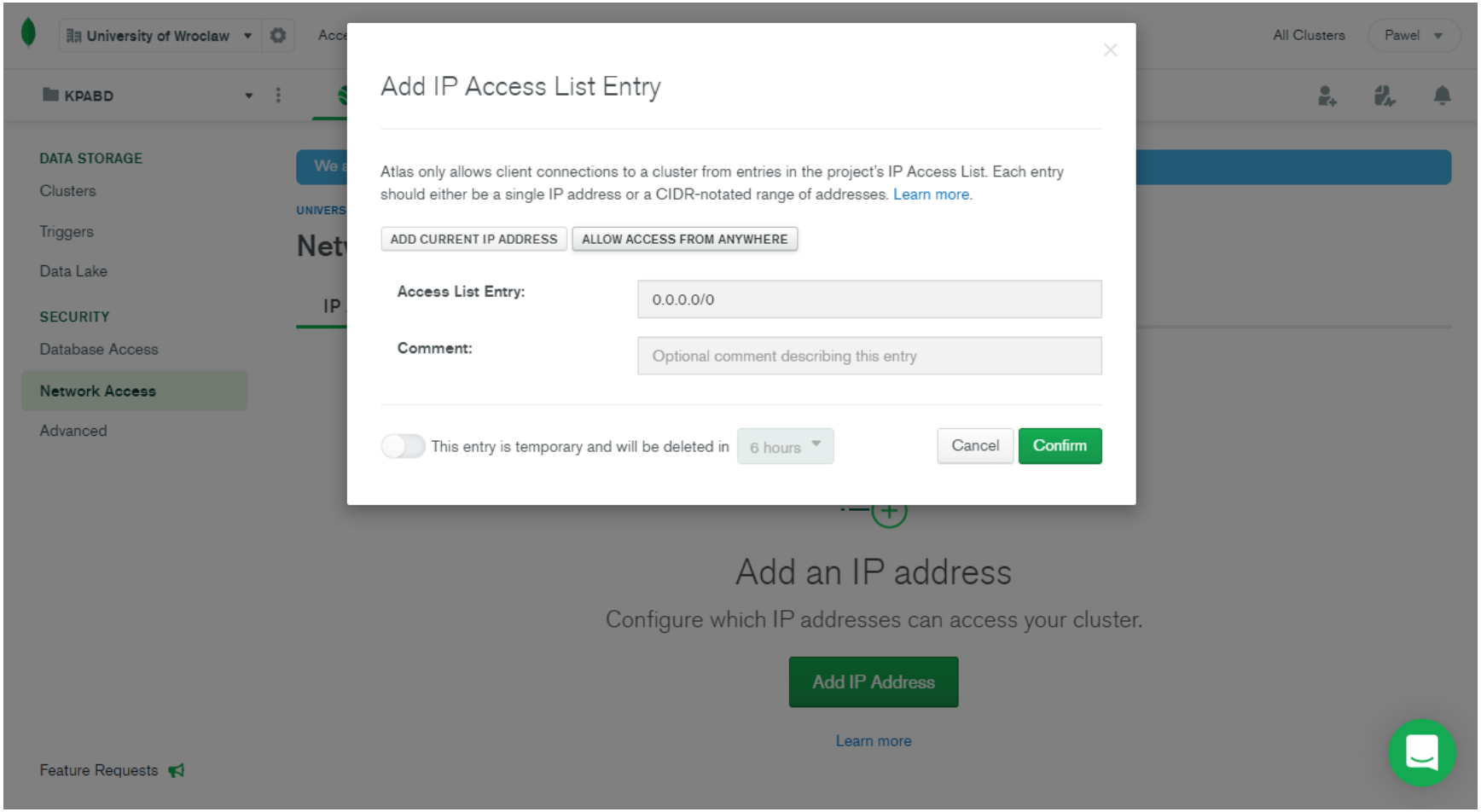
Establishing server (ATLAS)



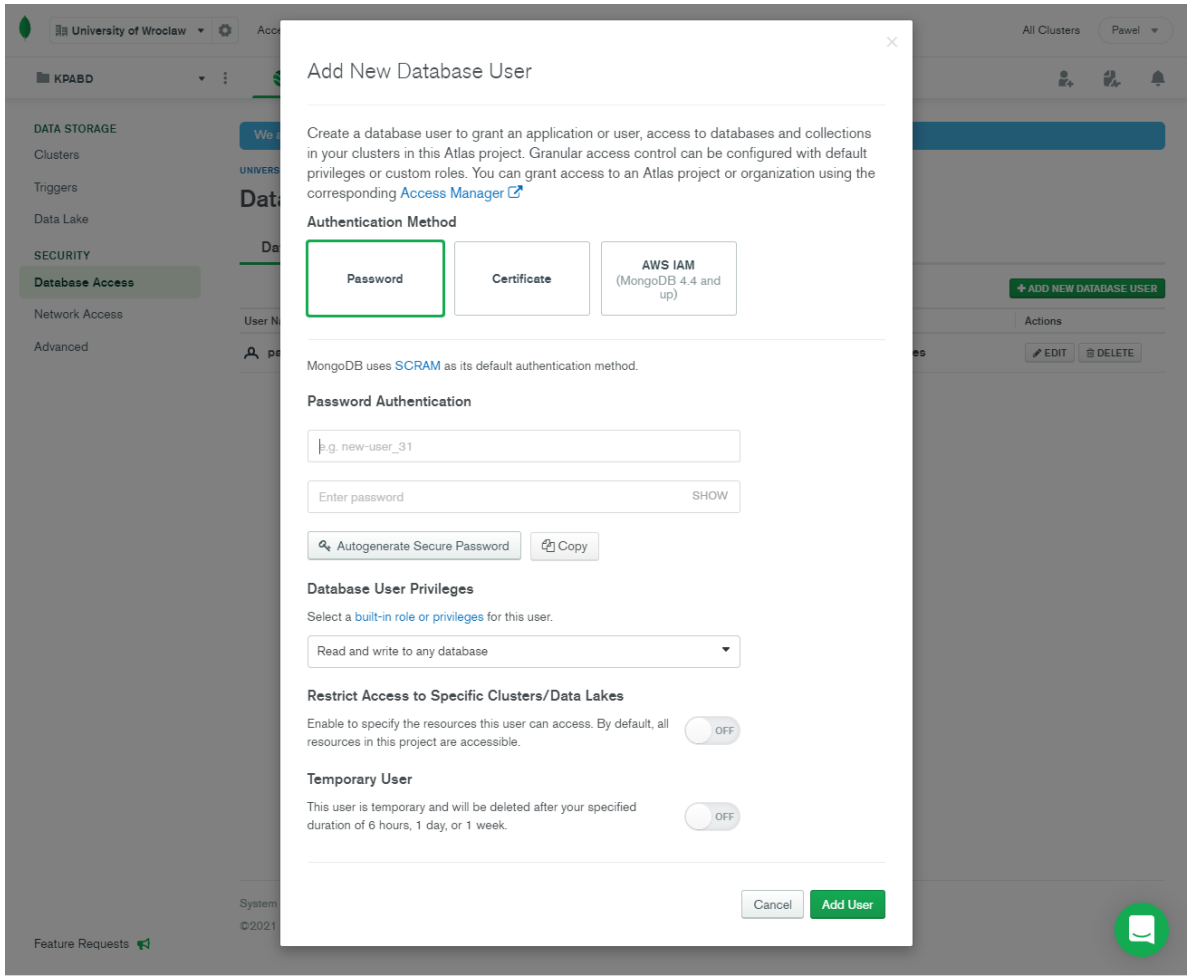
Establishing server (ATLAS)



Establishing server (ATLAS)



Establishing server (ATLAS)



Establishing server

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

Establishing client (shell)

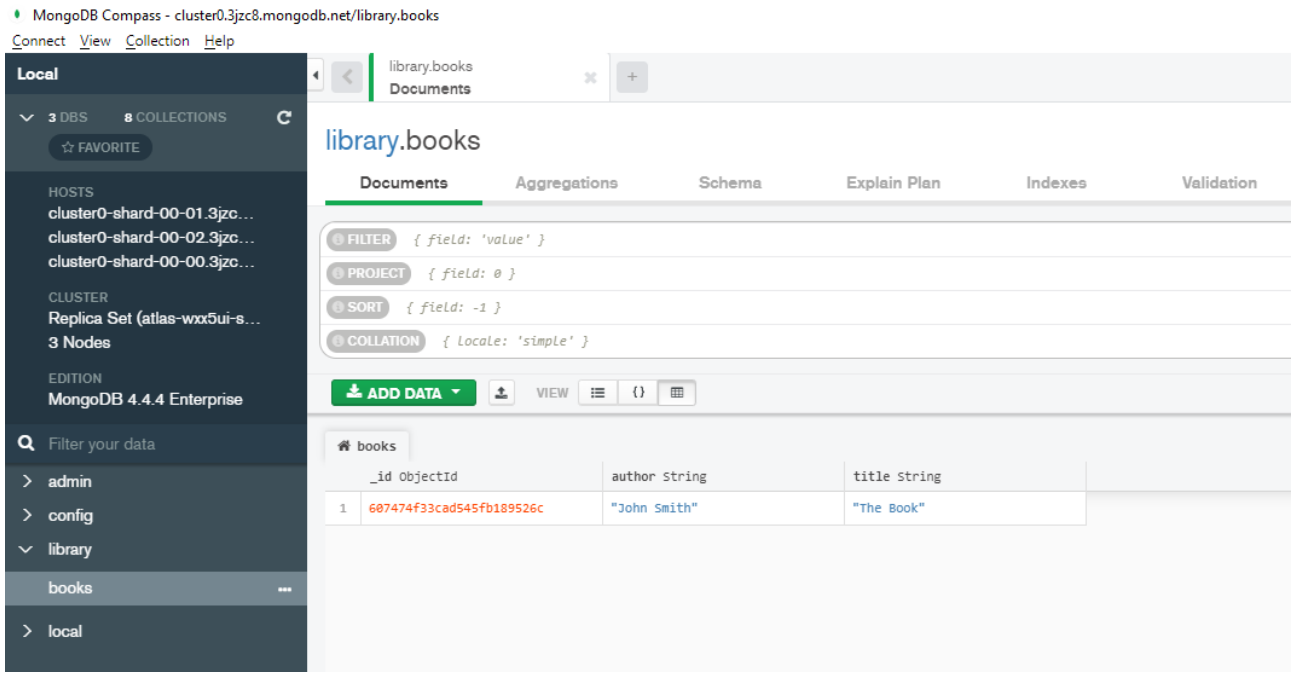
- mongo.exe
- JavaScript interpreter
- Multiline input is ok

```
> 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

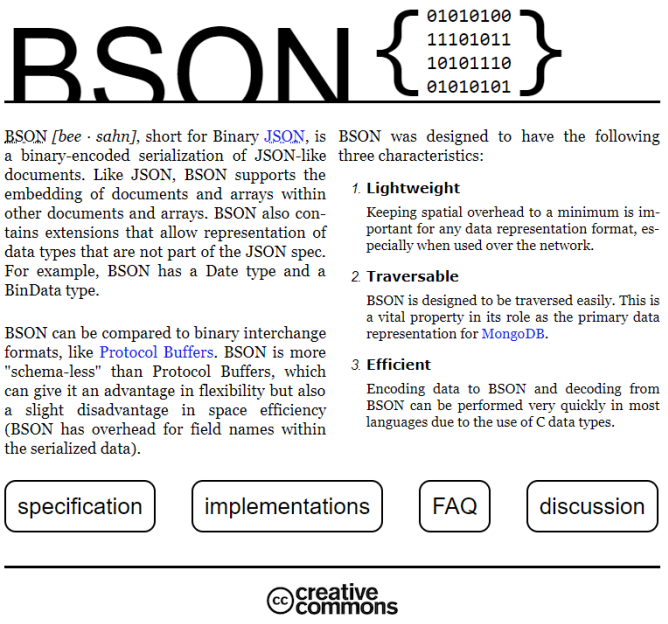


Structure

- 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/



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

- 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/

Quering 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'},
]
})
```

Quering 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()

Quering 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);

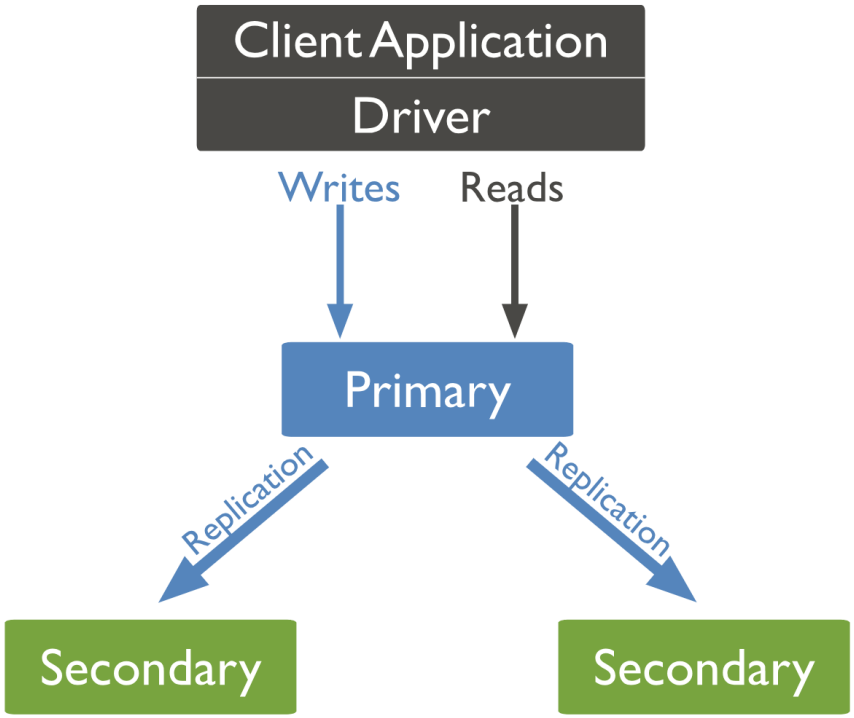
Quering data

- Iterating cursor

```
> var c = db.books.find({},{title:1});
> c.size()
> c.hasNext()
> c.forEach(function(d){ print(d.title); })
> c.hasNext()
```

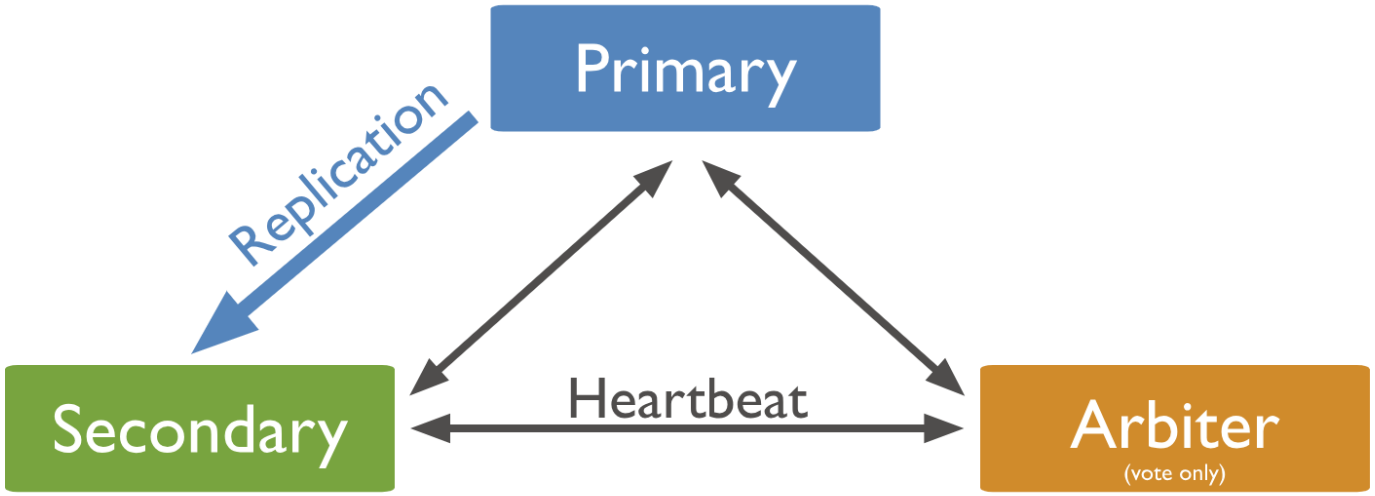
Replication

- Replica Set concept



Replication

- Role of arbiter
  - Doesn't have data, can be weak node
  - Support voting for primary in case there are even numer 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
  - Initiatiate a cluster
    - rs.initiate(rsConfig)

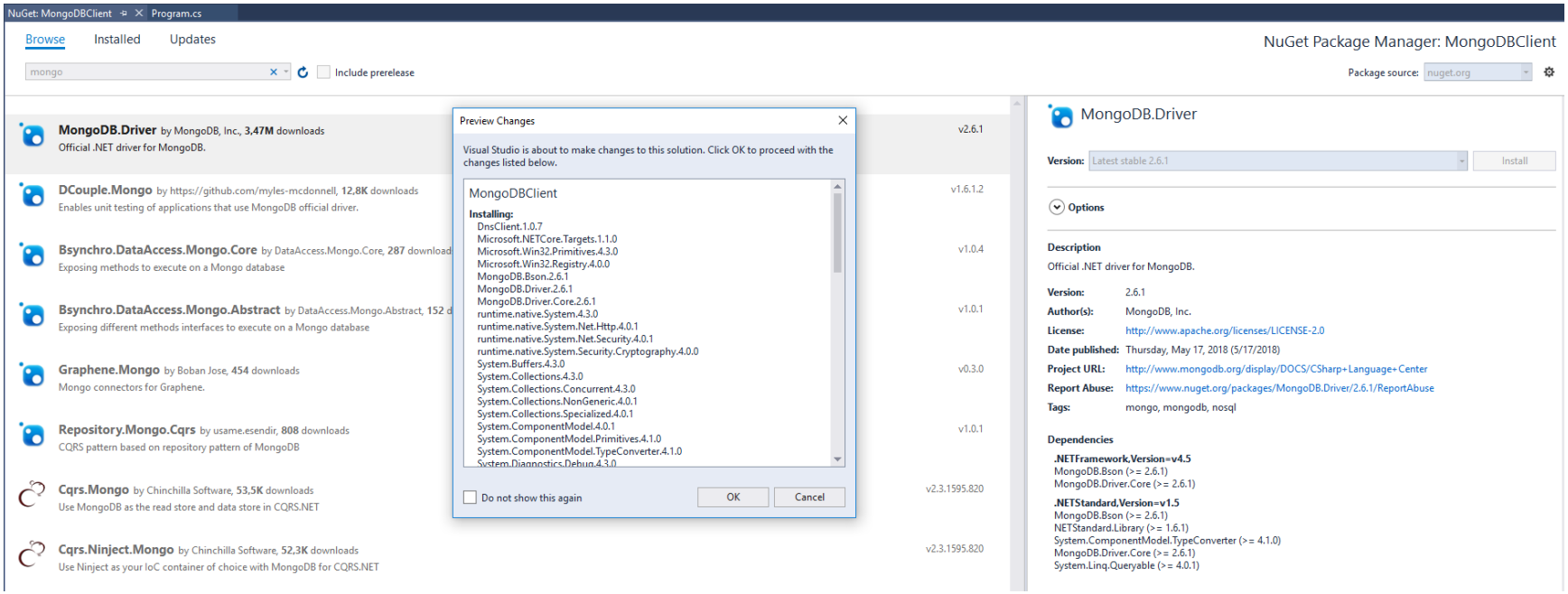
Replication

- 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



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/drumwurzel/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/