
STACK 2022
Do more with your data, with less
Derrick Chua, Senior Solutions Architect
MongoDB

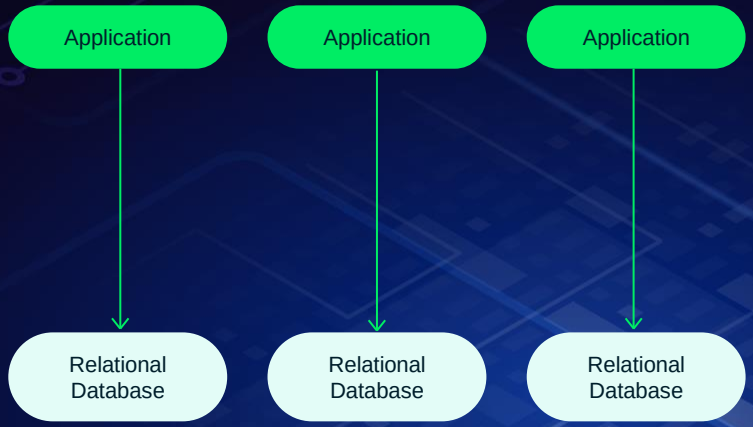
STACK 2022

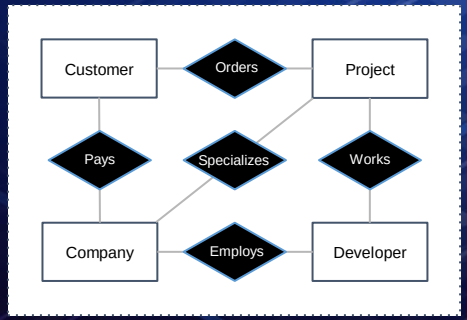
The typical data infrastructure is still built around legacy relational databases

Relational databases are optimized to solve a different set of problems

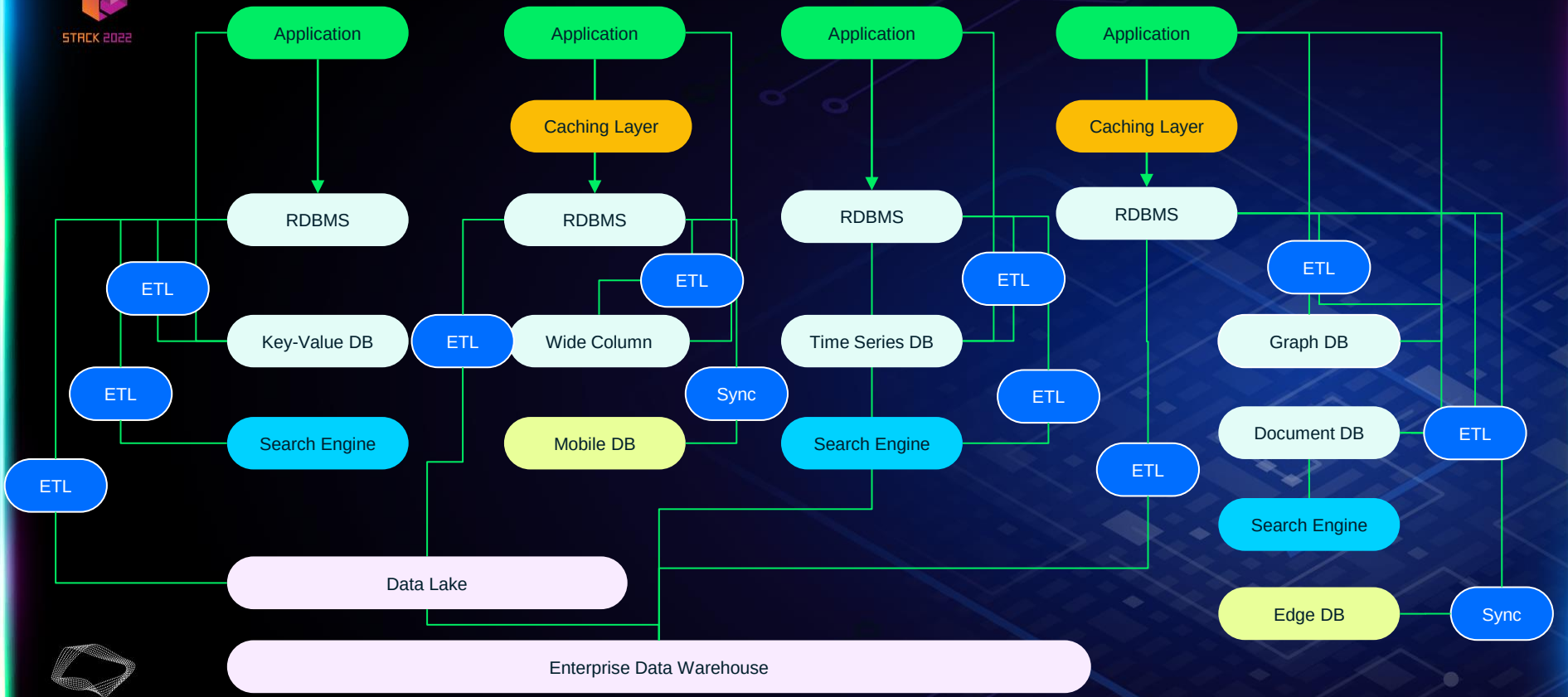
Data structures clash with modern data and the objects developers work with

Rigidity makes experimenting and iterating on applications difficult





GOVTECH SINGAPORE


Application
Application
Application
Application
Caching Layer
Caching Layer
RDBMS
RDBMS
RDBMS
RDBMS
ETL
ETL
ETL
ETL
ETL
Key-Value DB
Wide Column
Time Series DB
Graph DB
ETL
ETL
Sync
ETL
Search Engine
Mobile DB
Search Engine
Document DB
ETL
ETL
ETL
Search Engine
Data Lake
Edge DB
Sync
Enterprise Data Warehouse

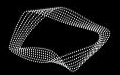
Fragmented developer experience

Multiple operational and security models to rationalize

Significant data integration effort required

Unnecessary data duplication

Developer Data Platform



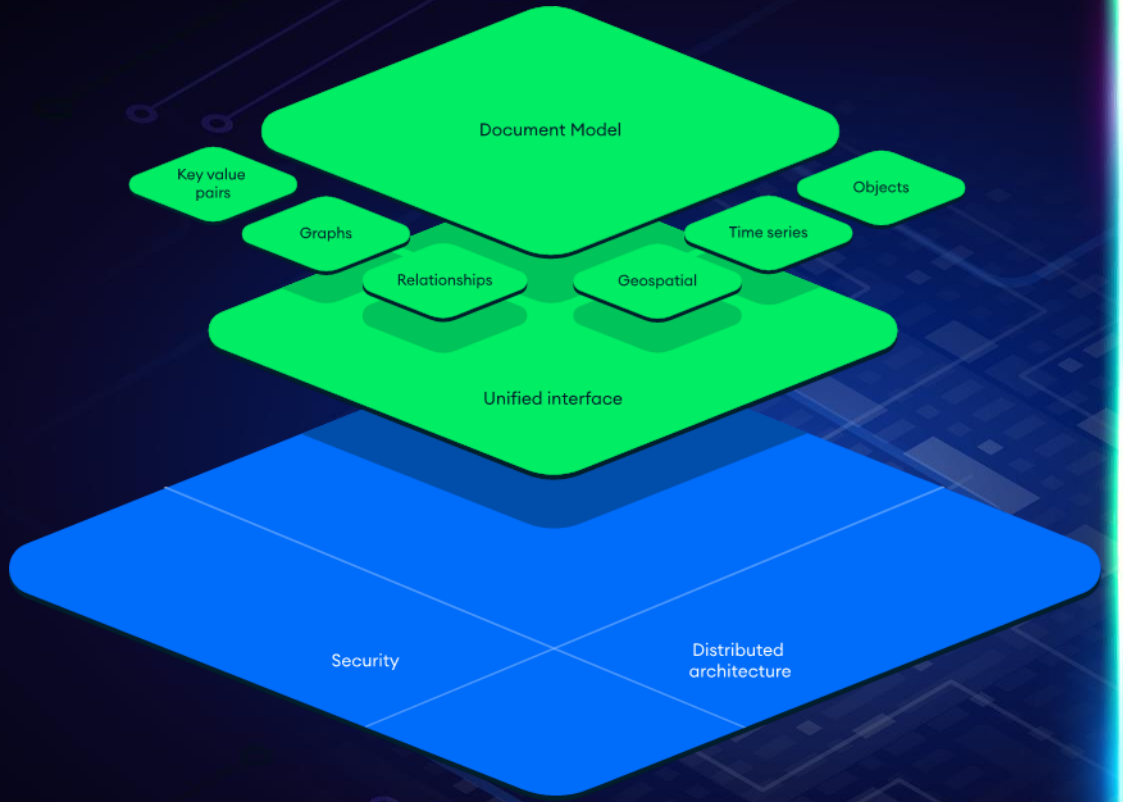
Built around a modern, distributed database built for developers

Data model that maps to how developers think/code; flexible while allowing data governance when needed

Strongly consistent, support for ACID transactions

One easy query interface for a variety of workloads

Advanced security & data distribution capabilities



Address a range of application use cases w/o adding complexity

Able to support full-text search functionality for delivering a fast and relevant user experience

Able to support data on mobile devices/at the edge w/o having to manually sync data

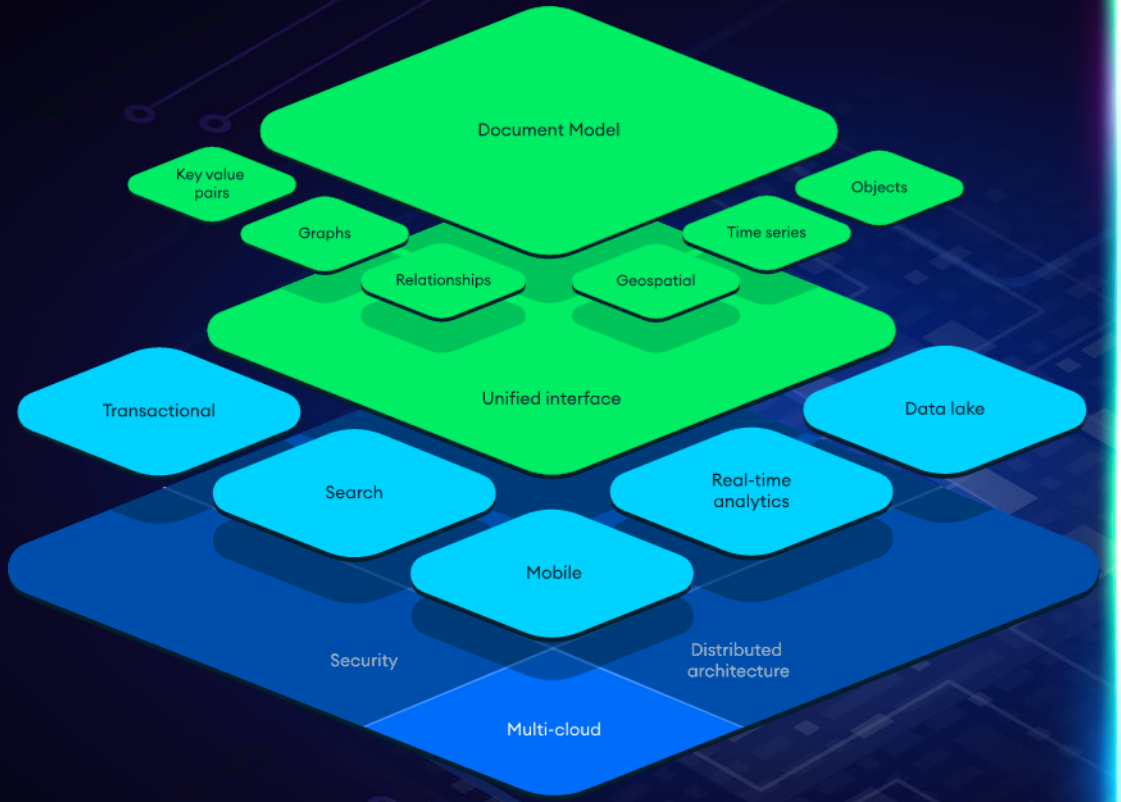
Able to deliver real-time analytics on live data w/o having to move data back & forth

Run anywhere—start on-premises, migrate to the cloud, multi-cloud

The flexibility to deploy in over 90 regions across 3 cloud providers

Database that extends across multiple clouds; get the best from each provider with no lock-in

Seamlessly migrate from on-premises to cloud w/o code rewrites



STACK 2022

MongoDB.

Do more with your data with

One interface.
For any application.
Anywhere.

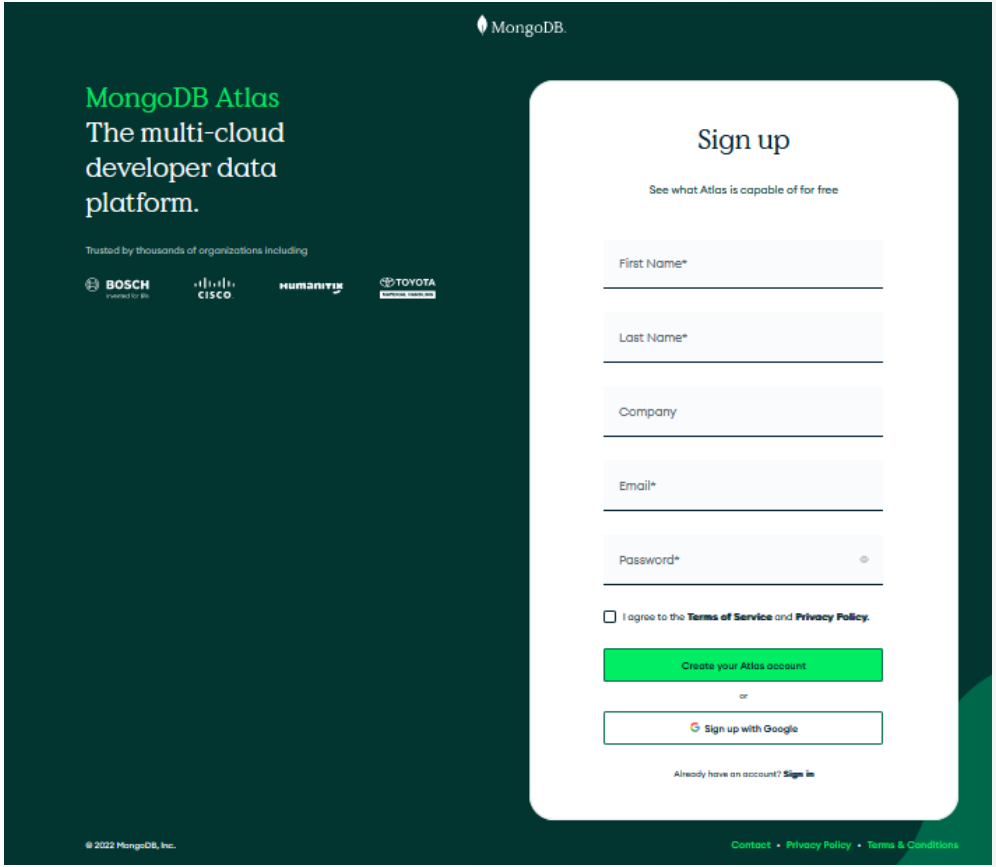
GOVTECH SINGAPORE

STACK 2022

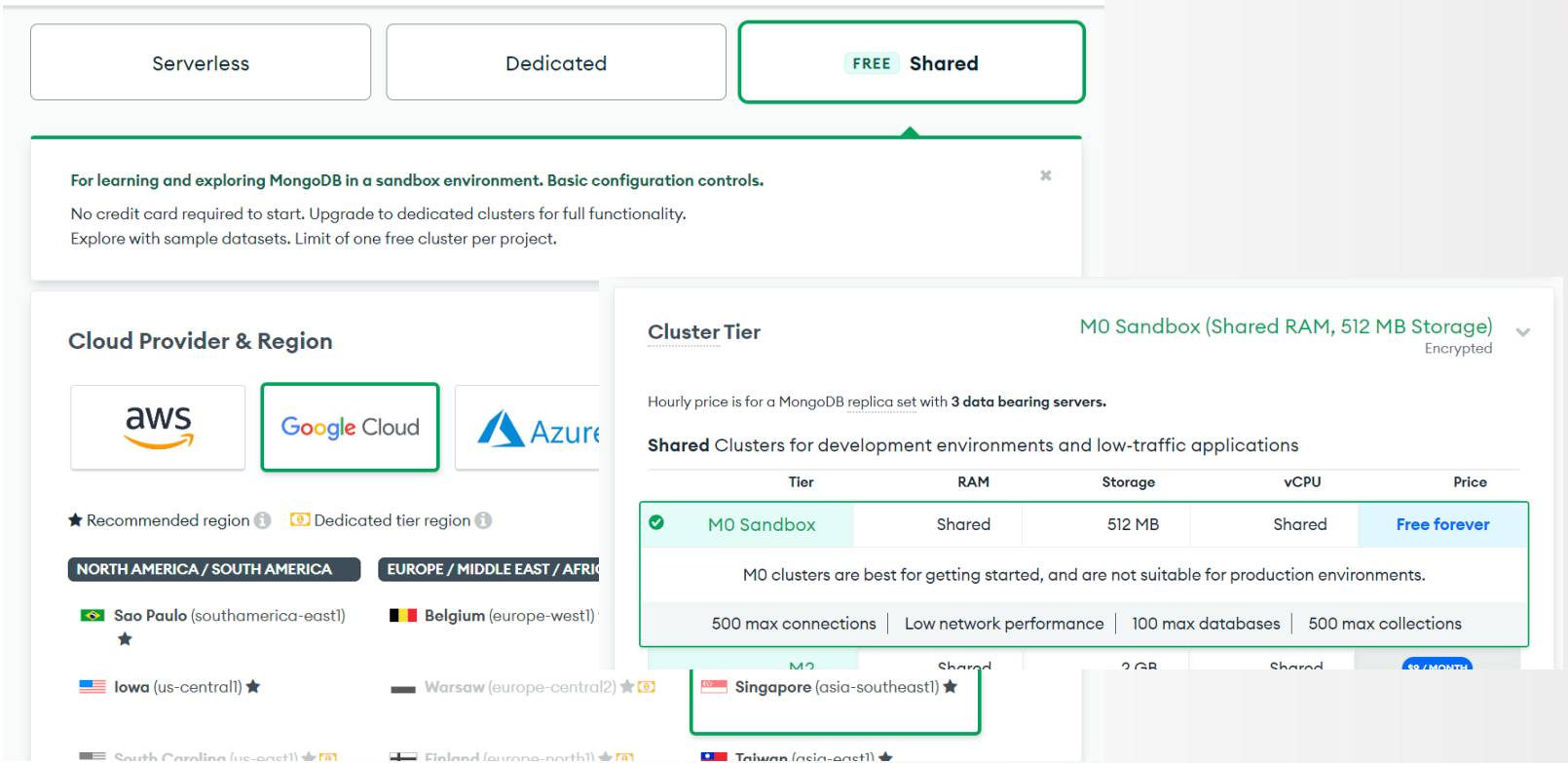
Hands On:

Get your free (forever)
Atlas Cluster

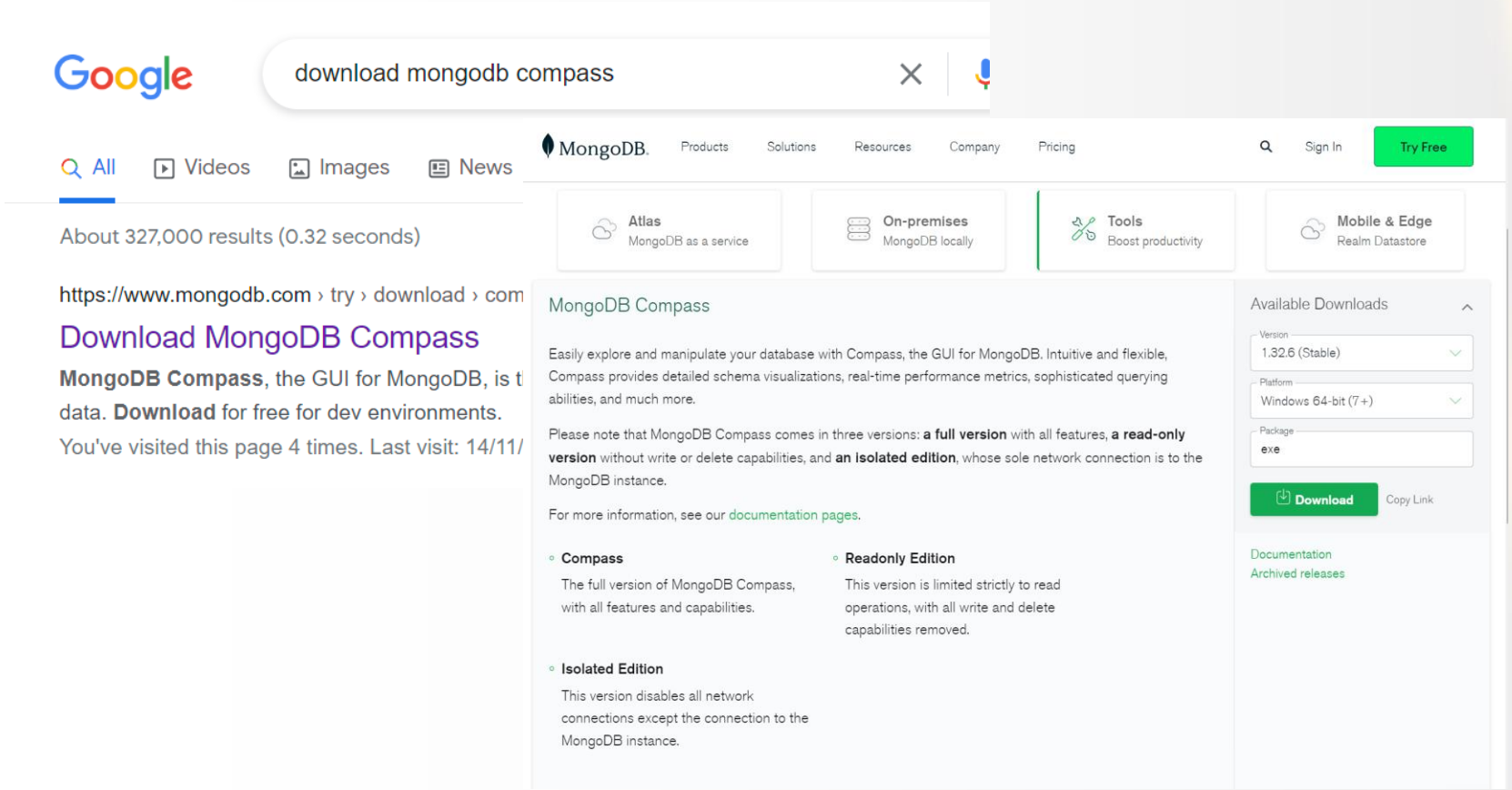
1. Sign up at cloud.mongodb.com



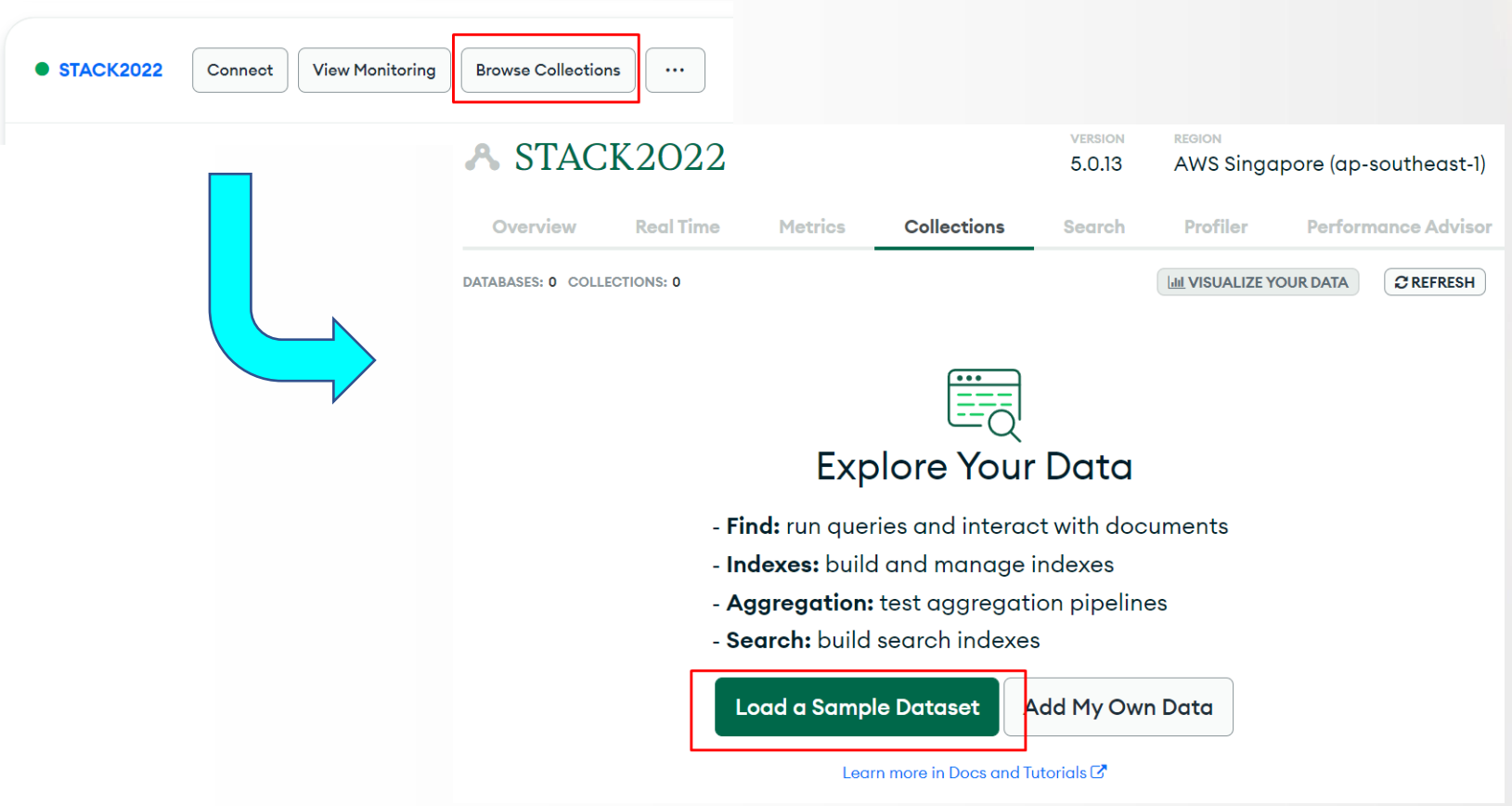
2. Create free forever M0 cluster



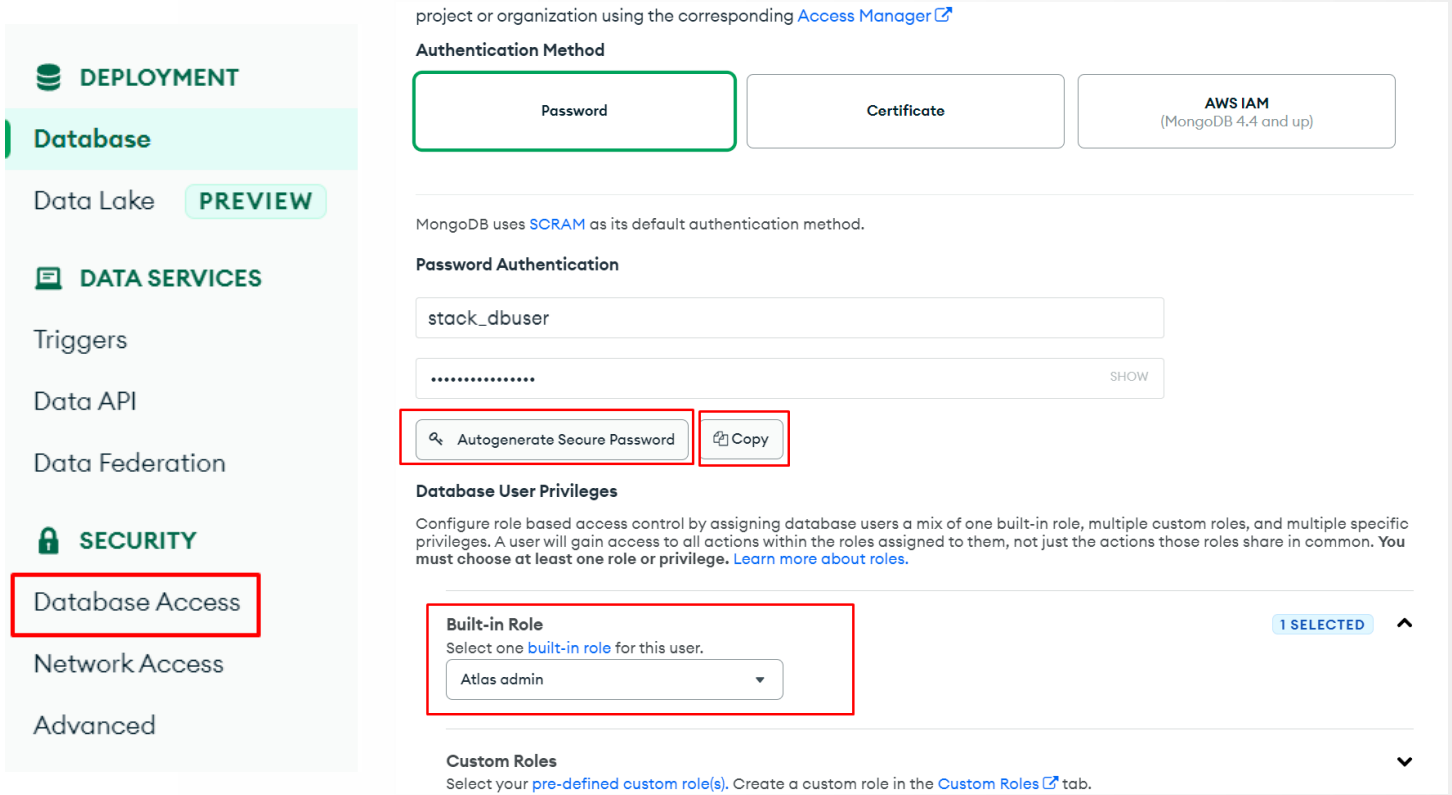
3. Install MongoDB Compass



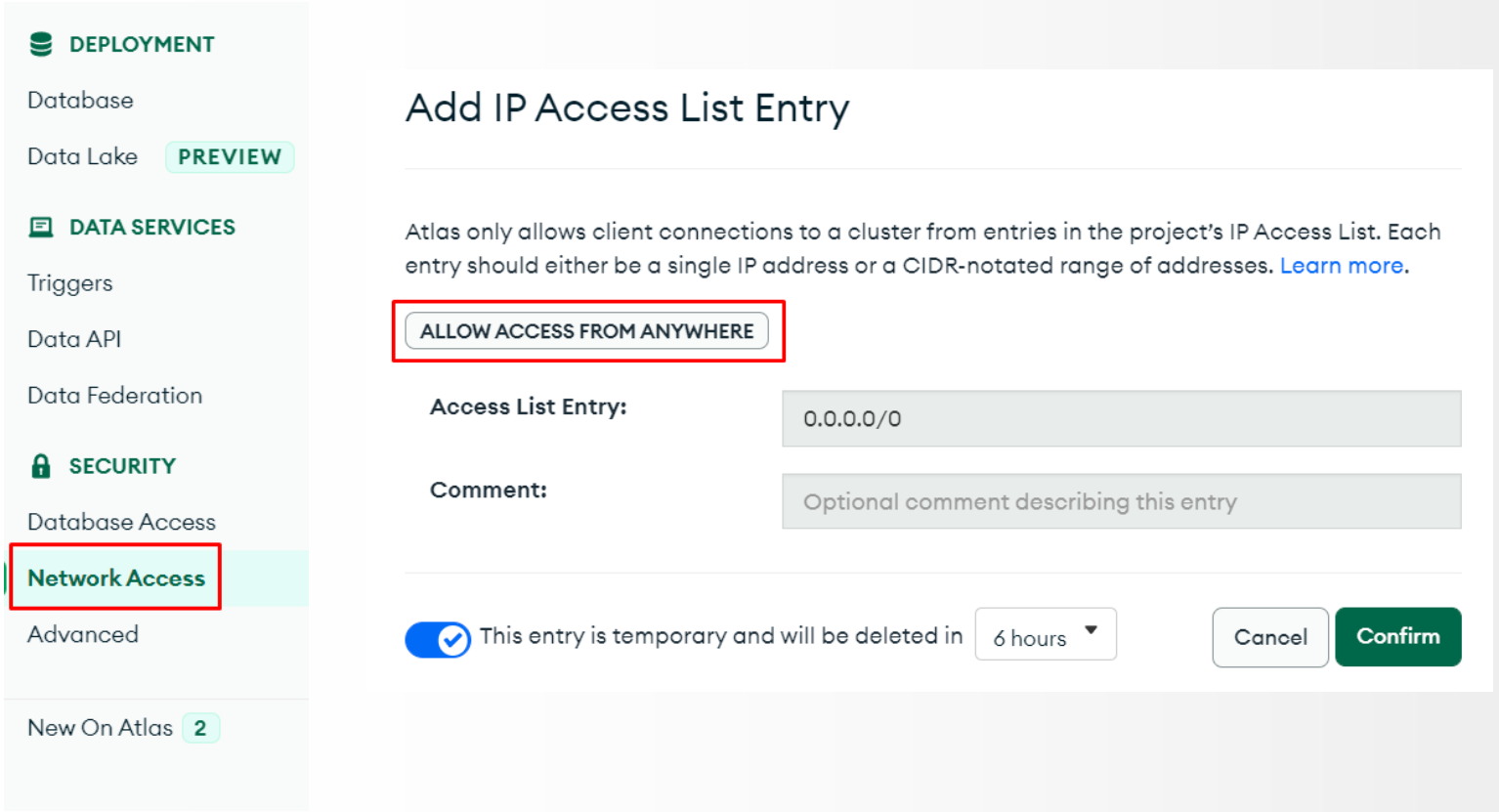
4. Load sample data



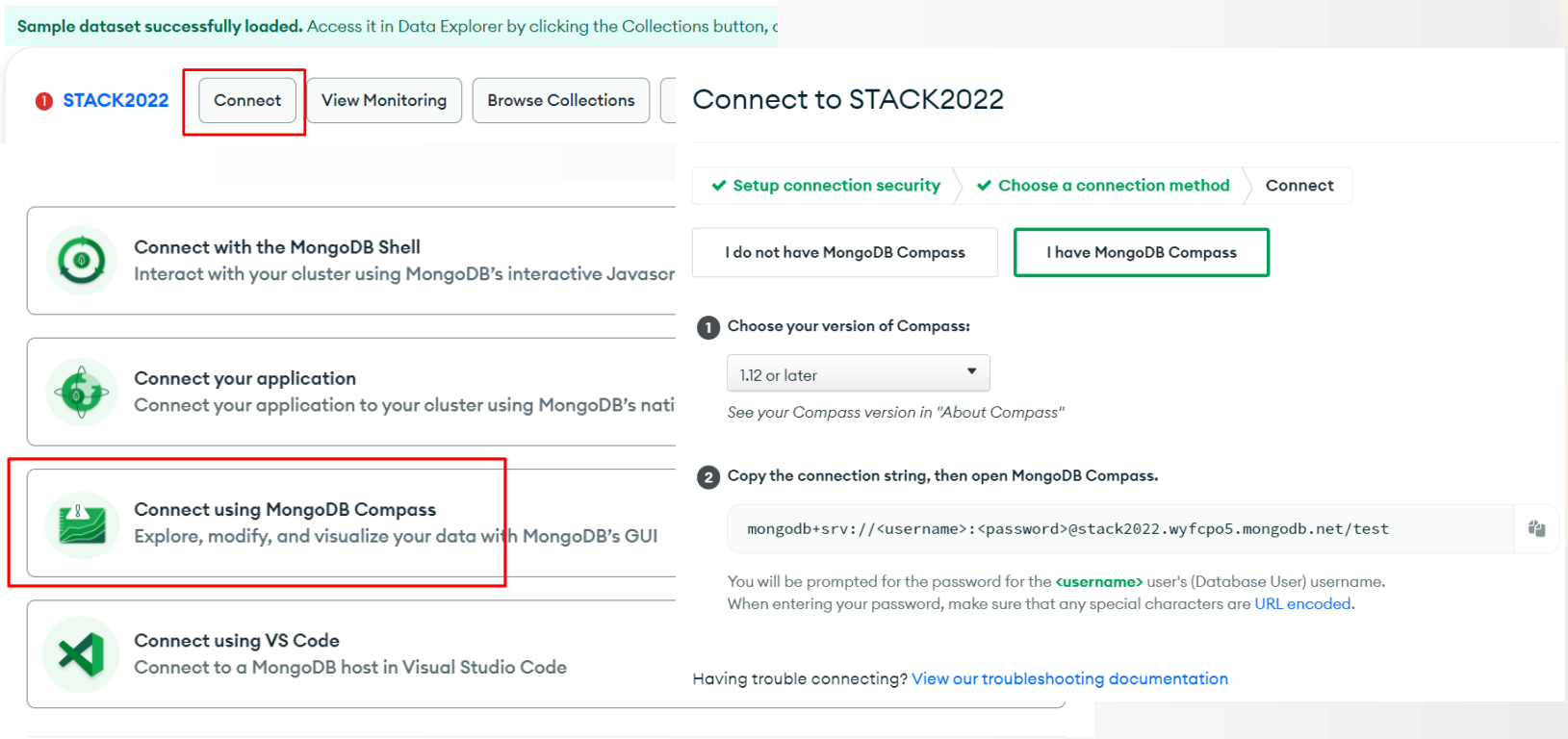
5. Add database user

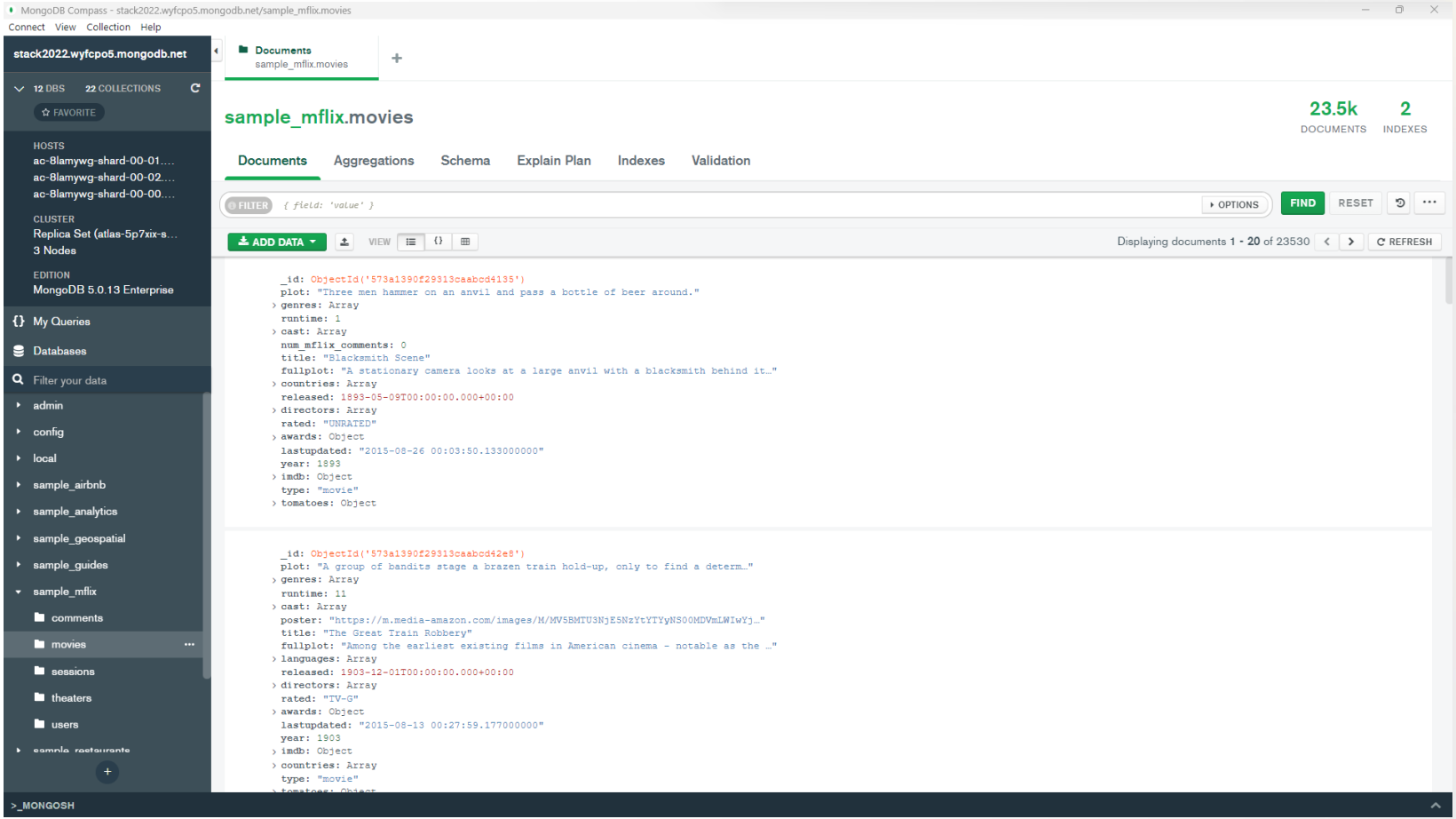


6. Open network access



7. View dataset with Compass




MongoDB Compass - stack2022.wyfcpo5.mongodb.net/sample_mflix.movies
Connect View Collection Help
stack2022.wyfcpo5.mongodb.net
12 DBS 22 COLLECTIONS
FAVORITE
HOSTS
ac-8lamywg-shard-00-01....
ac-8lamywg-shard-00-02....
ac-8lamywg-shard-00-00....
CLUSTER
Replica Set (atlas-5p7xix-s...
3 Nodes
EDITION
MongoDB 5.0.13 Enterprise
My Queries
Databases
Filter your data
admin
config
local
sample_airbnb
sample_analytics
sample_geospatial
sample_guides
sample_mflix
comments
movies
sessions
theaters
users
Documents
sample_mflix.movies
sample_mflix.movies
23.5k
DOCUMENTS
2
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
{ field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 20 of 23530
REFRESH
_id: ObjectId('573a1390f29313caabcd4135')
plot: "Three men hammer on an anvil and pass a bottle of beer around."
genres: Array
runtime: 1
cast: Array
num_mflix_comments: 0
title: "Blacksmith Scene"
fullplot: "A stationary camera looks at a large anvil with a blacksmith behind it…"
countries: Array
released: 1893-05-09T00:00:00.000+00:00
directors: Array
rated: "UNRATED"
awards: Object
lastupdated: "2015-08-26 00:03:50.133000000"
year: 1893
imdb: Object
type: "movie"
tomatoes: Object
_id: ObjectId('573a1390f29313caabcd42e8')
plot: "A group of bandits stage a brazen train hold-up, only to find a determ…"
genres: Array
runtime: 11
cast: Array
poster: "https://m.media-amazon.com/images/M/MV5BMTU3NjE5NzYtYTYyNS00MDVmLWIwYj…"
title: "The Great Train Robbery"
fullplot: "Among the earliest existing films in American cinema - notable as the …"
languages: Array
released: 1903-12-01T00:00:00.000+00:00
directors: Array
rated: "TV-G"
awards: Object
lastupdated: "2015-08-13 00:27:59.177000000"
year: 1903
imdb: Object
countries: Array
type: "movie"
>_MONGOSH

MongoDB

MongoDB: Database for Modern Apps

Cloud to Edge to On-Premise for your Workload

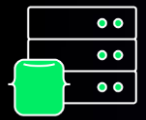
Document Model and MQL:
The fastest way to innovate and build REST APIs

Distributed Architecture and Global Scale
Freedom & flexibility

Atlas Developer Data Platform:
Unified experience for modern apps

Mission Critical and General Purpose

Expressive Query Language & Secondary Indexes



ACID Transactions

Enterprise Security & Management

Native data structures & Idiomatic drivers

Scale & Distribution

Laptop, your data center, multi-cloud

Traditional **Relational** Strengths

Additional **NoSQL** and **Big Data** Strengths and Innovations

The evolution of MongoDB

STACK 2022

2015

3.0 & 3.2

Doc-Level Concurrency
RAFT / Fast Failover
$lookup
Ops Manager
Compression
≤50 replicas
Aggregation ++
Encrypted and In-Memory storage engines
BI Connector
Compass

WiredTiger (Acquisition + Integration)

2016

3.4

Views Graph Processing Zones
++Aggregation ++ Auto-balancing ++ Linearizable Reads Decimal Intra-cluster Compression Log Redaction
Spark Connector ++ BI Connector ++

2017

3.6

Change Streams
Retryable Writes
Schema Validation Expressive $lookUp
Query Expressivity
Causal Consistency Consistent Sharded Secondary Reads
Query Advisor
End to End Compression
WiredTiger 1m+ Collections
MongoDB BI Connector ++
R Driver
Charts (post GA)
Atlas X-Region Replication Atlas Auto Storage Scaling

MongoDB Atlas

2018

4.0

Replica set transactions
Atlas global clusters
40% Faster Shard Migrations
Atlas HIPAA
Atlas LDAP
Atlas Audit
Atlas Enc. Storage Engine
Atlas Backup Snapshots
Type Conversions
Snapshot Reads
Non-Blocking Sec. Reads
SHA-2 & TLS 1.1+
Compass Agg Pipeline Builder
Compass Export to Code
Free Monitoring Cloud Service
Ops Manager K8s Beta

ACID (Transactions)

2019

4.2

Distributed Transactions
Client-Side Field Level Encryption
Materialized Views
Wildcard Indexes
Global PIT Reads
Large Transactions
Mutable Shard Key Values
Atlas Data Lake (Beta)
Atlas Auto Scaling (Beta)
Atlas Search (Beta)
Multi-CAs
Expressive Updates
Apache Kafka Connector
MongoDB Charts GA
Retryable Reads & Writes
New Index Builds
10x Faster stepDown
Storage Node Watchdog
Zstandard Compression

2020

4.4

Union
Custom Agg Expressions
Refinable Shard Keys
Compound Hashed Shard Keys
Mirrored Reads
Hedged Reads
Resumable Initial Sync
Time-Based Oplog Retention
Simultaneous Indexing
Hidden Indexes
Streaming Replication
Global Read/Write Concerns
Rust & Swift Drivers GA
TLS 1.3 & Faster Client Auth
OCSP Stapling
Kerberos Utility
Atlas Online Archive
Auto-Scaling
Schema Recommendations
AWS IAM Auth & Atlas x509
Federated Queries
Multi-cloud clusters

Multi-Cloud

2021-2022

5.x

Time Series collections
Clustered indexes
Window functions
Live resharding
Client-Side FLE KMIP & cloud KMS
Atlas Serverless (preview)
Atlas Search fast facets, function scores, synonyms
Long running snapshot reads
Sharded $lookup & $graphlookup
Majority write concern default
4x faster initial sync
Schema validation diagnostics
New MongoDB Shell GA
Resumable index builds
Rewritten Swift driver
Rewritten C# LINQ provider and .NET Analyzer
PyMongoArrow API
New accumulator operators
Charts on Data Lake
x509 certificate rotation
Auditing ++
Atlas K8S controller
Ops Manager migration wizard

Stable API & Rapid Releases

2022-2023

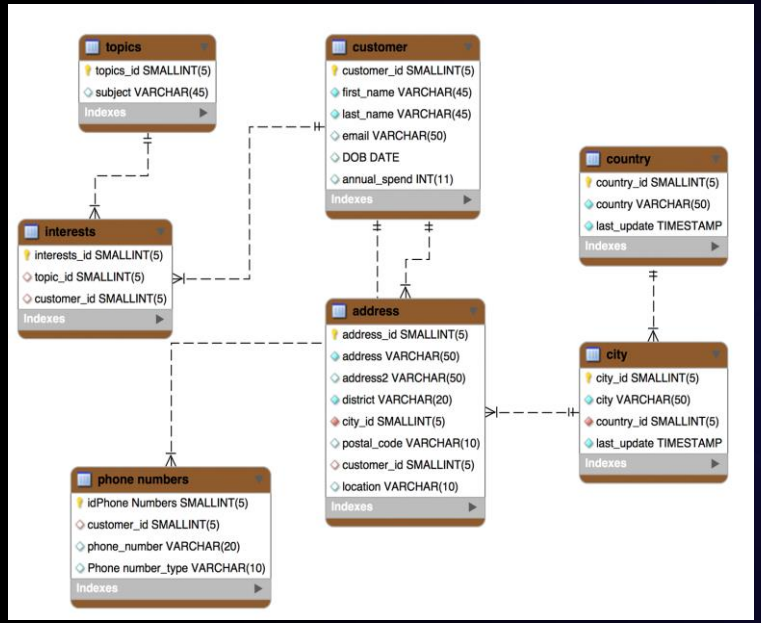
6.x

Queryable encryption (preview)
Atlas Serverless GA
Column store indexes
Cluster-to-cluster sync
Independent analytic nodes scaling
Flexible sync GA
Asymmetric sync preview
Kotlin and Flutter SDKs for Realm
$lookup perf improvements
Atlas CLI GA
Atlas Data API GA
Secondary index on measurements for time series
Time series read perf +
Atlas Data Lake with fully managed storage
Atlas Data Federation
Atlas Charts dashboard embedding
Atlas SQL interface
Encrypted audit log
Change streams improvements
Announcement of relational to MongoDB migrator

Developer Data Platform

GOVTECH SINGAPORE

Document model - intuitive and fast




```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
          51.5065752,-0.109081]}},
+    {...}
  ],
  "dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")
}
```

Tabular (Relational) Data Model

Related data split across multiple records and tables

Document Data Model

Related data contained in a single, rich document

Document model - adapt to changes flexibly

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
        51.5065752,-0.109081]}},
+   {...}
  ],
"dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")

}
```

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
        51.5065752,-0.109081]}},
+   {...}
  ],
  "phone" : [
    { "location" : "work",
      "number" : "+44-1234567890"},
+   {...}
  ],
  "dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")
}
```

Add new fields dynamically at runtime

Document data model

Naturally maps to objects in code

- Eliminates requirements to use ORMs
- Breaks down complex interdependencies between developer and DBAs teams

Represent data of any structure

- Polymorphic: each document can contain different fields
- Modify the schema at any time

Strongly typed for ease of processing

- Over 20 binary encoded JSON data types

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
          -0.109081, 51.5065752]}},
+   {...}
  ],
  "dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")
}
```

Access by idiomatic drivers in all major programming language

JSON Schema

Enforces strict schema structure over a complete collection for data governance & quality

- Builds on document validation introduced by restricting new content that can be added to a document
- Enforces presence, type, and values for document content, including nested array
- Simplifies application logic

Tunable

Enforce document structure, log warnings,
or allow complete schema flexibility

Queryable

Identify all existing documents that do not comply

Model and query data any way you need

All wrapped in a single API, giving a consistent experience for any workload

JSON Documents

Tabular

Key-Value

Text

Geospatial

Graph

File Storage

Events

SQL needed to insert a customer

```
import mysql.connector
from mysql.connector import errorcode

def addUser(connection, user):
    cursor = connection.cursor()

    customerInsert = (
        "INSERT INTO customer (first_name, last_name,
            email, "
        "DOB, annual_spend) VALUES "
        "(%(first)s, %(last)s, %(email)s, %(dob)s, %(spend)
            s)")

    customerData = {
        'first': user['name']['first'],
        'last': user['name']['second'],
        'email': user['email'],
        'dob': user['dob'],
        'spend': user['annualSpend']
    }

    cursor.execute(customerInsert, customerData)
    customerId = cursor.lastrowid

    cityQuery = ("SELECT city_id FROM city WHERE city = %(
        city)s")
    for address in user['address']:
        cursor.execute(cityQuery, {'city': address['city']}
            )
        city_id = cursor.fetchone()[0]

        addressInsert = (
            "INSERT INTO address (address, address2,
                district, "
            "city_id, postal_code, customer_id, location) "
            "VALUES (%(add)s, %(add2)s, %(dist)s, %(city)s
                , %(post)s, %(cust)s, %(loc)s)")
```

```
        addressData = {
            'add': address['number'],
            'add2': address['street'],
            'dist': address['state'],
            'city': city_id,
            'post': address['postalCode'],
            'cust': customerId,
            'loc':  address['location']
        }

        cursor.execute(addressInsert, addressData)

    topicQuery = ("SELECT topics_id FROM topics WHERE
        subject = %(subj)s")
    interestInsert = (
        "INSERT into interests (topic_id, customer_id) "
        "VALUES (%(topic)s, %(cust)s)")
```

```
    for interest in user['interests']:
        topicId = 0
        topicData = {
            'subj': interest['interest']
        }

        cursor.execute(topicQuery, topicData)
        row = cursor.fetchone()
        if row is None:
            topicInsert = ("INSERT INTO topics (subject)
                VALUES (%(subj)s)")
            cursor.execute(topicInsert, topicData)
            topicId = cursor.lastrowid
        else:
            topicId = row[0]

        interestData = {
            'topic': topicId,
            'cust': customerId
        }
        cursor.execute(interestInsert, interestData)

    phoneInsert = (
        "INSERT INTO `phone numbers` (customer_id,
            phone_number, `Phone number_type`) "
        "VALUES (%(cust)s, %(num)s, %(type)s)")
    for phoneNumber in user['phone']:
        phoneData = {
            'cust': customerId,
            'num': phoneNumber['number'],
            'type': phoneNumber['location']
        }
        cursor.execute(phoneInsert, phoneData)

    connection.commit()
    cursor.close()
    return customerId
```

CODE SNIPPET: HTTPS://GIT.IO/VPNXX

MongoDB requires just two lines of code

```
import mysql.connector
from mysql.connector import errorcode

def addUser(connection, user):
    cursor = connection.cursor()

    customerInsert = (
        "INSERT INTO customer (first_name, last_name,
            email, "
        "DOB, annual_spend) VALUES "
        "(%(first)s, %(last)s, %(email)s, %(dob)s, %(spend)
            s)")
```

```
    addressData = {
        'add': address['number'],
        'add2': address['street'],
        'dist': address['state'],
        'city': city_id,
        'post': address['postalCode'],
        'cust': customerId,
        'loc':  address['location']
    }

    cursor.execute(addressInsert, addressData)
```

```
    for interest in user['interests']:
        topicId = 0
        topicData = {
            'subj': interest['interest']
        }

        cursor.execute(topicQuery, topicData)
        row = cursor.fetchone()
        if row is None:
            topicInsert = ("INSERT INTO topics (subject)
                VALUES (%(subj)s)")
            cursor.execute(topicInsert, topicData)
```

```
def addUser(database, user):
  return database.customers.insert_one(user).inserted_id
```

```
    cityQuery = ("SELECT city_id FROM city WHERE city = %(
        city)s")
    for address in user['address']:
        cursor.execute(cityQuery, {'city': address['city']}
            )
        city_id = cursor.fetchone()[0]

        addressInsert = (
            "INSERT INTO address (address, address2,
                district, "
            "city_id, postal_code, customer_id, location) "
            "VALUES (%(add)s, %(add2)s, %(dist)s, %(city)s
                , %(post)s, %(cust)s, %(loc)s)")
```

```
            phone_number, `Phone Number_type`) "
        "VALUES (%(cust)s, %(num)s, %(type)s)")
    for phoneNumber in user['phone']:
        phoneData = {
            'cust': customerId,
            'num': phoneNumber['number'],
            'type': phoneNumber['location']
        }
        cursor.execute(phoneInsert, phoneData)

    connection.commit()
    cursor.close()
    return customerId
```

CODE SNIPPETS: SQL example and MongoDB example

Aggregations

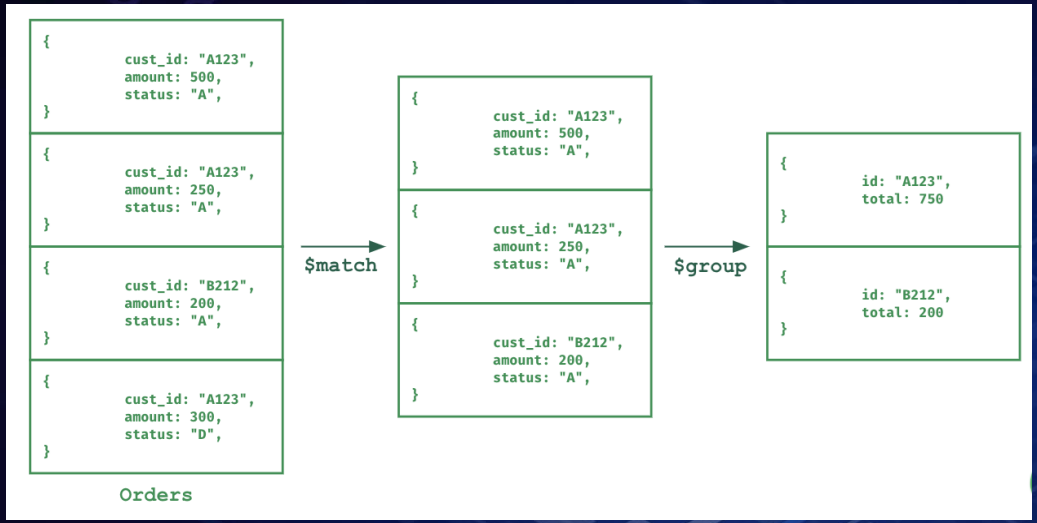
Advanced data processing pipeline for transformations
and analytics

Multiple stages

Similar to a unix pipe
Construct modular, composable processing pipelines

Rich Expressions

Example Aggregation Command on the Orders Collection:



$match stage
$group stage

```
db.orders.aggregate( [
     {$match: { status: "A" } },
       {$group: { _id: "$cust_id", total: { $sum: "$amount" } } }
] )
```

Aggregation features

A feature rich framework for data transformation and Analytics

Pipeline Stages

- $match
- $group
- $facet
- $geoNear
- $graphLookup
- $lookup
- $merge
- $project
- $search
- $sort
- $setWindowFields
- $unionWith
- $unwind
- ...and more

Operators

- Mathematical

  $add, $abs, $subtract, $multiply, $divide, $log, $log10, $stdDevPop, $stdDevSam, $avg, $sqrt, $pow, $sum, $zip, $convert, $round, etc.

- Array

  $push, $reduce, $reverseArray, $addToSet, $arrayElemAt, $slice, etc.

- Conditionals

  $and, $or, $eq, $lt, $lte, $gt, $gte, $cmp, $cond, $switch, $in, etc.

- Temporal

  Window Functions

  $dateAdd, $dateDiff, $dateSubtract, $dateTrunc

  $dateFromParts, $dateToParts, $dateFromString, $dateToString, $dayOfMonth, $isoWeek, $minute, $month, $year, etc.

- String

  $toUpper, $toLower, $substr, $strcasecmp, $concat, $split, etc.

- Literals

  $exp, $let, $literal, $map, $type, etc.

- Regex

  $regexFind, $regexMatch, etc

- Trigonometry

  $sin, $cos, $degreesToRadians, etc.

- Custom Aggregation Expressions

PRACTICAL MONGODB AGGREGATIONS EBOOK

GOVTECH SINGAPORE

Fully indexable

Fully featured secondary indexes - document optimized - extended beyond RDBMS experiences

Index Types

Primary Index
: Every Collection has a primary key index

Compound Index
: Index against multiple keys in the document

MultiKey Index
: Index into arrays

Wildcard Index
: Auto-index all matching fields, sub-documents & arrays

Text Indexes
: Support for text searches. Atlas Search offers Lucene-based inverted indexes

GeoSpatial Indexes
: 2d & 2dSphere indexes for spatial geometries

Clustered Indexes
: For time series collections, pre-sorted by timestamp for low latency querie

Index Features

TTL Indexes
: Single Field indexes, when expired delete the document

Unique Indexes
: Ensures value is not duplicated

Partial Indexes
: Expression based indexes, allowing indexes on subsets of data

Case Insensitive Indexes
: Supports text search using case insensitive search

Sparse Indexes
: Only index documents which have the given field

Transactional data guarantees

For many apps, single document transactions meet the majority of needs

```
  _id: 12345678
> name: Object
> address: Array
> phone: Array
  email: "john.doe.@mongodb.com"
  dob: 1966-07-30 01:00:00:000
v interests:Array
        0: "Cycling"
        1: "IoT"
```

MongoDB multi-doc ACID transactions

Multi-node transactional guarantees delivered at scale

- Multi-statement, familiar relational syntax
- Easy to add to any application
- Multiple documents in 1 or many collections and databases, across replica sets and sharded clusters

ACID guarantees

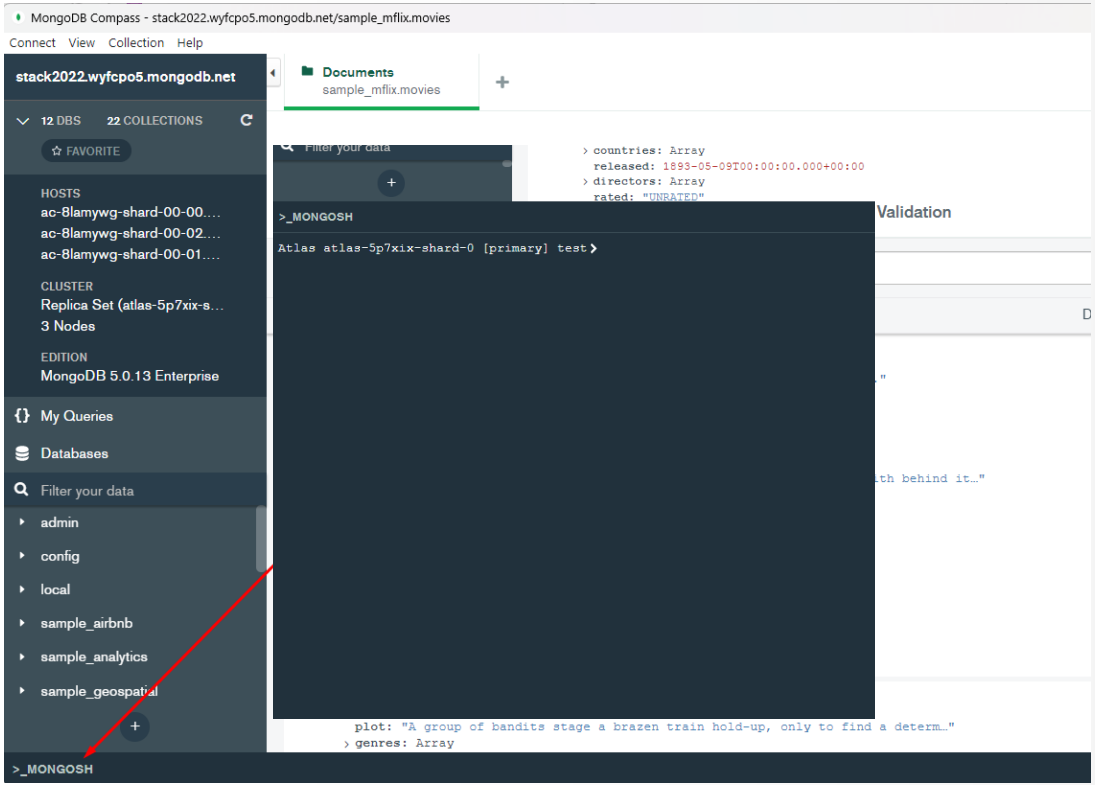
Snapshot isolation, all or nothing execution

```
with client.start_session() as s:
    s.start_transaction()
    collection_one.insert_one(doc_one, session=s)
    collection_two.insert_one(doc_two, session=s)
    s.commit_transaction()
```

Hands On:

MongoDB Flexible Schema and MQL

1. Activate mongosh in Compass

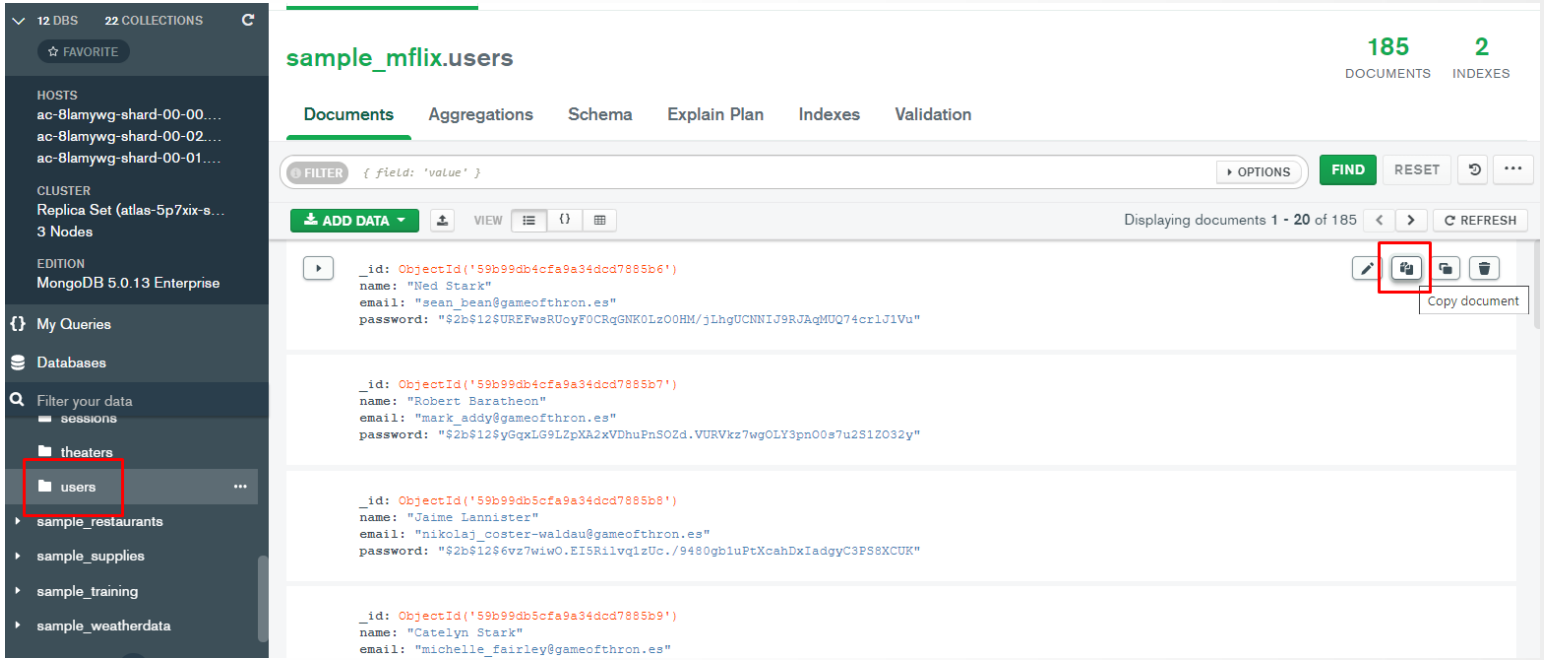


2. Switch context to “stackdb”

Note: “stackdb” does not exist but it is ok… trust me…

```
>_MONGOSH

> use stackdb

< 'switched to db stackdb'

Atlas atlas-5p7xix-shard-0 [primary] stackdb>
```

3. Copy a document from sample data



4. Insert copied document into “stackcoll”

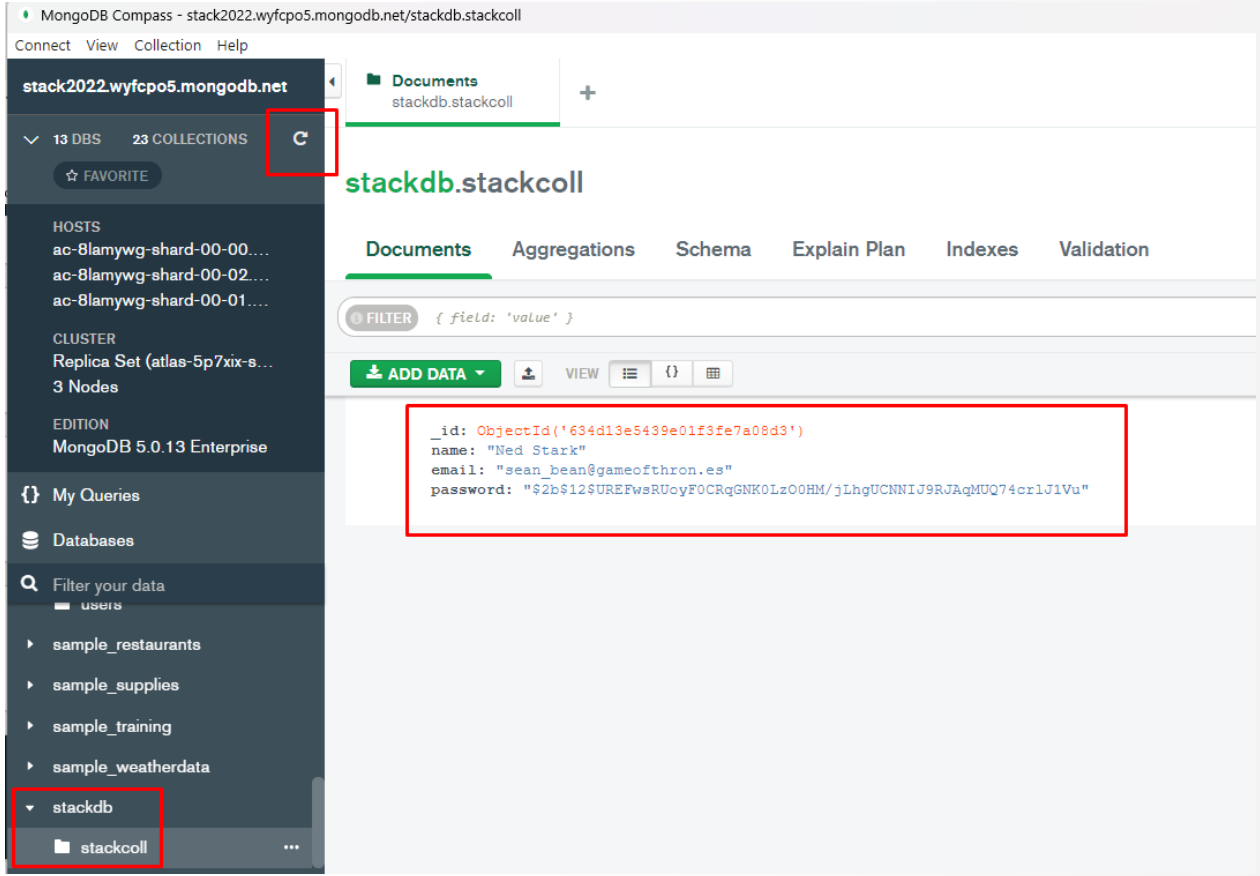
Note: “stackcoll” does not exist but it is ok… trust me again…

Remove this _id field

```
>_MONGOSH

> use stackdb
< 'switched to db stackdb'
Atlas atlas-5p7xix-shard-0 [primary] stackdb> db.stackcoll.insertOne({  "_id": {    "$oid": "59b99db4cfa9a34dcd7885b6"  },  "name": "Ned Stark",  "em
```

```
>_MONGOSH

> use stackdb
< 'switched to db stackdb'
> db.stackcoll.insertOne({ "name": "Ned Stark",  "email": "sean_bean@gameofthron.es",  "password": "$2b$12$UREFw
< { acknowledged: true,
    insertedId: ObjectId("634d13e5439e01f3fe7a08d3") }
Atlas atlas-5p7xix-shard-0 [primary] stackdb>
```

5. Refresh and check stackdb and stackcoll



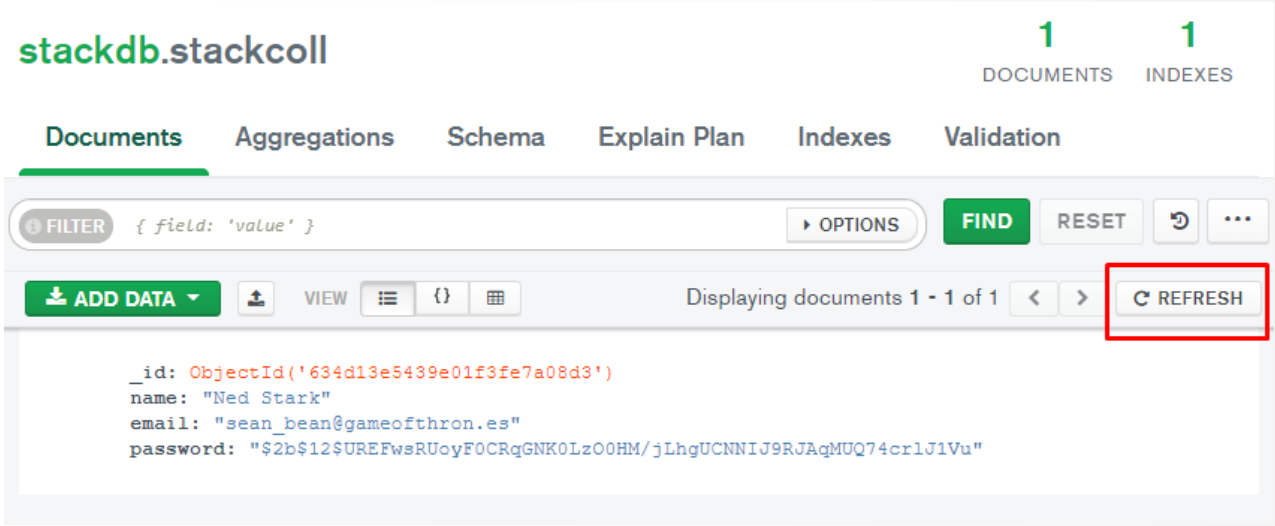
7. Update Ned's profile to include an array of interest

```
Atlas atlas-5p7xix-shard-0 [primary] stackdb> db.stackcoll.updateOne({name: "Ned Stark"}, {$set: {"interest": ["books", "food", "music"]}})
```

```
db.stackcoll.updateOne(
  {name: "Ned Stark"},
  {$set: {"interest": ["books", "food", "music"]}}
)
```

```
> db.stackcoll.updateOne({name: "Ned Stark"}, {$set: {"interest": ["books", "food", "music"]}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
Atlas atlas-5p7xix-shard-0 [primary] stackdb>
```

8. Refresh and check stackcoll

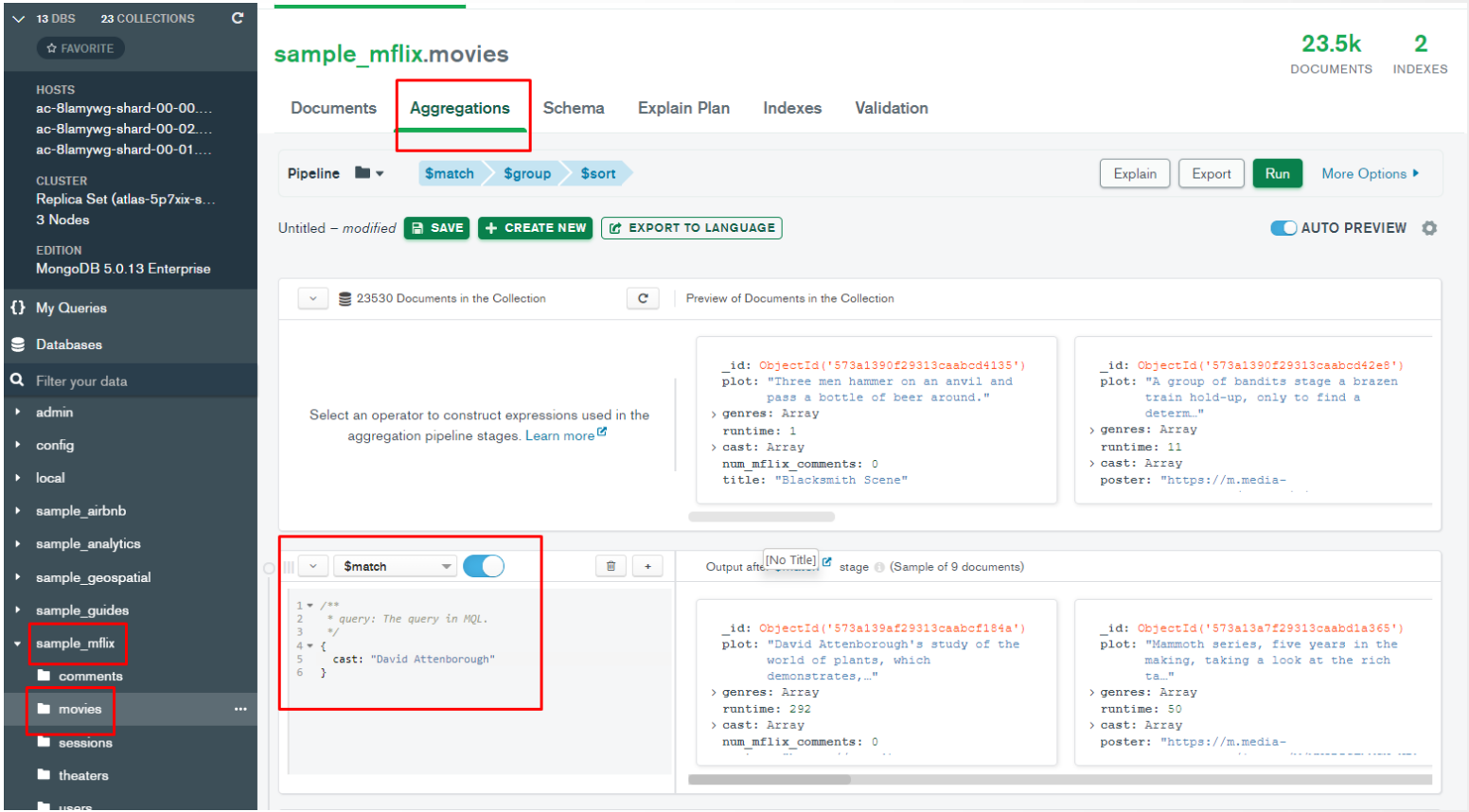



```
_id: ObjectId('634d13e5439e01f3fe7a08d3')
name: "Ned Stark"
email: "sean_bean@gameofthron.es"
password: "$2b$12$UREFwsRUoyF0CRqGNK0LzO0HM/jLhgUCNNIJ9RJAqMUQ74crlJ1Vu"
interest: Array
  0: "books"
  1: "food"
  2: "music"
```

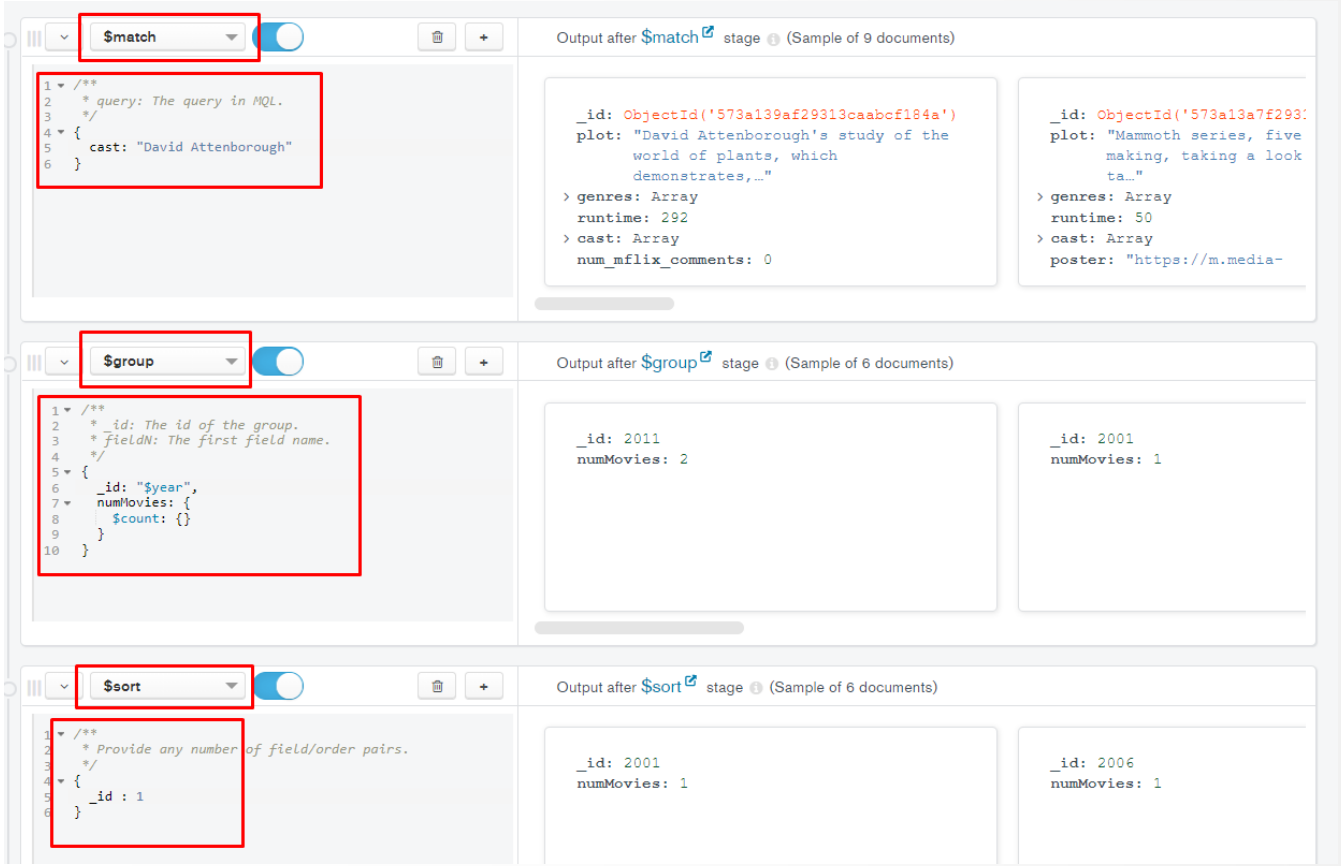
9: Query for movies featuring “David Attenborough”, made after 2010 and return only the title

```
> use sample_mflix
< 'switched to db sample_mflix'
> db.movies.find({cast: "David Attenborough", year: {$gt: 2010}}, {title: 1, _id: 0})
< { title: 'Madagascar' }
  { title: 'Frozen Planet' }
  { title: 'Bjèrk: Biophilia Live' }
Atlas atlas-5p7xix-shard-0 [primary] sample_mflix>
```

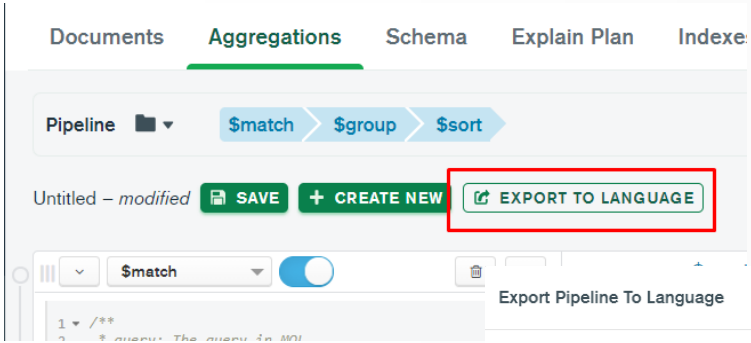
10. Find the number of movies featuring “David Attenborough” per year, sorted by year in ascending order (1)



10. Find the number of movies featuring “David Attenborough” per year, sorted by year in ascending order (2)



10. Find the number of movies featuring “David Attenborough” per year, sorted by year in ascending order (3)



Export Pipeline To Language

My Pipeline:

```
[{
    $match: {
        cast: 'David Attenborough'
    }
}, {
    $group: {
        _id: '$year',
        numMovies: {
            $count: {}
        }
    }
}, {
    $sort: {
        _id: 1
    }
}]
```

Export Pipeline To: PYTHON 3

```
[
    {
        '$match': {
            'cast': 'David Attenborough'
        }
    }, {
        '$group': {
            '_id': '$year',
            'numMovies': {
                '$count': {}
            }
        }
    }, {
        '$sort': {
            '_id': 1
        }
    }
]
```

Include Import Statements

Include Driver Syntax

10. Find the number of movies featuring “David Attenborough” per year, sorted by year in ascending order (4)

```
> db.movies.aggregate([{
   $match: {
    cast: 'David Attenborough'
   }
  }, {
   $group: {
    _id: '$year',
    numMovies: {
     $count: {}
    }
   }
  }, {
   $sort: {
    _id: 1
   }
  }])
< { _id: 2001, numMovies: 1 }
  { _id: 2006, numMovies: 1 }
  { _id: 2009, numMovies: 3 }
  { _id: 2011, numMovies: 2 }
  { _id: 2014, numMovies: 1 }
  { _id: '1995è', numMovies: 1 }
Atlas atlas-5p7xix-shard-0 [primary] sample_mflix>
```

MongoDB: HA and Scalability

MongoDB Replica Sets – self healing HA

Replica Set — 3 to 50 copies

Self-healing

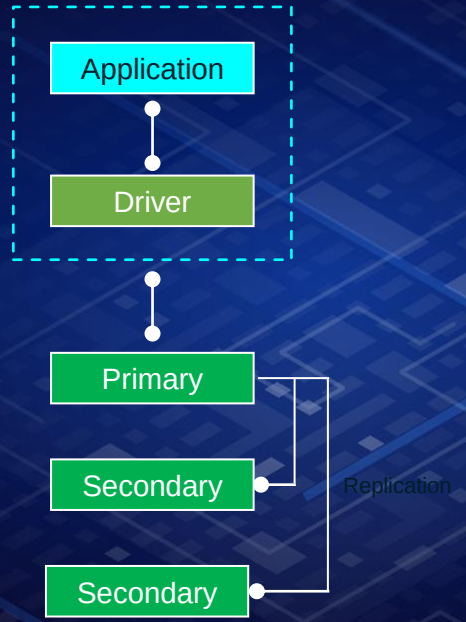
- Typical failover in 5 seconds or less
- Retryable reads and writes to catch temporary exceptions

Data center aware, tunable durability, and consistency

Addresses availability considerations:

- High Availability
- Disaster Recovery
- Maintenance

Workload Isolation: operational & analytics



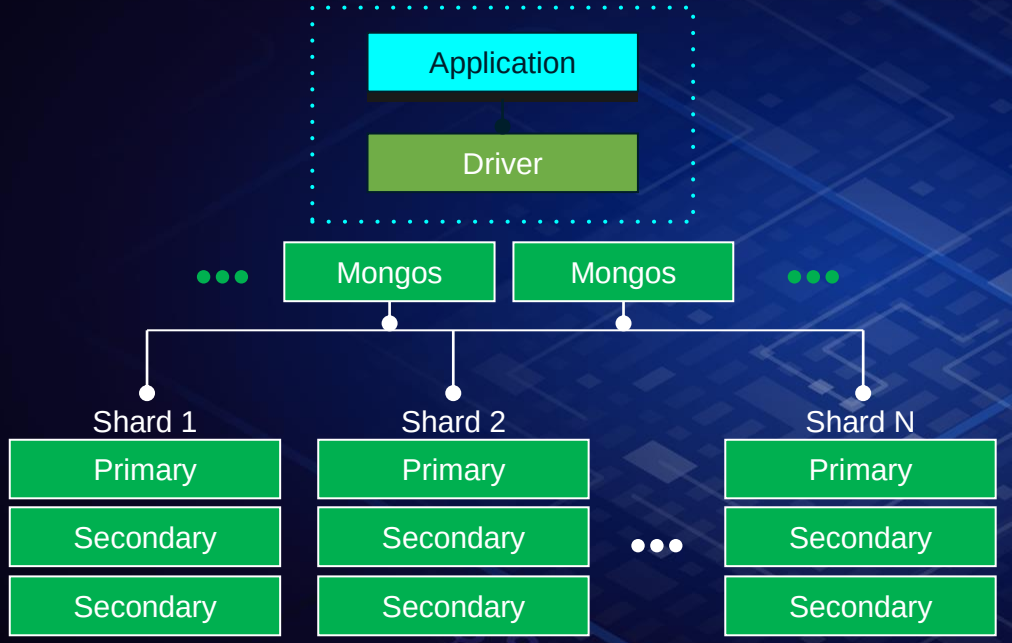
Automated sharding – scale with your modern apps

Horizontal scalability

Sharding

High availability

Replica sets



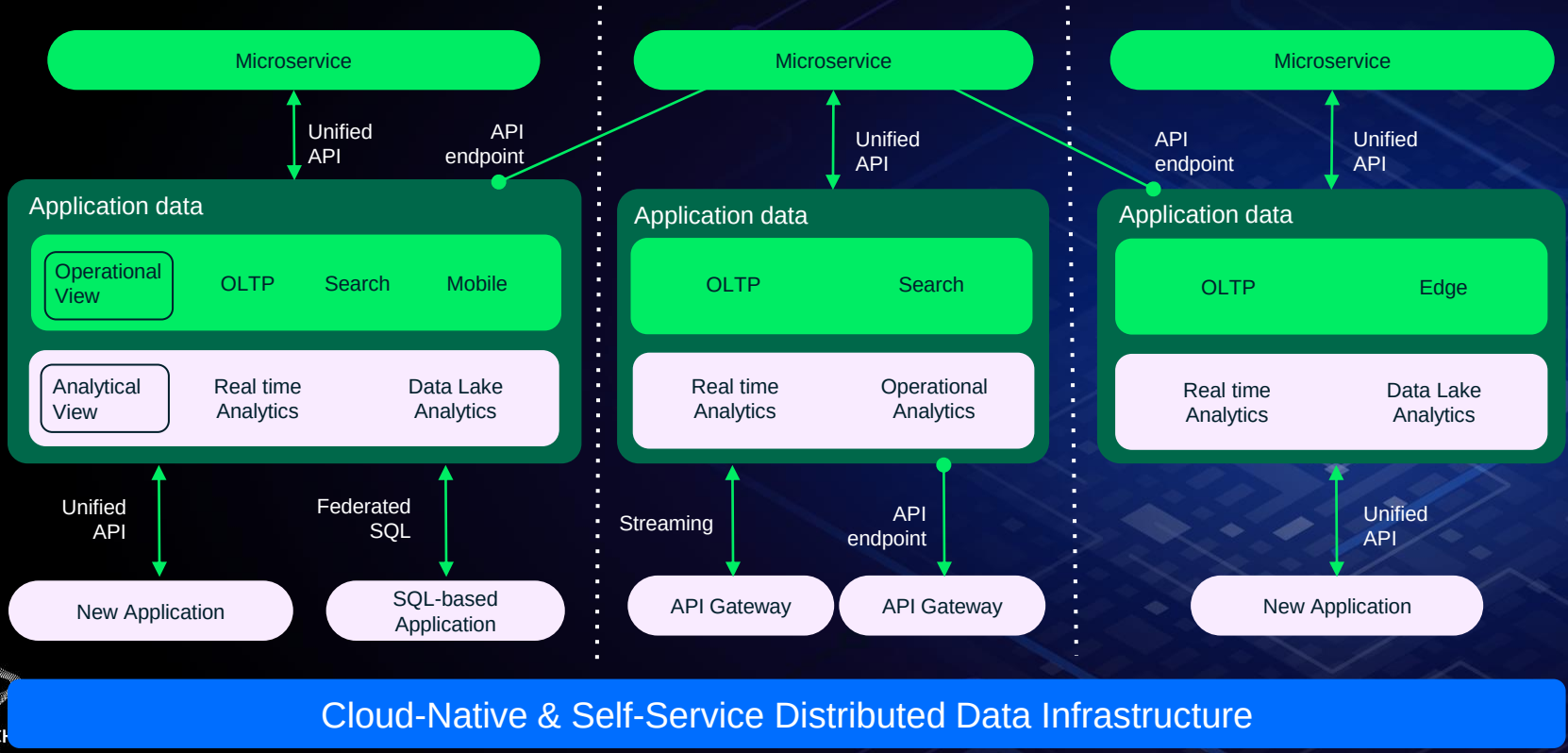
Cost effective at any scale

Native-Sharding for horizontal scale-out

- Automatically scale beyond the constraints of a single node
- Application transparent
- Scale, refine, rebalance, and reshard data at any time
- Unlike NoSQL systems that randomly spray data across a cluster, MongoDB exposes multiple data distribution policies (hashed, ranged, zoned) to optimize for query patterns and locality

Atlas Developer Data Platform

Our developer data platform allows you to support a wide range of application and analytics workloads



Atlas Search

Integrated and fully managed

Available with every 4.2+ cluster on Atlas

No need for a separate search architecture

Simple to use

Create indexes with the Atlas UI or API

$search queries are MongoDB aggregations

Rich search capabilities

Built on Apache Lucene

Fine-tune relevance with dozens of features

STACK 2022

Atlas Data Federation

Simplify Data Workflows

Seamlessly aggregate data and persist the results to their preferred storage location without complex data pipelines.

Power Real-Time Apps

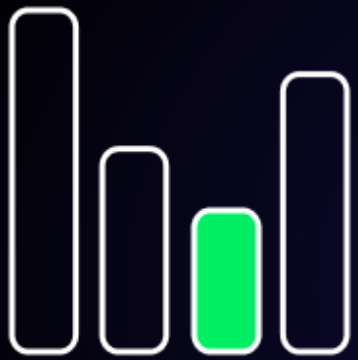
Federate queries across Atlas and AWS S3 without data movement or transformation.

Get Faster Insights

Convert MongoDB data to a columnar file format and output to S3 to be consumed by downstream teams.

GOVTECH
SINGAPORE

STACK 2022



Atlas Charts

Fast to visualize

Built for the document model

No ETL, data movement or duplication required

Easy to start and share

Integrated with MongoDB Atlas

Interactive dashboards with secure sharing

Powerful insights + experiences

No-code aggregation, runs on secondary nodes

Embed charts in apps via IFrame or SDK

GOVTECH
SINGAPORE

STACK 2022

Realm Mobile Database

Lightweight, fully-featured and object-oriented database embedded on device

Object-oriented database that developers find easy and intuitive to work with

Enables offline mode and on-device client-side compute

Live objects which means no need to write code to check the client-side data store for changes in user's data

Database can be encrypted with a single API call

GOVTECH SINGAPORE

Atlas Device Sync

Saves time and code

Automatic conflict resolution

No need to work with REST or an ORM

Optimized for mobile

Works for any mobile device or platform

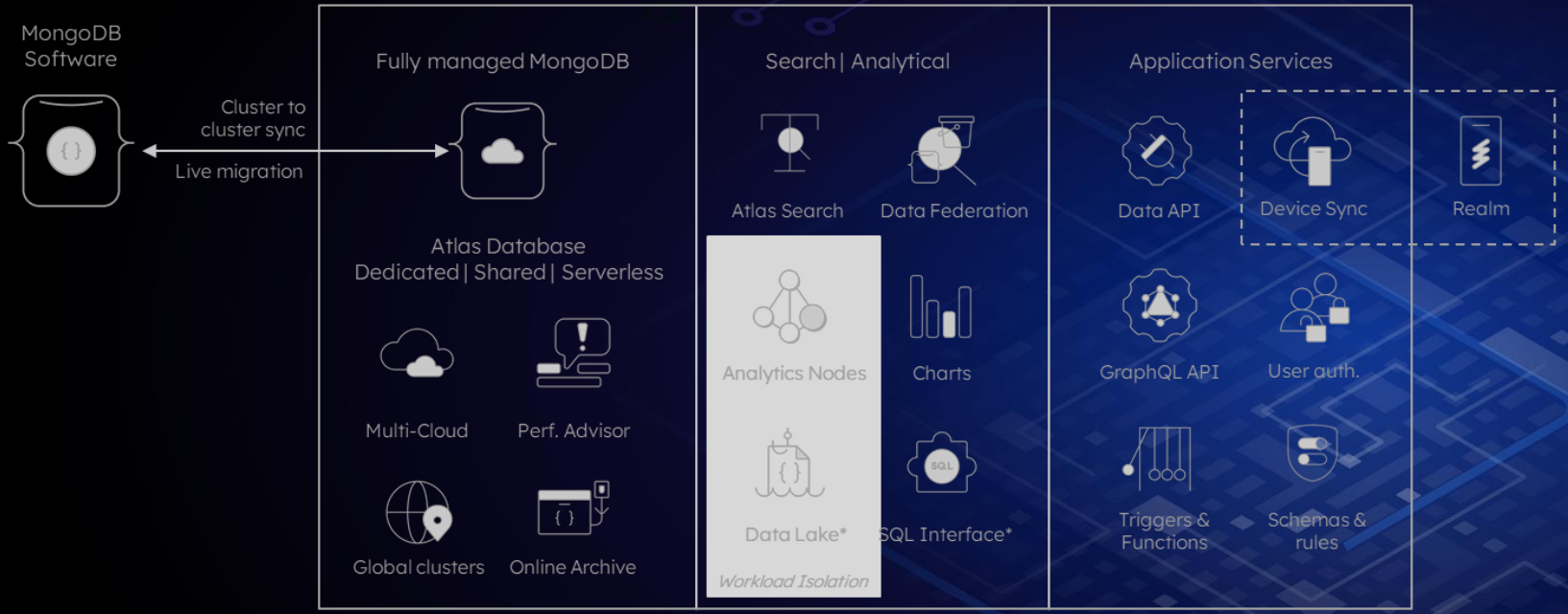
Resource-efficient, offline-first

Fully serverless

Native integration with MongoDB Atlas

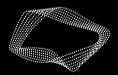
Scales with you as app usage changes

Atlas Developer Data Platform



Atlas Search

With a database you know exactly what you want. When you use search you are open to suggestions



Query vs Search

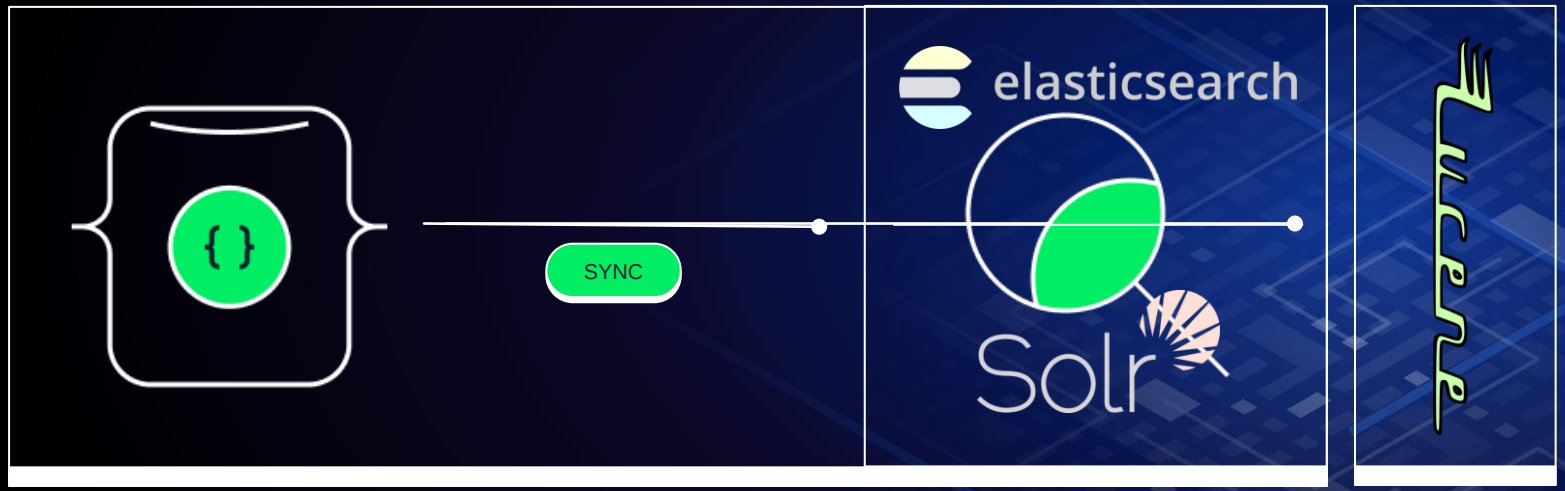
Query

```
find {title: "Batman Returns"}
//Title is exactly “Batman Returns”


sort {ratings: -1}
//Sorted by specific field, “ratings”
```

Search

```
$search {title: "batman returns"}
//Title has some variations of
“batman”, “returns” or both


//By default sorted according to
relevance score
```

Challenge with search

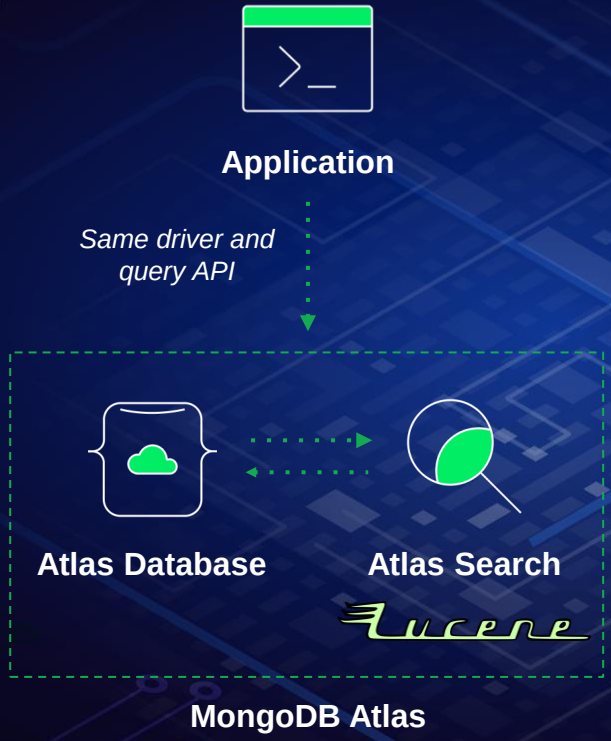


MongoDB Atlas: integrated DB and search

Higher developer productivity
Build database and search features using the same query API

Simplified data architecture
Automatic data synchronization, even as your data and schema changes

Fully managed platform
Get the security, performance, and reliability of Atlas



STACK 2022

Rich Search Features

Atlas Search offers many features to fine-tune your search results to help users find what they need, including:

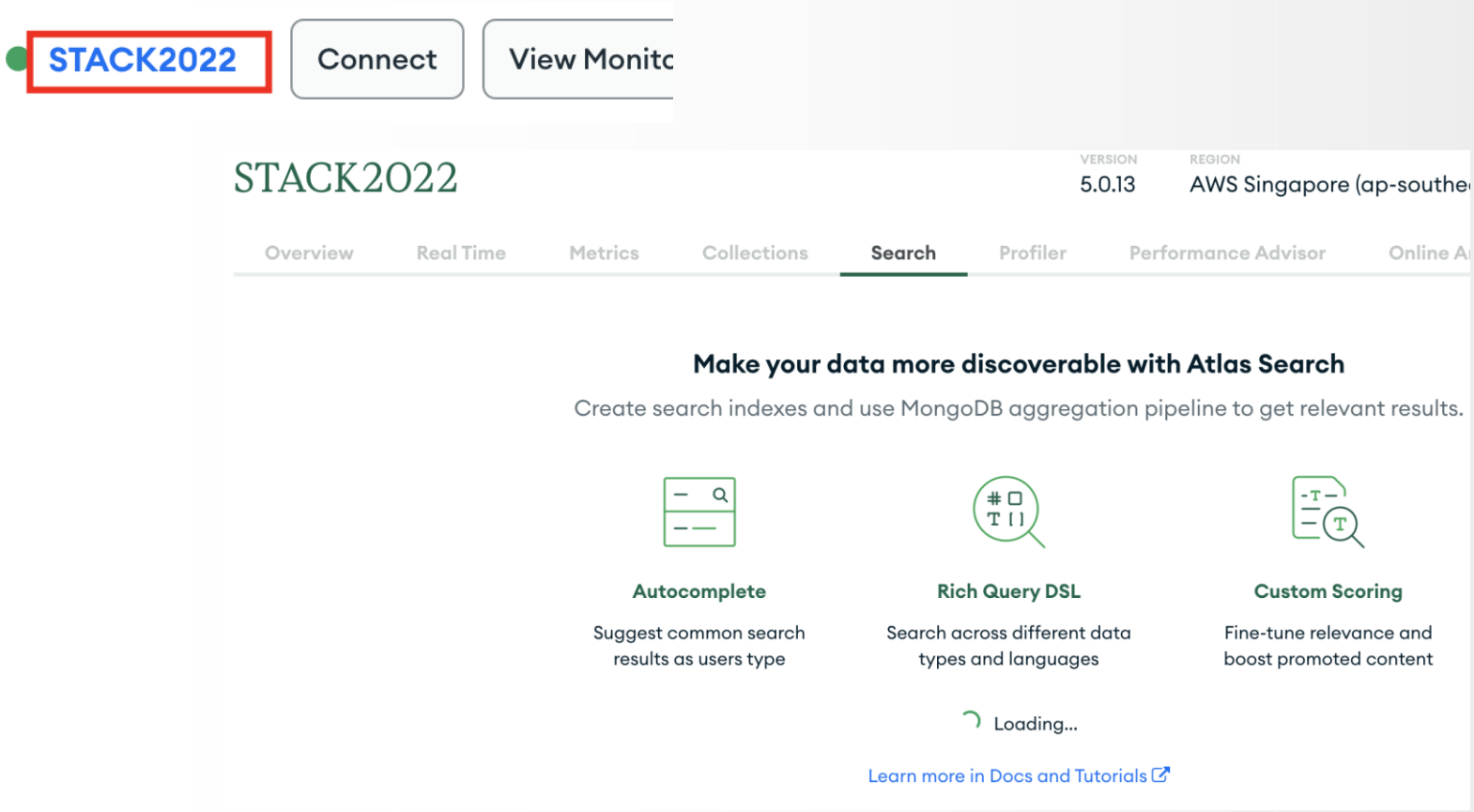
- Fuzzy search
- Autocomplete
- Highlighting
- Filters and facets
- moreLikeThis
- Multi-data type support (text, numerical, boolean, geospatial, etc.)
- Multi-language support
- Relevance based and custom scoring
- Synonyms

GOVTECH SINGAPORE

Hands On:

Atlas Search (Part 1)

1. Go to your cluster search page



2. Create search index

Make your data more discoverable with Atlas Search

Create search indexes and use MongoDB aggregation pipeline to get relevant results.

Autocomplete

Suggest common search results as users type

Rich Query DSL

Search across different data types and languages

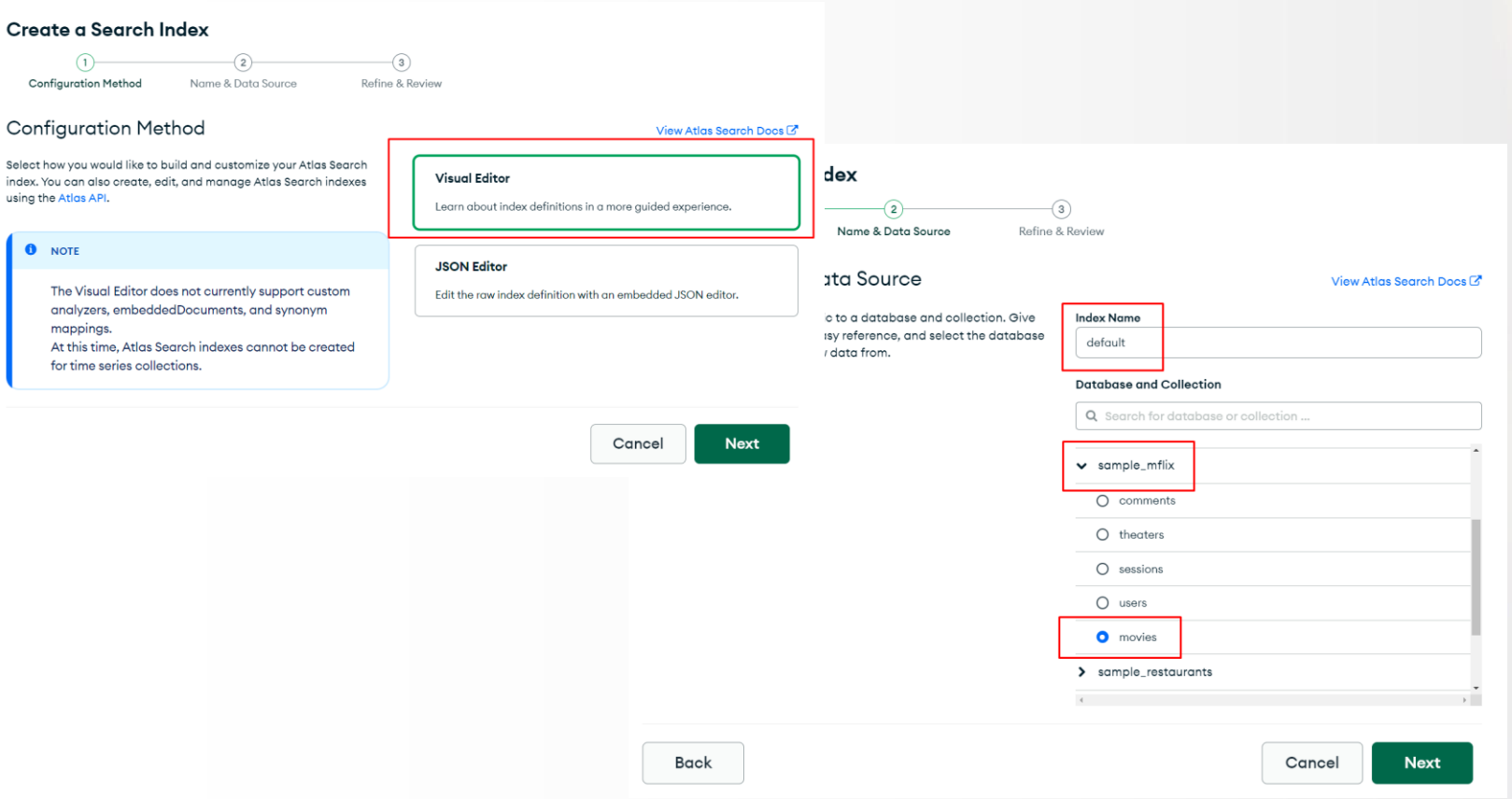
Custom Scoring

Fine-tune relevance and boost promoted content

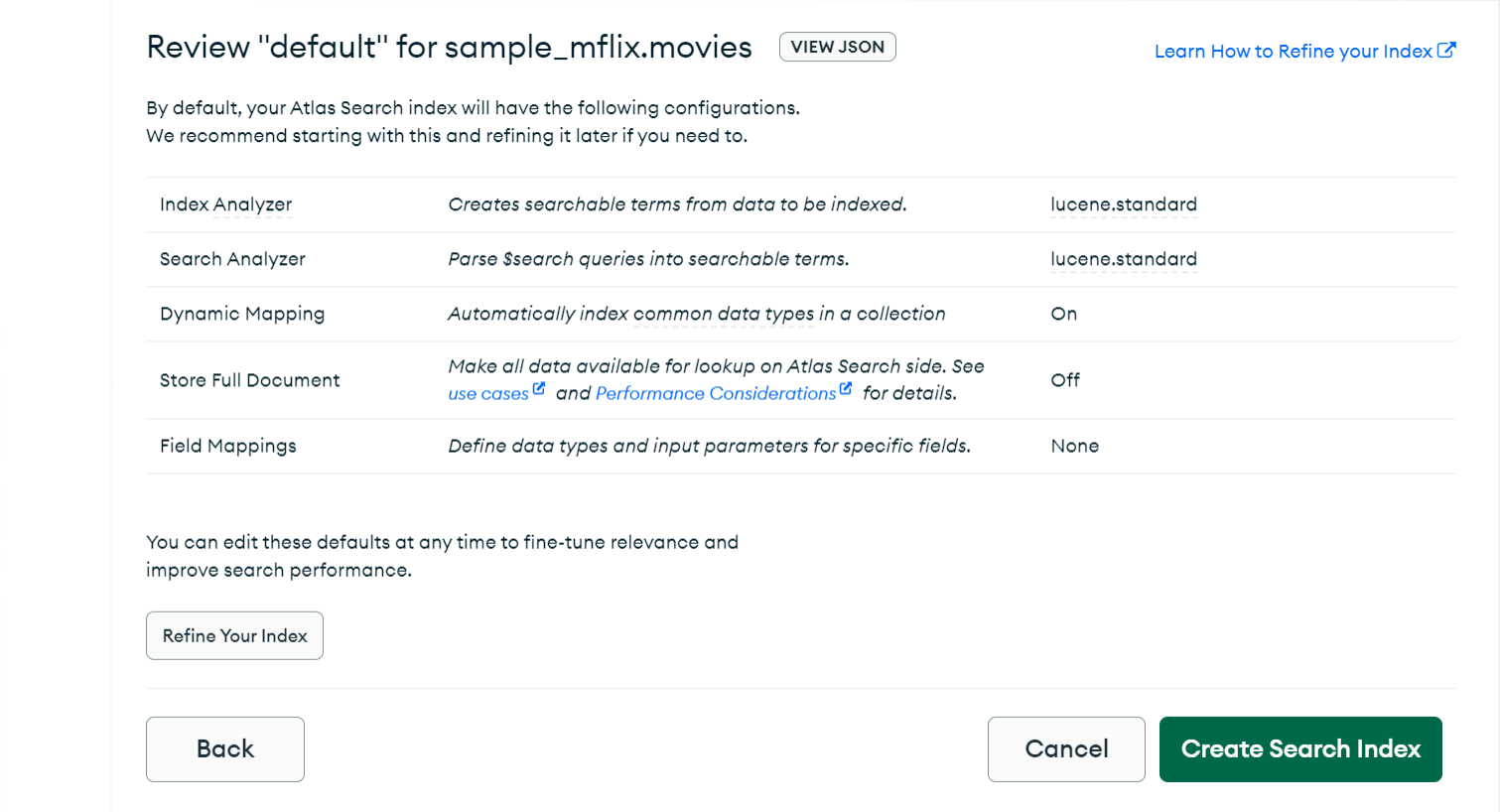
Create Search Index

Learn more in Docs and Tutorials

3. Select collection to index



4. Create search index (for real)

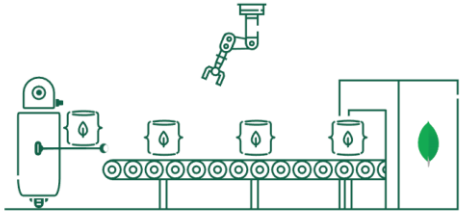


5. Wait

You're All Set!

Your Search index build is now in progress. You will receive an alert once it is complete

Indexes may take a few minutes or longer to build, depending on the size of your collection. You can see the status of your index in the **Search** tab.



Close

sample_mflix.movies

Name	Index Fields	Status	Size	Documents	Actions
default	[dynamic]	ACTIVE View status details	**Primary Node:** 42.94MB	**Primary Node:** 23,530 (100%) indexed of 23,530 total	QUERY •••

6. Search

sample_mflix.movies

Name	Index Fields	Status	Size	Documents	Actions
default	[dynamic]	ACTIVE View status details	Primary Node: 42.94MB	Primary Node: 23,530 (100%) indexed of 23,530 total	QUERY ···

Search Tester

Enter a term below to run a simple $search query on these sample documents from **sample_mflix.movies** using your search index and see the top 10 documents that are returned ranked by score. The Search Tester is limited to wildcard path queries. To run more complex queries, visit the aggregation pipeline.

batman RESET Search View Query Syntax

This search took 0.36 seconds

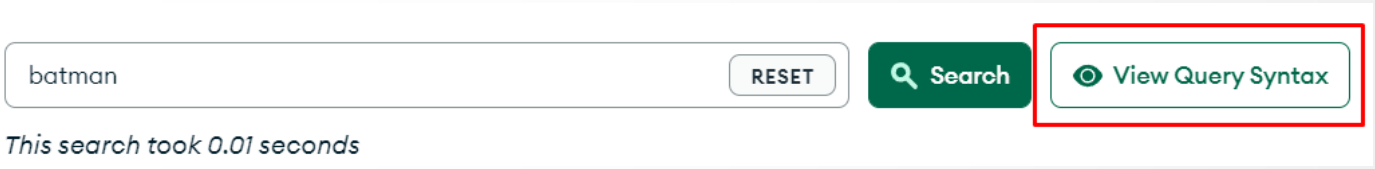
SCORE: 17.312782287597656 _id: "573a139af29313caabcf0774"

```
fullplot: "Batman and Robin are back working side-by-side to stop the villains of..."
▸ imdb: Object
year: 1997
```

SCORE: 15.905843734741211 _id: "573a1399f29313caabced8d4"

```
plot: "Batman is wrongly implicated in a series of murders of mob bosses actu..."
▸ genres: Array
runtime: 76
```

7. View query syntax



View Query Syntax

To execute the full $search query, connect to the mongo shell and copy the code below into an aggregation pipeline. You can also export the code to a programming language.

MongoDB API | Export to Language

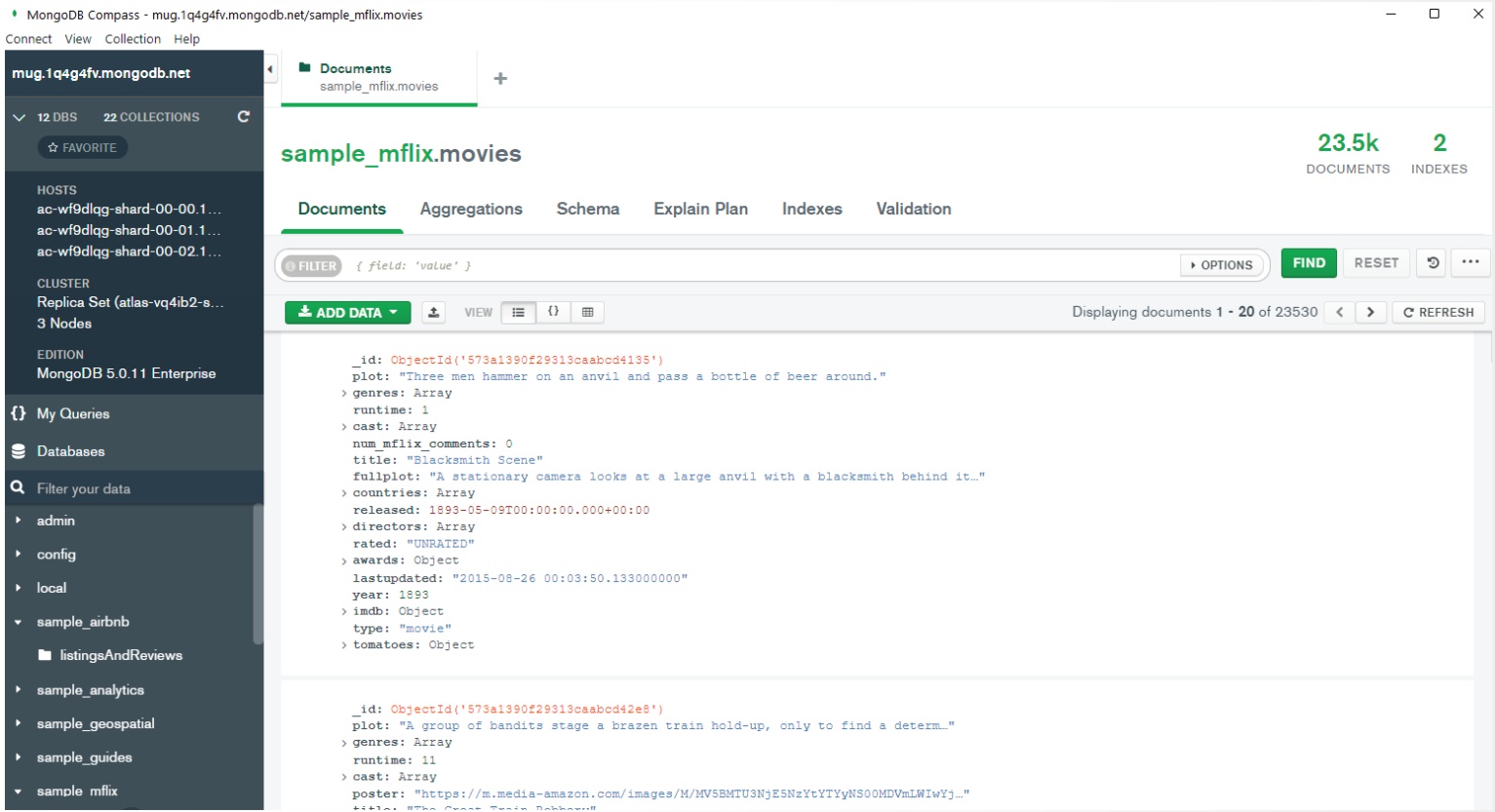
```
[
  {
    $search: {
      index: 'default',
      text: {
        query: 'batman',
        path: {
          'wildcard': '*'
        }
      }
    }
  }
]
```

STACK 2022

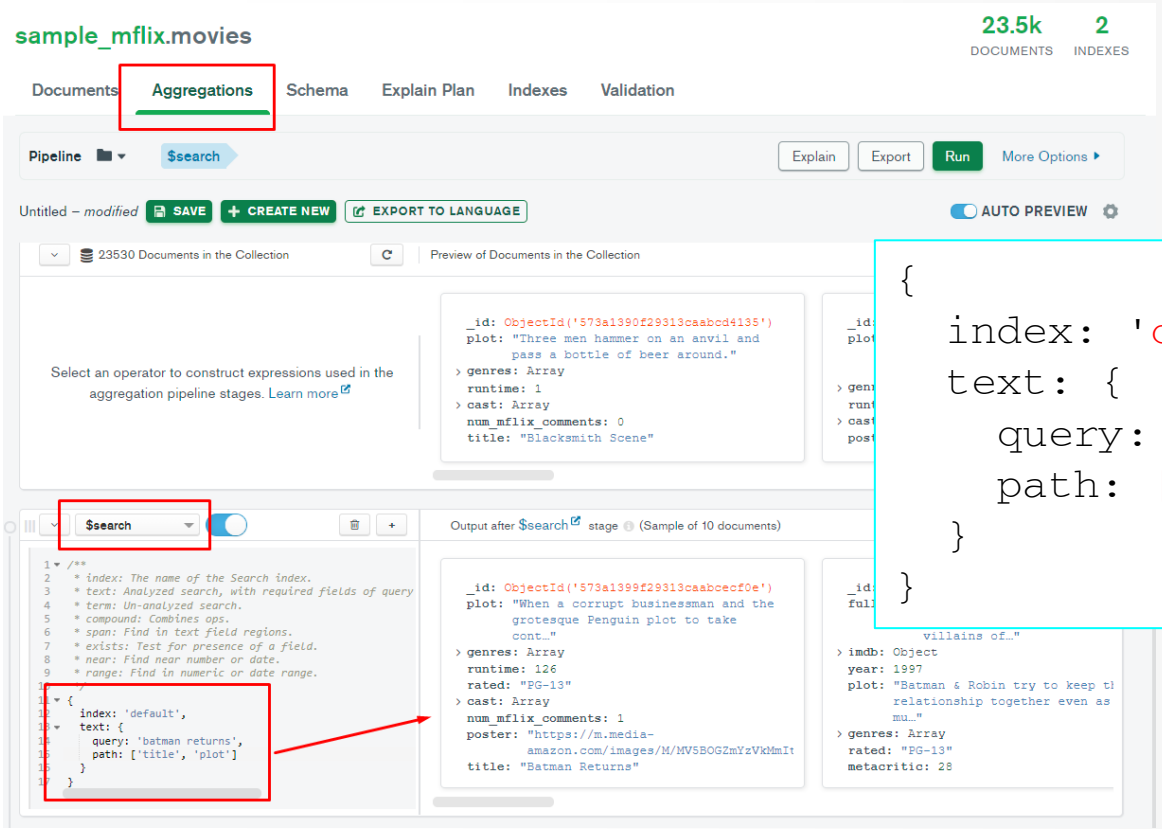
Hands On:

Atlas Search (Part 2)

1. Connect to cluster with Compass



2. Let's try a simple search query

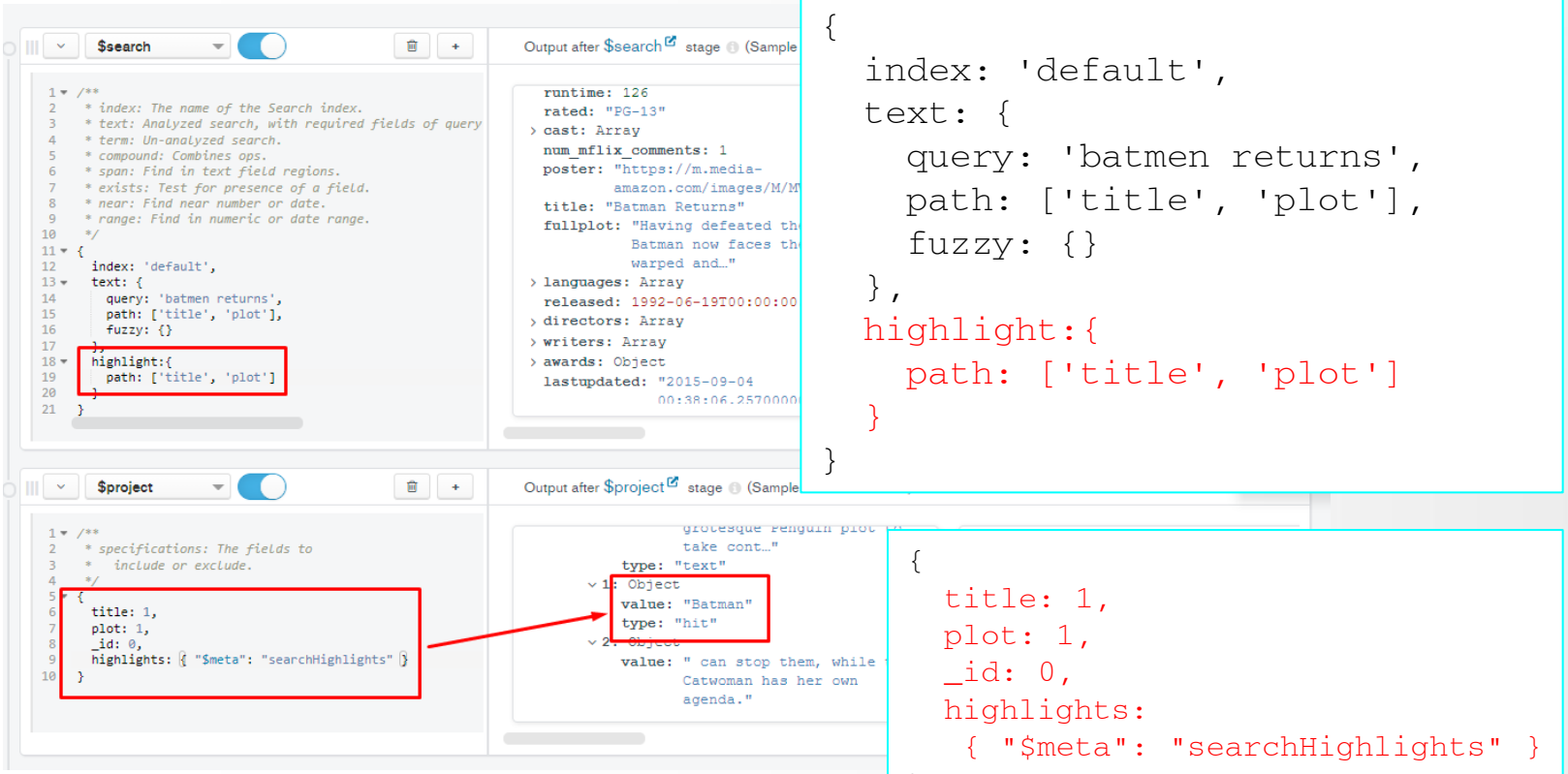



```
{
   index: 'default',
   text: {
      query: 'batman returns',
      path: ['title', 'plot']
   }
}
```

3. Add autocorrect to the search


```
{
  index: 'default',
  text: {
    query: 'batmen returns',
    path: ['title', 'plot'],
    fuzzy: {}
  }
}
```

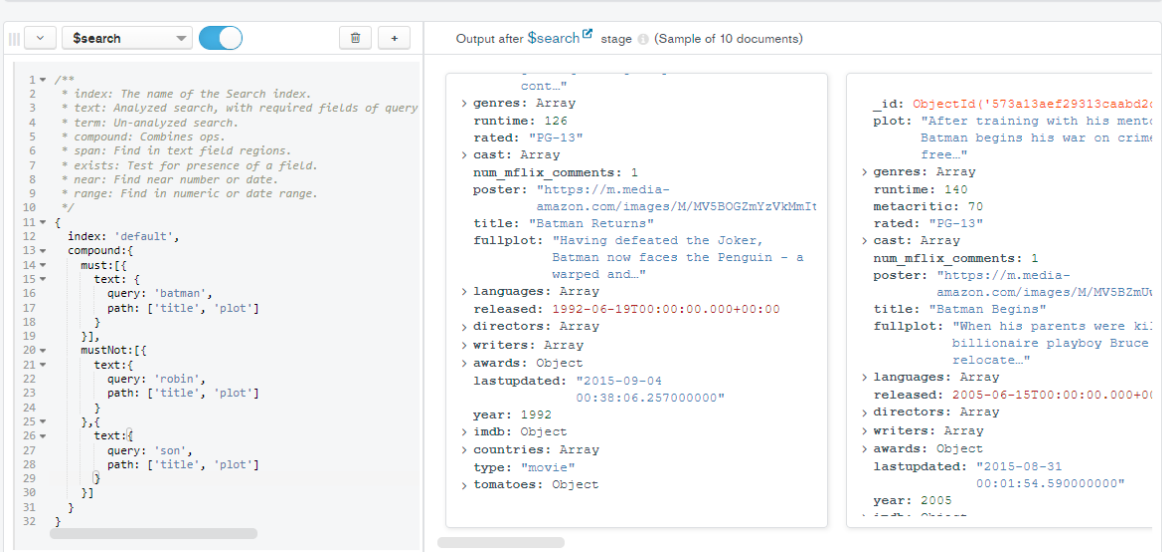
4. Now we add keyword highlighting




```
{
  index: 'default',
  text: {
    query: 'batmen returns',
    path: ['title', 'plot'],
    fuzzy: {}
  },
  highlight:{
    path: ['title', 'plot']
  }
}
```

```
{
  title: 1,
  plot: 1,
  _id: 0,
  highlights:
    { "$meta": "searchHighlights" }
}
```

5. I like Batman but not Robin




```
{
  index: 'default',
  compound:{
    must:[{
      text: {
        query: 'batman',
        path:
          ['title', 'plot']
      }
    }],
    mustNot:[{
      text:{
        query: 'robin',
        path:
          ['title', 'plot']
      }
    },{
      text:{
        query: 'son',
        path:
          ['title', 'plot']
      }
    }]
  }
}
```

STACK 2022

Thank You for Attending

We would love your feedback

Scan the QR code on the right handside to provide us with your feedback.

Your feedback and comments will help us improve our future events, making them more relevant to you.

GOVTECH
SINGAPORE

STACK 2022

MongoDB Day Singapore

MongoDB Days is coming to the sunny shores of Singapore! Join us on **Wednesday, 23 Nov 2022** at Pan Pacific Singapore for an exciting full day of presentations designed to provide MongoDB users with technical education! Hear from leading customers on their use case, engage with our partners and MongoDB team to learn how our developer data platform can help you build and deploy mission-critical applications at scale.

We hope to see you at the event!

GOVTECH SINGAPORE

Event Details

Date: Wednesday, 23 November 2022

Time: 9:00 AM - 5:00 PM

Venue: Pan Pacific Hotel

Scan the QR Code to find out more!

STACK 2022

THANK YOU

Brought to you by
Government Technology Agency