WELCOME

# MongoDB

## Atlas Workshop
## go/mongodbatlas

**Bhavik Bhatt**
Solutions Architect

**Soheyl Rafi**
Solutions Architect

# Terraform

Tony Wands

# Repo

github.com/chaitanyavaranasi-mdb/AtlasWorkshop/tree/main

or

shorturl.at/xHX78

shorturl.at/xHX78

# Exercise 0

Create MongoDB cluster -> M0
(if not already done)
Add sample data
Enable Network Access
Create User w/ read & write access

shorturl.at/xHX78

# Mongo Exercise 1

## Rich Queries - MQL

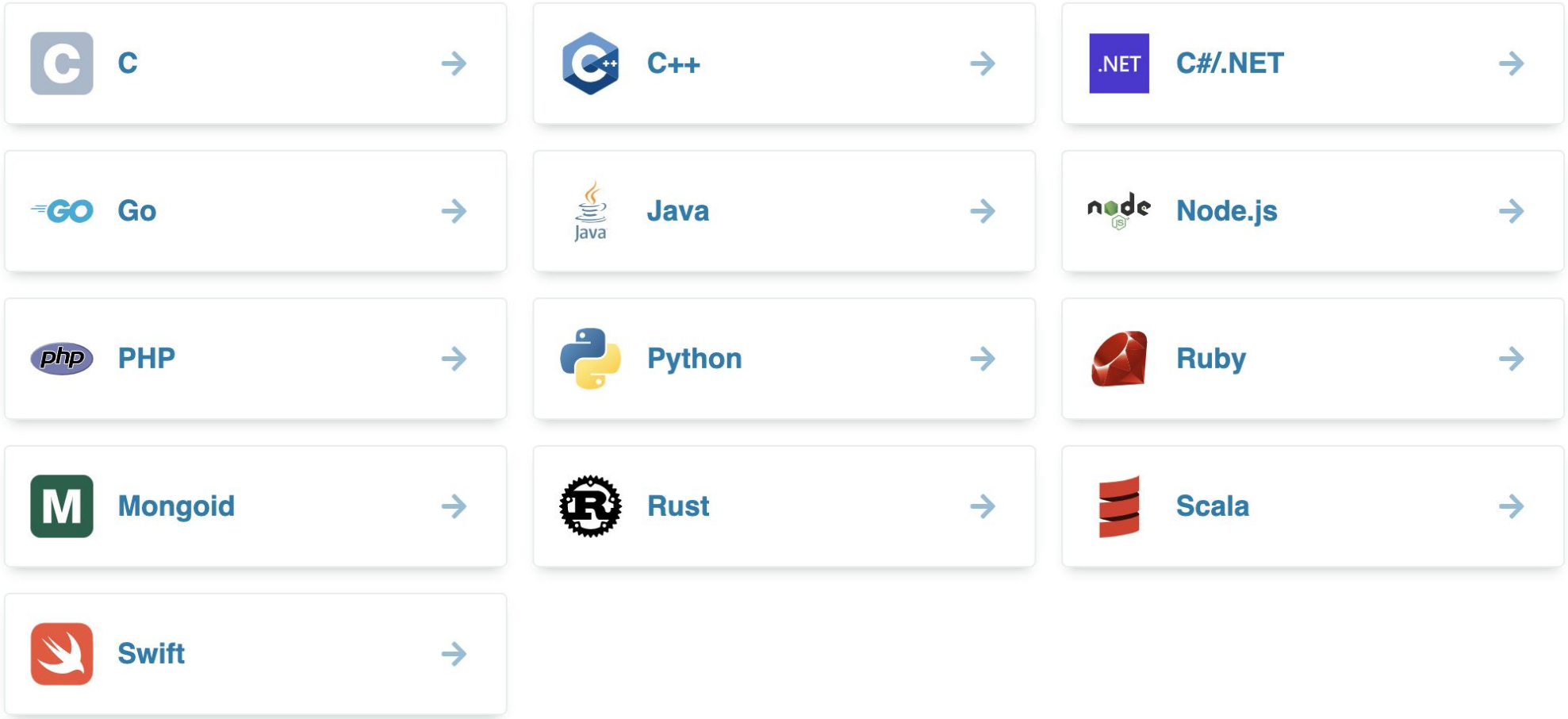
# MongoDB Query API: rich and expressive

| | |
|---|---|
| **Expressive queries** | • Find anyone with phone # “1-212…”<br>• Check if the person with number “555…” is on the “do not call” list |
| **Geospatial** | • Find the best offer for the customer at geo coordinates of 42nd St. and 6th Ave |
| **Text search** | • Find all tweets that mention the firm within the last 2 days |
| **Aggregation** | • Count and sort number of customers by city, compute min, max, and average spend |
| **Native binary JSON support** | • Add an additional phone number to Mark Smith’s record without rewriting the document<br>• Update just 2 phone numbers out of 10<br>• Sort on the modified date |
| **JOIN ($lookup)** | • Query for all San Francisco residences, lookup their transactions, and sum the amount by person |
| **Graph queries ($graphLookup)** | • Query for all people within 3 degrees of separation from Mark |

## MongoDB

```
{
     customer_id : 1,
     first_name : "Mark",
     last_name : "Smith",
     city : "San Francisco",
     phones: [
         {
            number : "1-212-777-1212",
            type : "work"
         },
         {
            number : "1-212-777-1213",
            type : "cell"
         }
     ]
}
```

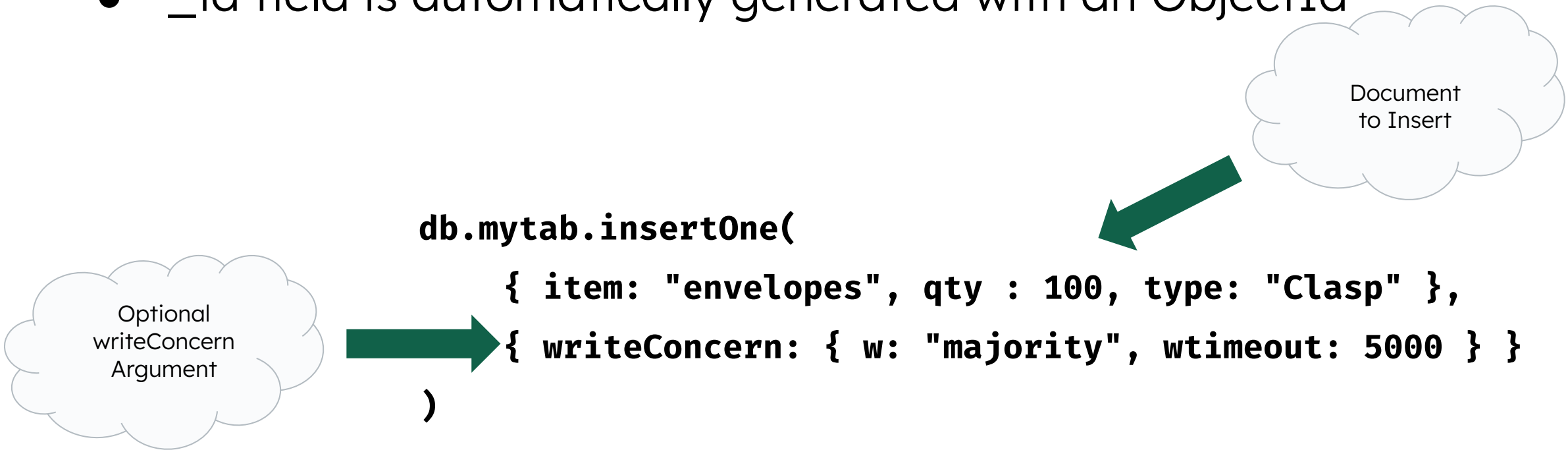
# Intuitive: client drivers




- Common CRUD capabilities but idiomatic to each language
- Uniform HA & Failover capabilities across all

# insertOne()

- Inserts document(s) into a collection
- Can optionally define the writeConcern of the operation
- _id field is automatically generated with an ObjectId



# find() and findOne()

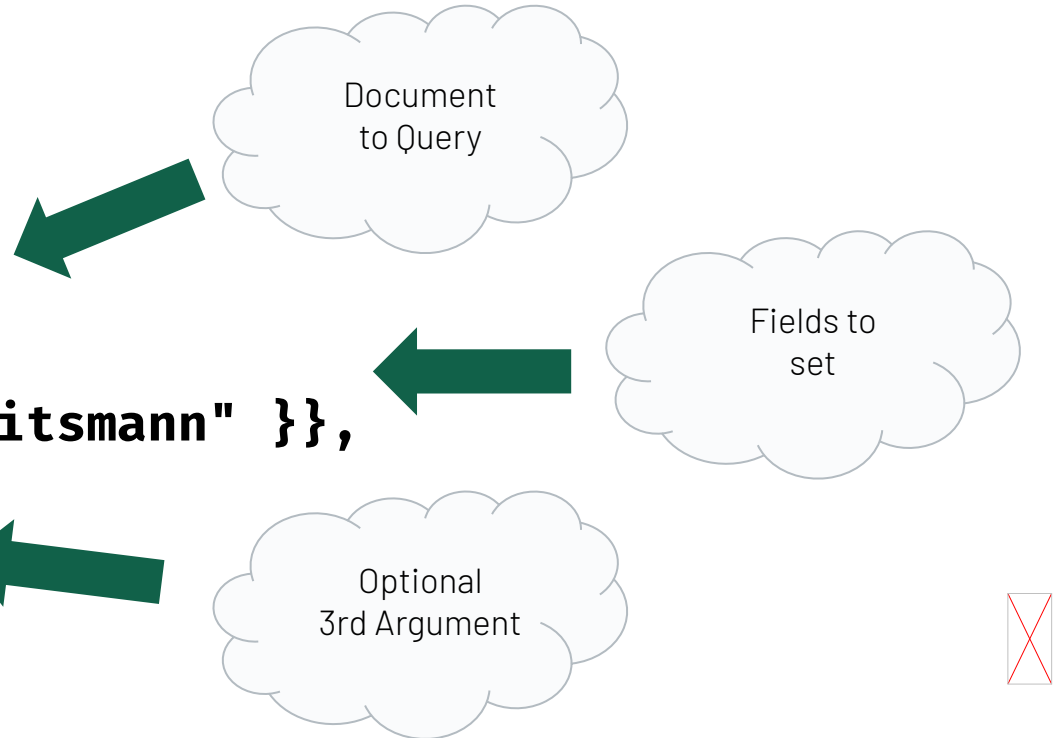
- Get documents from the collection (table)
- Can filter by content (query) and by what is returned (project)
- `findOne()` returns just the first document

```
db.mytab.find({x:1, y:2}, {a:1, b:1, c:1})
```

# updateMany() and updateOne()

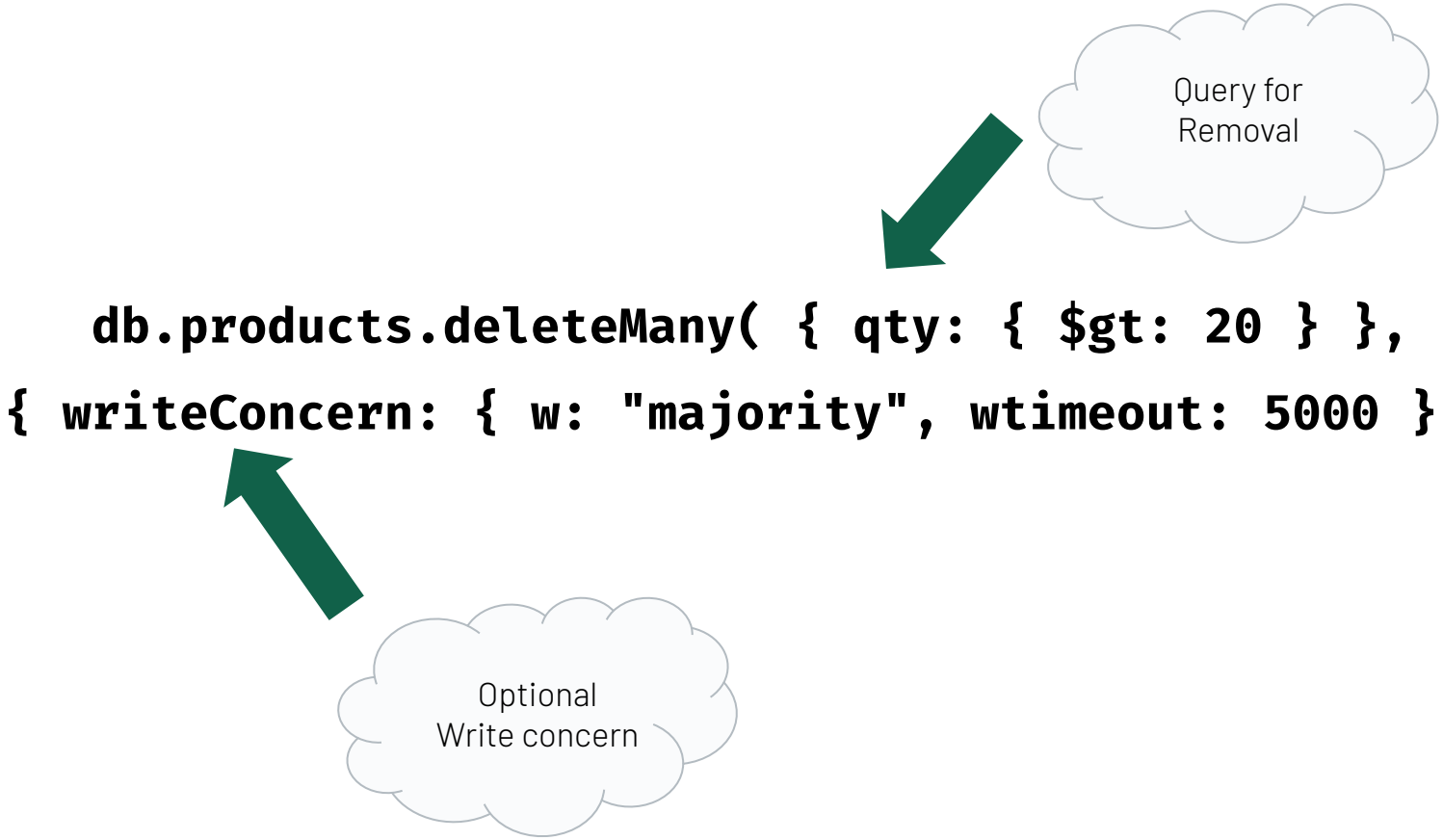
- Modifies existing document(s) in a collection
- Use with a field update operator (MQL)
- Query for the documents you want to update
- Can optionally upsert, multi-update, define writeConcern, etc.

```
db.books.updateOne(
    { author: “Tom Gleitsmann” },
    { $set: { author: "Thomas Gleitsmann" }},
 { multi: true }
)
```



# deleteMany()

- Deletes a document from the collection
- Takes an optional writeConcern argument



# Comparison Query Operators

| | | |
|---|---|---|
| $lt | : | Exists and less than |
| $lte | : | Exists and less than or equal to |
| $gt | : | Exists and greater than |
| $gte | : | Exists and greater than or equal to |
| $ne | : | Does not exist or does but not equal to |
| $in | : | Exists and in a set |
| $nin | : | Does not exist or not in a set |

```
{ age : { $gte : 21 }}
{ temp : { $gt: 10, $lt: 30 }}
```

# Field Update Operators (well some…)

| | | |
|---|---|---|
| $currentDate | : | Sets the value of a field to current date |
| $inc | : | Increments value |
| $min | : | Update if field is less than $min value |
| $max | : | Update if field is greater than $max value |
| $mul | : | Multiply field value |
| $set | : | Set value of field |
| $unset | : | Removes field from document |

```
{ $set : { bestDB : “MongoDB”}}
{ $mul: { amount : 5, ... } }
```

shorturl.at/xHX78

# Hands-On: Compass

# sample_airbnb.listingsAndReviews

5.6k DOCUMENTS  4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter {"address.market":"Hong Kong", "property_type" : "Apartment"} Reset Find </> More Options

ADD DATA EXPORT COLLECTION 1 – 20 of 444


```
_id: "10893326"
listing_url: "https://www.airbnb.com/rooms/10893326"
name: "Designer Apartment in Sheung Wan"
summary: "We live in a beautiful and spacious one bedroom apartment in the heart…"
space: ""
description: "We live in a beautiful and spacious one bedroom apartment in the heart…"
neighborhood_overview: "Often described as the new SOHO, there are chic coffee shops and inter…"
notes: ""
transit: "The area is well served by minibuses, City Bus, taxis, and the old sty…"
access: ""
interaction: ""
house_rules: ""
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "1"
maximum_nights: "1125"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
calendar_last_scraped: 2019-03-11T04:00:00.000+00:00
first_review: 2016-03-06T05:00:00.000+00:00
last_review: 2019-02-11T05:00:00.000+00:00
accommodates: 2
```


```
use sample_airbnb
db.listingsandreviews.find({"address.mar
ket":"Hong Kong","property_type" :
"Apartment"})
```

sample_airbnb.listingsAndReviews

# sample_airbnb.listingsAndReviews

5.6k DOCUMENTS  4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter { "bedrooms": {"$gte": 3}, "price": {"$gte": 1400, "$lte": 1500}}  Reset  Find  </>  More Options

ADD DATA  EXPORT COLLECTION  1 - 7 of 7


```
_id: "1176693"
listing_url: "https://www.airbnb.com/rooms/1176693"
name: "BEST REVIEWS*BEST MALLS*SAFE STAY*DIMSUM*CWB*MTR"
summary: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
space: "FACTS - located in the heart of one of Hong Kong's most bustling place…"
description: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
neighborhood_overview: "This enigmatic city of skyscrapers, ancient traditions and heavenly fo…"
notes: "Please note that not every neighbor is happy about the New Sharing Eco…"
transit: "HOW TO GET TO CAUSEWAY BAY: By Public transport: MTR: Island Line: Tin…"
access: "The entire apartment with all amenities is accessible for my guests. A…"
interaction: "I will respond to your questions within very short notice. My reservat…"
house_rules: "Early check-in and late check-out is possible for an additional fee. P…"
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1124"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
```


```
use sample_airbnb
db.listingsandreviews.find({ "bedrooms":
{"$gte": 3}, "price": {"$gte": 1400,
"$lte": 1500}})
```

# sample_airbnb.listingsAndReviews

5.6k DOCUMENTS  4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter { "amenities": { "$all": ["Wifi", "Kitchen"]}} Reset Find </> More Options

ADD DATA EXPORT COLLECTION 1 – 20 of 4767


```
_id: "10047964"
listing_url: "https://www.airbnb.com/rooms/10047964"
name: "Charming Flat in Downtown Moda"
summary: "Fully furnished 3+1 flat decorated with vintage style.  Located at the…"
space: "The apartment is composed of 1 big bedroom with double sized bed, a gu…"
description: "Fully furnished 3+1 flat decorated with vintage style.  Located at the…"
neighborhood_overview: "With its diversity Moda- Kadikoy is one of the most colorfull neighbou…"
notes: ""
transit: ""
access: ""
interaction: ""
house_rules: "Be and feel like your own home, with total respect and love..this woul…"
property_type: "House"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1125"
cancellation_policy: "flexible"
last_scraped: 2019-02-18T05:00:00.000+00:00
calendar_last_scraped: 2019-02-18T05:00:00.000+00:00
first_review: 2016-04-02T04:00:00.000+00:00
last_review: 2016-04-02T04:00:00.000+00:00
accommodates: 6
```


```
use sample_airbnb
db.listingsandreviews.find({
"amenities": { "$all": ["Wifi",
"Kitchen"]}})
```

# sample_airbnb.listingsAndReviews

5.6k DOCUMENTS  4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter {":": 3}, "price": {"$gte": 1400, "$lte": 1500}, "amenities": { "$all": ["Wifi", "Kitchen"]}} Reset Find </> More Options

ADD DATA | EXPORT COLLECTION

1 – 4 of 4

```
_id: "1176693"
listing_url: "https://www.airbnb.com/rooms/1176693"
name: "BEST REVIEWS*BEST MALLS*SAFE STAY*DIMSUM*CWB*MTR"
summary: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
space: "FACTS - located in the heart of one of Hong Kong's most bustling place…"
description: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
neighborhood_overview: "This enigmatic city of skyscrapers, ancient traditions and heavenly fo…"
notes: "Please note that not every neighbor is happy about the New Sharing Eco…"
transit: "HOW TO GET TO CAUSEWAY BAY: By Public transport: MTR: Island Line: Tin…"
access: "The entire apartment with all amenities is accessible for my guests. A…"
interaction: "I will respond to your questions within very short notice. My reservat…"
house_rules: "Early check-in and late check-out is possible for an additional fee. P…"
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1124"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
calendar_last_scraped: 2019-03-11T04:00:00.000+00:00
first_review: 2013-07-04T04:00:00.000+00:00
last_review: 2018-07-29T04:00:00.000+00:00
accommodates: 8
```

```
use sample_airbnb
db.listingsandreviews.find({"address.market":"Hong Kong", "property_type" : "Apartment", "bedrooms": {"$gte": 3}, "price": {"$gte": 1400, "$lte": 1500}, "amenities": { "$all": ["Wifi", "Kitchen"]}})
```

# sample_airbnb.listingsAndReviews

5.6k DOCUMENTS  4 INDEXES

Documents | Aggregations | Schema | **Explain Plan** | Indexes | Validation

Filter {"address.market":"Hong Kong", "property_type" : "Apartment", "bedrooms": {"$gte": 3}, ' Reset Explain </> More Options

VIEW VISUAL TREE RAW JSON

## Query Performance Summary Learn more

Documents Returned: **4**

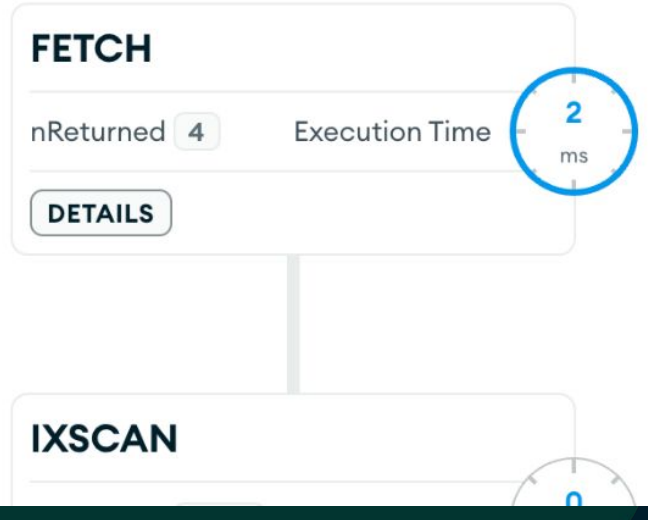
Index Keys Examined: **3626**

Documents Examined: **3626**

Actual Query Execution Time (ms): **14**

Sorted in Memory: **no**

Query used the following index: PROPERTY_TYPE ↑ ROOM_TYPE ↑ BEDS ↑




```
use sample_airbnb
db.listingsandreviews.find({"address.market":"Hong Kong", "property_type" : "Apartment", "bedrooms": {"$gte": 3}, "price": {"$gte": 1400, "$lte": 1500}, "amenities": { "$all": ["Wifi", "Kitchen"]}}).explain()
```

shorturl.at/xHX78

# 5 min break

shorturl.at/xHX78

# Mongo Exercise 2

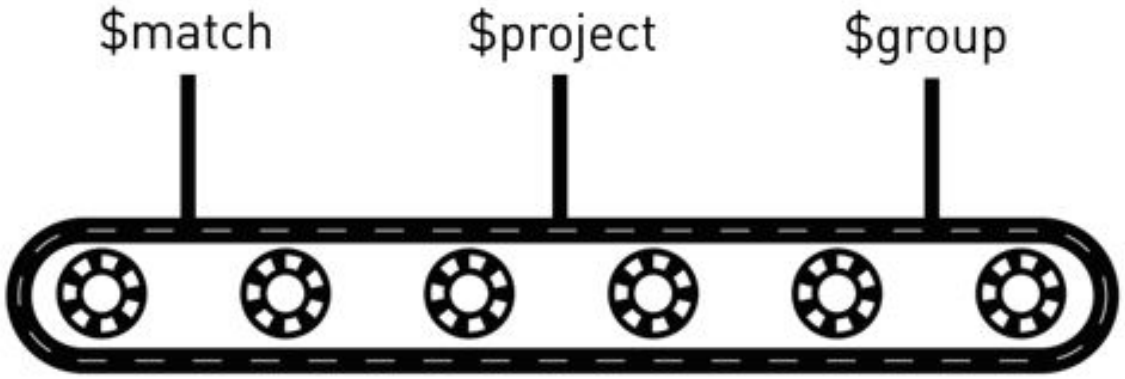
## Aggregation Framework

# Aggregations 101

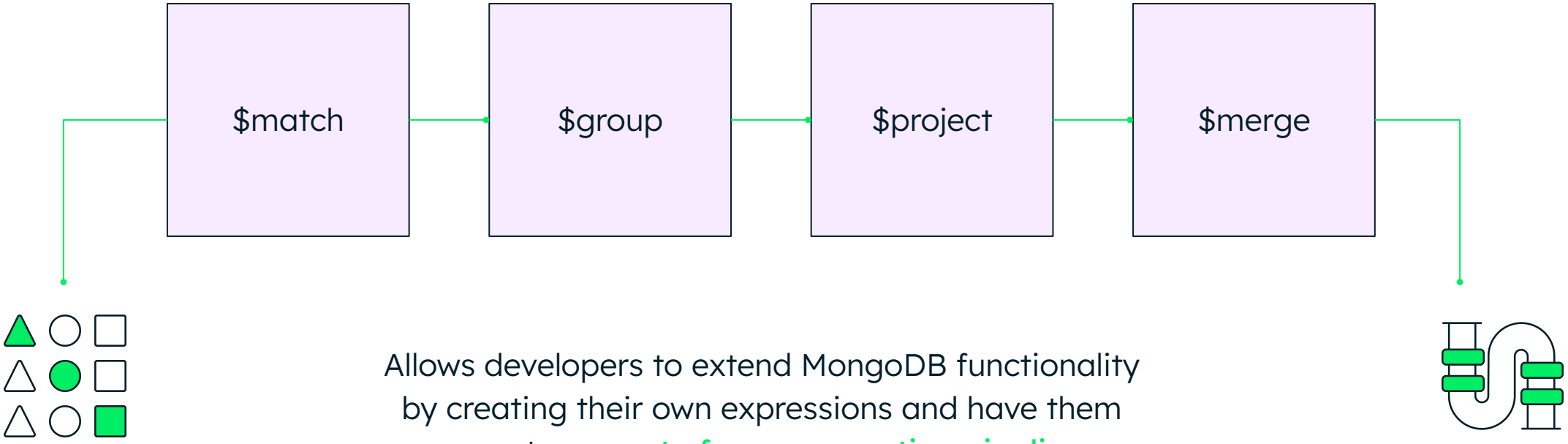
- Until now, we have only filtered data
  - What Documents
  - What Fields
  - What Order
- Aggregation allows us to compute new data
  - Calculated fields
  - Summarised and grouped values
  - Reshape documents

# Aggregation -> Pipeline

- Each transformation is a single step known as a stage.
- Compared to one huge SQL style statement this is
  - Easier to understand
  - Easier to debug
  - Easier for MongoDB to rewrite and optimize

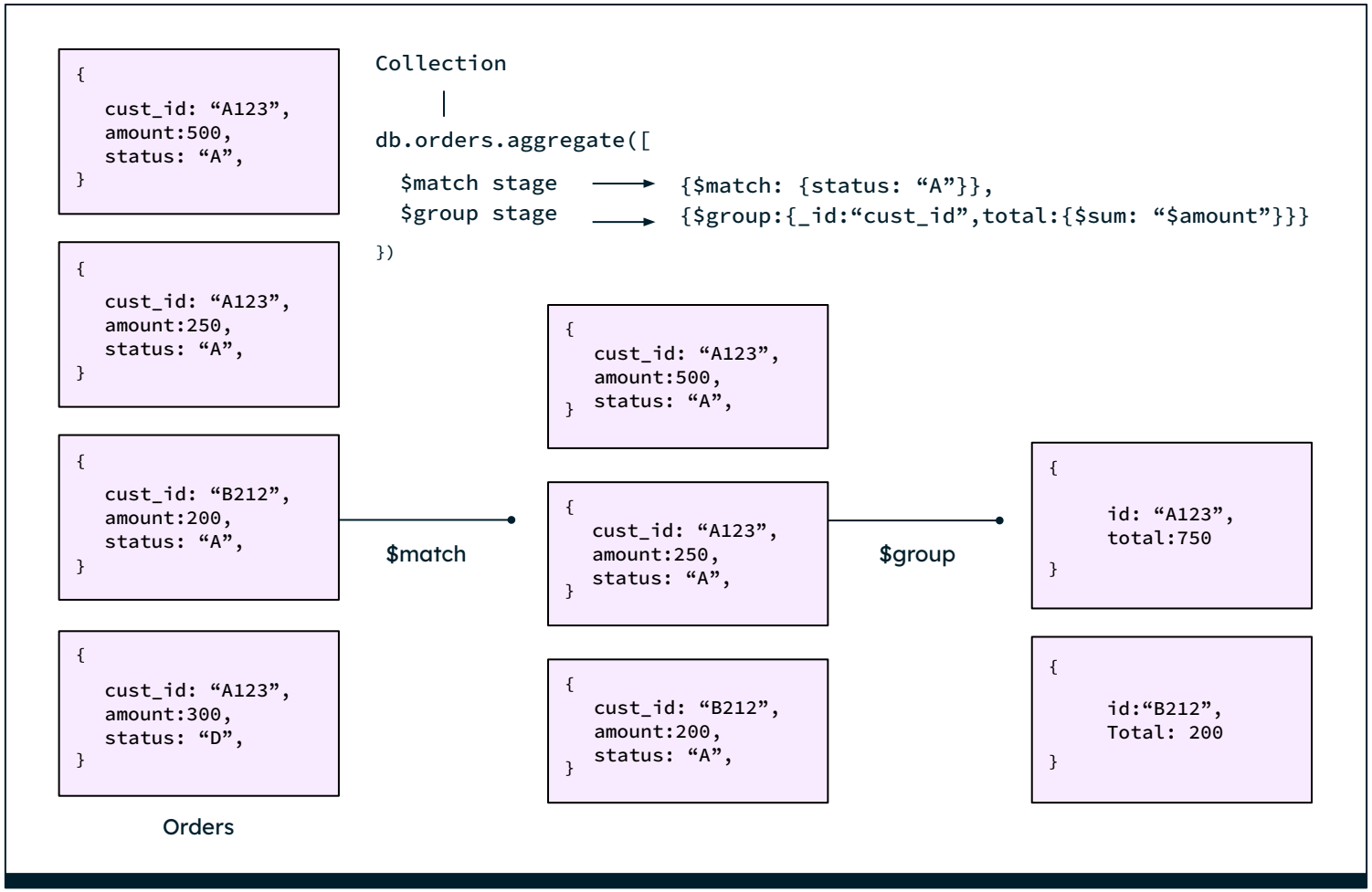


# Aggregation pipelines




Allows developers to extend MongoDB functionality by creating their own expressions and have them execute as part of an aggregation pipeline

# Aggregation pipelines



# Stages we know

- $match equivalent to find(query)
- $project equivalent to find({},projection)
- $sort equivalent to find().sort(order)
- $limit equivalent to find().limit(num)
- $skip equivalent to find().skip(num)
- $count equivalent to find().count()

When these are used at the start of a pipeline, they are transformed to a find() by the query optimizer.

# Comparing Aggregation Syntax

## Find the name of the host in Canada with the most "total listings":

Using find()

```
db.listingsAndReviews.find(
    {"address.country":"Canada"},
    {"_id":0, "host.host_total_listings_count":1, "host.host_name":1}
).sort({"host.host_total_listings_count":-1}).limit(1)
```

Using aggregate()

```
db.listingsAndReviews.aggregate([
    {$match:{"address.country":"Canada"}},
    {$sort: {"host.host_total_listings_count":-1 }},
    {$limit:1},
    {$project:{"_id":0, "host.host_total_listings_count":1, "host.host_name":1}}
])
```

# Dollar Overloading

```
{$match: {a: 5}}
```

Dollar on left means a stage name - in this case a **$match** stage

```
{$set: {b: "$a"}}
```

Dollar on right of colon "**$a**" refers to the value of field a

```
{$set: {area: {$multiply: [5,10]}}
```

**$multiply** is an expression name left of colon

```
$map:{input: "$quizzes", as: "grade",in: { $add: [ "$$grade", 2 ] }}}
```

**$$grade** refers to a variable in $map statements

Think as a programmer - not as a database shell.

Define variables. Doing this helps you keep track of brackets.

It also helps you really understand the Object concept.

```
//Do it THIS way for ease of testing and debugging

> no_celebs = {$match:{"user.followers_count":{$lt:200000}}}

> name_only = {$project:{"user.name":1, "user.folowers_count":1,_id:0}}

> most_popular = {$sort: {"user.followers_count":-1}}

> first_in_list = {$limit:1}

> pipeline = [no_celebs,name_only,most_popular,first_in_list]

> db.twitter.aggregate(pipeline)
```

shorturl.at/xHX78

# Hands-On: MongoDB Shell

# Exercise 1

Using the Shell or Compass/Atlas Aggregation Builder and the Airbnb listings data (`sample_airbnb.listingsAndReviews`) read the following question and output the answer. You will need to use a `$set` stage and a `$project` stage with expressions.

To keep in mind...

1. The price for the basic rental is in the **$price** field; for that price, you can have the number of guests provided in **$guests_included** field.
2. The property may take more guests in total (maximum guests is in **$accommodates** property).
3. You have to pay **$extra_people** dollars per person for every person more than **$guests_included**

Requirement:

- Calculate how many extra guests you can have for each property and add that as a field using `$set`
- Calculate how much it would cost with these extra guests. `$project` the basic price and the price if fully occupied with `$accommodates` people.

# ANSWER: Exercise 1

Using the Airbnb listings data:

- Calculate how much it would cost for these extra guests - they cost "$extra_people" each and $project the basic price and the price if fully populated

```
addextra = { $set: { numguestsextra : { $subtract:
["$accommodates","$guests_included"]}}}

db.listingsAndReviews.aggregate([addextra]).pretty()

extraguestcost = { $multiply : ["$extra_people","$numguestsextra"]}

finaloutput = { $project: { price: 1 , maxprice: { $add: [
"$price",extraguestcost]}}}

db.listingsAndReviews.aggregate([addextra,finaloutput]).pretty()
```

# Exercise 2

Calculate how many Airbnb properties there are in each country, ordered by count, list the one with the largest count first. You should continue working with namespace **`sample_airbnb.listingsAndReviews`**.

- **Hint:** to count things, you add the explicit value 1 to an accumulator using $sum

# ANSWER: Exercise 2

Calculate how many Airbnb properties there are in each country, ordered by count, list the one with the largest count first.

```
groupfield = "$address.country"
groupstage = { $group: { _id: groupfield, count:{$sum:1}}}
sortstage = {$sort:{count:-1}}
pipe = [groupstage,sortstage]
db.listingsAndReviews.aggregate(pipe).pretty()
```

or

```
db.listingsAndReviews.aggregate([{$sortByCount:"$address.country"}]).pretty()
```

# Exercise 3

What are the top 10 most popular amenities across all listings?

# ANSWER: Exercise 3

What are the top 10 most popular amenities across all listings?

```
unwindstage = { $unwind:"$amenities"}
group = { $group: { _id: "$amenities", count: { $sum: 1 } } }
sortstage = {$sort:{count:-1}}
limitstage = { $limit : 10 }
pipe=[unwindstage,group,sortstage, limitstage]
db.listingsAndReviews.aggregate( pipe ).pretty()
```

shorturl.at/xHX78

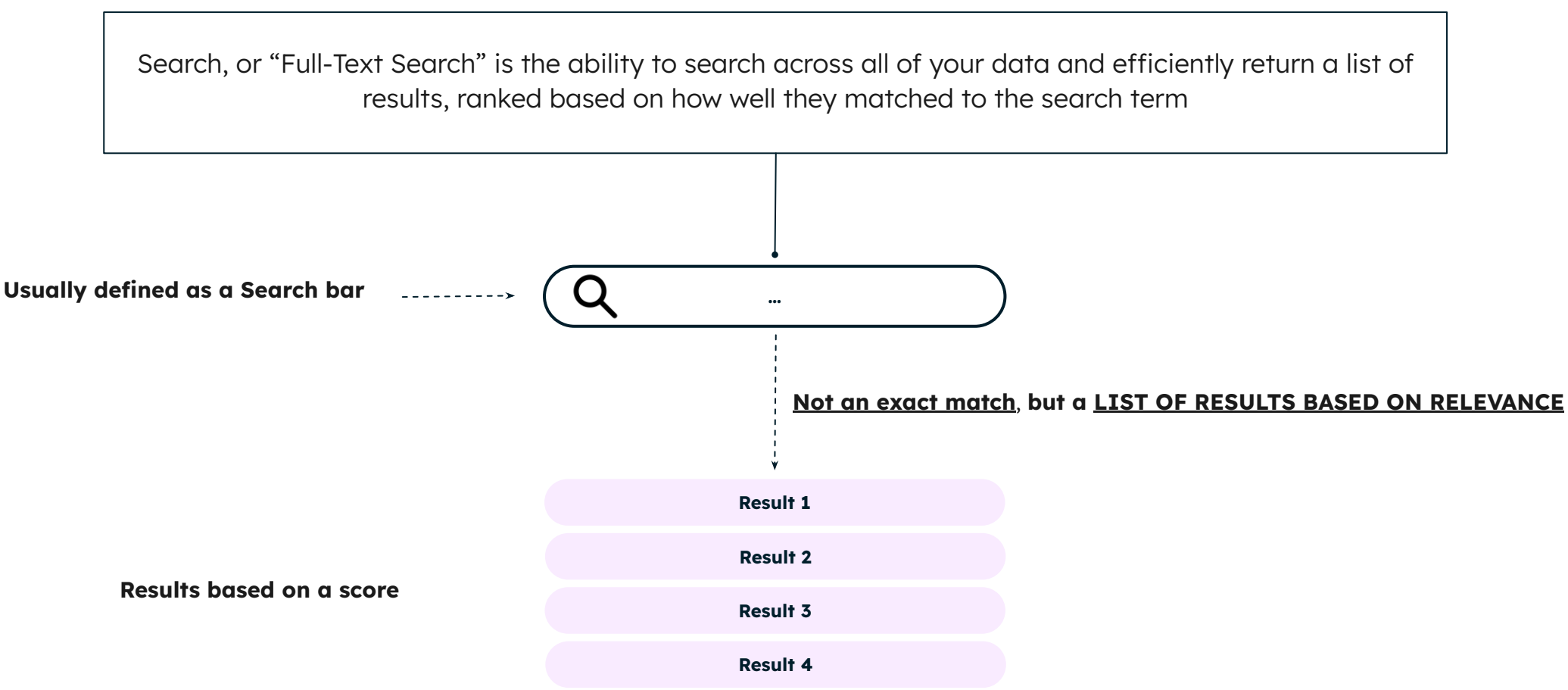
# 5 min break

shorturl.at/xHX78

# Search

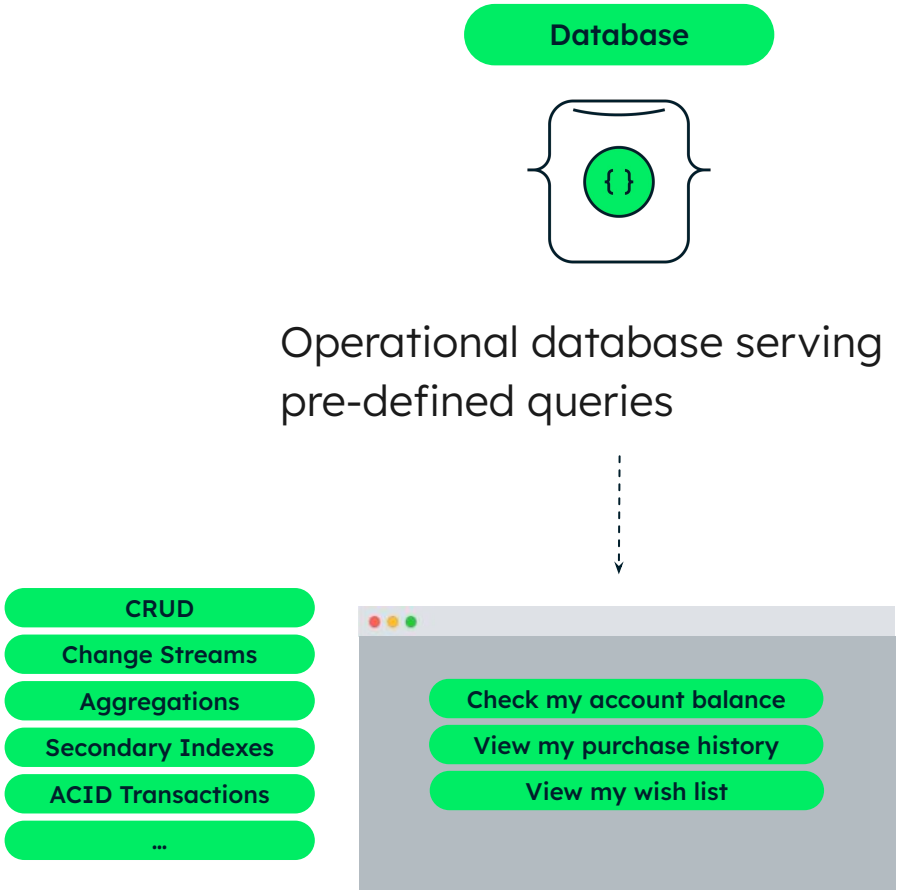
# What is Search?

Search, or “Full-Text Search” is the ability to search across all of your data and efficiently return a list of results, ranked based on how well they matched to the search term



# How developers build app search today

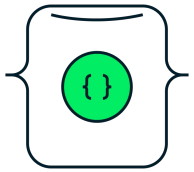
Stand up a database to provide the application's persistence layer



# How developers build app search today

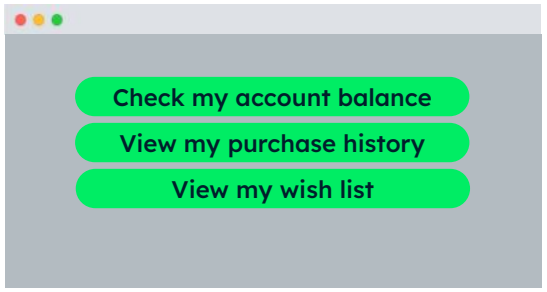
Bolt-on a search engine to power search across application data

**Database**

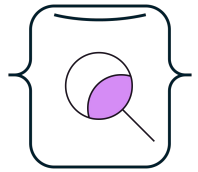


Operational database serving pre-defined queries

- CRUD
- Change Streams
- Aggregations
- Secondary Indexes
- ACID Transactions
- ...




**Search Engine**

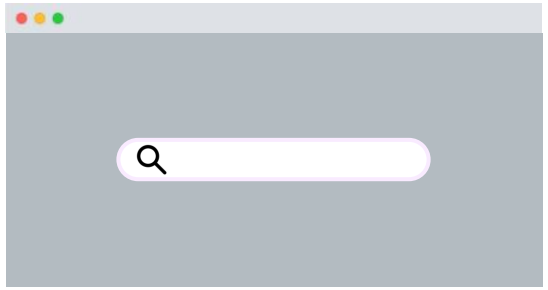


Lucene

elasticsearch

Solr

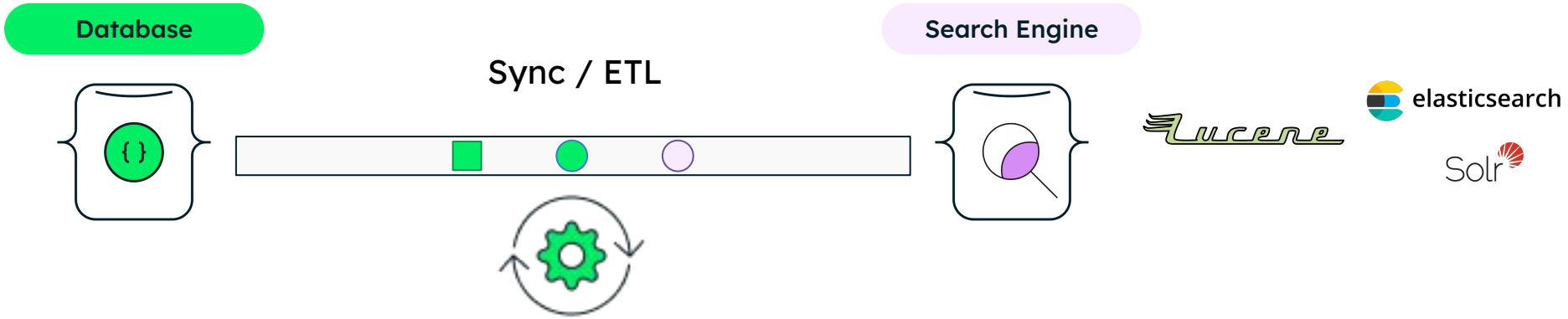
Search engine inferring intent from free form, natural language queries



- Fuzzy matching
- Autocomplete
- Facets and counts
- Synonyms
- Scoring / Weighting
- moreLikeThis

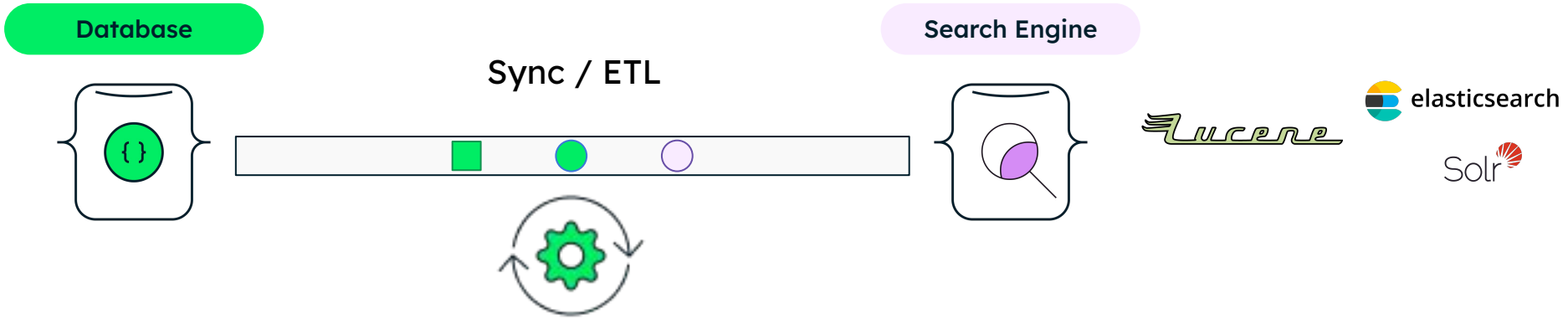
# How developers build app search today

Stand up a replication mechanism to keep the systems in sync



# How developers build app search today

## Architectural complexity in our application stack




### Developer friction

Different query APIs and drivers for database and search, coordinate schema changes

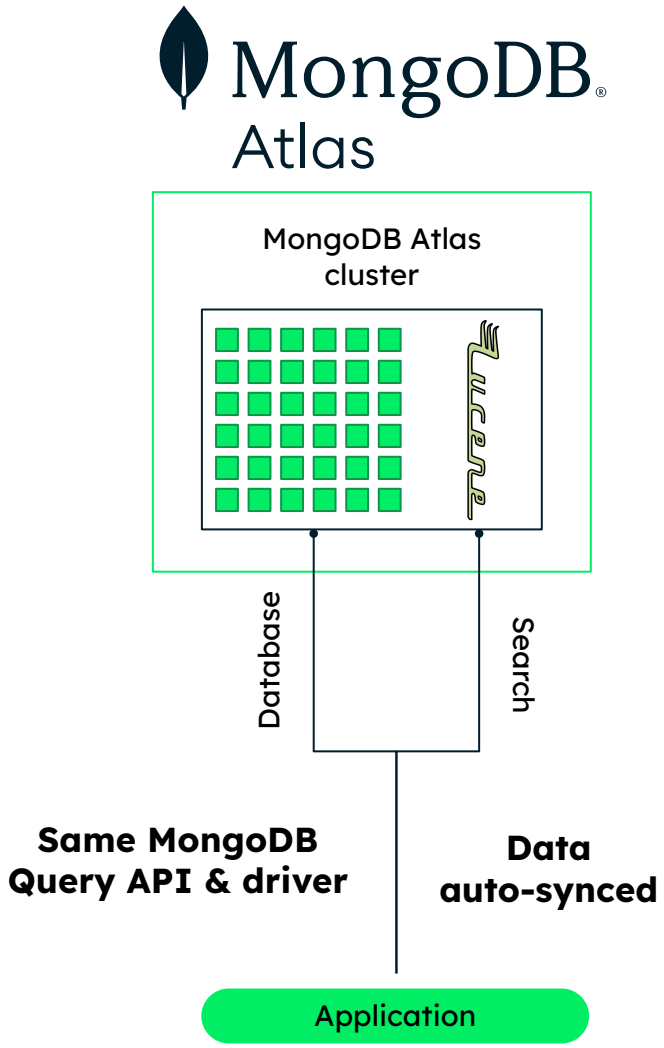
### Pay the sync tax

Requires its own systems and skills. Recovering sync errors can consume 10% of a developer's time

### Operational overhead

More to provision, secure, upgrade, patch, back up, monitor, scale, etc.

# How does Atlas Search address that pain?




Embeds a fully managed Apache Lucene index right alongside the database

**Improved developer productivity**

Build database and search features using the same query API and driver

**Simplified data architecture**

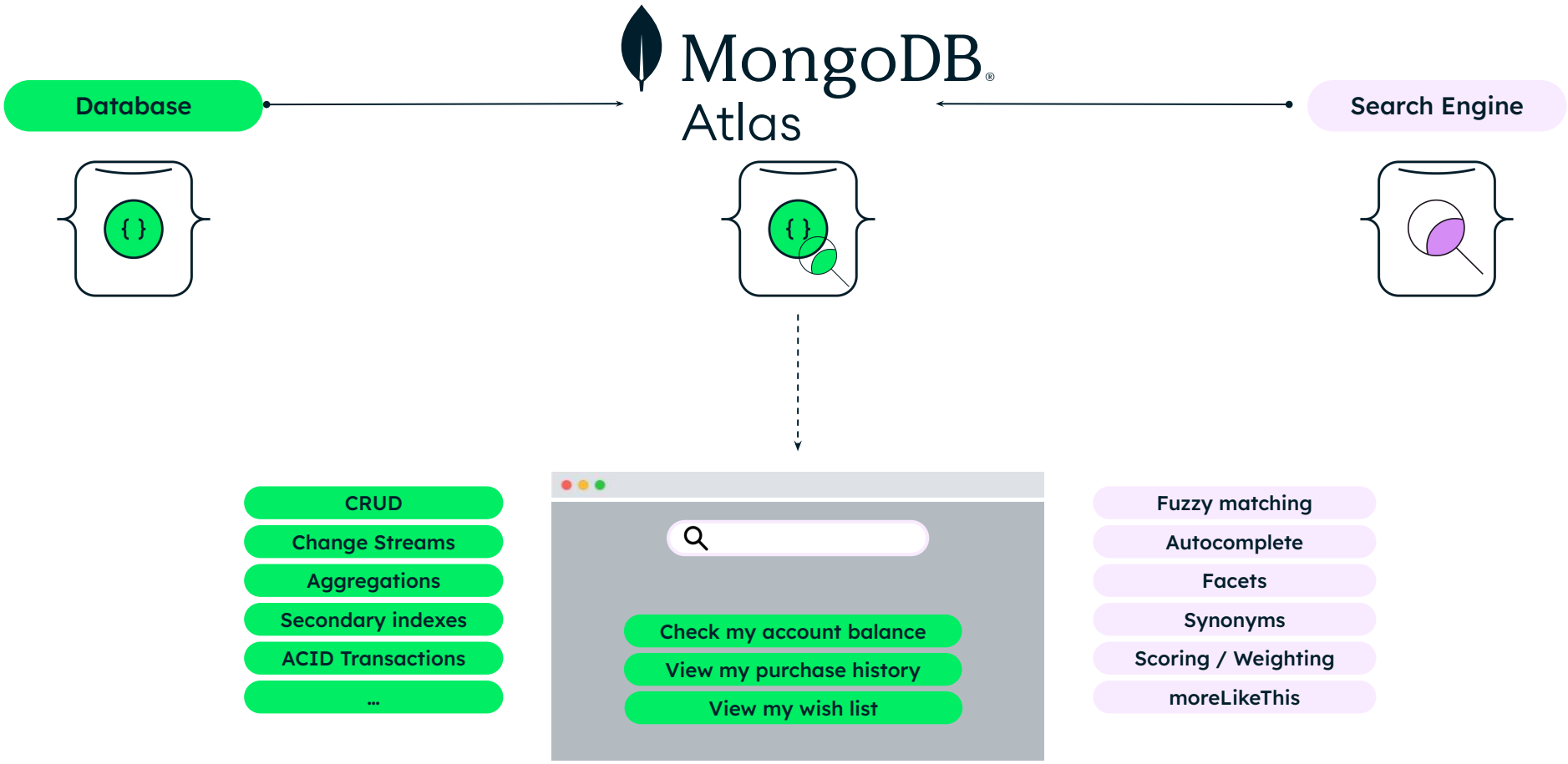
Automatic data synchronization, even as your data and schema changes

**Fully managed platform**

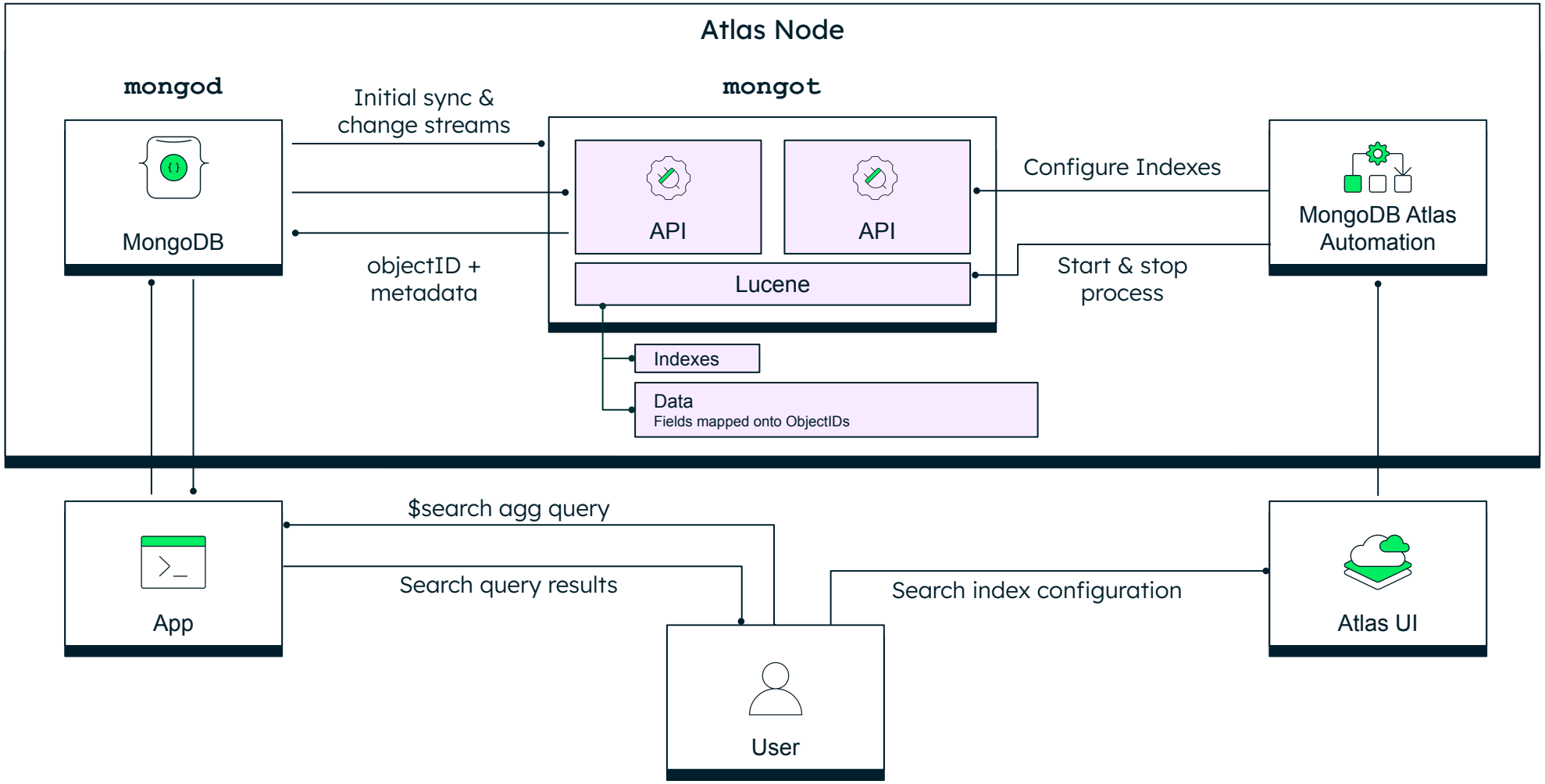
Get the security, performance, and reliability of Atlas

# Unifying database and search

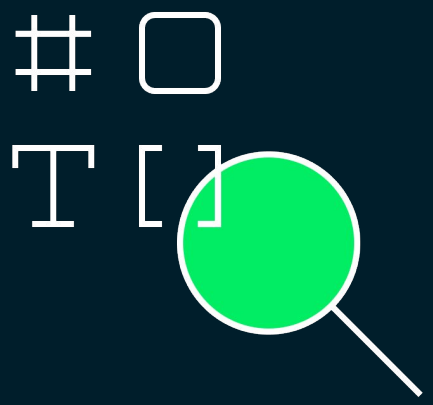
Compress 3 systems into 1, ship search features **30%-50% faster**



# Atlas Search architecture



# Rich search features



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

- Fuzzy search
- Autocomplete
- Filters
- Synonyms
- Facets
- Multi-data type support
- Multi-language support
- Highlighting
- Custom scoring
- Rich query language
- More Like This

# Analyzers

We start with an original query

Lions and tigers and bears, oh my!

**Analyzers** apply parsing and language rules to the query

**Tokens** are the individual terms that are queried; think of these as the words that matter in a search query

# Analyzers

We start with an original query

**Analyzers** apply parsing and language rules to the query

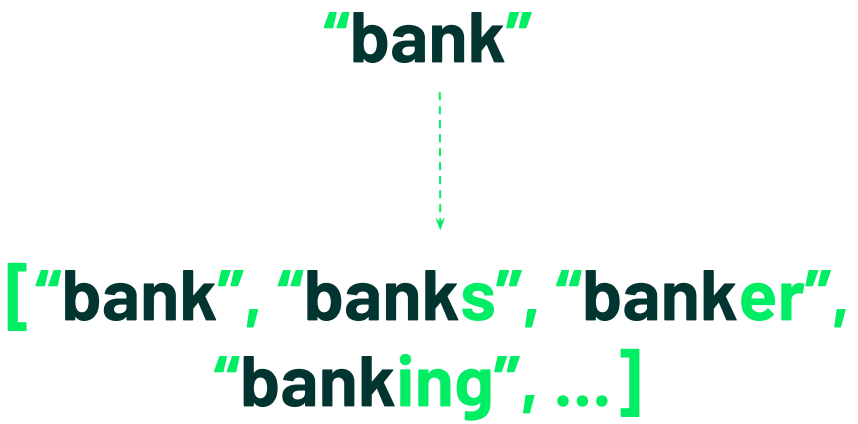
**Tokens** are the individual terms that are queried; think of these as the words that matter in a search query

# Analyzers

Analyzers also support language rules

A search for...

...can return results for related words



# Other supported search features

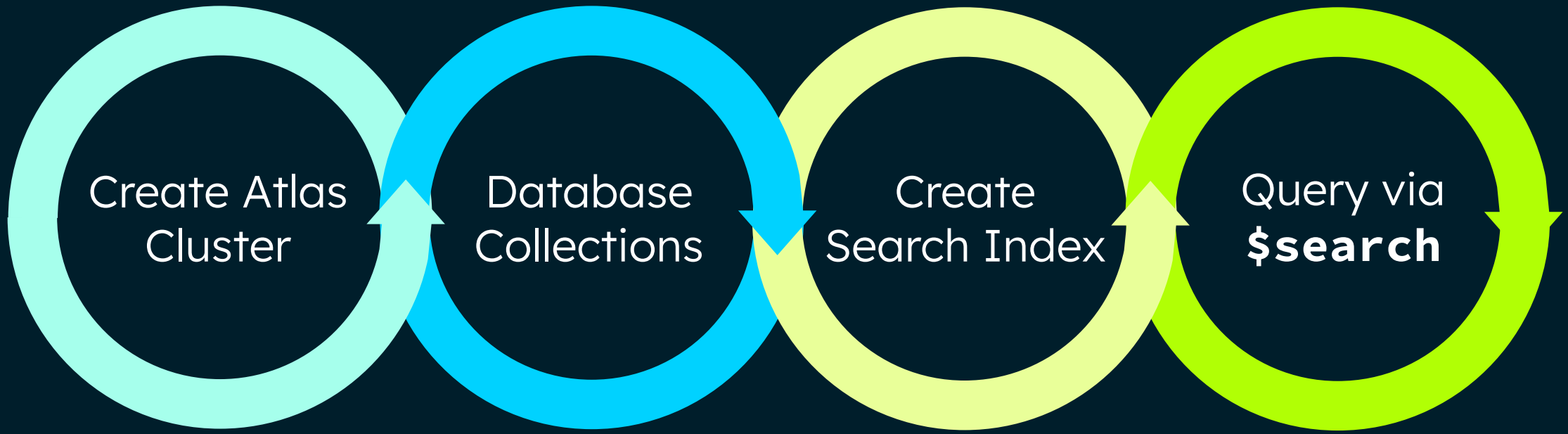
Search features are all about helping customers make their data more **discoverable** and **relevant** for end users

- **Support for 30+ languages**
- Autocomplete
- Highlighting
- Fuzzy matching (typo tolerance)
- Faceting
- Multiple data types
- Synonyms
- Custom scoring
- Rich query language
- More like this

| | | | | |
|---|---|---|---|---|
| lucene.arabic | lucene.armenian | lucene.basque | lucene.bengali | lucene.brazilian |
| lucene.bulgarian | lucene.catalan | lucene.cjk | lucene.czech | lucene.danish |
| lucene.dutch | lucene.english | lucene.finnish | lucene.french | lucene.galician |
| lucene.german | lucene.greek | lucene.hindi | lucene.hungarian | lucene.indonesian |
| lucene.irish | lucene.italian | lucene.latvian | lucene.lithuanian | lucene.norwegian |
| lucene.persian | lucene.portuguese | lucene.romanian | lucene.russian | lucene.sorani |
| lucene.spanish | lucene.swedish | lucene.turkish | lucene.thai | |

Each language analyzer has built-in rules based on the language's usage pattern

# How do I search data in atlas?



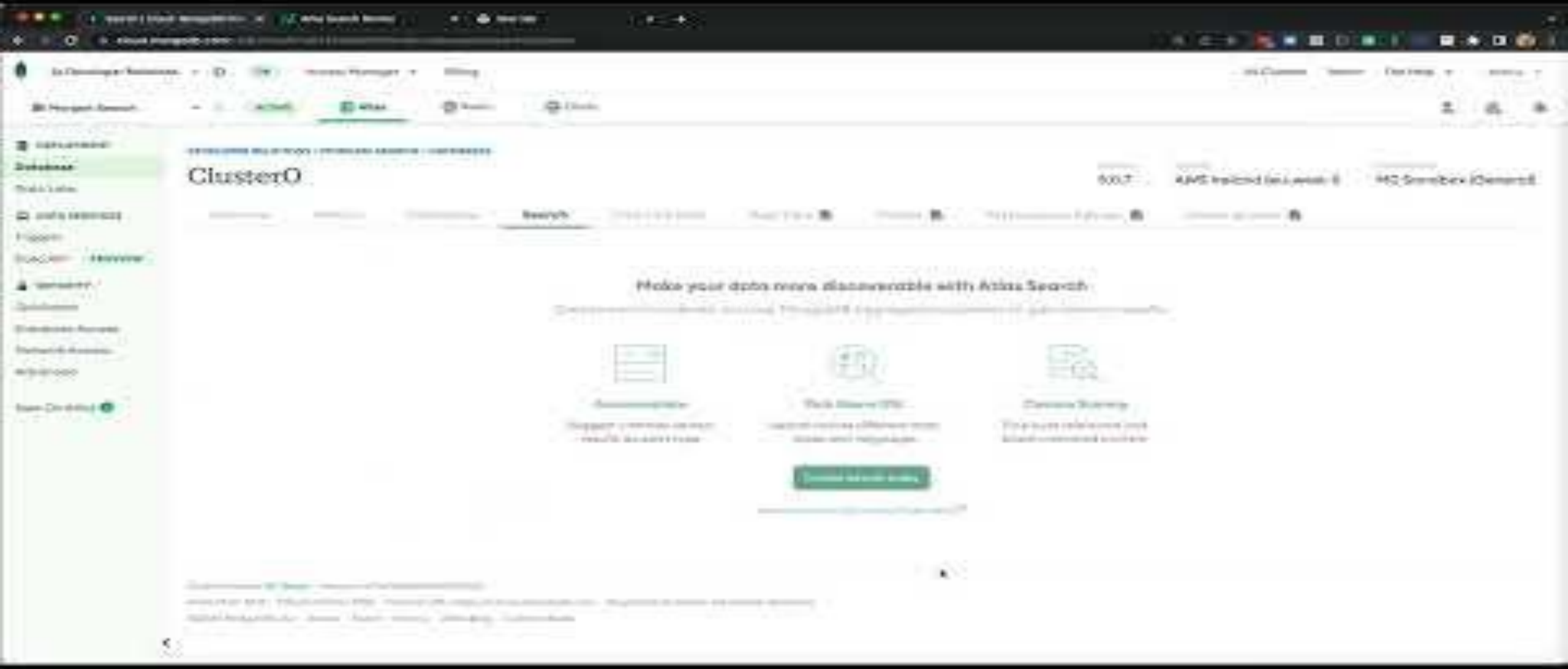

shorturl.at/xHX78

shorturl.at/xHX78

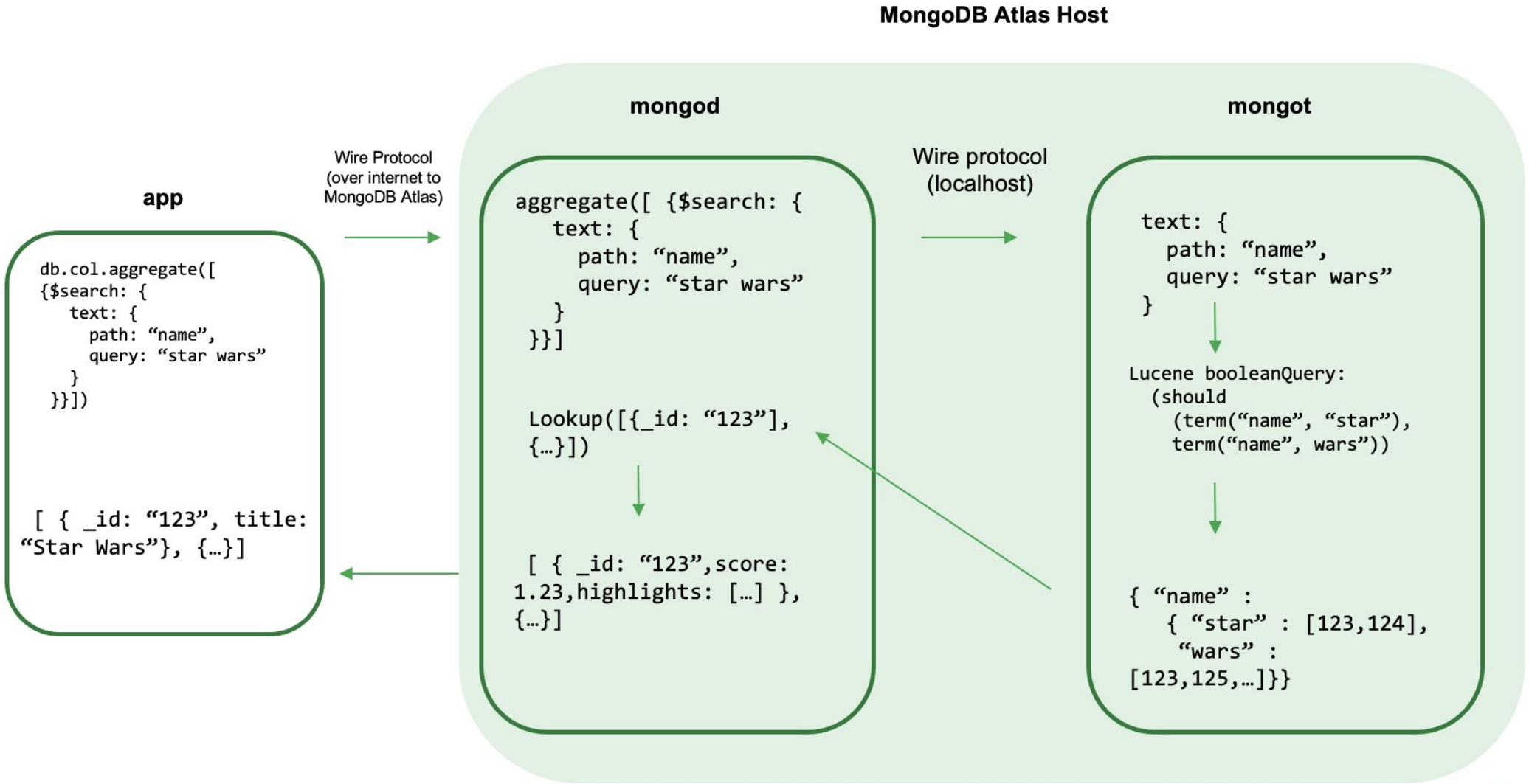
# Search Exercise 1

Configuring Search Indexes

1. Default Search Index
2. Autocomplete on Title Index


ClusterO
Search
Make your data more discoverable with Atlas Search

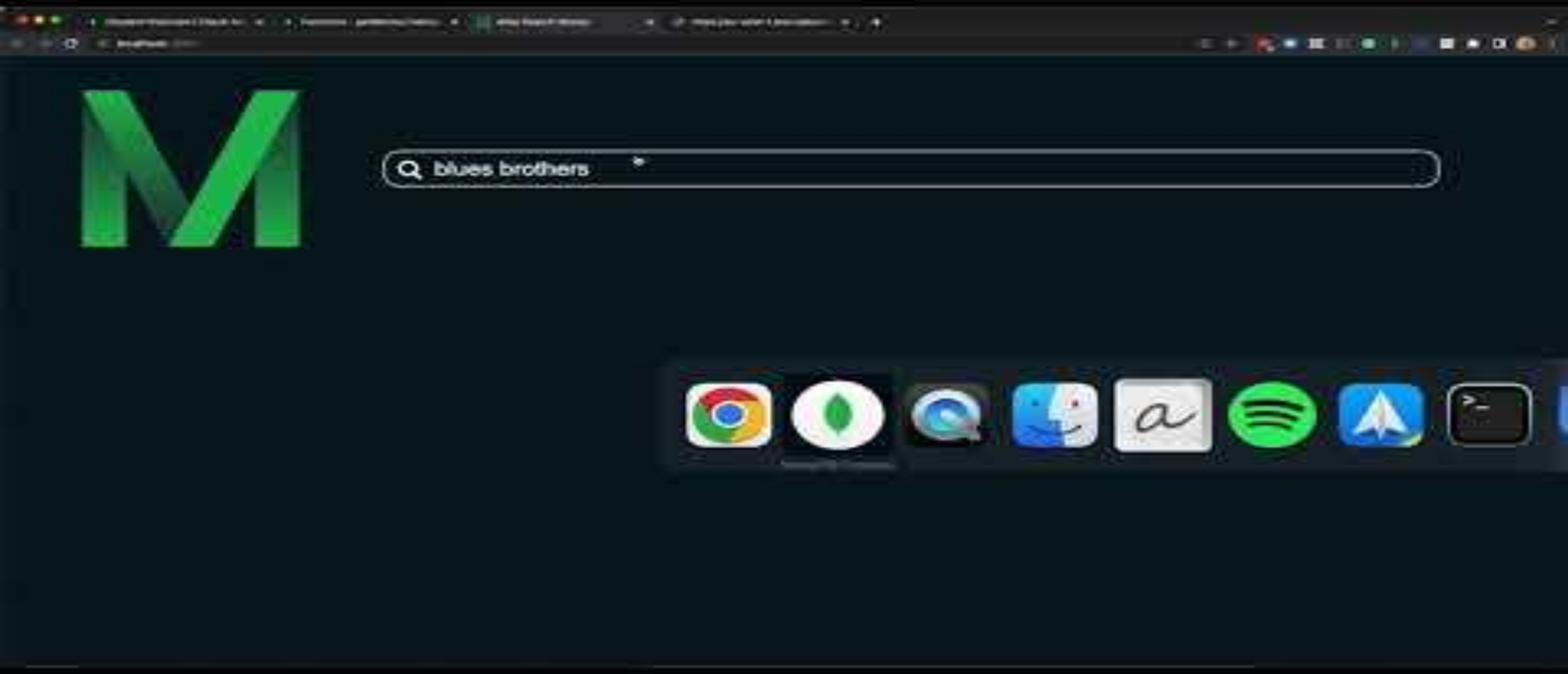
# Query lifecycle



shorturl.at/xHX78

# Search Exercise 2

Our first Search aggregations in Atlas UI & searching from our app


blues brothers

shorturl.at/xHX78

# Self Serve Exercise 3 (Optional)

Integrate your endpoints into a web app

localhost:3000
Click to go back, hold to see history

M

bollywood

## Hollywood Shuffle

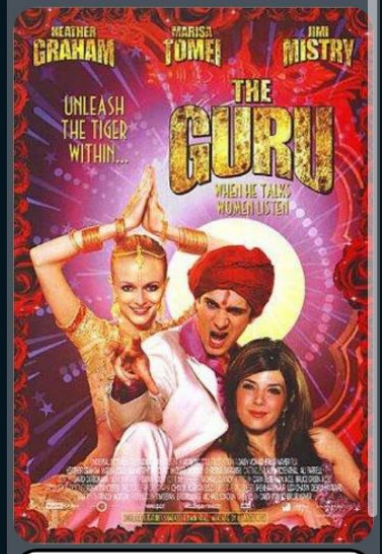

Score: 7.507

Year: 1987

Rating: 7

He just has to convince **Hollywood** that

## The Guru



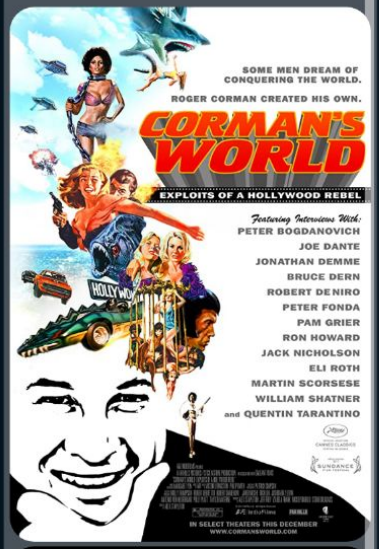

Score: 7.221

Year: 2002

Rating: 5.4

Bored with **Bollywood** movies but

## Corman's World: Exploits of a Hollywood Rebel

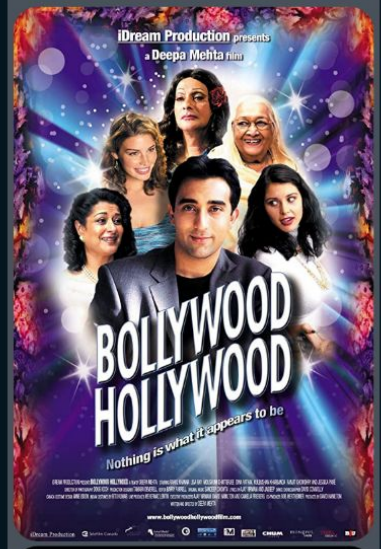



Score: 7.068

Year: 2011

Rating: 7.6

## Bollywood/Hollywood




Score: 6.904

Year: 2002

Rating: 6

# Self Serve Demo 4

## Search feature tour

atlassearchrestaurants.com

github.com/mongodb-developer/WhatsCooking

atlassearchrestaurants.com

MongoDB. Aggregation | Function Score 💯 | Synonyms | Data & Indexes

# Atlas Search Demo: Restaurant Finder

What's Cooking

restaurants...

burger

Find

$search

**Geospatial Search Options**

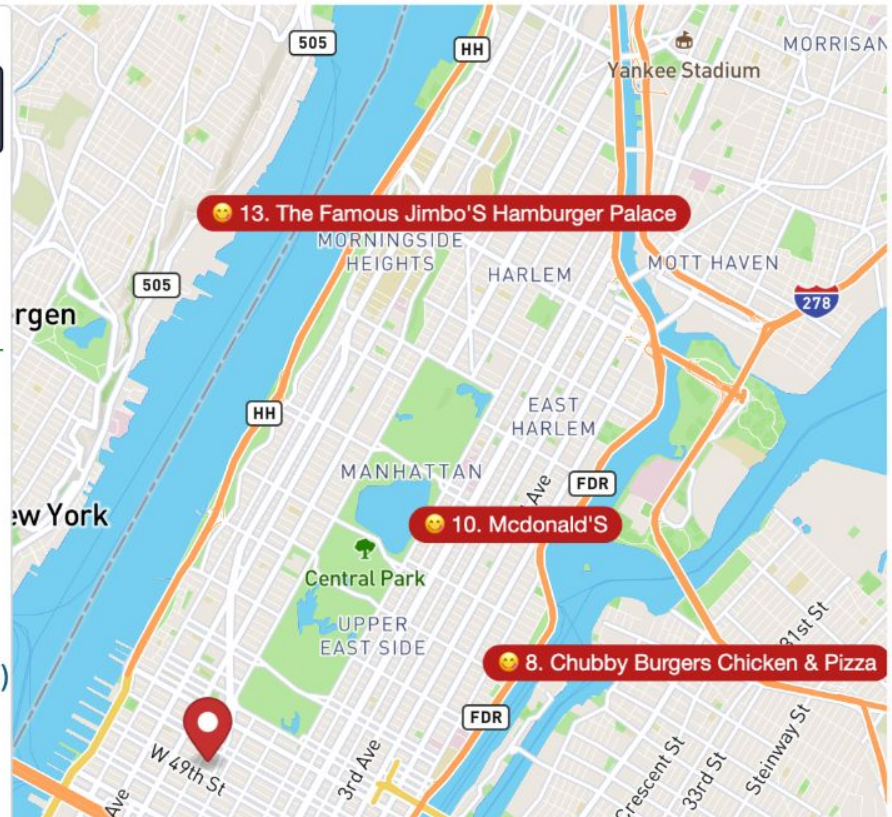
near | geoWithin

**Average Star Rating:**

★★★★★

Overall Count: 5813

- [ ] American (5380)
- [ ] Chinese (0)
- [ ] French (0)
- [ ] Hamburgers (433)
- [ ] Italian (0)
- [ ] Japanese (0)




**score: 1.8**

**White Castle**

Hamburgers
Myrtle Avenue Brooklyn

131 reviews

Show Menu 👓

**score: 1.8**

**Mcdonald'S**

Hamburgers
Atlantic Avenue Brooklyn

```
{ $search :

/ / optional, defaults to "default"

index: < indexName >

{
  "text": {
    "query": "burger",
    "path": "menu",
    "synonyms": "MenuSynonyms"
  }
}

highlight :
```

# Vector Search Demo

# Thank you!

# Appendix