

Technology Icebreaker

<https://tinyurl.com/mdb-welcome>

At the end of today, you will have the tools
to build an app like this!

MongoDB & Google Cloud Workshop



Jim Blackhurst, Solutions Architect, MongoDB

Ronan Bohan, Solutions Architect, MongoDB

Natasha Wilson, Marketing Northern Europe, MongoDB

PJ Singh, Director Partners NEUR, MongoDB

Bradley Wilson Hunt, Service Delivery Manager, FACEIT

Emanuele Massara, Software Engineer, FACEIT



Logistics

Rest Rooms: Outside this room, on the right

WIFI: GoogleGuest-V4 (no password)

Create accounts:

- cloud.mongodb.com
- cloud.google.com

Detailed Agenda

- 9.00 - 9.20 Networking**
 - with breakfast snacks, tea, coffee
- 9.20 - 9.30 Logistics and program**
- 9.30 - 9.45 MongoDB & GCP Overview**
- 9.45 - 10.00 Technical Overview**
 - Best Way to Work with Data
 - Intelligently Place Data Where You Need It
 - Freedom to Run Anywhere
- 10.00 - 10.15 Atlas Overview and Demo**
- 10.15 - 10.30 Break**
- 10.30 - 11.00 FacelT Presentation**
- 11.00 - 11.30 CRUD**
- 11.30 - 12.30 Change Streams on Google Cloud**
 - Concepts
 - Build demo together
- 12.30 - 13.00 MongoDB Charts**
- 13.00 - 13.15 Closing Remarks with Q&A**
 - MongoDB, Google & attendees
- 13.15 - 14.00 Lunch / Networking**

MongoDB and GCP Overview

PJ Singh - Director Partners NEUR

Who is MongoDB?

 1500+ Employees

 100+ Countries

 15,000+ Customers

 70m+ Downloads

 40 Offices Globally

 Nasdaq: MDB



Create competitive advantage



Reduce risk



Accelerate time to value



Lower costs

Intelligent Operational Data Platform



Best for modern data

Intelligent placement

Run Anywhere

HSBC 

AutoTrader

 BARCLAYS

 CHALLENGE YOUR GAME

 compare the market™

 RBS

 HM Revenue & Customs

MongoDB Atlas on GCP



Google Cloud Platform



mongoDB®

MongoDB Technical Overview

Cases for Change

MongoDB is a modern, operational database that supports a polyglot data strategy – on-premise and in the cloud. This allows us to drive several business critical topics with our customers.

Cloud Data Strategy

Leveraging the right data platforms as part of your overall cloud strategy helps to avoid vendor lock-in.

Legacy Modernization

Current legacy technology stacks can't cope with the range of new business requirements – we can help you modernise in a highly efficient and effective way.

Mainframe Offloading

Reduce cost and MIPS on legacy mainframes and enable data to be leveraged for new use cases.

Single View

Provide a holistic view of data entities (e.g. customer) across multiple underlying, disconnected source systems.

Operational Intelligence

Solving the problem of deriving value from existing EDW or Hadoop-based data lake solutions in real-time

Internet of Things (IoT)

MongoDB can help you overcome Scalability & Performance issues that are not being met by many current IoT solutions.

Distributed Ledger/Blockchain

MongoDB is the ideal database/persistence layer for enterprise-grade Distributed Ledger applications.

Leading modern general purpose database



Creative Cloud



HR Mobile Application



Real-Time Travel Search



Mobile Drug Applications



Mobile Banking



Multi-Screen TV



Internet of Things Platform



Predictive Messaging



Background Checks as a Service



Order Capture



Cryptocurrency Platform



Logistics Modernization



E-Commerce System



Entry Decision System



Social Security Benefits Program



Mobile App for Patient Data



Product Catalog



Gaming Platform



Single View of Patient



Genetic Analysis



Online Lending



Online Tax Returns



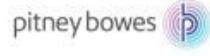
Swap Equities Management



Real-Time Analytics



E-Commerce Personalization



E-Commerce Platform



Marketing Cloud



Video Streaming



Log Metadata Store



Social Media Management



Website Platform



Product Catalog



Identity Theft Protection



CHALLENGE YOUR GAME
Gaming Platform



Subscriber QoS



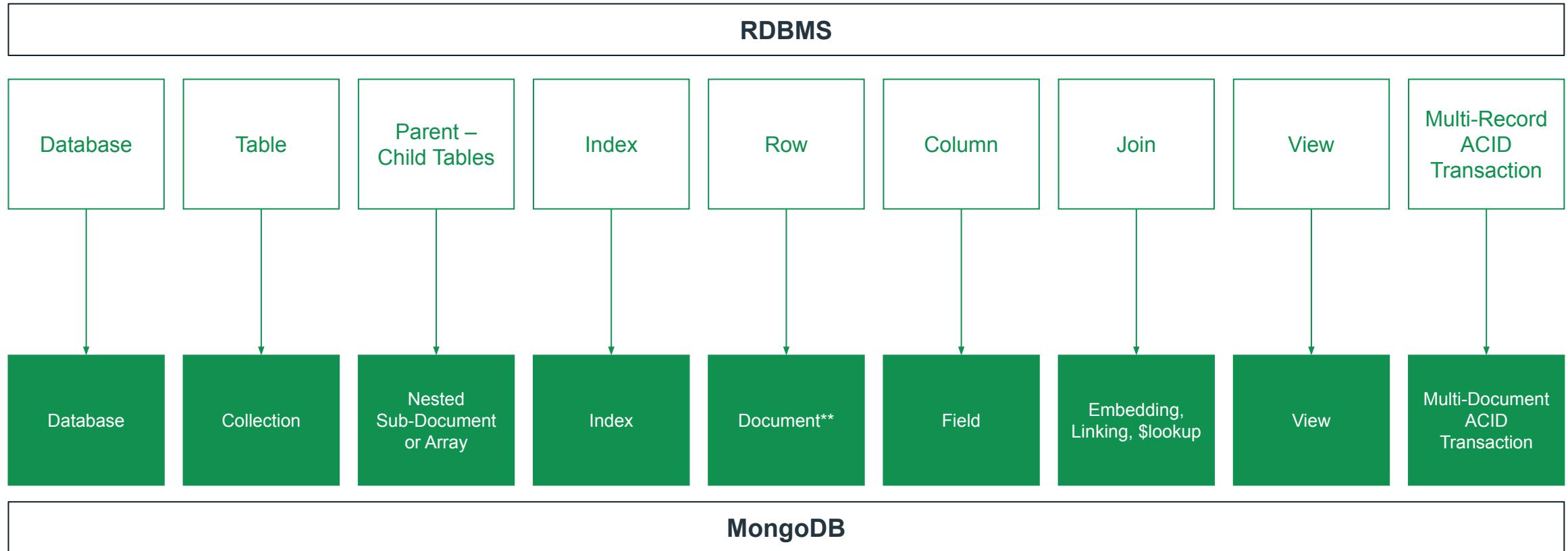
Shopping Cart

The evolution of MongoDB (2019)

	3.2	3.4	3.6	4.0	4.2
<p>Document Validation \$lookup Fast Failover Simpler Scalability Aggregation ++ Encryption At Rest In-Memory Storage Engine BI Connector MongoDB Compass APM Integration Auto Index Builds Backups to File System</p> <p>Linearizable reads Intra-cluster compression Read only views Log Redaction Graph Processing Decimal Collations Faceted Navigation Aggregation ++ Auto-balancing ++ ARM, Power, zSeries BI & Spark Connectors ++ Compass ++ LDAP Authorization Encrypted Backups Cloud Foundry Integration</p> <p>Change Streams Retryable Writes Expressive Array Updates Query Expressivity Causal Consistency Consistent Sharded Sec. Reads Ops Manager ++ Query Advisor Schema Validation End to End Compression IP Whitelisting Default Bind to Localhost Sessions WiredTiger 1m+ Collections Expressive \$lookUp R Driver Atlas Cross Region Replication Atlas Auto Storage Scaling</p> <p>Replica Set Transactions Atlas Global Clusters Atlas HIPAA Atlas LDAP Atlas Audit Atlas Enc. Storage Engine Atlas Backup Snapshots Type Conversions 40% Faster Shard Migrations Snapshot Reads Non-Blocking Sec. Reads SHA-2 TLS 1.1+ Compass Agg Pipeline Builder Compass Export to Code Charts Beta Free Monitoring Cloud Service Ops Manager K8s Beta MongoDB Stitch GA MongoDB Mobile Beta</p> <p>Distributed Transactions Global Point in Time Reads Large Transactions Mutable Shard Key Values Atlas Data Lake Atlas Auto Scaling Atlas Full-Text Search Atlas ISO Compliance Atlas Service Broker Field Level Encryption Multi-CAs & Online Rotation On-Demand Materialized Views Wildcard Indexes Agg Pipeline ++ Expressive Updates Apache Kafka Connector MongoDB Charts GA Retryable Reads & Writes New Index Builds 10x Faster stepDown Storage Node Watchdog Zstandard Compression Ops Manager Headless Backup Ops Manager K8s GA Ops Manager Single Agent</p>					

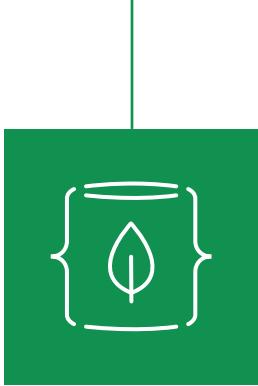
Some Terminology

A comparison



Why MongoDB?

Intelligent Operational Data Platform



**Best way to
work with data**

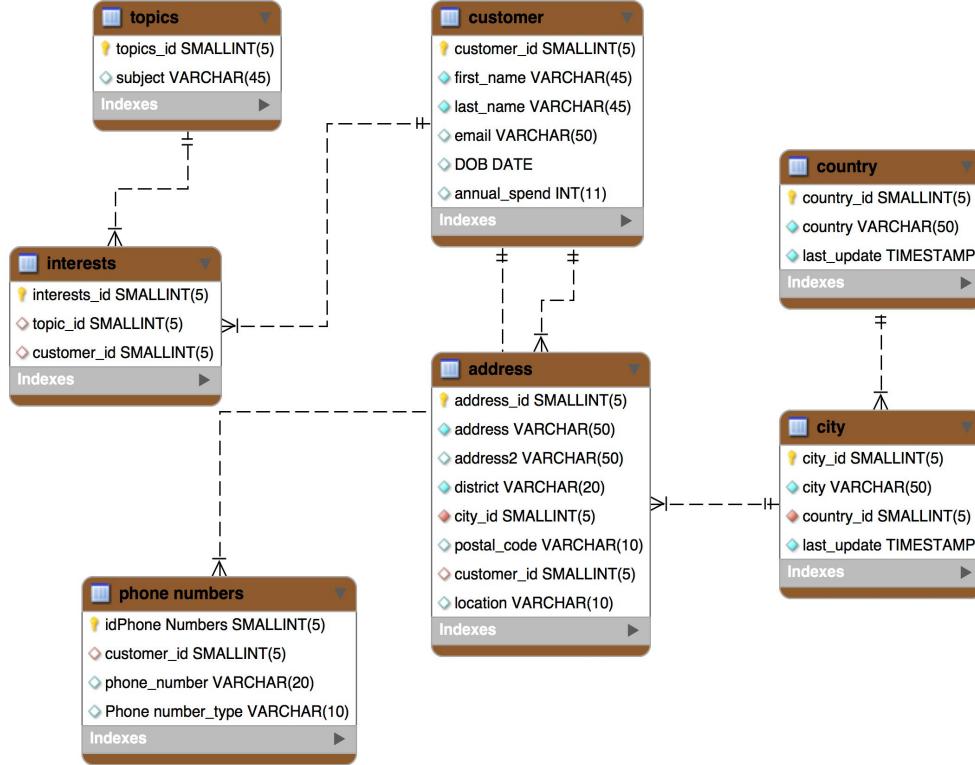


**Intelligently put data
where you need it**



**Freedom to
run anywhere**

Easy: Contrasting data models



Tabular (Relational) Data Model

Related data split across multiple records and tables

```
{
  "_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")
}
```

Document Data Model

Related data contained in a single, rich document

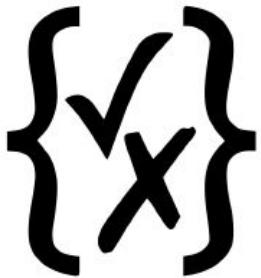
Flexible: Adapt to change

```
{  
  "_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

Flexible: Govern



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

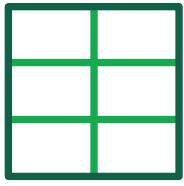
Queryable: identify all existing documents that do not comply

Versatile:

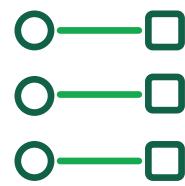
Multiple data models, rich query functionality



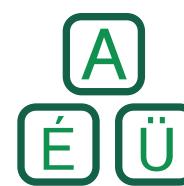
JSON Documents



Tabular



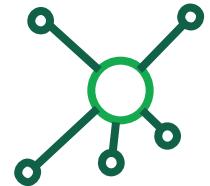
Key-Value



Text



Geospatial



Graph

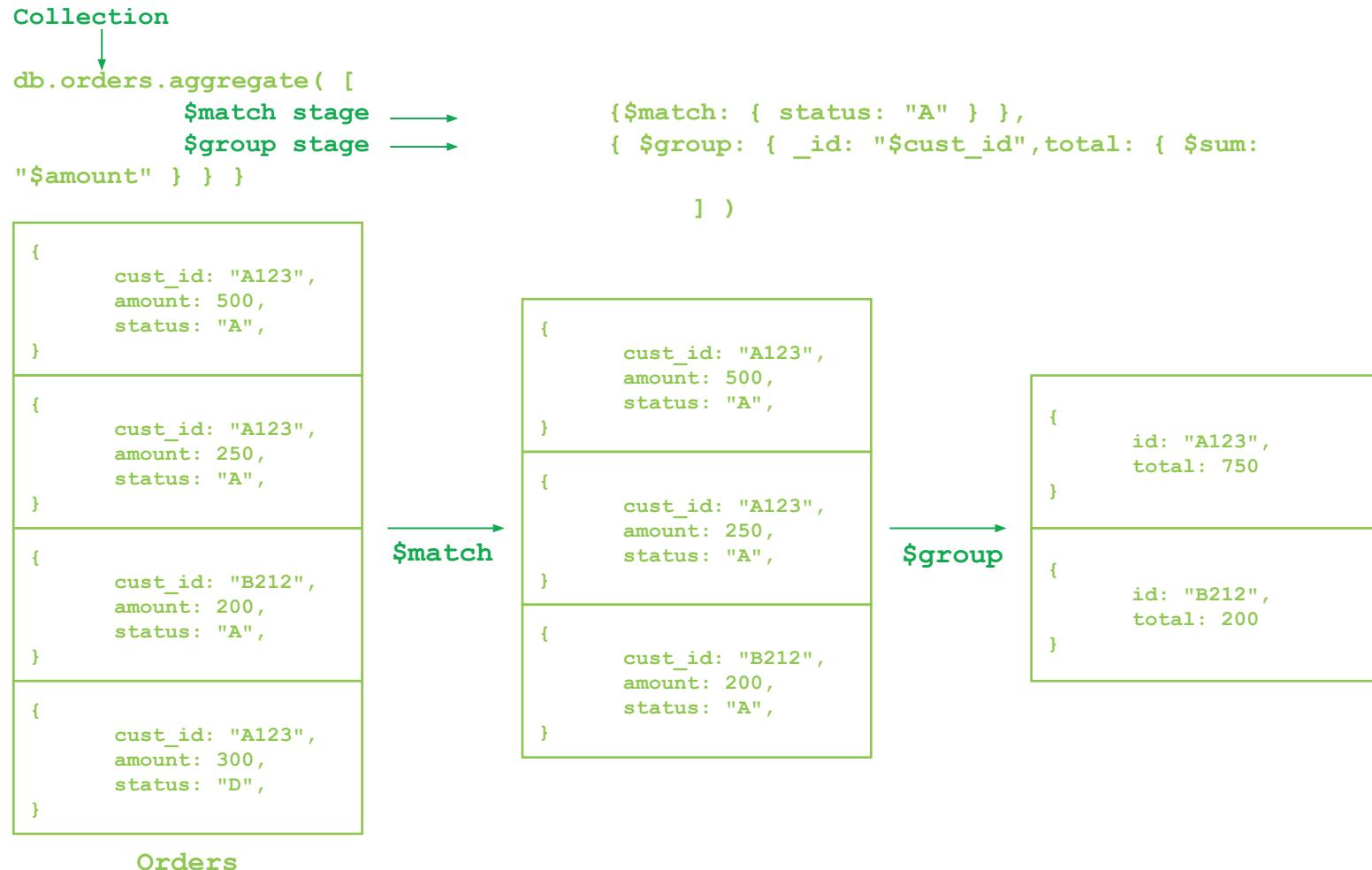
Rich Queries

Point | Range | Geospatial | Faceted Search | Aggregations | JOINs | Graph Traversals

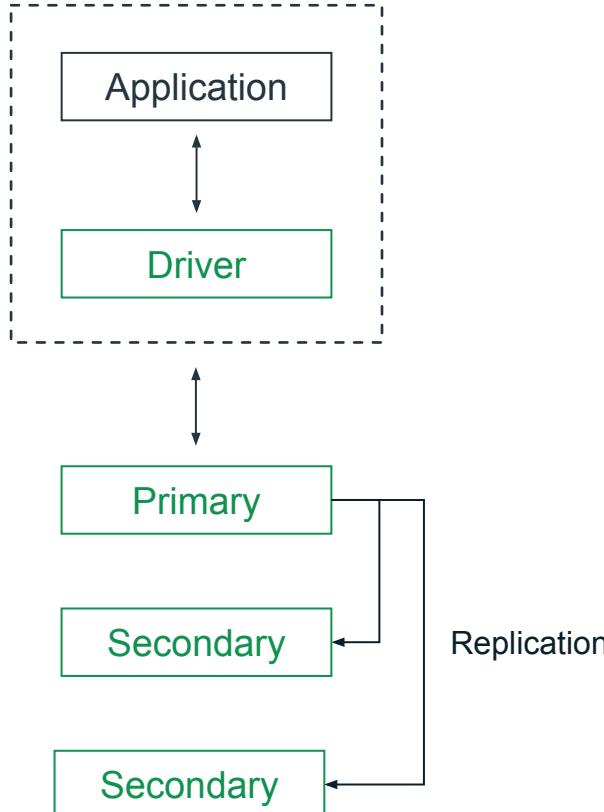
Aggregations

Advanced data processing pipeline for transformations and analytics

- Multiple stages
- Similar to a unix pipe
 - Build complex pipeline by chaining commands together
- Rich Expressions



MongoDB Replica Sets



Replica Set – 2 to 50 copies

Self-healing

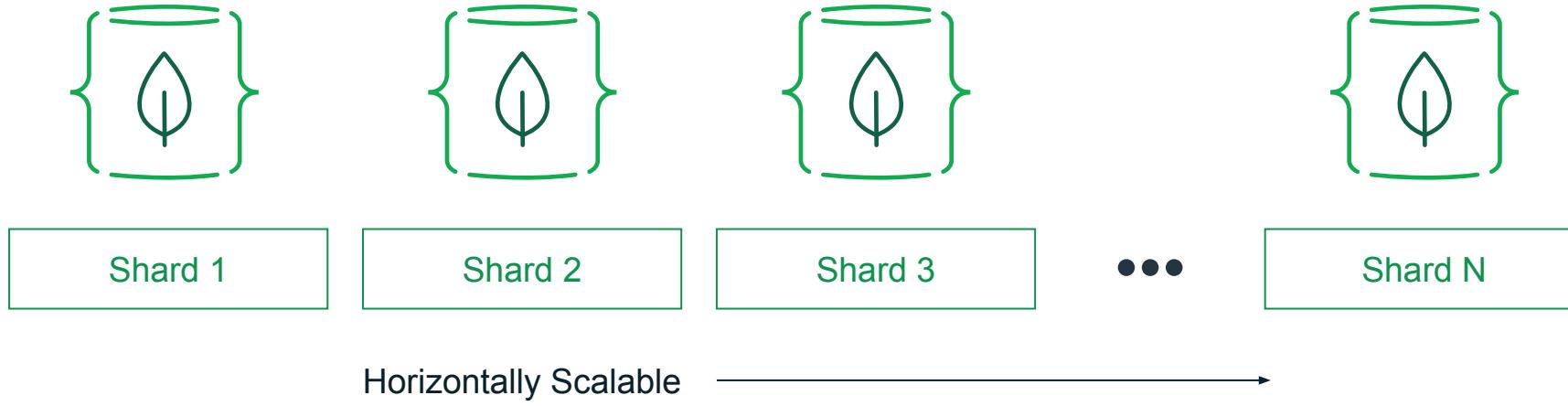
Data Center Aware

Addresses availability considerations:

- High Availability
- Disaster Recovery
- Maintenance

Workload Isolation: operational & analytics

Scaling MongoDB: Automatic Sharding

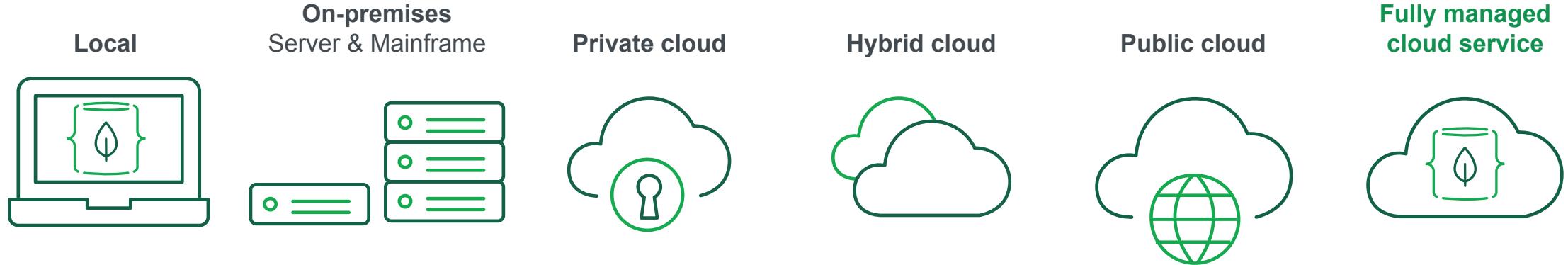


Multiple sharding policies: hashed, ranged, zoned

Increase or decrease capacity as you go

Automatic balancing for elasticity

Freedom to run anywhere



- Database that runs the same everywhere
- Leverage the benefits of a multi-cloud strategy
- Global coverage
- Avoid lock-in

Convenience: same codebase, same APIs, same tools, wherever you run

Atlas

unlocks **agility**
and **reduces**
cost



Self-service and
elastic



Global and highly
available



Secure by default



Comprehensive
monitoring

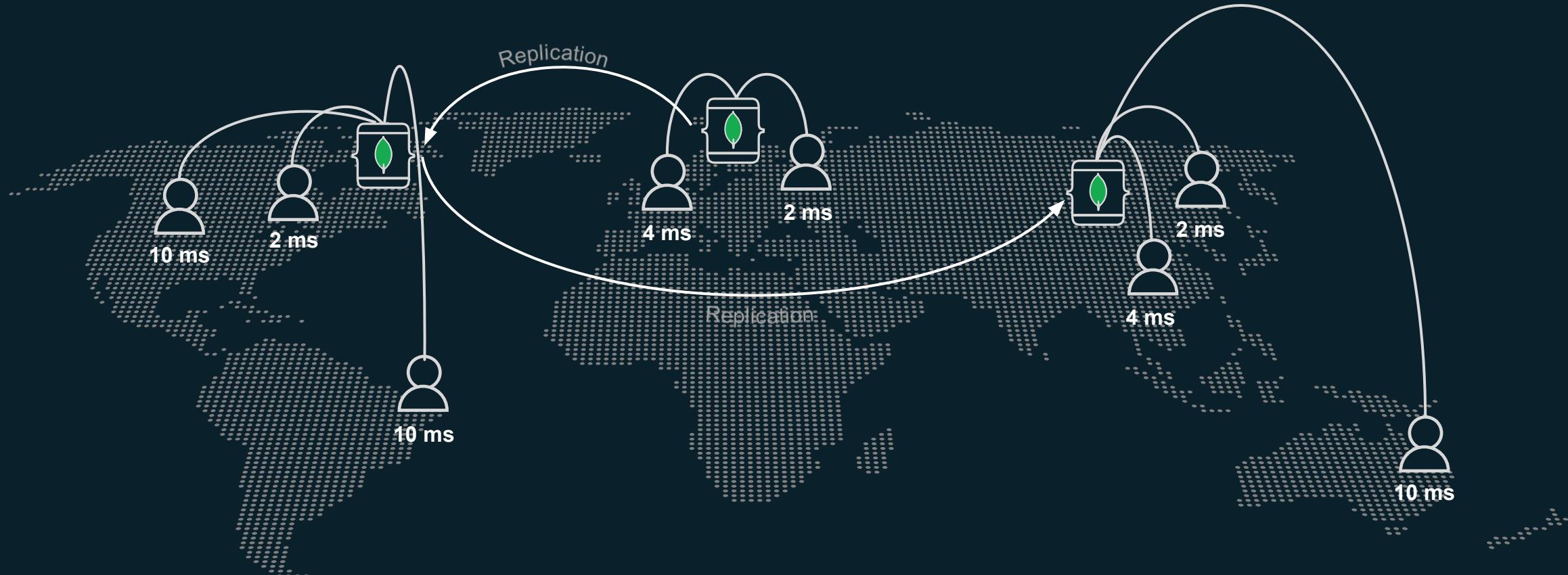


Managed backup



Cloud agnostic

Serving global audiences with MongoDB



Atlas Demo

Sign Up for Free

<https://cloud.mongodb.com/>

The screenshot shows the MongoDB Atlas sign-up interface. At the top, there's a navigation bar with links for DOCS, LEARN, WHAT'S MONGODB?, BLOG, and LOGIN. On the right side of the header are icons for phone, search, and a green 'Get MongoDB' button. Below the header, the main navigation menu includes SOLUTIONS, CLOUD, CUSTOMERS, RESOURCES, and ABOUT US. The main content area features the MongoDB logo and the tagline 'FOR GIANT IDEAS'. A large heading 'MongoDB Atlas' is displayed, followed by a paragraph describing the service: 'Move faster with an automated cloud MongoDB service built for agile teams who'd rather spend their time building apps than managing databases. Available on AWS, Azure, and GCP.' Below this text is a green 'Get started free' button. Further down, there's a link 'Already have an account? Log in here →'. A modal window titled 'Cloud Provider & Region' is open, showing options for AWS, Google Cloud, and Azure. It lists regions for each provider, with 'N. Virginia (us-east-1)' selected under AWS. The modal also includes a section for 'Configure cross-region replication (M10 and up)' with a toggle switch set to 'OFF'.

Break

(15 mins)

- Create an account on MongoDB Atlas (<http://cloud.mongodb.com>)
 - Deploy a free M0 cluster in GCP Belgium
- Create a user (in 'Database Access') and add entry to IP whitelist
 - Install Sample Data into cluster
- Download Compass from
tinyurl.com/mongodb-compass
(Optional)

FACEIT on MongoDB Atlas

Let us tell you a story

Who are we?

Emanuele Massara

<https://uk.linkedin.com/in/emanuelemassara>

<https://twitter.com/emas80>



Bradley Wilson-Hunt

<https://uk.linkedin.com/in/bradley-wilson-hunt-342153109>

[https://twitter.com/braderzwh](https://twitter.com;braderzwh)

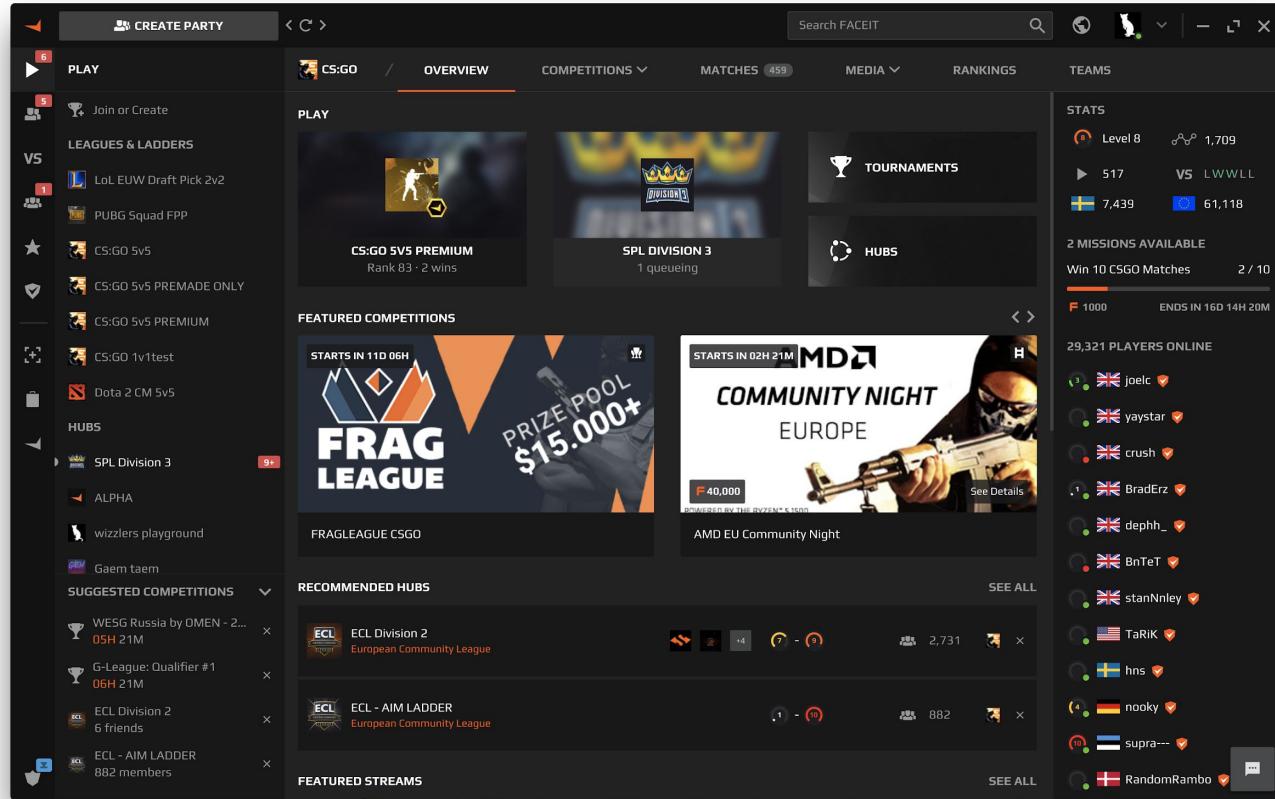


What is FACEIT?

The screenshot shows the FACEIT website interface for the PUBG SQUAD FPP league. The top navigation bar includes options like 'CREATE PARTY', 'PLAY', 'OVERVIEW', 'LADDERS' (which is currently selected), and 'MATCHES'. A search bar and a profile icon are also present. The main content area displays the 'SEASON 2 (ONGOING)' ladder for the Gold division. The ladder table lists players with their names, nationalities, and performance metrics: Position, Player, Matches, Avg. Placement, Kills, K/D Ratio, Points, and Prizes. The top player is 'HalloSenpai' from Russia, followed by 'Houstonrdy' and 'dannyZdog'.

Position	Player	Matches	Avg. Placement	Kills	K/D Ratio	Points	Prizes
1	Hallosenpai	109	4.06	279	3.32	2,032	F 20,000
2	Houstonrdy	108	4.54	243	3.08	1,840	F 15,000
3	dannyZdog	106	4.74	154	1.77	1,630	F 10,000
4	atxcry	85	4.02	170	2.88	1,553	F 5,000
5	ScoobyDoo	103	5.28	165	1.96	1,404	F 4,500
6	Sampulax	85	4.69	176	2.51	1,367	F 4,000
7	zetNN	70	4.16	154	2.61	1,250	F 3,500

The leading competitive platform in esports



15,000,000
USERS



125,000
DAILY PEAK
CONCURRENT USERS

40,000,000
MONTHLY
SESSIONS

3,000,000
MONTHLY UNIQUE
VISITORS

A bit of context

Migrating FACEIT platform from AWS to GCP

- Business Target to reduce costs
- Starting a strong partnership

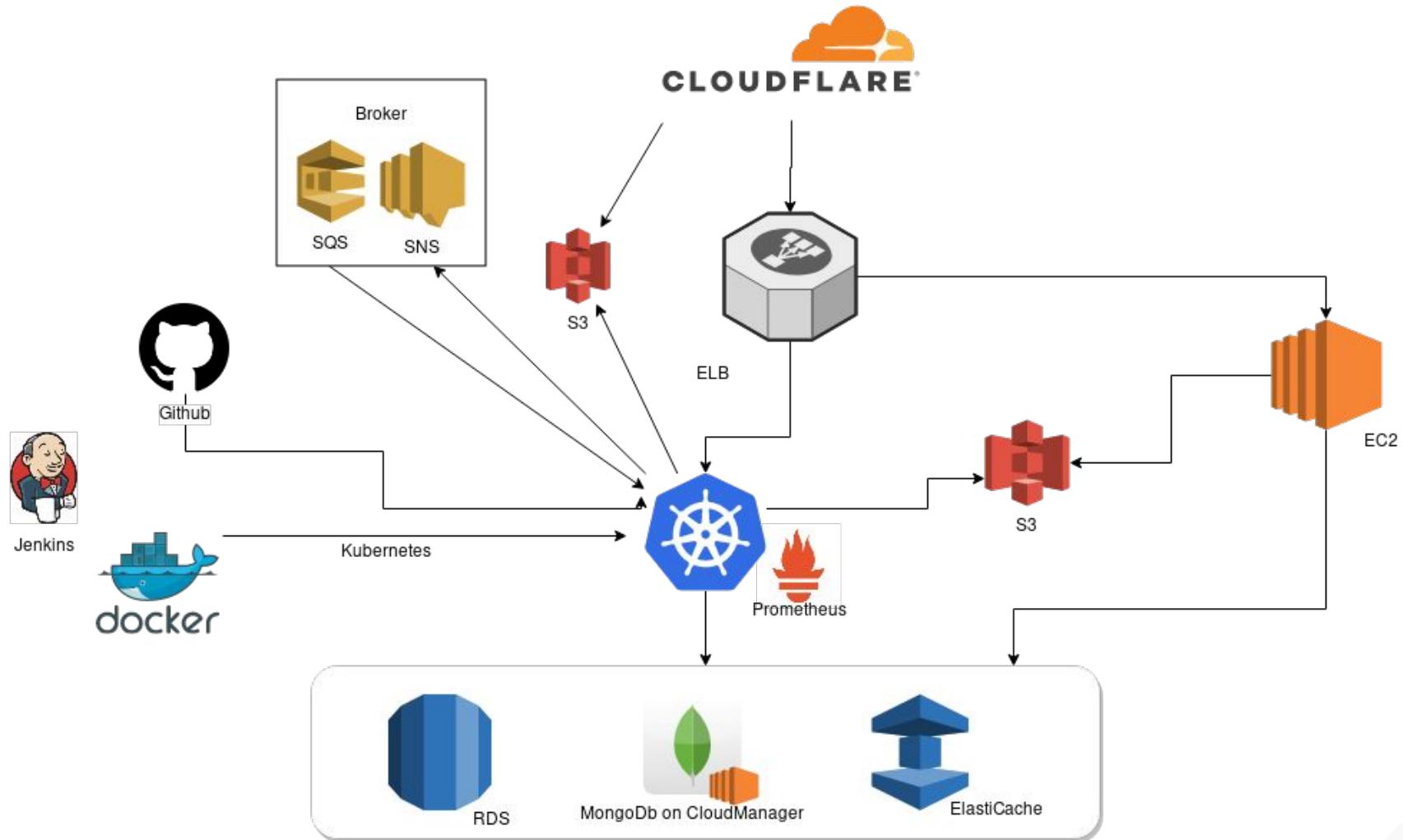
Opportunity to review the setup

- Small company that grew fast

Small ops team (2), many devs (30-40)

- Fast pace of development

Architecture Overview back then



mongoDB setup

mongoDB main database

Started with mongoDB 2.2

- Went through several updates and migrations

CloudManager, 2 replicaset, mongoDB 3.2

- r3.8xlarge (32VCPU, 244GB of RAM), 1.2TB data (uncompressed)
- r3.2xlarge (8CPU, 61GB of RAM), 1TB data (uncompressed)

20+ services (and databases) to migrate

Why mongoDB Atlas?

We evaluated different options:

- Kubernetes
- GCP

Good relations and support

- Current setup was the result of a migration

How did we do it?

Downtime not an option

- 20+ services that need to be swapped at the same time
- All running into Kubernetes

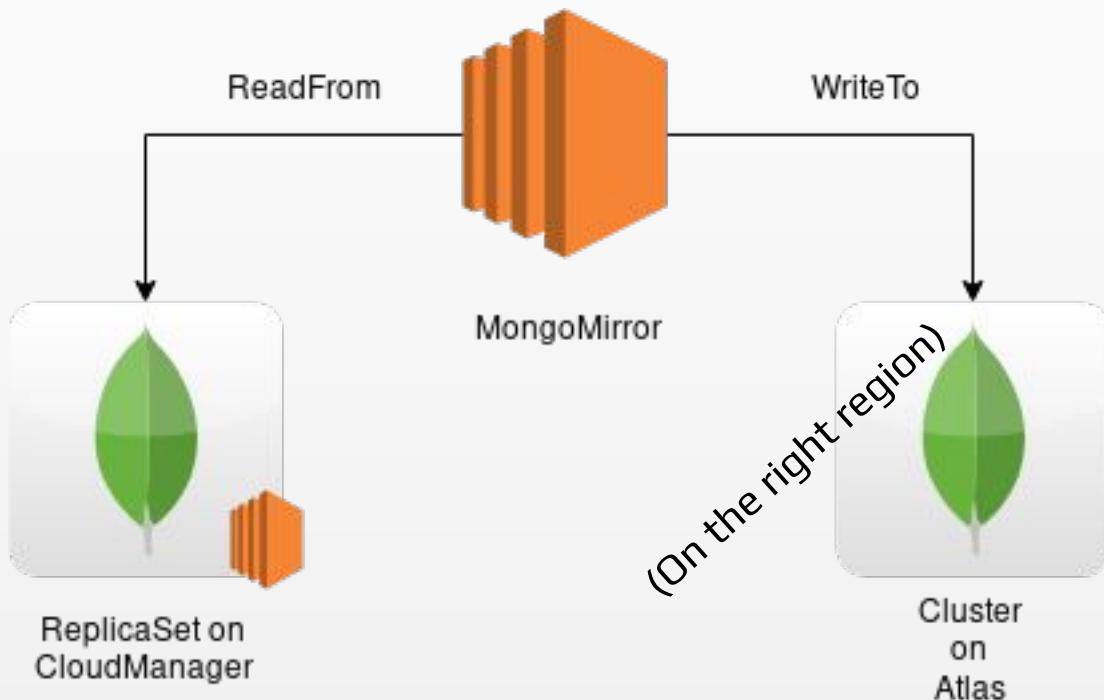
Steps

- Create all the mongo users calling Atlas API
- Start MongoMirror
 - We could split 2 rs in multiple clusters on Atlas
- Restart the applications

Issues faced

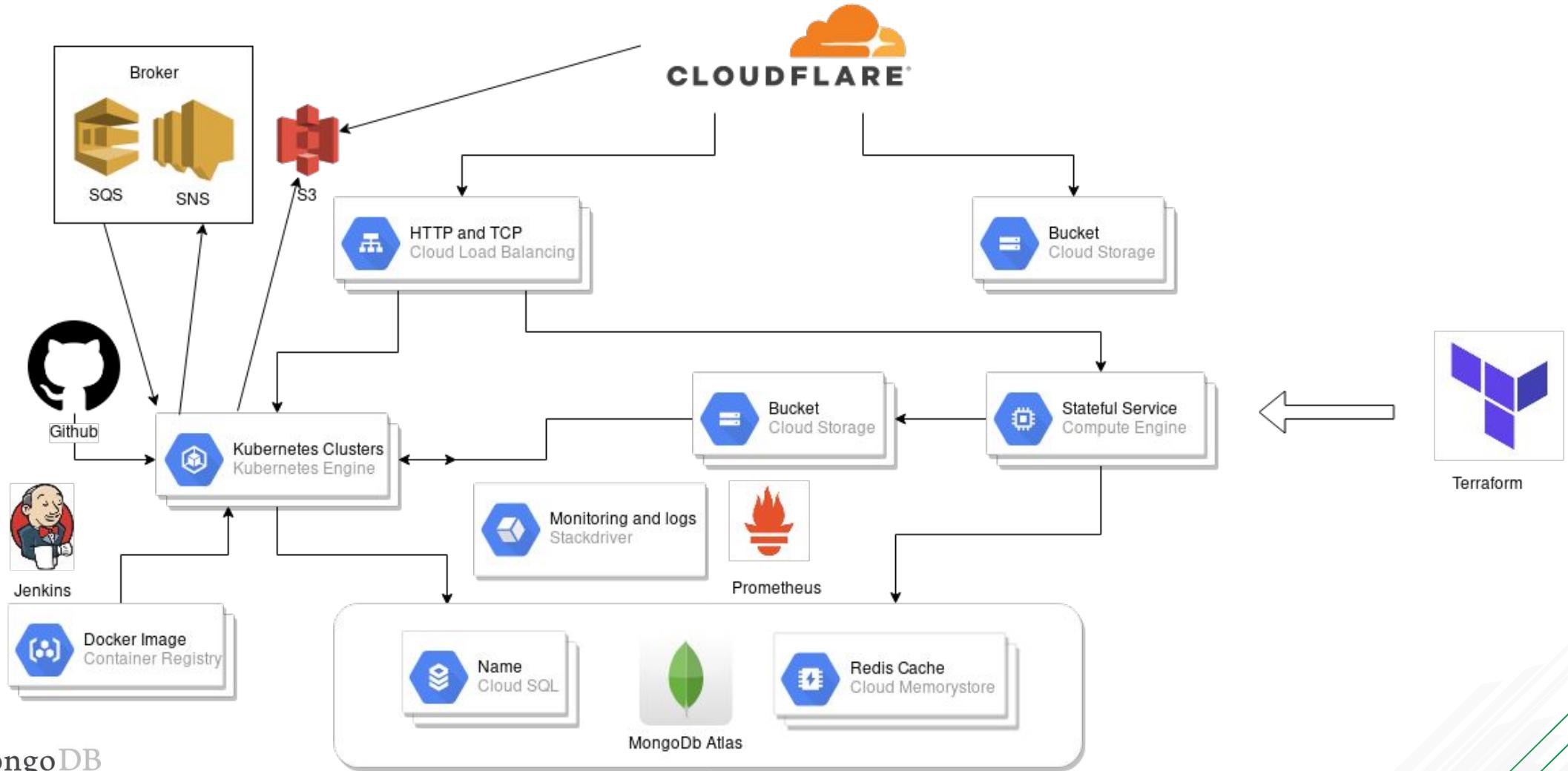
- Initially moved to the wrong region
- problems with indexes, we had to recreate them!

Mongo Mirror



- Instance running
 - We run it on AWS
- You can run it on all the databases, or just on a single one
- It syncs the data and it creates the indexes
- It took between 8 and 12 hours to sync each cluster, because of the indexes

The new FACEIT



Current mongoDB setup

7 clusters for prod

- From M30 to M50
- Split from the initial 2 based on use case/usage
- About 1,8TB (uncompressed)

Fully monitored on Atlas

- Disk size auto increase
- Backups
- Minor version updates and security fixes

Upgrade to 3.4 after the migration

- By simply pressing a button
- 3.6 planned for January, 4.0 for 2020/Q2

How Atlas has improved

Rolling index builds

- It was just announced when we migrated

Maintenance windows

- Still by project

Backups Snapshots on GCP

Query Profiler

Index suggestions

They listen to the feedbacks!



New things we want to try

4.0 and 4.2 migration by 2020/Q2

Terraform for the clusters setup

- UI is good, but scripting is better!

Database user credentials

- Not completely satisfied on how we are handling it now
- Stronger integration with vault.

Transactions

Global Clusters

Thank you!

Ps. We're hiring!

<https://corporate.faceit.com/available-jobs/>

Atlas Workshop

EXERCISE: MONGODB COMPASS & CRUD

PreReqs for Exercise...

Deploy a MongoDB Atlas Cluster

- free-tier M0
- Go to <https://www.mongodb.com/cloud/atlas>

@Google in Belgium

Secure the cluster, e.g. user and IP whitelisting

Understand the basic features of Atlas

Let's do it!

Download and Install Compass Enterprise:

- tinyurl.com/mongodb-compass
- Connect to Atlas with Compass

or

Use Atlas Data Explorer & Aggregation Builder

- Used in the web browser, no install
- Can do 80+% of this session

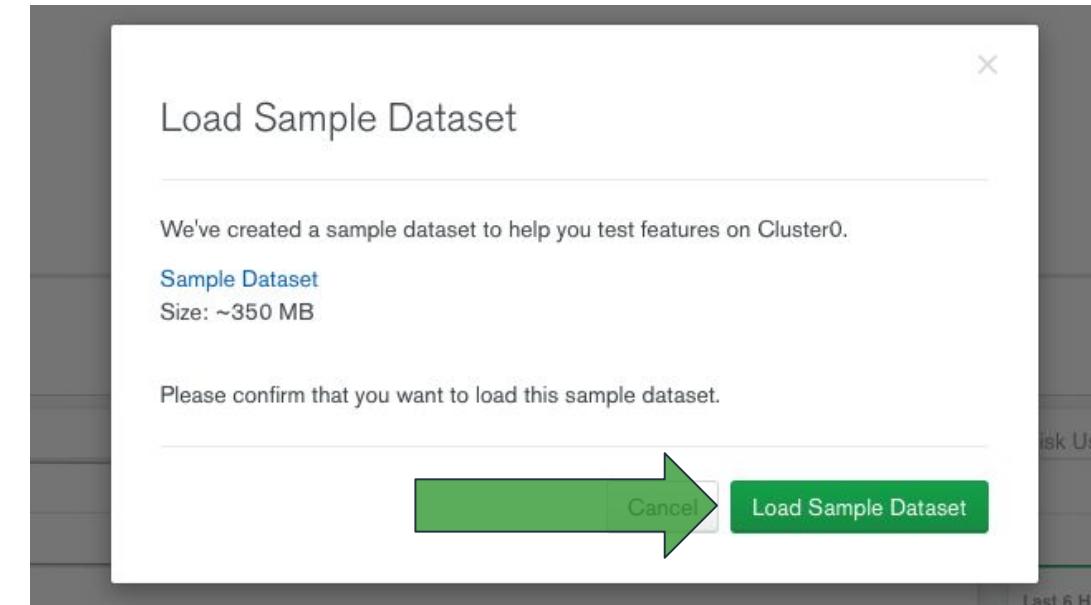
Import the sample data

The screenshot shows the MongoDB Cloud interface for Cluster0. The cluster details are as follows:

- Cluster0** (Status: Green)
- Version 4.0.9
- CONNECT**, **METRICS**, **COLLECTIONS** buttons
- BACKUPS**: Inactive
- INSTANCE SIZE**: M50 (General)
- REGION**: AWS / N. Virginia (us-east-1)
- TYPE**: Sharded Cluster (highlighted with a green arrow)
- LINKED STITCH APP**: None Linked
- BI CONNECTOR**: Disabled

A context menu is open over the cluster details, listing the following options:

- Edit Configuration
- Command Line Tools
- Migrate Data to this Cluster
- Load Sample Dataset** (highlighted with a green arrow)
- Download Logs
- Test Failover
- Pause Cluster
- Terminate



We will now be using `sample_mflix.movies` database from now on

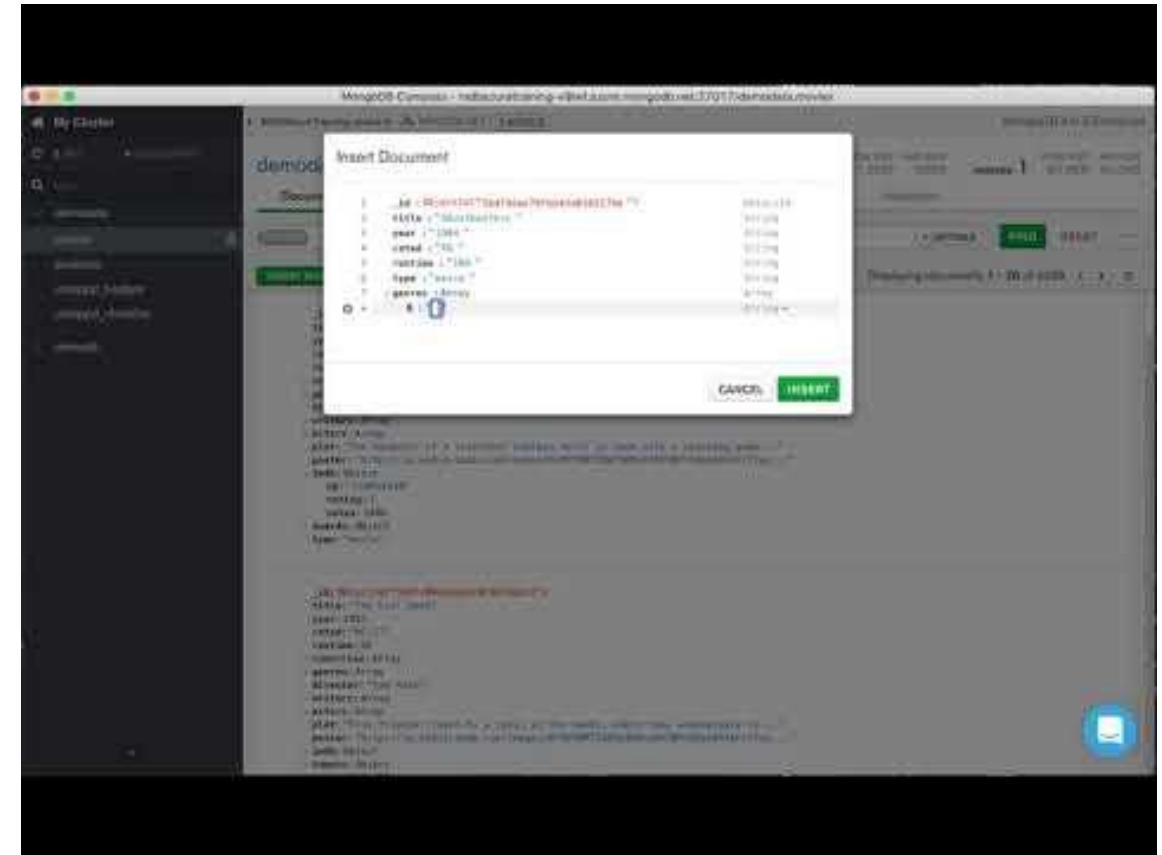
Connection Strings

Note: If you click “Connect” on the cluster in Atlas, then choose “Compass,” it will automatically detect the connection string and fill out the details for you when you launch Compass

Key	Value
Server	<<insert your server details here>>
SRV Record	Yes
Authentication	Username / Password
Username	<<insert your username that you created>>
Password	<<insert your password that you created>>
Auth DB	admin

CRUD: Let's create

```
db.movies.insertOne(  
{  
    "title": "Ghostbusters",  
    "year": 1984,  
    "rated": "PG",  
    "runtime": 105,  
    "type": "movie",  
    "genres": ["comedy", "action"] ,  
    "director": "Ivan Reitman",  
    "writers": ["Dan Aykroyd", "Harold Ramis"]  
}  
)
```



CRUD: LET'S READ...

```
db.movies.findOne()
```

```
db.movies.find().pretty()
```

```
db.movies.find({title:"Ghostbusters"})
```

The screenshot shows the MongoDB Compass interface connected to a MongoDB cluster named 'MDBAzureTraining-shard-0'. The 'demodata' database is selected, and the 'movies' collection is viewed. The 'Documents' tab is active, showing two documents. The first document is for 'Show Boat' (1951), and the second is for 'The Evil Dead' (1981). Both documents are displayed in a pretty-printed JSON format, showing fields such as title, year, rated, runtime, countries, genres, director, writers, actors, plot, poster, and awards.

```
_id:ObjectId("5b9fc096a3ae5c0c947db0dc")
title:"Show Boat"
year:1951
rated:"PASSED"
runtime:108
> countries:Array
> genres:Array
> director:"George Sidney"
> writers:Array
> actors:Array
plot:"The daughter of a riverboat captain falls in love with a charming gamb...)"
poster:"http://ia.media-imdb.com/images/MV5BMjQ2NTQyMjI1NV5BMjQyNzUxXzZTn...)"
> imdb:Object
> awards:Object
type:"movie"

_id:ObjectId("5b9fc096a3ae5c0c947db0dc2")
title:"The Evil Dead"
year:1981
rated:"NC-17"
runtime:85
> countries:Array
> genres:Array
> director:"Sam Raimi"
> writers:Array
> actors:Array
plot:"Five friends travel to a cabin in the woods, where they unknowingly re...)"
poster:"http://ia.media-imdb.com/images/MV5MTIwMDk4NzBhNzQxXzZTn...)"
> imdb:Object
> tomato:Object
metacritic:70
> awards:Object
type:"movie"
```

CRUD: LET'S UPDATE...

```
db.movies.updateOne (  
  {  
    title: "Ghostbusters"  
  },  
  {  
    $set: {  
      imdb: { id: "tt0087332", rating: 7.8, votes: 312798 }  
    }  
  }  
)
```

The screenshot shows the MongoDB Compass interface with the following details:

- Filter:** {title:"Ghostbusters"}
- Options:** Options, Find, Reset, More
- Display:** Insert Document, View (List, Table), Displaying documents 1 - 1 of 1
- Document Structure:**
 - _id: ObjectId("5bd74eae70fda43a0185279e")
 - title : "Ghostbusters "
 - year : 1984
 - rated : "PG "
 - runtime : 105
 - type : "movie "
 - director : "Ivan Reitman "
 - writers : Array
 - > genres : Array
 - + Add Array Element To genres
 - + Add Field After genres
- Buttons:** Cancel, Update

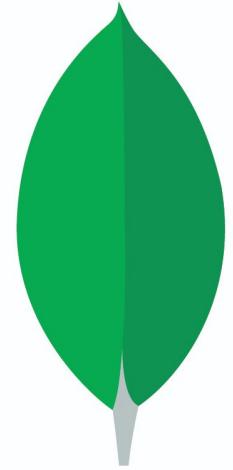
CRUD: LET'S DELETE...

The screenshot shows the MongoDB Compass interface. At the top, there's a search bar with the filter condition `{title: "Ghostbusters"}`. Below the search bar are buttons for `OPTIONS`, `FIND`, `RESET`, and three dots. The main area displays a single document in a list view. The document details are as follows:

- `_id: ObjectId("5bd74eae70fda43a0185279e")`
- `title: "Ghostbusters"`
- `year: 1984`
- `rated: "PG"`
- `runtime: 105`
- `type: "movie"`
- `director: "Ivan Reitman"`
- `> writers: Array`
- `> genres: Array`
- `> imdb: Object`

At the bottom right of the interface, it says "Displaying documents 1 - 1 of 1".

```
db.movies.deleteOne (  
 {  
   title: "Ghostbusters"  
 }  
)
```



mongoDB®



CRUD: Final Quiz

Query	What's the Question?

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	“Comedy” and “Drama”

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	“Comedy” and “Drama”
db.movies.find({"imdb.rating": {\$gt: 8.0}, rated: "PG"})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	“Comedy” and “Drama”
db.movies.find({"imdb.rating": {\$gt: 8.0}, rated: "PG"})	IMDB Rating > 8.0 and PG Rating

CRUD: Final Quiz

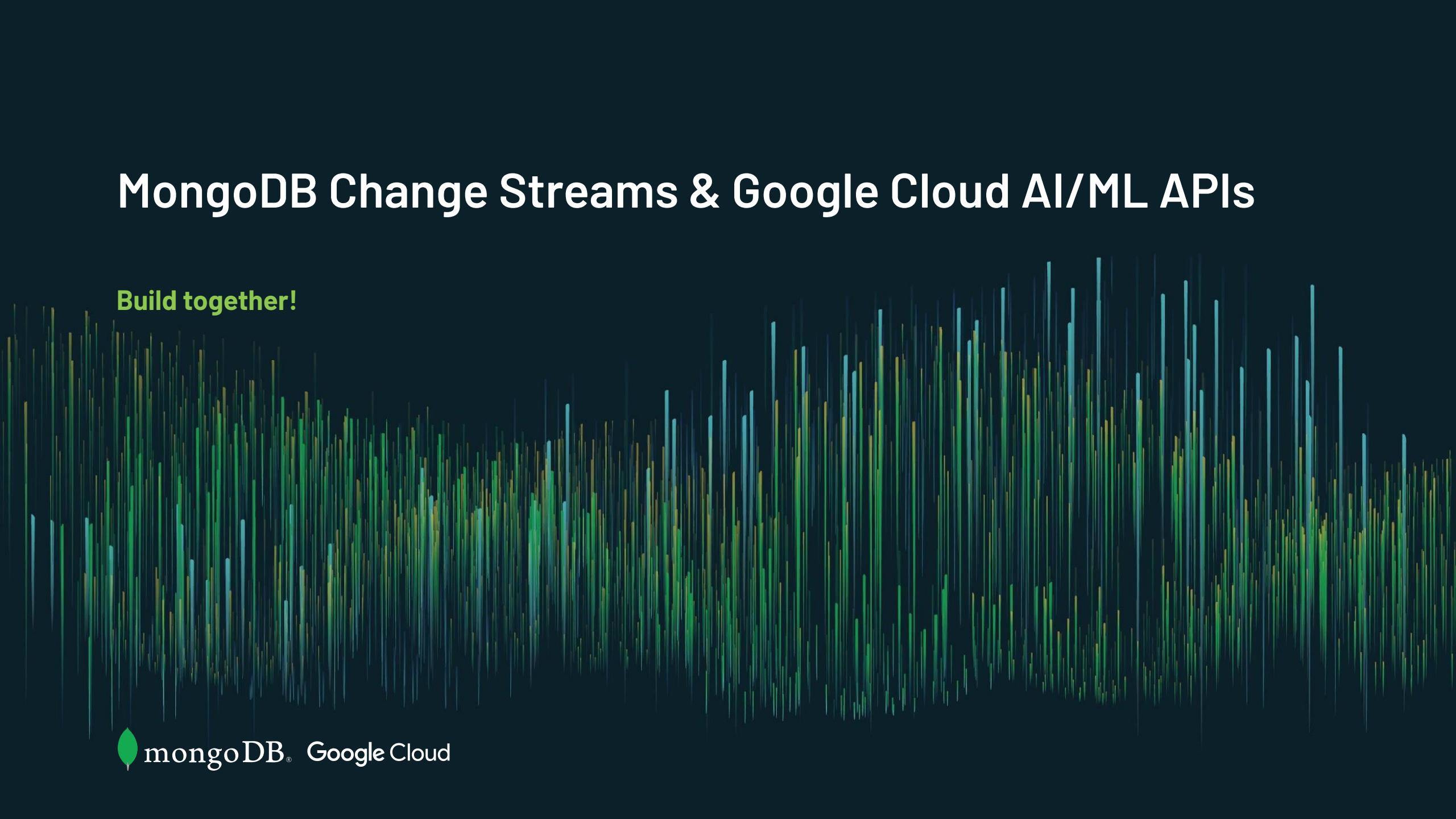
Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	“Comedy” and “Drama”
db.movies.find({"imdb.rating": {\$gt: 8.0}, rated: "PG"})	IMDB Rating > 8.0 and PG Rating
db.movies.find({title: {\$regex: '^Dr. Strangelove'}})	

CRUD: Final Quiz

Query	What's the Question?
db.movies.find({year: 1987})	movies from 1987
db.movies.find({genres: "Comedy"})	“Comedy” as one of their genres
db.movies.find({genres: ["Comedy"]})	“Comedy” as only genre
db.movies.find({genres: {\$in: ["Comedy", "Drama"]}})	“Comedy” or “Drama”
db.movies.find({genres: {\$all: ["Comedy", "Drama"]}})	“Comedy” and “Drama”
db.movies.find({"imdb.rating": {\$gt: 8.0}, rated: "PG"})	IMDB Rating > 8.0 and PG Rating
db.movies.find({title: {\$regex: '^Dr. Strangelove'}})	Title starting with “Dr. Strangelove”

MongoDB Change Streams & Google Cloud AI/ML APIs

Build together!



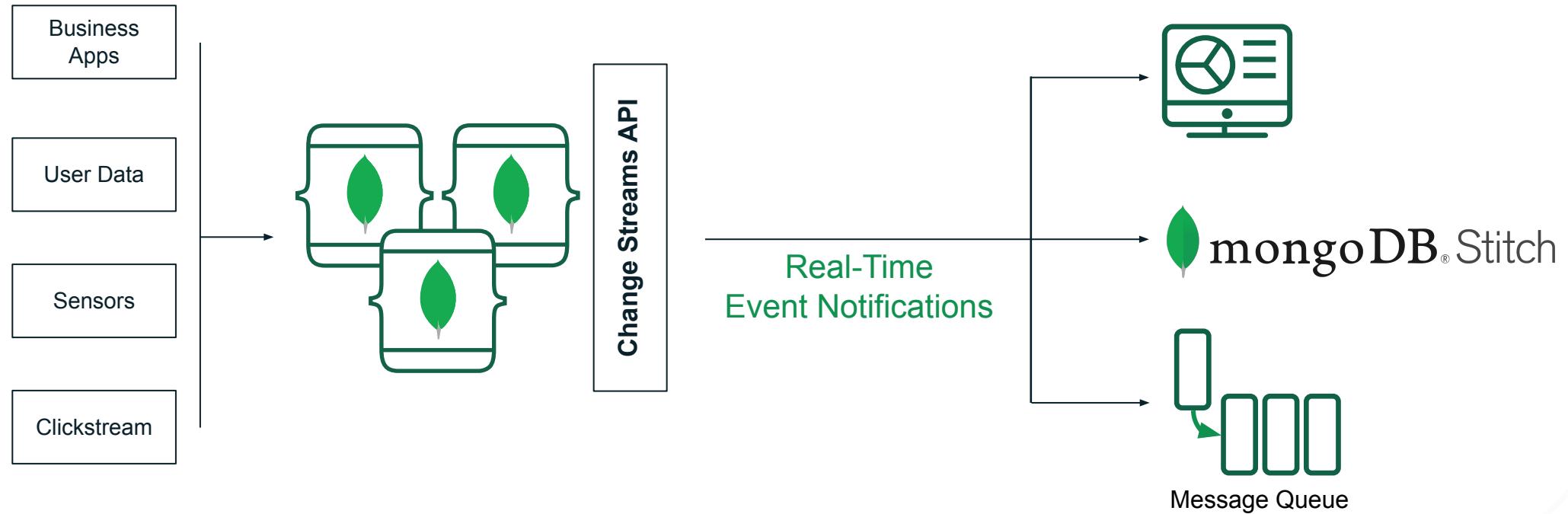
mongoDB® Google Cloud

After this exercise, you should be able to...

- Understand what MongoDB Change Streams are
- Deploy compute instances within Google Cloud
- Find and enable AI/ML APIs within Google Cloud
- Connect to Change Streams
- Call authenticated Google Cloud APIs and update MongoDB documents accordingly

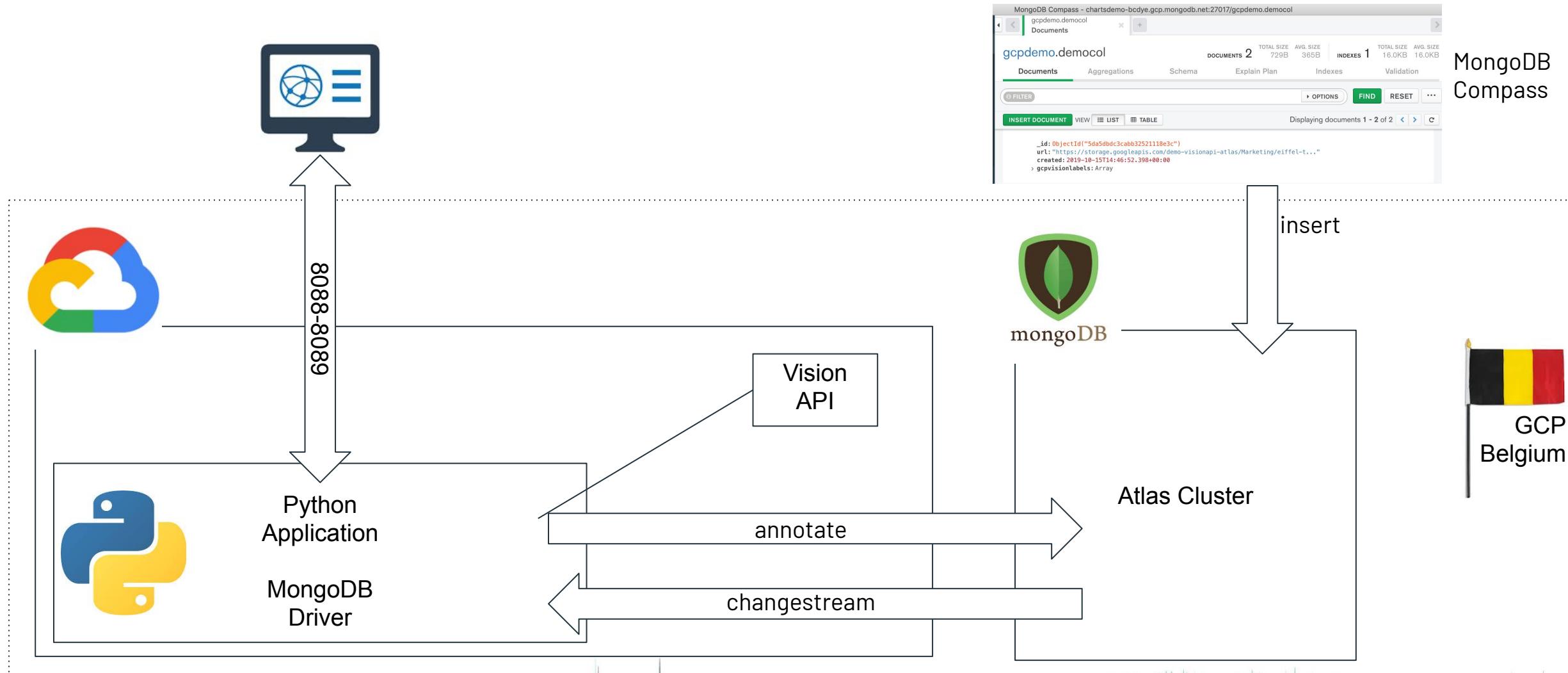


Versatile: MongoDB Change Streams



Enabling developers to build
reactive, real-time services

GCP MongoDB Integration: What we will build





Change Stream in Python

```
21
22     # configure connection to mongodb
23     conn = pymongo.MongoClient(cfg['DEFAULT']['_URI'])
24     handle = conn[cfg['DEFAULT']['_DBNAME']][cfg['DEFAULT']['_COLNAME']]
25
26     " . . . "
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74     # connect to a change stream
75     change_stream = handle.watch()
76     # every change in the db
77     for change in change_stream:
78         # can be insert, update, replace (Compass)
79         if change["operationType"] == "insert":
80             # make sure it had a URL attribute
81             if "url" in change["fullDocument"]:
```

Homework: Change Streams & GCP AI/ML APIs

- Follow along at:
 - <https://github.com/rbohan/MongoDBAtlas-GCP-AIMLv2>
 - Shortened Link: <https://tinyurl.com/mdb-code>
- Scroll to the “Low Level Readme” section to follow along step-by-step

Make it pretty

Host your image with
Google Cloud Cloud
Storage Buckets

Google Cloud CLIs

Natural Language API

Document design

Extra Credit

Find another Google
Cloud AI/ML API

Include authentication

Google Assistant

Instance authentication
for API rather than .json
file

**Deadline:
COB, Monday December 16, 2019**

To: natasha@mongodb.com
Subject: GCP London contest
Include: github/gitlab address.

Prize to Win

EXERCISE: AGGREGATION FRAMEWORK

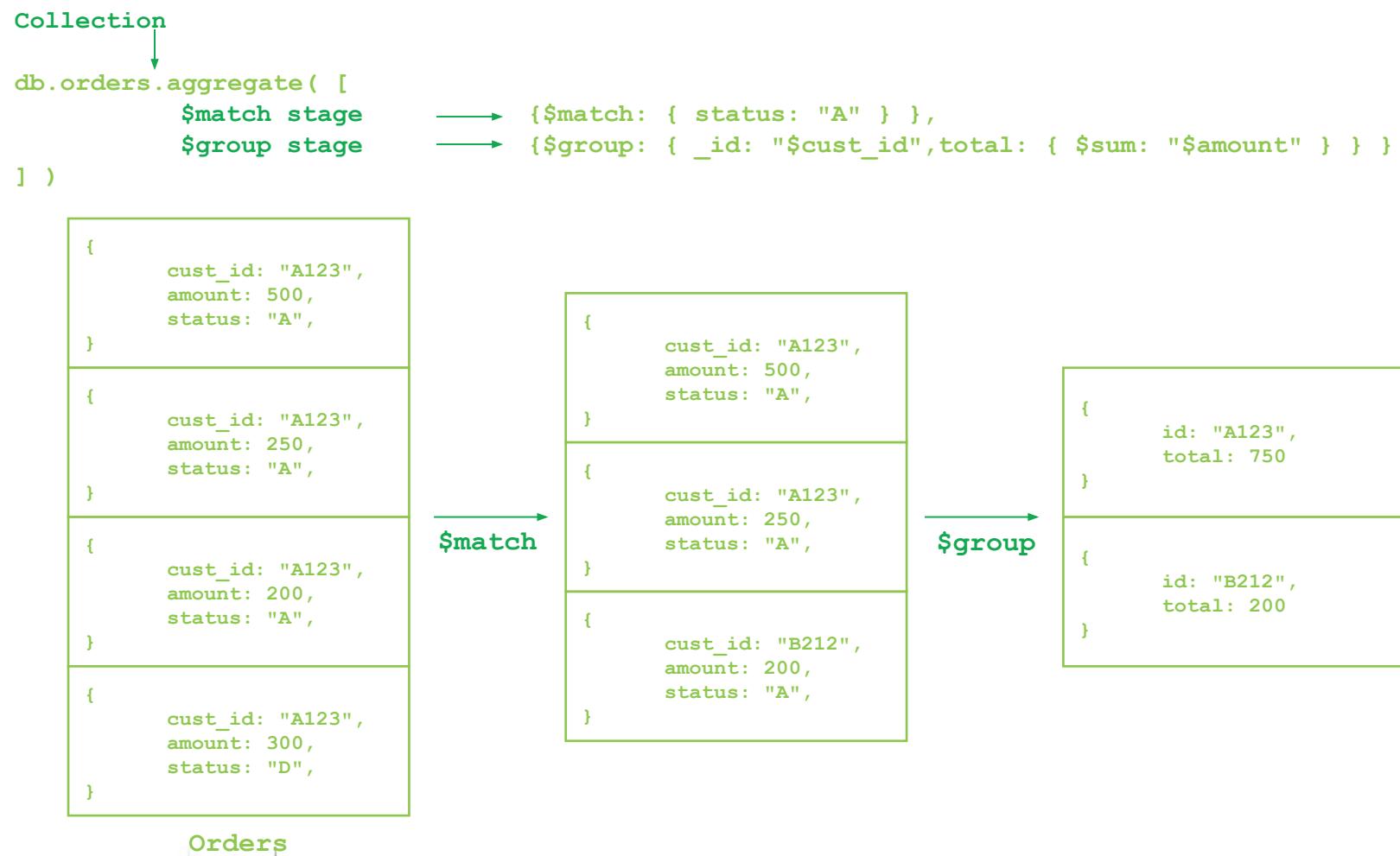
At the end of this Exercise, you should be able to...

- Have a basic understanding of what the aggregation framework is
- Write some basic queries using the aggregation framework

Aggregations

Advanced data processing pipeline for transformations and analytics

- Multiple stages
- Similar to a unix pipe
 - Build complex pipeline by chaining commands together
- Rich Expressions



Aggregation Features

A feature rich analytical framework

Pipeline Stages

- \$match
- \$group
- \$facet
- \$geoNear
- \$graphLookup
- \$lookup
- \$project
- \$sort
- \$unwind

Operators

- Mathematical
 - \$add, \$abs, \$subtract, \$multiply, \$divide, \$log, \$log10, \$stdDevPop, \$stdDevSam, \$avg, \$sqrt, \$pow, \$sum, \$zip, etc.
- Array
 - \$push, \$reduce, \$reverseArray, \$addToSet, \$arrayElemAt, \$slice, etc.
- Conditionals
 - \$and, \$or, \$eq, \$lt, \$lte, \$gt, \$gte, \$cmp, \$cond, \$switch, \$in, etc
- Date
 - \$dateFromParts, \$dateToParts, \$dateFromString, \$dateToString, \$dayOfMonth, \$isoWeek, \$minute, \$month, \$year, etc.
- String
 - \$toUpperCase, \$toLowerCase, \$substr, \$strcasecmp, \$concat, \$split, etc.
- Laterals
 - \$exp, \$let, \$literal, \$map, \$type, etc

Let's play with
Aggregation Pipeline
Builder in Compass
or Atlas

Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
},  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```

Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
  },  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```

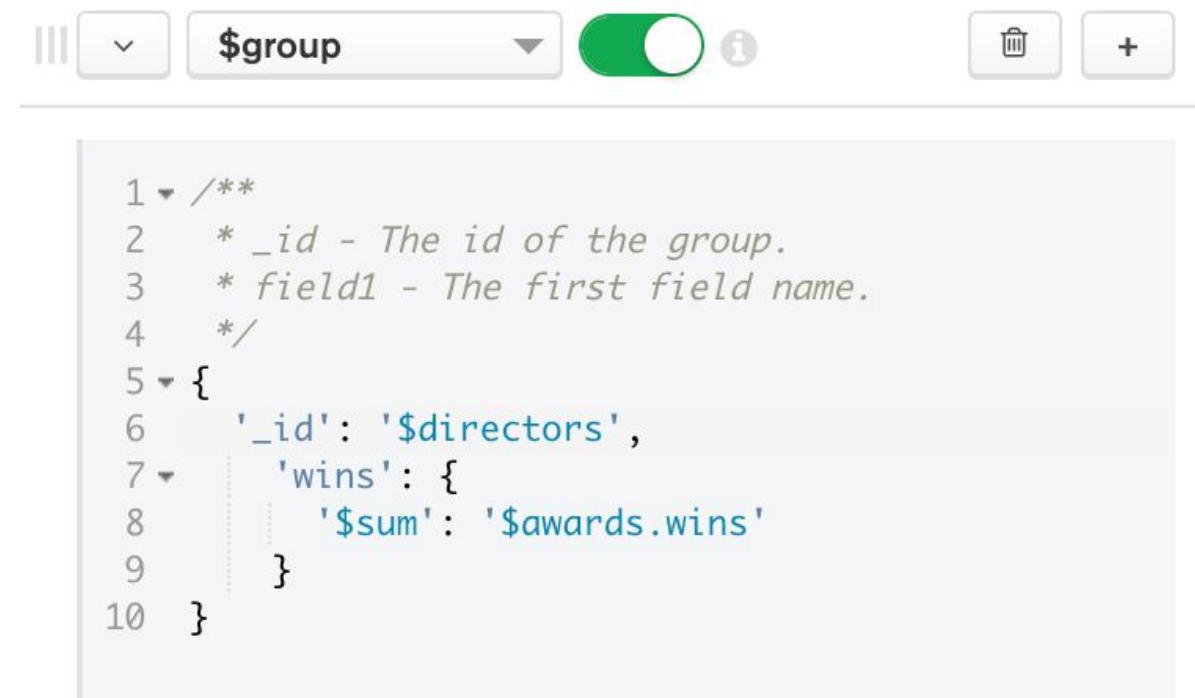
The screenshot shows the MongoDB aggregation pipeline editor interface. At the top, there is a toolbar with a dropdown menu, a search bar containing '\$unwind', a toggle switch (which is green), and three buttons: a trash can, a plus sign, and an info icon. Below the toolbar, the pipeline stages are listed in a code-like syntax. Stage 1 is a comment block starting with '1 /**'. Stage 2 is the '\$unwind' stage, which includes documentation for 'path', 'includeArrayIndex', 'preserveNullAndEmptyArrays', and a path value of '\$directors'. Stage 3 is the '\$group' stage, and Stage 4 is the '\$sort' stage.

```
1 /**
2  * path - Path to the array field.
3  * includeArrayIndex - Optional name for index.
4  * preserveNullAndEmptyArrays - Optional
5  *   toggle to unwind null and empty values.
6 */
7 {
8   path: '$directors'
9 }
```

Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
},  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```



The screenshot shows the MongoDB aggregation pipeline editor interface. At the top, there are buttons for '\$group' (which is selected), '\$group' (dropdown), a toggle switch (green), and a help icon. Below the buttons is a code editor area with numbered lines. The code is identical to the one on the left, representing an aggregation pipeline.

```
1 /**
2  * _id - The id of the group.
3  * field1 - The first field name.
4 */
5 {
6   '_id': '$directors',
7   'wins': {
8     '$sum': '$awards.wins'
9   }
10 }
```

Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
},  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```

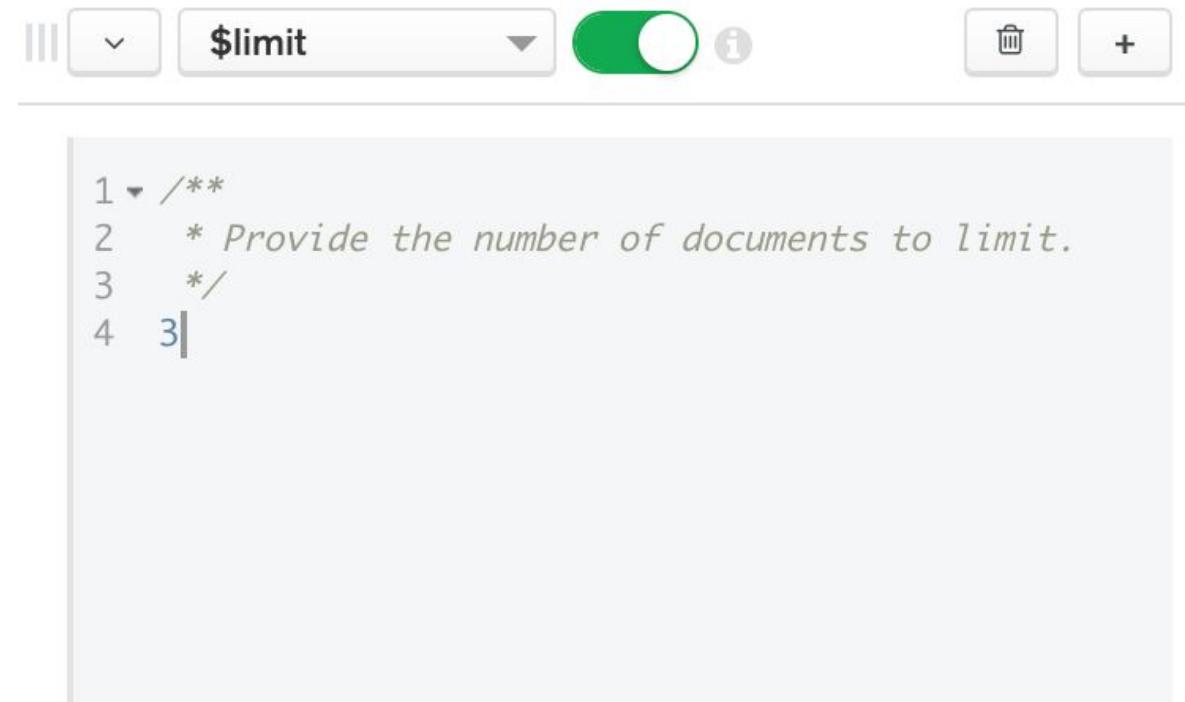
The screenshot shows a MongoDB aggregation pipeline editor interface. At the top, there are buttons for '\$sort' (with a dropdown arrow), a toggle switch (which is green), and a help icon. Below this is a code editor area containing the following code:

```
1 /*  
2   * Provide any number of field/order pairs.  
3   */  
4 {  
5   'wins': -1  
6 }
```

Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
},  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```



Sample Aggregation

Top 3 directors and their win count

```
[  
  { '$unwind': {  
    'path': '$directors'  
  },  
  { '$group': {  
    '_id': '$directors',  
    'wins': {  
      '$sum': '$awards.wins'  
    }  
  }  
},  
  { '$sort': { 'wins': -1 } },  
  { '$limit': 3 }  
]
```

Aggregation Output

```
{ "_id" : "Steven Spielberg", "wins" : 699 }  
{ "_id" : "Martin Scorsese", "wins" : 587 }  
{ "_id" : "Alfonso Cuarèn", "wins" : 577 }
```

MongoDB Charts



MongoDB Charts: Create, Visualize, Share

```
{
  "_id": {"$oid": "5abf2c3dc09c8d2dd5852cf2"},  

  "saleDate": {"$date": "2017-11-08T19:06:53.449Z"},  

  "items": [  

    {  

      "name": "envelopes",  

      "tags": ["stationary", "office", "general"],  

      "price": {"$numberDecimal": "9.83"},  

      "quantity": 10  

    },  

    {  

      "name": "pens",  

      "tags": ["office", "writing", "school", "stationary"],  

      "price": {"$numberDecimal": "73.62"},  

      "quantity": 2  

    },  

    {  

      "name": "laptop",  

      "tags": ["office", "school", "electronics"],  

      "price": {"$numberDecimal": "595.72"},  

      "quantity": 4  

    },  

    {  

      "name": "notepad",  

      "tags": ["office", "writing", "school"],  

      "price": {"$numberDecimal": "34.65"},  

      "quantity": 3  

    },  

    ],  

    "storeLocation": "Seattle",  

    "customer": {  

      "gender": "M",  

      "age": 45,  

      "email": "ugai@we.so",  

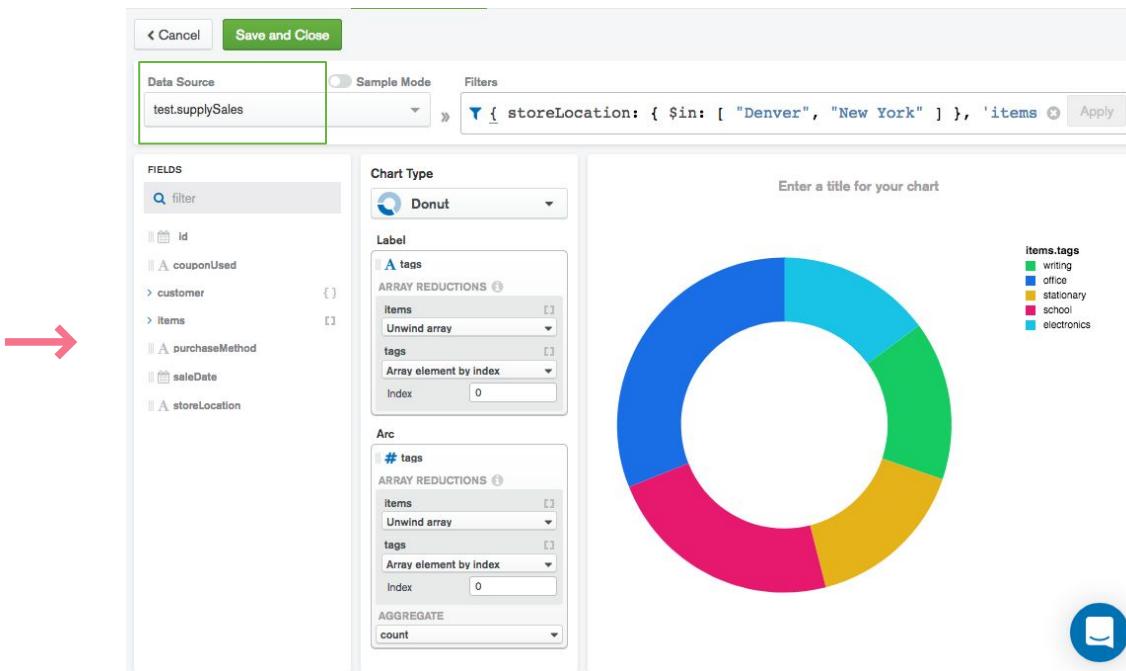
      "satisfaction": 4  

    },  

    "couponUsed": false,  

    "purchaseMethod": "Online"  

  ]
```



Work with complex data

Connect to data sources securely. Filter. Sample. Visualize.



Share dashboards and collaborate

Exercise 5: Charts Visualisation - Step 1

Add the Data Collection as a Data Source

1. In MongoDB Atlas, click on **Charts** on the left side
2. Click the **Data Sources** tab
3. Click **New Data Source**
4. Select your Atlas Deployment in your project
5. Click **Connect**
6. Select the `sample_mflix.movies` collection
7. Click Set **Permissions**, leave the permissions as the default
8. Click **Publish Data Source**

Exercise 5: Charts Visualisation - Step 2

Create a New Dashboard

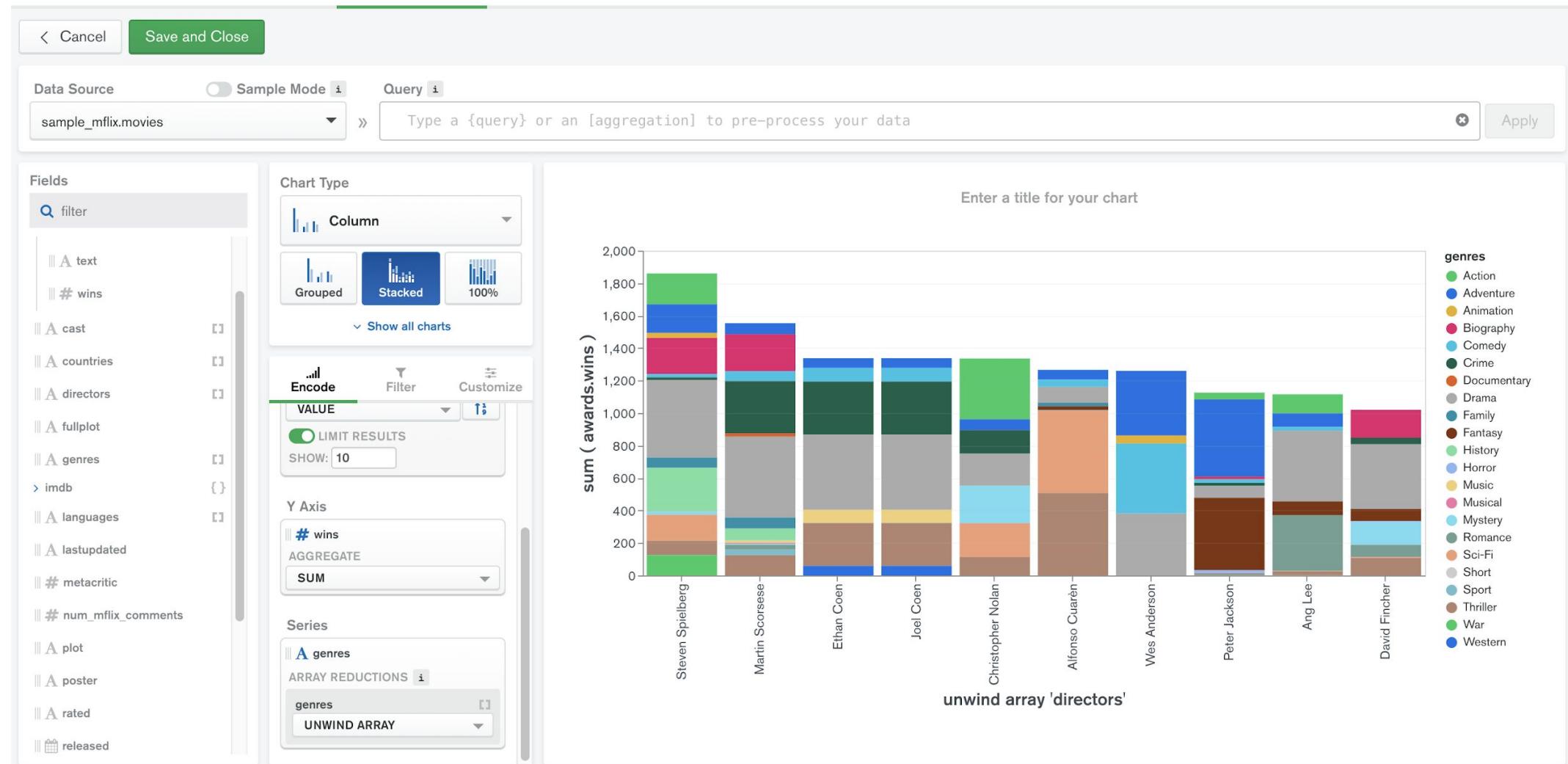
1. Click the **Dashboards** tab
2. Click the **New Dashboard** button
3. Enter the Title: Movie Details
4. Click **Create**

Exercise 5: Charts Visualisation - Step 3

Create Chart Showing Directors with the Most Awards

1. Click **Add Chart**
2. In the Data Source dropdown, select `movies.movies`
3. Select Chart Type: **Column / Stacked**
4. X Axis: `directors`
 - a. Array Reduction: Unwind Array
 - b. Sort By: Descending
 - c. Limit: 10
5. Y Axis: `awards.wins`
 - a. Aggregate: sum
6. Series: `genres`
 - a. Array Reduction: Unwind Array
7. Title: Directors with Most Awards, Split by Genre
8. Click **Save and Close**

Charts Workshop - Summary



Thank you!



mongoDB® Google Cloud

Your MongoDB Team:

PJ Singh (Director Partners NEUR)

pj.singh@mongodb.com

Jim Blackhurst (Solutions Architect)

jim@mongodb.com

@jimthree

Ronan Bohan (Solutions Architect)

ronan@mongodb.com