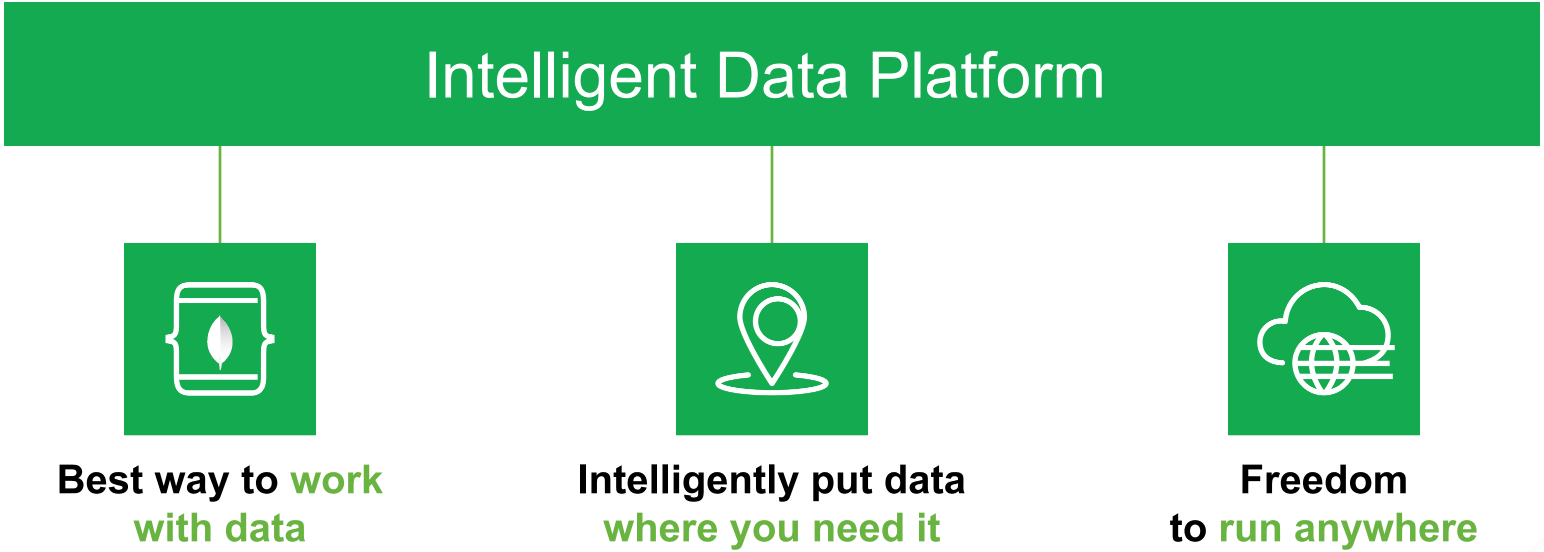
What’s New: MongoDB 4.2

Sam Harley | Senior Solutions Architect

MongoDB: Built for Developer Productivity



Intelligent Data Platform

Best way to work with data

Intelligently put data where you need it

Freedom to run anywhere

The evolution of MongoDB

3.2

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

3.4

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

3.6

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

4.0

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

4.2

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

mongoDB 4.2

ACID Transactions

- Distributed Trx
- Global PiT Reads
- Mutable Shard Key Values

mongoDB 4.2

ACID Transactions

- Distributed Trx
- Global PiT Reads
- Mutable Shard Key Values

Query & Analytics

- Materialized Views
- Wildcard Indexes
- Atlas Data Lake

mongoDB 4.2

ACID Transactions

- Distributed Trx
- Global PiT Reads
- Mutable Shard Key Values

Query & Analytics

- Materialized Views
- Wildcard Indexes
- Atlas Data Lake

Resilience & Scale

- Retryable R/W
- 10x faster stepDown
- Zstandard

mongoDB 4.2

ACID Transactions

- Distributed Trx
- Global PiT Reads
- Mutable Shard Key Values

Query & Analytics

- Materialized Views
- Wildcard Indexes
- Atlas Data Lake

Resilience & Scale

- Retryable R/W
- 10x faster stepDown
- Zstandard

Enterprise Security

- Field Level Encryption
- Multiple CAs
- 3x Lower Auditing Overhead

mongoDB 4.2

ACID Transactions

- Distributed Trx
- Global PiT Reads
- Mutable Shard Key Values

Query & Analytics

- Materialized Views
- Wildcard Indexes
- Atlas Data Lake

Resilience & Scale

- Retryable R/W
- 10x faster stepDown
- Zstandard

Enterprise Security

- Field Level Encryption
- Multiple CAs
- 3x Lower Auditing Overhead

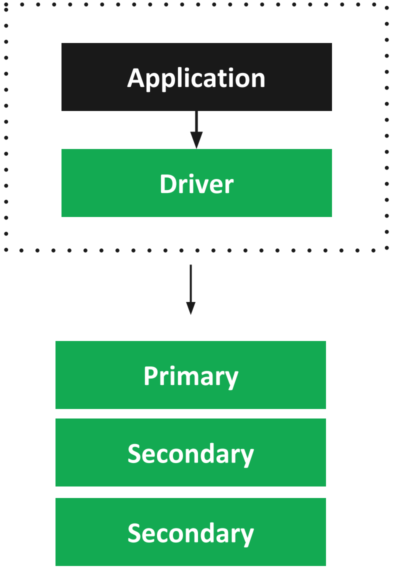
Run Anywhere

- Atlas Auto-Scale
- Atlas Full Text Search
- K8s Integration

Multi-Document ACID Transactions

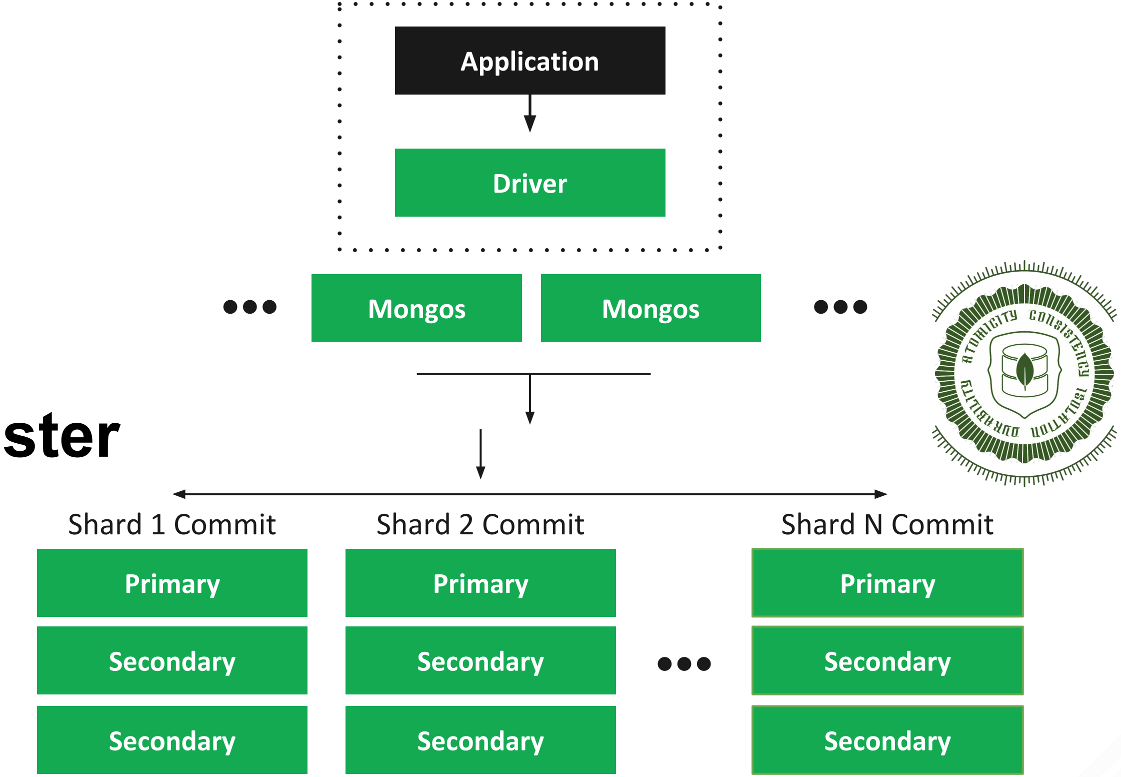
Transactions in 4.0

Replica Set



Distributed Transactions in 4.2

Sharded Cluster



Maintains Transaction Design Goals

Just like relational transactions

- Multi-statement, familiar relational syntax
- Easy to add to any application
- Multiple documents in 1 or many collections and databases
- No difference between transactions in a replica set or across a sharded cluster

ACID guarantees

- Snapshot isolation, all or nothing execution
- No performance impact for single document operations

MongoDB Transactions Syntax

```
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()
```

Natural for developers

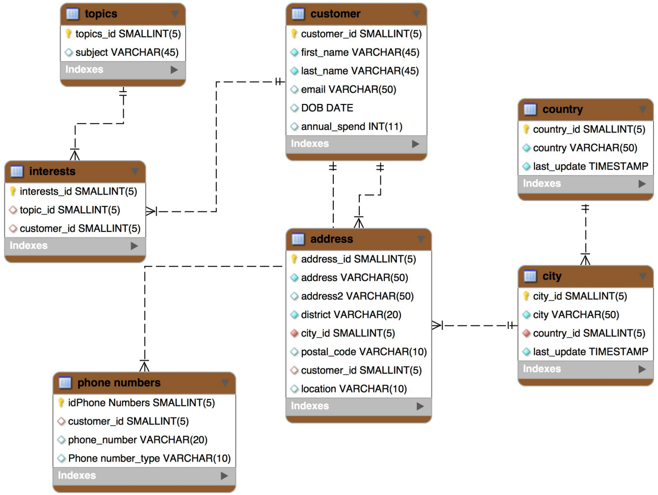
- Idiomatic to the programming language
- Familiar to relational developers
- Simple

MongoDB Transactions Syntax

```
try (ClientSession clientSession = client.startSession()) {

           clientSession.startTransaction();

           collection.insertOne(clientSession, docOne);

           collection.insertOne(clientSession, docTwo);

           clientSession.commitTransaction();

        }
```

Data Models and Transactions

Different databases take **different approaches**



topics
topics_id SMALLINT(5)
subject VARCHAR(45)
Indexes

customer
customer_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
DOB DATE
annual_spend INT(11)
Indexes

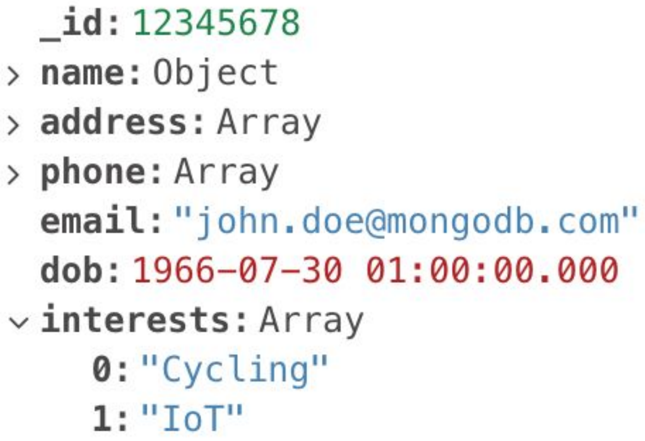
country
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes

interests
interests_id SMALLINT(5)
topic_id SMALLINT(5)
customer_id SMALLINT(5)
Indexes

address
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
customer_id SMALLINT(5)
location VARCHAR(10)
Indexes

city
city_id SMALLINT(5)
city VARCHAR(50)
country_id SMALLINT(5)
last_update TIMESTAMP
Indexes

phone numbers
idPhone Numbers SMALLINT(5)
customer_id SMALLINT(5)
phone_number VARCHAR(20)
Phone number_type VARCHAR(10)
Indexes



```
  _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"
```

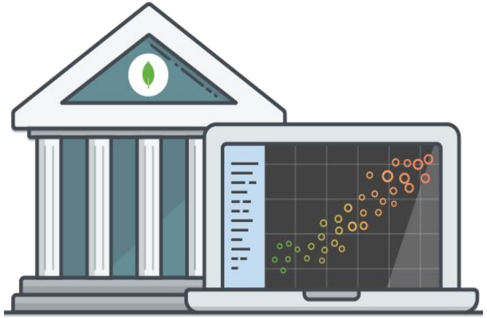
Tabular (Relational) Database

Related data split across multiple records and tables.
Multi-record transactions essential

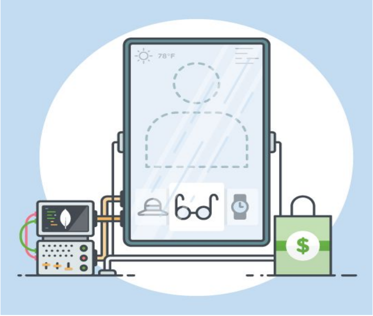
Document Database

Related data contained in a single, rich document.
Transaction scoped to the document

Distributed Transaction Examples



Payment & Trading
Updating security positions



Event Processing
Insert orders, notify suppliers

Telco Billing
Insert CDR, update monthly plan

Ease of Development

Large Transactions

- More than 16MB
- Committed in <60 seconds (default)

Transactions Diagnostics

- Metrics exposed across logs

Error Handling

- Driver side helpers, callback API

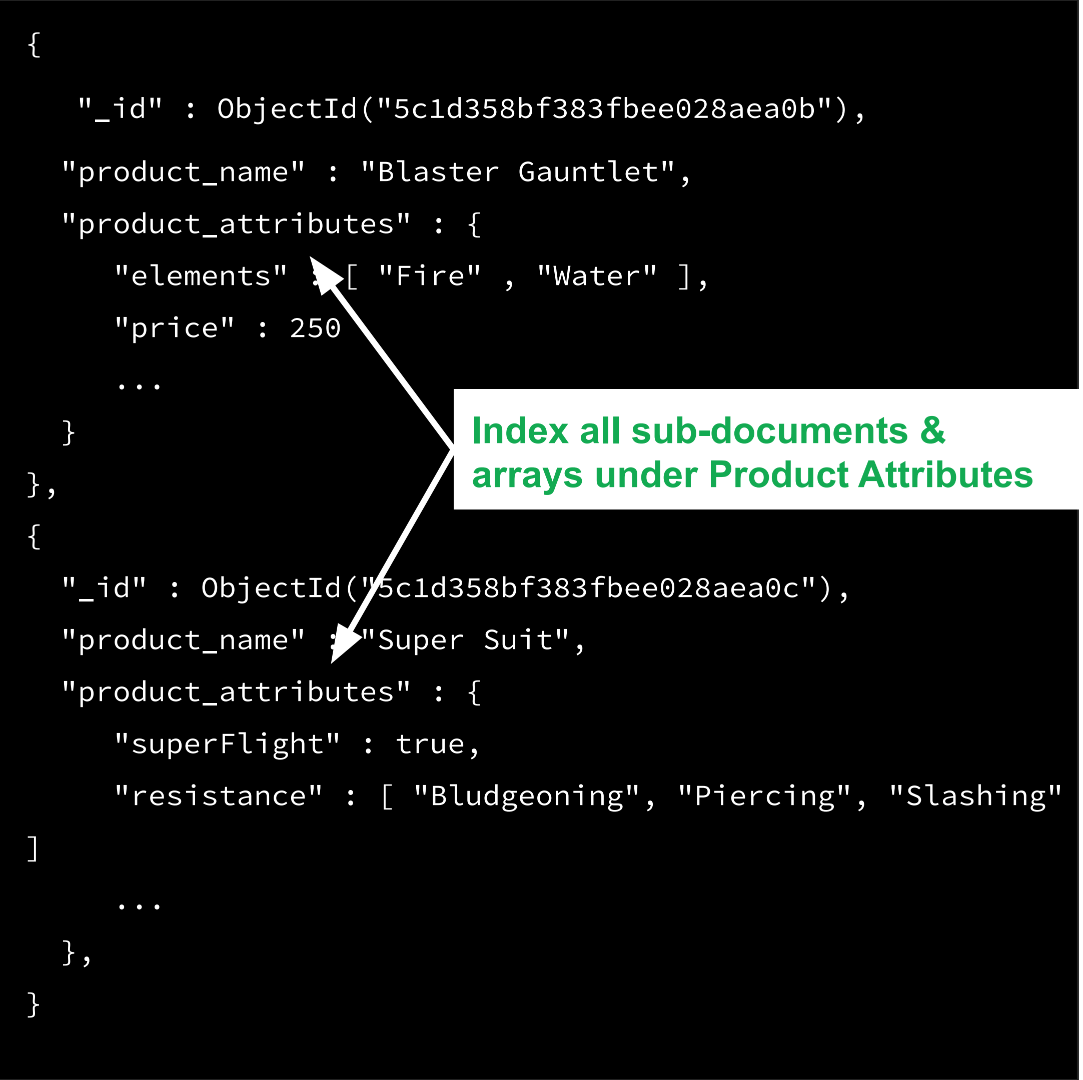
Query & Indexing

On-Demand Materialized Views

- Faster insights on your data: pre-compute and store results of common analytics queries
- With $merge stage aggregation pipeline outputs with existing result sets to increment and enrich views
  - Updated each time the pipeline is run
  - Output to sharded and unsharded collections
  - Define indexes on each view
- With uniqueKey, control how documents are added to the view: Insert, Replace, Merge

$merge Syntax

```
{ $merge: {
    to: "<output-collection>",
    on: { <field1>: 1, ... },
    whenNoMatch: <"insert" | "ignore" | "fail">,
    whenMatch: <"replace" | "keepExisting" | "fail" | "merge" | [ ] >
} }
```



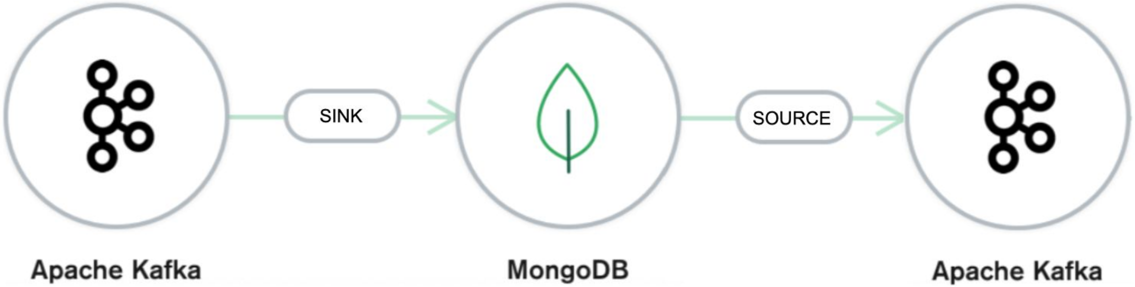
```
{
   "_id" : ObjectId("5c1d358bf383fbee028aea0b"),
  "product_name" : "Blaster Gauntlet",
  "product_attributes" : {
     "elements" : [ "Fire" , "Water" ],
     "price" : 250
     ...
  }
},
{
  "_id" : ObjectId("5c1d358bf383fbee028aea0c"),
  "product_name" : "Super Suit",
  "product_attributes" : {
     "superFlight" : true,
     "resistance" : [ "Bludgeoning", "Piercing", "Slashing"
]
     ...
  },
}
```

Index all sub-documents & arrays under Product Attributes

Wildcard Indexes

- Allow more natural data modeling, avoids pre-defining indexes for every access pattern
  - Polymorphic document structures: Product catalogs, CMS
  - Ad-hoc queries & data exploration
- Define a filter that indexes all matching fields, sub-documents, and arrays
  - Sparse index, omit specific fields
  - Covered queries & collations
  - Strongly consistent: updated atomically with base data

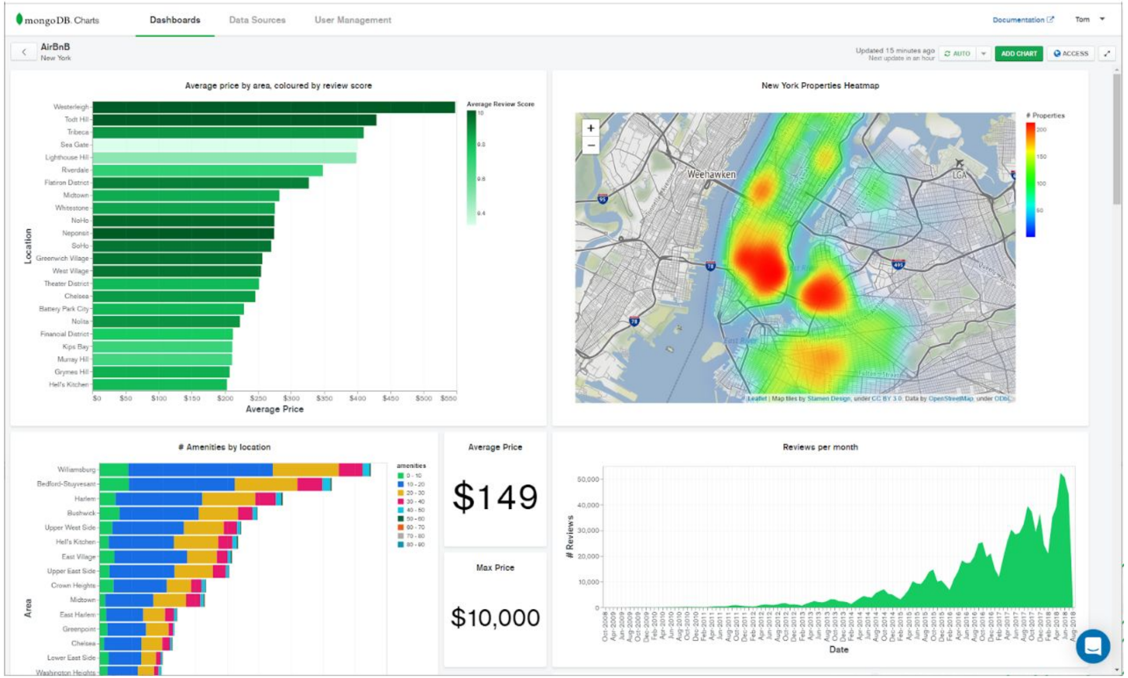
MongoDB Connector for Apache Kafka (Beta)



- Build robust data pipelines for microservices and Event Driven Architectures
- Developed and supported by MongoDB engineers, verified by Confluent
- Supports MongoDB as a sink and a source for Kafka
- Integrate with Change Streams and Atlas triggers to create fully reactive, event driven pipelines

GitHub Confluent Hub

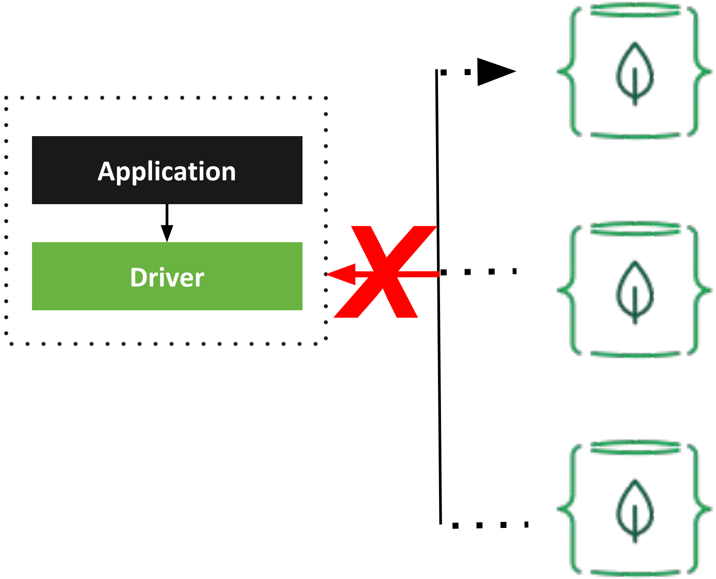
MongoDB Charts General Availability



- The fastest and easiest way to create, share, and embed visualizations of MongoDB data
- Built for the MongoDB document model, run as a service in Atlas or downloadable to run on-premise
- New: geospatial analytics
- Workload isolation
- Embedding charts into web apps to create rich user experiences

Resilience & Scalability

Retryable Reads & Writes



- Moves more of the error handling code from app to drivers and the server
- Failed reads automatically retried by the driver
  - ○ Network errors, primary elections
  - ○ Triggered after 30 seconds, retried once, honors read preferences
- Retryable writes introduced in 3.6 now default for all drivers

Zstandard Compression

- Create more efficient storage infrastructure with lower overhead

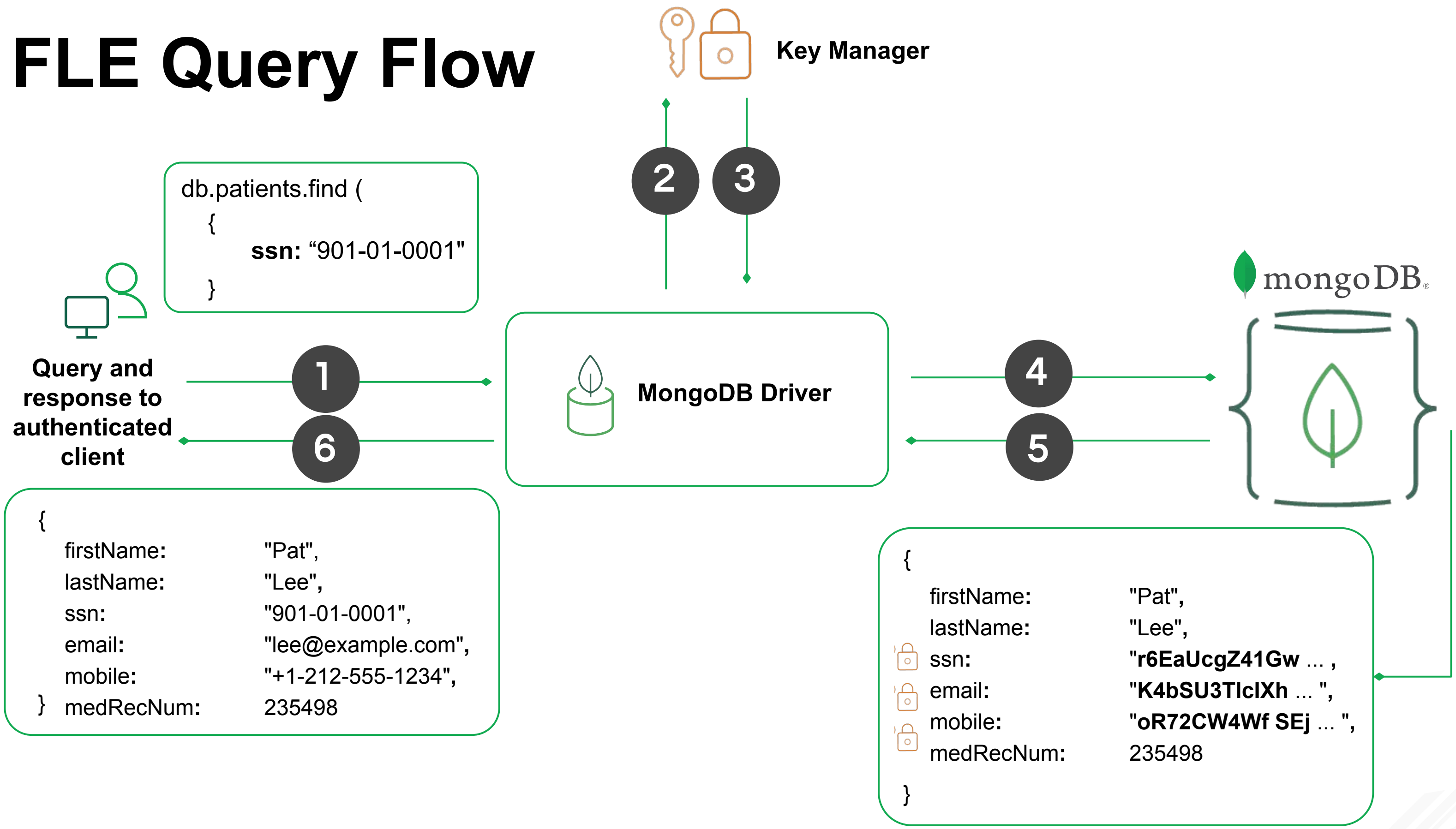
- Up to 55% storage size reduction versus snappy, with lower CPU overhead than zlib
    - Applies to collections and journal
    - WiredTiger and Encrypted storage engine

Enterprise Security

Client-Side Field Level Encryption

- Individual document fields encrypted by own key
- Database only sees ciphertext
- Many Advantages
    - Easy: Automatic and Transparent
    - Separation of Duties: (simplifies move to service-based systems as no service engineers ever see plaintext)
    - Compliant: Regulatory “right to be forgotten”
    - Fast: Minimal performance penalty

FLE Query Flow



```
db.patients.find (
   {
         ssn: “901-01-0001"
   }
```

```
{
   firstName:        "Pat",
   lastName:         "Lee",
   ssn:              "901-01-0001",
   email:            "lee@example.com",
   mobile:           "+1-212-555-1234",
}  medRecNum:        235498
```

```
{
   firstName:        "Pat",
   lastName:         "Lee",
   ssn:              "r6EaUcgZ41Gw ... ,
   email:            "K4bSU3TlclXh ... ",
   mobile:           "oR72CW4Wf SEj ... ",
   medRecNum:        235498
}
```

Encrypted fields always stored, transmitted, and retrieved as ciphertext

Run Anywhere

MongoDB Atlas — Global Cloud Database

Self-service & elastic

Deploy, modify, and upgrade on demand with best-in-class operational automation

Automated database maintenance

Database and infrastructure resources as code

Scale up, out, or down in a few clicks or API calls

Global & cloud-agnostic

Available in 60+ regions across AWS, Azure, GCP

Global clusters for read/write anywhere deployments and multi-region fault tolerance

Easy migrations with a consistent experience across cloud providers

Enterprise-grade security & SLAs

Network isolation, VPC peering, end-to-end encryption, and role-based access controls

Encryption key management, LDAP integration, granular database auditing

SOC 2 / Privacy Shield / HIPAA

Guaranteed reliability with SLAs

Comprehensive monitoring

Deep visibility into 100+ KPIs with proactive alerting

Real-time performance tracking and Performance Advisor

APIs to integrate with monitoring dashboards

Managed backup

Point-in-time data recovery

Queryable backup snapshots

Consistent snapshots of sharded deployments

Cloud data mobility

Stitch: Serverless platform services

Simple, serverless functions for backend logic, service integrations, and APIs

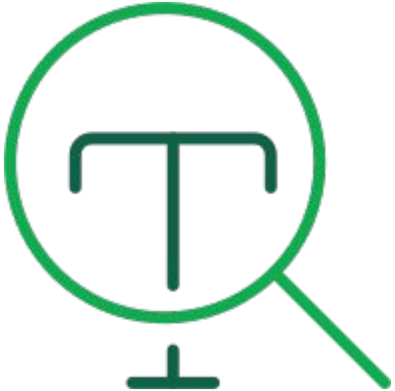
Database access from your frontend secured by straightforward, field-level access rules

Database and authentication triggers to react to changes in real time

MongoDB Atlas Auto Scaling (mid-Summer)

- Elastic scaling of instance sizes so your provisioned capacity responds to demand
  - Monitors key resource utilization metrics
  - Toggle on and off via UI or API
  - Cap peak instance sizes to control costs
- Rolling restarts across replica sets to maintain app availability
- Auto-storage scaling available since 2018

MongoDB Atlas Full Text Search (Beta)



- Adds Full Text Search as a fully managed service to your Atlas cluster
  - Power of Lucene 8, without provisioning and running a separate search platform
  - Integrated with MongoDB Query Language, so no separate APIs to learn
  - Dynamic and static indexing supporting fuzzy & wildcard search, Boolean & compound queries, language analyzers, scoring and snippets
  - Configured via Atlas Data Explorer or API

MongoDB Atlas Data Lake

Analyze data in any format on S3 using the MongoDB Query Language

Multiple Formats, No Schema

Auto-Scale, At Any Scale

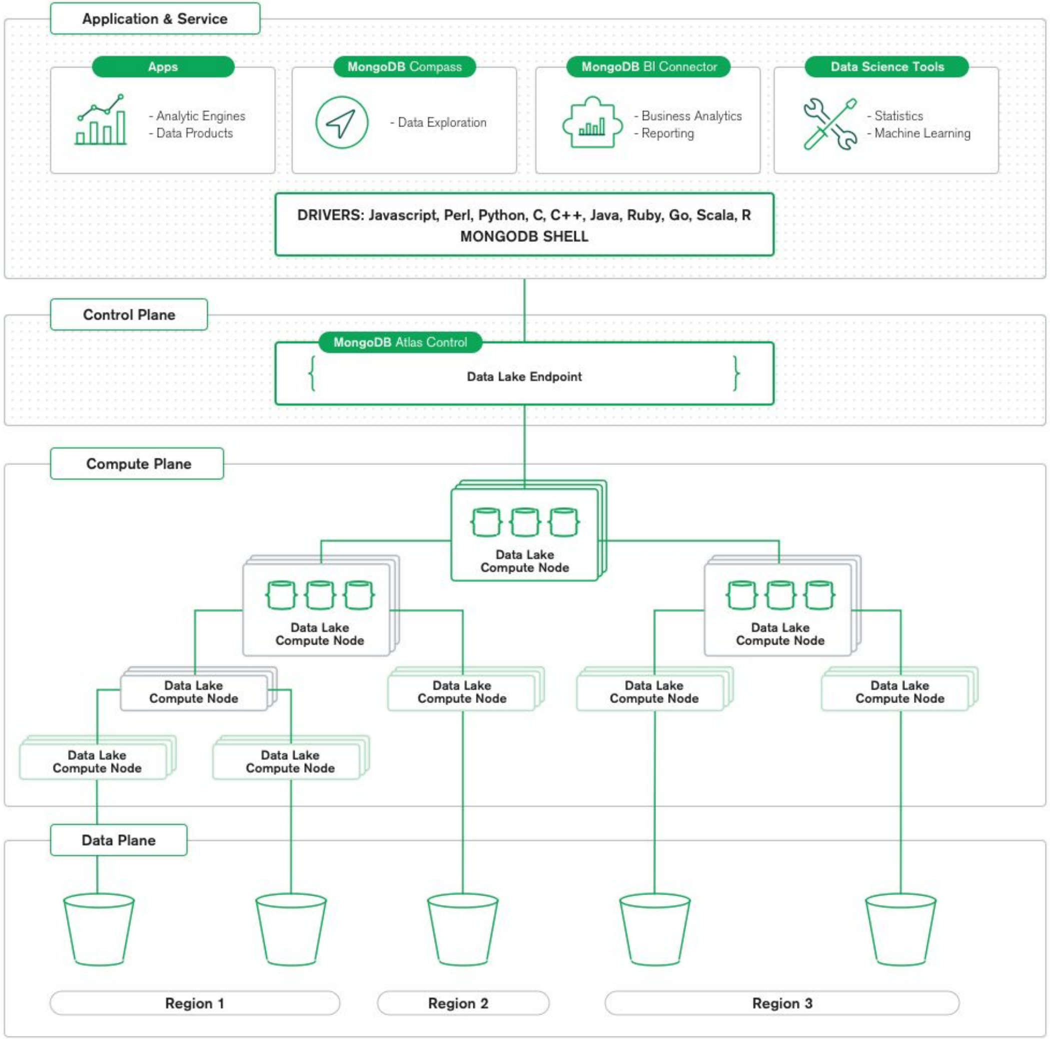
Best Tools, High Productivity

Integrated with Atlas, Single UI, Billing, Permissioning

Serverless, No Infrastructure Management

On-Demand, Usage-Based Pricing

MongoDB Atlas Data Lake Architecture



mongoDB

Use Cases

Data Lake Analytics

➔ explore all of your rich data naturally

➔ get to data as it lands via streams or microservices

➔ democratize access across diverse user groups

Data Products and Services

➔ monetize data

➔ market research, data- and insight-as-a-service

➔ snapshots, time series analysis, predictive analytics to innovate faster

Active Archives

➔ historical analysis against data assets retained in long term cold storage

➔ cost-effective data strategy

Complementing your Data Warehouse

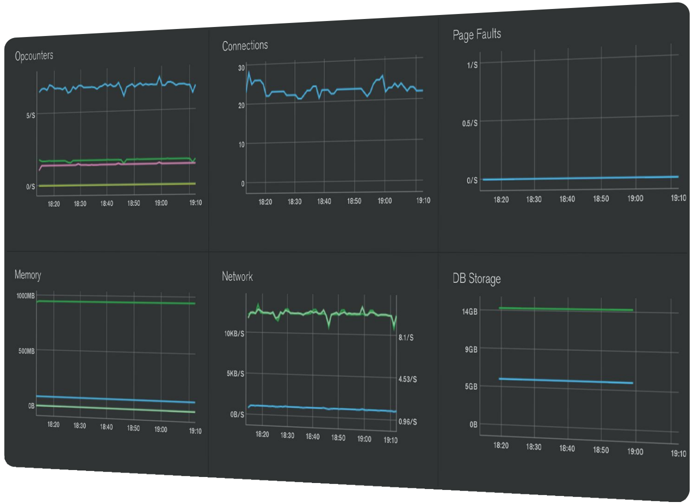
	Data Warehouse	MongoDB Atlas Data Lake
Data sets	Highly curated, cleansed, filtered, aggregated data, ingested via ETL processes from operational databases and applications	Vast pools of raw data, stored in its native form, ingested from logs, sensors, devices, streams, APIs, and operational databases
Data structures	Pre-defined and fixed, tabular schema, with well defined constraints and relationships	Dynamic and flexible schema, rich data structures of any shape
Query patterns	Highly optimized for specific reporting and BI purposes including dashboards, statistics and predictive modeling, regressions, and decision trees	Ad-hoc, data exploration and discovery, machine learning workloads, dashboards
Data storage	Specialized hardware, dedicated clusters of optimized compute and storage	General purpose cloud storage accessed on-demand by serverless compute instances
Consumers	Business analysts, data scientists	Business analysts, developers, data scientists, data engineers

Run by You, With MongoDB Tools

On-Prem Cluster

Mainframe

Self-managed in the cloud



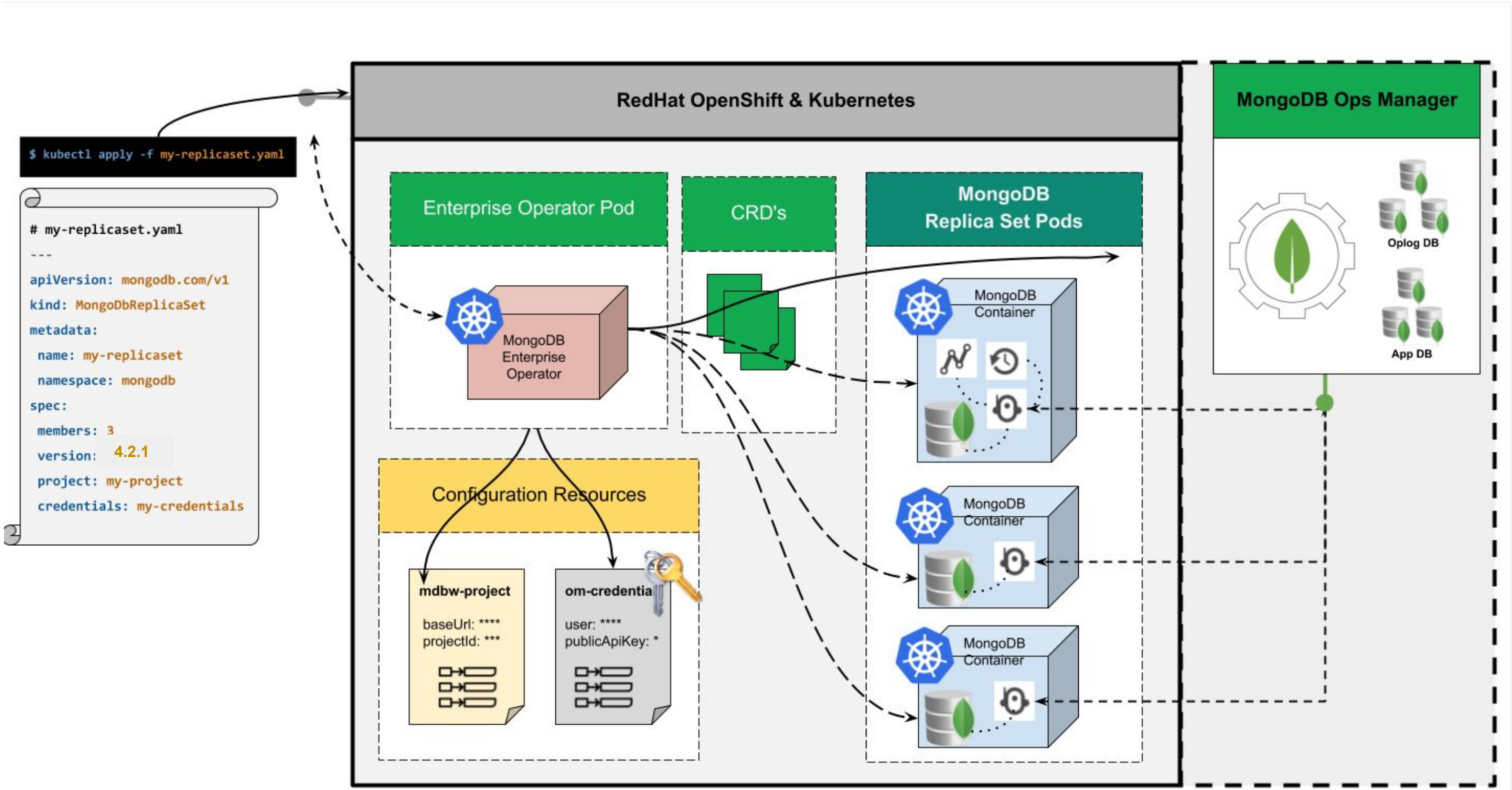
MongoDB Ops Manager

- Automation: Provision, Upgrade, Scale
- Monitoring & Alerting
- Continuous Backup & PiT Restore
- Patching
- Performance Advice
- Kubernetes integration

kubernetes

Pivotal Cloud Foundry

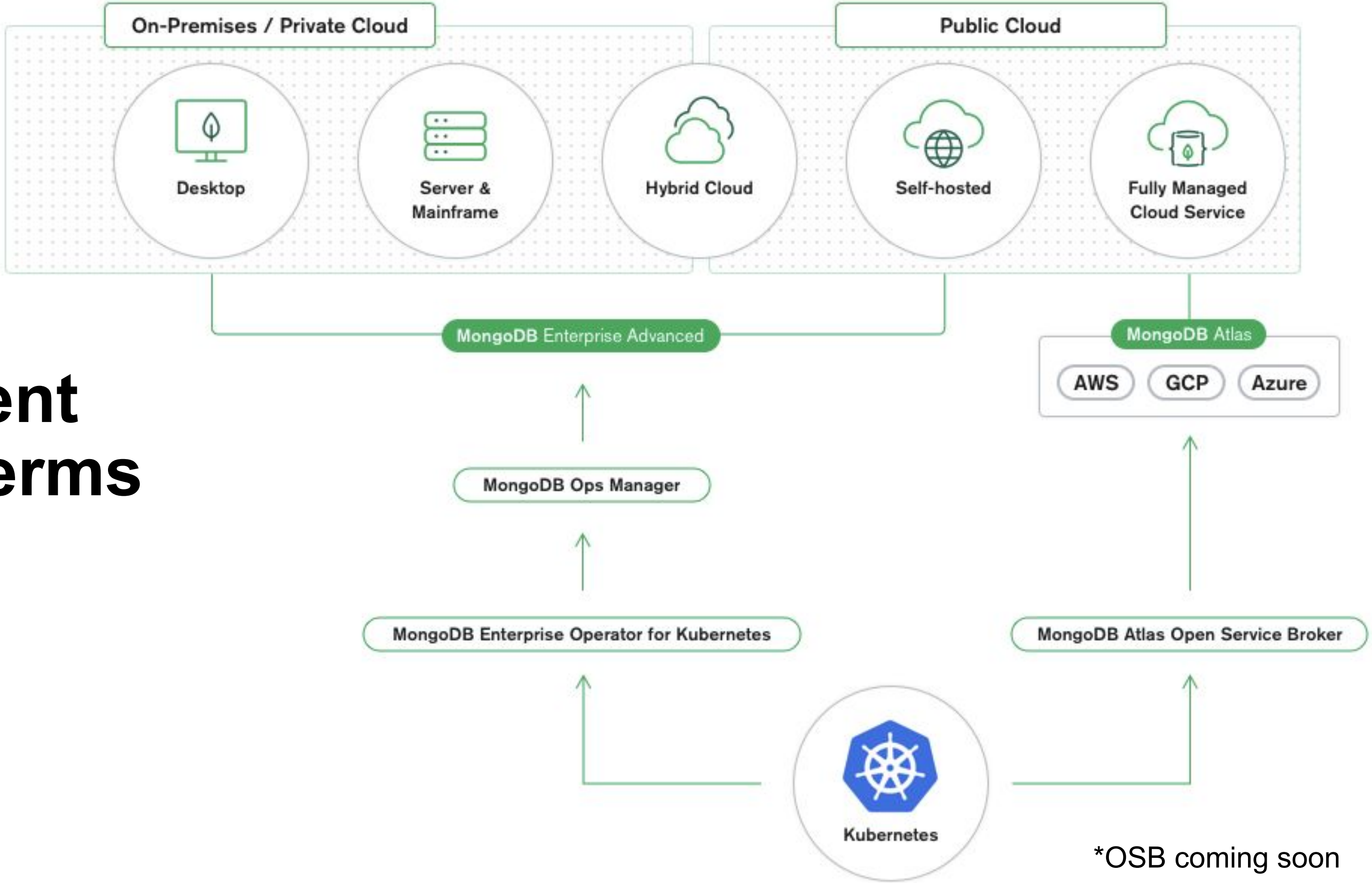
MongoDB Enterprise Operator for Kubernetes



```
$ kubectl apply -f my-replicaset.yaml
```

```
# my-replicaset.yaml
---
apiVersion: mongodb.com/v1
kind: MongoDbReplicaSet
metadata:
 name: my-replicaset
 namespace: mongodb
spec:
 members: 3
 version: 4.2.1
 project: my-project
 credentials: my-credentials
```

Deployment on your terms


On-Premises / Private Cloud
Public Cloud
Desktop
Server & Mainframe
Hybrid Cloud
Self-hosted
Fully Managed Cloud Service
MongoDB Enterprise Advanced
MongoDB Atlas
AWS
GCP
Azure
MongoDB Ops Manager
MongoDB Enterprise Operator for Kubernetes
MongoDB Atlas Open Service Broker
Kubernetes

*OSB coming soon

Q&A