MongoDB Server 4.0: What's New

June 2018

Table of Contents

Introduction

MongoDB is designed to meet the demands of modern apps with a technology foundation that enables you through:

1. The document data model – presenting you **the best way to work with data**.
2. A distributed systems design – allowing you to **intelligently put data where you want it**.
3. A unified experience that gives you the **freedom to run anywhere** – allowing you to future-proof your work and eliminate vendor lock-in.

MongoDB 4.0 is a significant milestone in the evolution of MongoDB. It builds upon our core foundations above with capabilities that allow you to ditch the costly and complex sprawl of legacy relational databases and niche NoSQL stores that are just going to slow you down. Instead, MongoDB 4.0 enables you to standardize your apps on a single, modern data platform. Here's how.

What's New: MongoDB Server 4.0 and Tools

- **Multi-document ACID transactions** enforce snapshot isolation to provide a consistent view of data, and all-or-nothing execution to maintain data integrity. It's now even easier to address a complete range of use cases with MongoDB.
- **Type conversions** allow you to run sophisticated transformations natively in the database, preparing data for BI and machine learning, while eliminating costly, slow, and fragile ETL processes
- The **improved shard balancer** allows up to 40% faster data migrations as clusters elastically expand and contract in response to dynamic workloads.
- The **aggregation pipeline builder** allows developers and data analysts to construct, optimize, and deploy sophisticated processing pipelines that transform, aggregate, and analyze MongoDB data, all from the simple and intuitive MongoDB Compass GUI.
- **MongoDB Charts** (beta) is a new native tool that enables you to create and share visualisations of your MongoDB data, without needing to move your data to

Figure 1: MongoDB evolution over past 3 years

other systems or leverage third-party tools. It's the fastest and easiest way to get insights into your operational data, in real time.

What's New: MongoDB Cloud Services

- **Global clusters** in MongoDB Atlas allows you to deploy geographically distributed, fully managed data platforms that provide low latency reads and writes to users anywhere, with data placement controls for regulatory compliance.
- **Atlas Enterprise: Security and HIPAA Compliance**. MongoDB Atlas is the most secure cloud database service available anywhere. It now offers new security controls including LDAP integration, the encrypted storage engine with bring-your-own key management, database-level auditing, and the HIPAA compliance program. Customers get the benefits of an on-demand, fully managed database service while satisfying complex security and regulatory compliance standards.
- The new free **monitoring cloud service** provides MongoDB community users with visualized metrics from the `serverStatus` command, making it easy for them to get deep and valuable insights into the health and performance of their data platform.
- **Kubernetes and Red Hat OpenShift integration** with MongoDB Ops Manager allows you to seamlessly deploy and manage your MongoDB workloads inside your cluster. Take full advantage of everything Ops Manager provides including automation, monitoring and backup capabilities.

Beyond the Server

- The **MongoDB Stitch serverless platform** is now GA. Stitch streamlines application development with simple, secure access to data and services from the client – getting your apps to market faster while reducing operational costs.
- **MongoDB Mobile** (beta) brings the power of MongoDB to your client. Same database, same access patterns – from IoT to Mobile to Web.

These and other features of MongoDB 4.0 are discussed in this guide to what's new.

Getting Started

MongoDB Server 4.0 is available today, ready for production apps.

- You can download it to run on your own infrastructure, or spin it up in the cloud using the on-demand MongoDB Atlas managed database service.
- You can sign-up for free, web-based MongoDB University training on 4.0, where you will see all of the new features in action.
- The Major Version Upgrade service from MongoDB global consulting is designed to accelerate your transition to MongoDB 4.0. You will receive guidance from a consulting engineer on the necessary steps to upgrade, get a walk through of the upgrade process, and help on evaluating the upgrade in a testing environment.

MongoDB 4.0: Best Way to Work with Data

Multi-Document ACID Transactions

Previewed back in February 2018, multi-document ACID transactions are now GA and ready for use in production apps. With snapshot isolation and all-or-nothing execution, transactions extend MongoDB ACID data integrity guarantees to multiple statements and multiple documents across one or many collections and databases. They feel just like the transactions you are familiar with from relational databases, easy to add to any application that needs them, and do not impact the performance of operations that don't require them. With multi-document transactions it's easier than ever for all developers to address a complete range of use cases with MongoDB. For many of them, simply knowing that they are available will provide critical peace of mind that they can meet any requirement in the future.

In MongoDB 4.0 transactions work within a replica set, and MongoDB 4.2 will support transactions across a sharded cluster* (see the Safe Harbor statement at the end of the Guide).

Data Models and Transactions

Because documents can bring together related data that would otherwise be modeled across separate parent-child tables in a tabular schema, MongoDB's atomic single-document operations provide transaction semantics that meet the data integrity needs of the majority of applications. One or more fields may be written in a single operation, including updates to multiple sub-documents and elements of an array. The guarantees provided by MongoDB ensure complete isolation as a document is updated; any errors cause the operation to roll back so that clients receive a consistent view of the document. These existing document atomicity guarantees will meet 80-90% of an application's transactional needs.

However, some developers and DBAs have been conditioned by 40 years of relational data modeling to assume multi-record transactions are a requirement for any database, irrespective of the data model they are built upon. Some are concerned that while multi-document transactions aren't needed by their apps today, they might be in the future. And for some workloads, support for ACID transactions across multiple documents is required.

Where are Multi-Document Transactions Useful?

There are use cases where transactional ACID guarantees need to be applied to a set of operations that span multiple documents, most commonly with apps that deal with the exchange of value between different parties and require "all-or-nothing" execution. Back office "System of Record" or "Line of Business" (LoB) applications are the typical class of workload where multi-document transactions can be useful. Examples include:

- Moving funds between bank accounts, payment processing systems, or trading platforms – for instance where new trades are added to a tick store, while simultaneously updating the risk system and market data dashboards.
- Transferring ownership of goods and services through supply chains and booking systems – for example, inserting a new order in the orders collection and decrementing available inventory in the inventories collection.
- Billing systems – for example adding a new Call Detail Record and then updating the monthly call plan when a cell phone subscriber completes a call.

MongoDB already serves these use cases today, and the introduction of multi-document transactions makes it

easier as the database automatically handles multi-document transactions for you. Before their availability, the developer would need to programmatically implement transaction controls in their application. To ensure data integrity, they would need to ensure that all stages of the operation succeed before committing updates to the database, and if not, roll back any changes. This adds complexity that slows down the rate of application development. MongoDB customers in the financial services industry have reported they were able to cut 1,000+ lines of code from their apps by using multi-document transactions.

In addition, implementing client side transactions can impose performance overhead on the application. For example, after moving from its existing client-side transactional logic to multi-document transactions, a global enterprise data management and integration ISV experienced improved MongoDB performance in its Master Data Management solution: throughput increased by 90%, and latency was reduced by over 60% for transactions that performed six updates across two collections. Beyond client side code, a major factor in the performance gains came from write guarantees. For each individual write operation in the transaction, the app previously had to wait for acknowledgement from the majority write concern that the operation had been propagated across a majority of replicas. Only once this acknowledgement was received could it then progress to the next write in the transaction. With multi-document transactions, the app only has to wait for the majority acknowledgement when it comes to commit the transaction.

Multi-Document ACID Transactions in MongoDB

Transactions in MongoDB feel just like transactions you are used to in relational databases. They are multi-statement, with similar syntax (e.g. `start_transaction` and `commit_transaction`), making them familiar to anyone with prior transaction experience.

The Python code snippet as follows shows a sample of the transactions API.

```
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()
```

The following snippet shows the transactions API for Java.

```
try (ClientSession clientSession
           = client.startSession()) {
        clientSession.startTransaction();
        collection.insertOne(clientSession, docOne);
        collection.insertOne(clientSession, docTwo);
        clientSession.commitTransaction();
    }
```

As these examples show, transactions use regular MongoDB query language syntax, and are implemented consistently whether the transaction is executed across documents and collections in a replica set, and with MongoDB 4.2, across a sharded cluster*.

Transactions can apply to operations against documents contained in one, or in many, collections and databases. Through snapshot isolation, transactions provide a consistent view of data, and enforce all-or-nothing execution to maintain data integrity. During its execution, a transaction is able to read its own uncommitted writes, but none of uncommitted writes will be seen by other operations outside of the transaction. Uncommitted writes are not replicated to secondary nodes until the transaction is committed to the database. Once the transaction has been committed, it is replicated and applied atomically to all secondary replicas.

Taking advantage of the transactions infrastructure introduced in MongoDB 4.0, the new snapshot read concern ensures queries and aggregations executed within a read-only transaction will operate against a single, isolated snapshot on the primary replica. As a result, a consistent view of the data is returned to the client, irrespective of whether that data is being simultaneously modified by concurrent operations. Snapshot reads are especially useful for operations that return data in batches with the `getMore`command.

Transactions Best Practices

As noted earlier, MongoDB's existing document atomicity guarantees will meet 80-90% of an application's transactional needs. They remain the recommended way of

enforcing your app's data integrity requirements. For those operations that do require multi-document transactions, there are several best practices that developers should observe.

Creating long running transactions, or attempting to perform an excessive number of operations in a single ACID transaction can result in high pressure on WiredTiger's cache. This is because the cache must maintain state for all subsequent writes since the oldest snapshot was created. As a transaction always uses the same snapshot while it is running, new writes accumulate in the cache throughout the duration of the transaction. These writes cannot be flushed until transactions currently running on old snapshots commit or abort, at which time the transactions release their locks and WiredTiger can evict the snapshot. To maintain predictable levels of database performance, developers should therefore consider the following:

- By default, MongoDB will automatically abort any multi-document transaction that runs for more than 60 seconds. Note that if write volumes to the server are low, you have the flexibility to tune your transactions for a longer execution time. To address timeouts, the transaction should be broken into smaller parts that allow execution within the configured time limit. You should also ensure your query patterns are properly optimized with the appropriate index coverage to allow fast data access within the transaction.
- There are no hard limits to the number of documents that can be read within a transaction. As a best practice, no more than 1,000 documents should be modified within a transaction. For operations that need to modify more than 1,000 documents, developers should break the transaction into separate parts that process documents in batches.
- In MongoDB 4.0, a transaction is represented in a single oplog entry, therefore must be within the 16MB document size limit. While an update operation only stores the deltas of the update (i.e., what has changed), an insert will store the entire document. As a result, the combination of oplog descriptions for all statements in the transaction must be less than 16MB. If this limit is exceeded, the transaction will be aborted and fully rolled back. The transaction should therefore be decomposed into a smaller set of operations that can be represented in 16MB or less.
- When a transaction aborts, an exception is returned to the driver and the transaction is fully rolled back. Developers should add application logic that can catch and retry a transaction that aborts due to temporary exceptions, such as a transient network failure or a primary replica election. The commit operations are retryable write operations. If the commit operation operation encounters an error, MongoDB drivers retry the operation a single time.
- DDL operations, like creating an index or dropping a database, block behind active running transactions on the namespace. All transactions that try to newly access the namespace while DDL operations are pending, will not be able to obtain locks, aborting the new transactions.

You can review all best practices in the documentation for multi-document transactions.

Transactions and Their Impact to Data Modeling in MongoDB

Adding transactions does not make MongoDB a relational database – many developers have already experienced that the document model is superior to the relational one.

All best practices relating to MongoDB data modeling continue to apply when using multi-document transactions, or to other relational-type features such as fully expressive JOINs (via the $lookup aggregation pipeline stage). Where practical, all data relating to an entity should be stored in a single, rich document structure. Just moving tabular data normalized for relational tables into MongoDB will not allow users to take advantage of MongoDB's natural, fast, and flexible document model, or its distributed systems architecture.

The RDBMS to MongoDB Migration Guide describes the best practices for moving an application from a relational database to MongoDB.

The Path to Transactions

Our path to transactions represents a multi-year engineering effort, beginning over 3 years ago with the

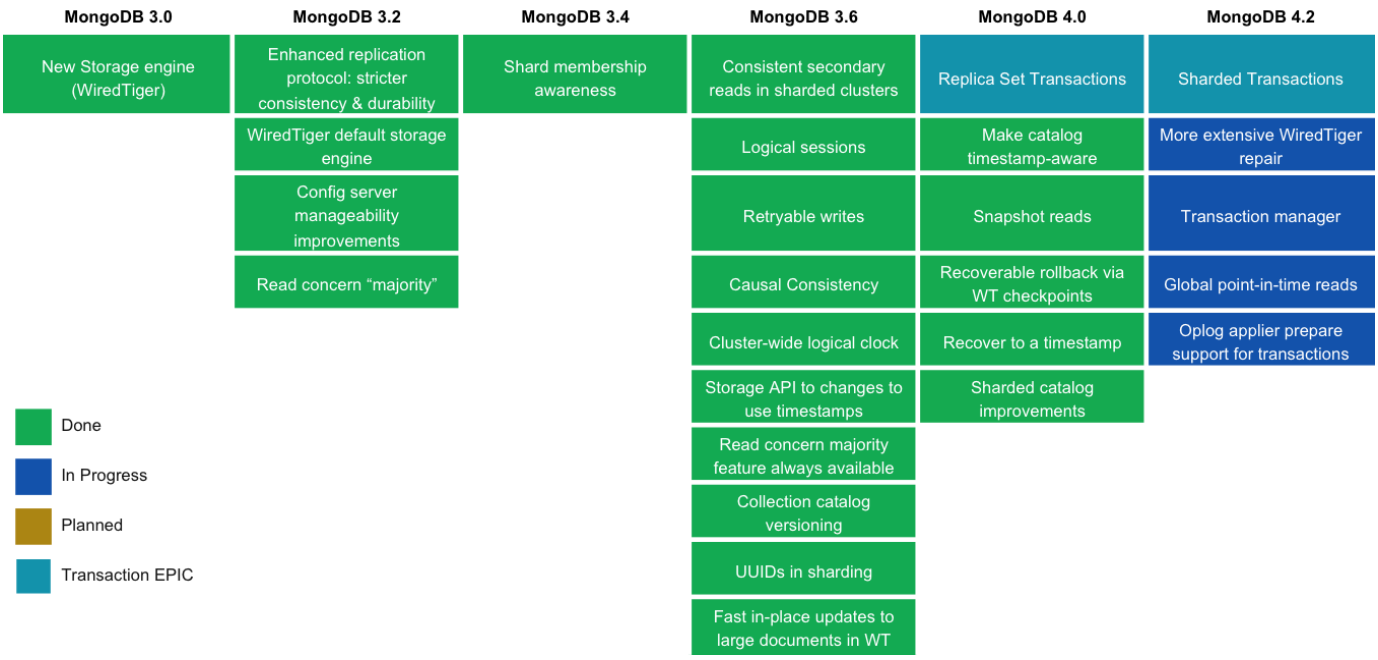

Figure 2: The path to transactions – multi-year engineering investment

integration of the WiredTiger storage engine. We've laid the groundwork in practically every part of the platform – from the storage layer itself to the replication consensus protocol, to the sharding architecture. We've built out fine-grained consistency and durability guarantees, introduced a global logical clock, refactored cluster metadata management, and more. And we've exposed all of these enhancements through APIs that are fully consumable by our drivers. We are feature complete in bringing multi-document transactions to replica sets, and 90% done on implementing the remaining features needed to deliver transactions across a sharded cluster.

Figure 2 presents a timeline of the key engineering projects that have enabled multi-document transactions in MongoDB, with status shown as of June 2018. The key design goal underlying all of these projects is that their implementation does not sacrifice the key benefits of MongoDB – the power of the document model and the advantages of distributed systems, while imposing no performance impact to workloads that don't require multi-document transactions.

Take a look at our multi-document ACID transactions page where you can hear directly from the MongoDB engineers who have built transactions, review code snippets, and access key resources and documentation to get started.

Aggregation Pipeline Type Conversions

Developers and data scientists rely on the MongoDB aggregation pipeline for its power and flexibility in creating sophisticated data processing pipelines that serve complex queries and real-time insights. Enhancements to the aggregation pipeline in MongoDB 4.0 now make it easier and faster to work with complex, multi-structured data.

One of the major advantages of MongoDB over rigid tabular databases is it's flexible data model. Data can be written to the database without first having to predefine its structure. This helps you to build apps faster, and respond easily to rapidly evolving application changes. It is also essential in supporting initiatives such as single customer view or operational data layers to support real-time analytics where data is ingested from multiple source systems. Of course, with MongoDB's schema validation, this flexibility is fully tunable, enabling you to enforce strict controls on data structure, type, and content when you need more control.

So while MongoDB makes it easy to ingest data without complex cleansing of individual fields, it means working with this data can be more difficult when a consuming application expects uniform data types for specific fields across all documents. Handling different data types pushes

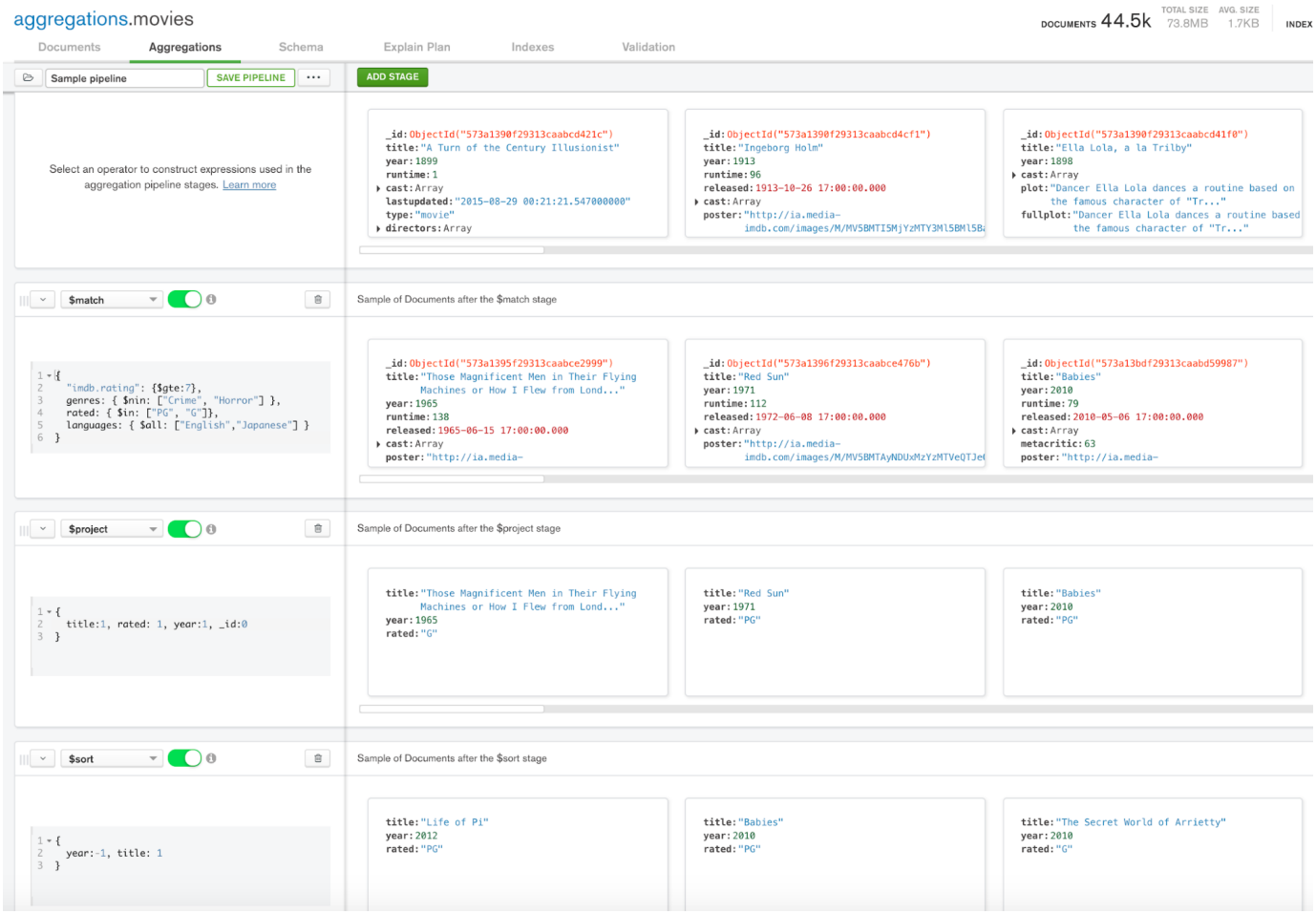

Figure 3: Graphically create sophisticated aggregation pipelines with MongoDB Compass

more complexity to the application, and available ETL tools have provided only limited support for transformations.

With MongoDB 4.0, you can maintain all of the advantages of a flexible data model, while prepping data within the database itself for downstream processes. The new $convert operator enables the aggregation pipeline to transform mixed data types into standardized formats **natively within the database**. Ingested data can be cast into a cleansed format and exposed to multiple consuming applications – such as the MongoDB BI and Spark connectors for high-performance visualizations, advanced analytics and machine learning algorithms, or directly to a UI. Casting data into cleansed types makes it easier for your apps to to process, sort, and compare data. For example, financial data inserted as a long can be converted into a decimal, enabling lossless and high precision processing. Similarly dates inserted as strings from some source systems and dates from others can be standardized into the native date type.

When $convert is combined with over 100 different operators available as part of the MongoDB aggregation pipeline, you can reshape and transform your documents in whichever way you need without having to incur the complexity, fragility, and latency of running data through external ETL processes.

MongoDB Compass: Aggregation Pipeline Builder

MongoDB Compass, the GUI for MongoDB, makes it easy for anyone to explore and manipulate their data. They can visualize the structure of data in MongoDB, run ad hoc queries and evaluate their performance, view and create indexes, build data validation rules, and more.

The new aggregation pipeline builder extends Compass's existing point and click visual query editor. Developers and data analysts can now construct sophisticated data processing pipelines straight from the intuitive Compass

GUI. They can easily build multi-stage pipelines by dragging and dropping stages onto a canvas, inspect the results at each stage to optimize performance, and then export the pipeline to native code. The aggregation pipeline is one of MongoDB's most powerful capabilities – and with the pipeline builder, it's accessible to developers and data analysts, even if they aren't familiar with the MongoDB query language.

Compass also now includes **export to language**. Using this new feature, you build your query as usual in Compass – including complex, multi-parameter queries and aggregation pipelines – then click a button to export it as native code in the driver language of your choice. This brings the power of Compass's intuitive, visual, auto-completing query builder directly to your favorite programming environment.

MongoDB Charts

Available now as a public beta, MongoDB Charts is a new tool that enables you to quickly and easily create and share visualisations of your MongoDB data in real time, without needing to move your data into other systems, or leverage third-party tools. Because Charts natively understands the MongoDB document model, you can create charts from data that varies in shape or contains nested documents and arrays, without needing to first map the data into a flat, tabular structure.

Existing approaches of using traditional BI tools to visualize non-tabular data structures present a number of drawbacks:

- Data movement between the database and BI platform adds latency.
- Sampling a subset of documents or tables to infer a tabular schema can compromise data fidelity by missing documents with sparsely populated fields.
- Unwinding embedded data such as arrays and subdocuments into tabular rows and columns to perform aggregations and analytics increases both computational overhead and data duplication.
- While enabling deep analytics and rich visualizations, conventional SQL-based BI suites add a large learning curve to users that slows time to value, and often command high prices for commercial licensing.

Using Charts, you can quickly and easily visualize your data, place multiple charts onto a single dashboard, and then share that dashboard with key stakeholders to support collaborative analysis and decision making. When you connect to a live data source, MongoDB Charts will keep your visualizations and dashboards up to date with the most recent data. Charts will automatically generate an aggregation pipeline from your chart design, which is then executed on your MongoDB server. With MongoDB's workload isolation capabilities – enabling you to separate your operational from analytical workloads in the same cluster – you can use Charts for a real-time view **without having any impact on production workloads**.

Initial chart types supported include bar, column, line, area, scatter, and donut charts. When you need more complex visualizations, or want to blend data from MongoDB with other data sources, the MongoDB Connector for BI provides integration with all leading SQL-based BI platforms such as Tableau, Qlik, SAP Business Objects, and more.

You can get started by trying out MongoDB Charts in beta.

MongoDB 4.0: Distribute Data Where you Need It

40% Faster Data Migrations

Very few of today's workloads are static. For example, the launch of a new product or game, or seasonal reporting cycles can drive sudden spikes in load that can bring a database to its knees unless additional capacity can be quickly provisioned. If and when demand subsides, you should be able to scale your cluster back-in, rightsizing for capacity and cost.

To respond to these fluctuations in demand, MongoDB enables you to elastically add and remove nodes from a sharded cluster in real time, automatically rebalancing the data across nodes, without you having to intervene. The sharded cluster balancer, responsible for evenly distributing data across the cluster, has been significantly improved in

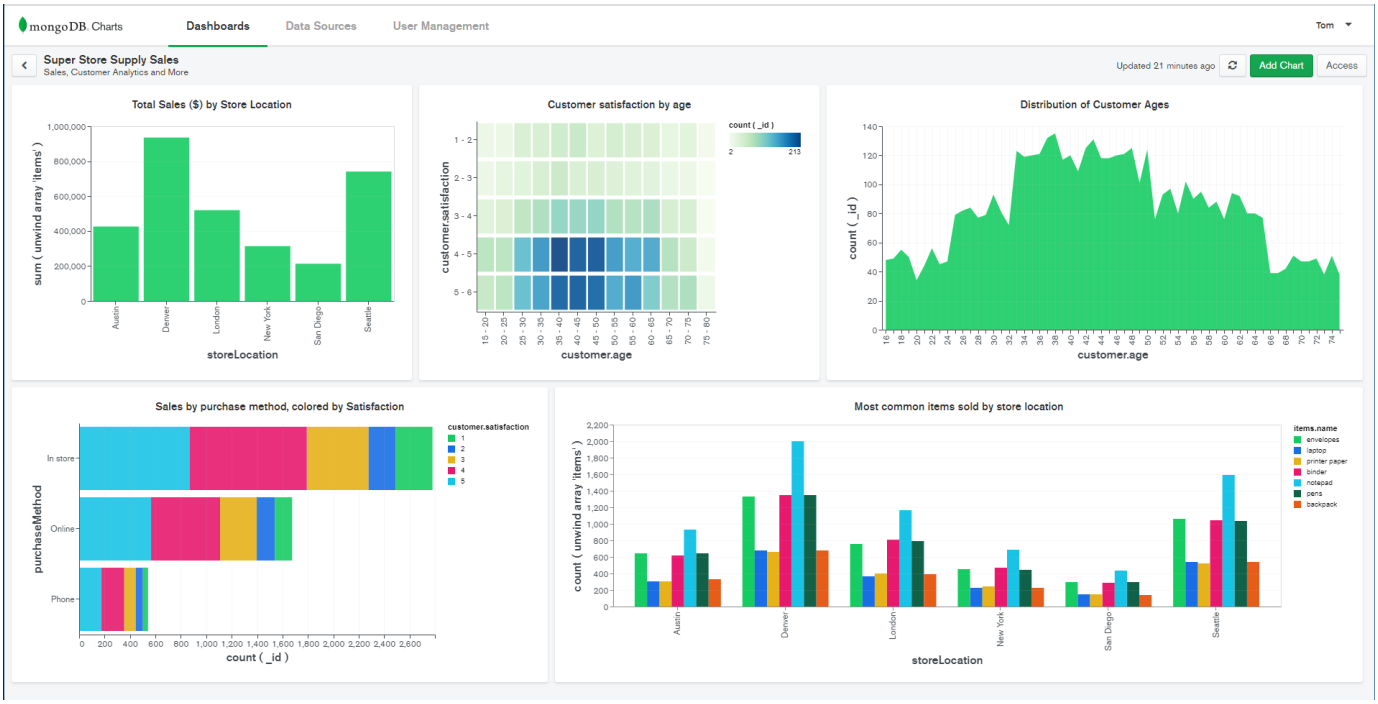

Figure 4: Creating rich graphs and dashboards directly from the Charts UI

MongoDB 4.0. By concurrently fetching and applying documents, shards can complete chunk migrations up to 40% faster, allowing you to more quickly bring new nodes into service at just the moment they are needed, and scale back down when load returns to normal levels.

Non-Blocking Secondary Reads

Many MongoDB users read from secondary replica set members to scale read throughput out across their cluster, or to reduce network latency when using the nearest read preference.

To ensure that reads can never return data that is not in the same causal order as the primary replica, MongoDB blocks readers while oplog entries are applied in batches to the secondary. This can cause secondary reads to have variable latency, which becomes more pronounced when the cluster is bulk loading new data. Why does MongoDB need to block secondary reads? When you apply a sequence of writes to a document, then MongoDB is designed so that each of the nodes must show the writes in the same causal order. So if you change field "A" in a document and then change field "B", it is not possible to see that document with changed field "B" and not changed field "A". Eventually consistent systems suffer from this behavior, but MongoDB does not, and never has.

By taking advantage of storage engine timestamps and snapshots implemented for multi-document ACID transactions, secondary reads in MongoDB 4.0 become non-blocking. Queries against secondary replicas now read from a snapshot of data while replication batches are simultaneously applied to the secondary. Reading from that snapshot guarantees a consistent view of the data, and since applying current writes via a replication batch doesn't change the earlier snapshots, we can relax the lock that replication uses to allow secondary reads.

With non-blocking secondary reads, you now get predictable, low read latencies and increased throughput from the replica set, while maintaining a consistent view of data. Workloads that see the greatest benefits are those where data is batch loaded to the database, and those where distributed clients are accessing low latency local replicas that are geographically remote from the primary replica. For more detail, read our blog on secondary reads in MongoDB 4.0.

SHA-2 Authentication

Having the ability to distribute data to where it's needed enables you to build powerful new classes of application. However, you must also be confident that your data is secure, wherever it is stored. Rather than build security controls back in the application, you should be able to rely on the database to implement the mechanisms needed to protect sensitive data and meet the needs of apps in regulated industries. To address these requirements, MongoDB offers extensive access control, encryption, and auditing features.

With MongoDB 4.0, authentication has been updated to the latest SHA-2 family (SHA-256), providing a stronger alternative to SHA-1, and building upon some of the most advanced security capabilities of any database. Read more in the Authentication section of the docs.

Freedom to Run Anywhere

As an open source database, MongoDB can be run anywhere — from laptops and mainframes to a private cloud, public cloud, or to the fully managed, on-demand MongoDB Atlas database service. The developer experience is entirely unaffected by the deployment model chosen; similarly, those teams that want to maintain responsibility for running their own databases can also leverage a unified set of tools that deliver the same experience across different environments.

MongoDB 4.0 brings a number of enhancements, whether you want to consume your database as a service in Atlas, or run it yourself on your own infrastructure.

Global Clusters

New to MongoDB Atlas, Global Clusters allow organizations with distributed applications to easily geo-partition a single fully managed database to provide low latency reads and writes anywhere in the world. With this update, data can be associated and directed to nodes in specified cloud regions, keeping it in close proximity to nearby application servers and local users, and helping with compliance to data regulations like the General Data Protection Regulation (GDPR). Global Clusters also allow organizations to easily replicate data worldwide for multi-region fault tolerance and providing fast, responsive access to any data set, anywhere. Developers can set up Global Clusters — now available on Amazon Web Services, Microsoft Azure, and Google Cloud Platform — with just a few clicks in the MongoDB Atlas UI.

MongoDB Atlas takes care of the deployment and management of infrastructure and database resources required to ensure that data is written to and read from different regions. For example, you may have an accounts collection that you want to distribute among your three regions of business, North America, Europe, and Asia. Atlas will recommend a templated configuration of zones, each with unique region mappings, to help reduce latency for distributed application servers. You also have the option of creating your own zones by choosing cloud regions in an easy to use, visual interface. Atlas will then automatically deploy your configuration and shard the accounts collection, ensuring that data tagged for a specific zone is appropriately routed. When performing reads and writes, you just need to include a combination of the country code and the account identifier in the query, and Atlas will ensure all operations are directed to the right shard in the right data zone.

Zones can also contain different numbers of shards. For example, if there are more application users in North America, you can provision more shards in that region and scale them on demand.

Atlas Enterprise Security and HIPAA Readiness

MongoDB Atlas launched with a robust set of security features including network isolation, always-on authentication, IP whitelisting, fully managed disaster recovery, and encryption for data in transit and at rest. These built-in security controls help protect tens of thousands of databases supporting applications across financial services, manufacturing, energy, healthcare, and more. New enterprise security features establish MongoDB Atlas as the most secure database as a service while allowing enterprises to exercise more control by enabling integration with existing security platforms. These features include:

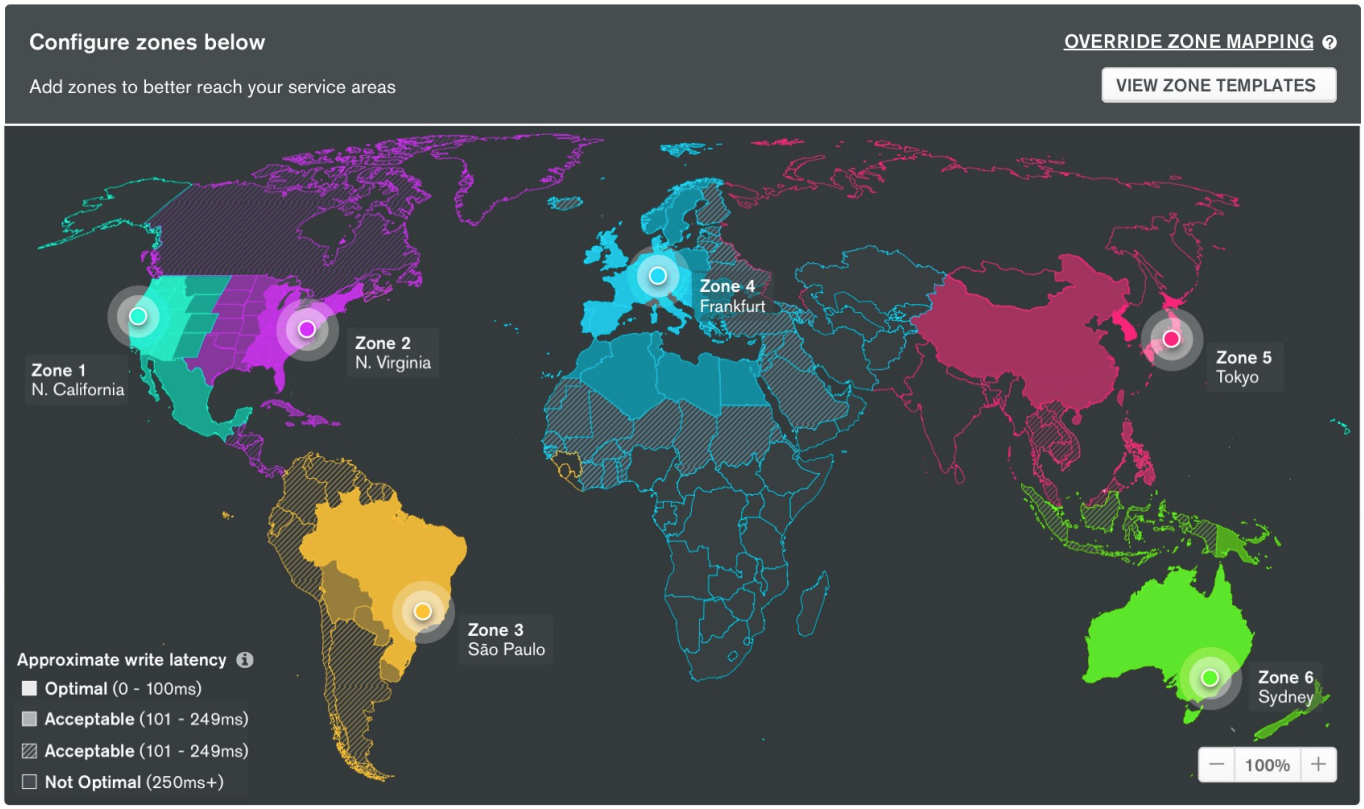

Figure 5: Serving always-on, globally distributed, write-everywhere apps with MongoDB Atlas Global Clusters

- **LDAP Integration**: MongoDB Atlas offers integration with customer LDAP servers to simplify access control and granular permissions management for users and applications.
- **Bring your own Key Management Service (KMS)**: MongoDB Atlas supports the encrypted storage engine, a native encryption option for WiredTiger. With this option enabled, all database files are written to the filesystem encrypted and customers can control creating, importing, or rotating keys via an integration with their KMS of choice.
- **Database-level auditing**: For regulatory compliance, security administrators can use MongoDB Atlas to retrieve audit logs tracking any operation taken against the database.

Bringing the power of MongoDB Atlas to Protected Health Information

MongoDB is committed to bringing the ease and power of MongoDB Atlas to the next generation of healthcare applications. For organizations subject to the requirements of the Health Insurance Portability and Accountability Act of 1996 (HIPAA), MongoDB Atlas now enables covered entities and their business associates to use a secure MongoDB Atlas environment to process, maintain, and store protected health information under an executed Business Associate Agreement with MongoDB, Inc.

You can learn more these new features from the MongoDB Atlas security documentation.

Free MongoDB Monitoring Cloud Service

With the 4.0 release, the MongoDB database can natively push monitoring metadata directly to the MongoDB Monitoring Cloud. Once enabled, you will be shown a unique URL that you can navigate to in a web browser, and instantly see monitoring metrics and topology information collected for your environment. You can share the URL to provide visibility to anyone on your team.

The free monitoring service is available to all MongoDB users, without needing to install an agent, navigate a paywall, or complete a registration form. You will be able to see the metrics and topology about your environment from the moment free monitoring is enabled. You can enable

free monitoring easily using the MongoDB shell, MongoDB Compass, or by starting the mongod process with the new `db.enableFreeMonitoring()` command line option, and you can opt out at any time.

With the Monitoring Cloud Service, the collected metrics enable you to quickly assess database health and optimize performance, all from the convenience of a powerful browser-based GUI. Monitoring features include:

- Environment information: Topology (standalone, replica sets including primary and secondary nodes). MongoDB version.
- Charts with 24 hours of data for the following metrics, updated every minute: Database operations per second (averaged to the minute), including commands, queries, updates, deletes, getMores, inserts and replication operations for replica set secondaries.
- Operation execution time.
- Queues.
- Replication lag.
- Network I/O.
- Memory (resident and virtual).
- Hardware: Process CPU, disk % utilization, Disk % free space

Learn more by reviewing the MongoDB Free Monitoring docs.

Kubernetes and Red Hat OpenShift

Today more DevOps teams are leveraging the power of containerization, and technologies like Kubernetes and Red Hat OpenShift, to manage containerized database clusters. To support teams building cloud-native apps with Kubernetes and OpenShift, we are introducing a Kubernetes Operator (beta) that integrates with Ops Manager, the enterprise management platform for MongoDB. This operator enables a user to deploy and manage MongoDB clusters from within the Kubernetes API, without having to connect separately to Ops Manager. For example, a Kubernetes user managing a Kubernetes cluster using the kubectl tool can simply submit a yaml file like the one below to easily deploy a 2 shard MongoDB cluster.

```
apiVersion: mongodb.com/v1beta1
kind: MongoDbShardedCluster
metadata:
    name: k8sdemo
    namespace: mongodb
spec:
    shardCount: 2
    mongodsPerShardCount: 3
    mongosCount: 2
    configServerCount: 3
    version: 4.0.1

    persistent: false

    project: my-project
    credentials: my-credentials
```

Note that the MongoDB cluster can also be configured to use persistent storage. This is the recommended approach for any production deployment.

With this Kubernetes integration, you can consistently and effortlessly run and deploy workloads wherever they need to be, standing up the same database configuration in different environments, all controlled from a single pane of glass. Operations teams can also offer developers on-demand MongoDB-as-a-Service, that provides a cloud-native, fully managed database, alongside other products and services, managed by Kubernetes and OpenShift.

To learn more, read our MongoDB and Kubernetes blog, and join our Slack channel on #enterprise-kubernetes.

Beyond the Server

MongoDB Stitch

The MongoDB Stitch serverless platform facilitates application development with simple, secure access to data and services from the client – getting your apps to market faster while reducing operational costs.

Stitch represents the next stage in the industry's migration to a more streamlined, managed infrastructure. Virtual Machines running in public clouds (notably AWS EC2) led the way, followed by hosted containers, and serverless offerings such as AWS Lambda and Google Cloud Functions. These still required backend developers to implement and manage access controls and REST APIs to provide access to microservices, public cloud services, and

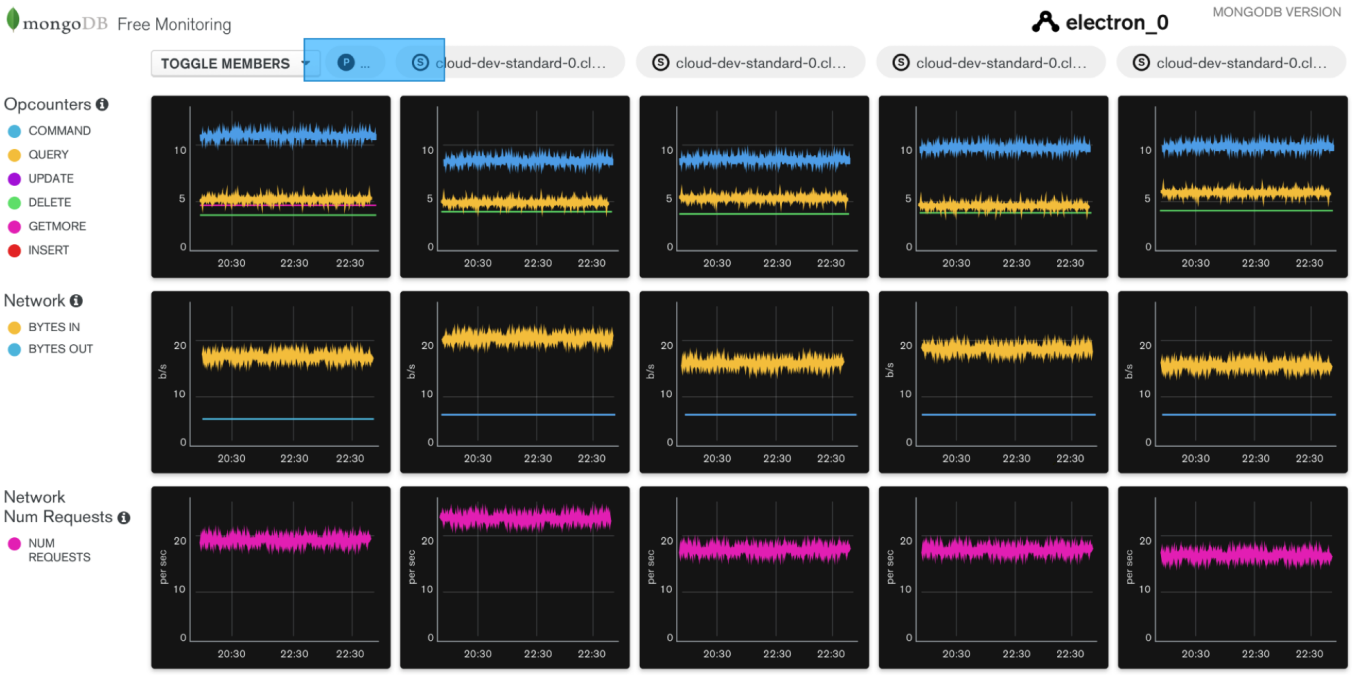

Figure 6: Gaining deep insight into database status with the MongoDB Monitoring Cloud Service

of course data. Frontend developers were held back by needing to work with APIs that weren't suited to rich data queries.

The Stitch serverless platform addresses these challenges by providing four services:

- **Stitch QueryAnywhere**. Exposes the full power of working with documents in MongoDB and the MongoDB query language, directly from your web and mobile application frontend code. A powerful rules engine lets developers declare fine-grained security policies.
- **Stitch Functions**. Allows developers to run simple JavaScript functions in Stitch's serverless environment, making it easy to create secure APIs or to build integrations with microservices and server-side logic. Enables integration with popular cloud services such as Slack and Twilio, enriching your apps with a single Stitch method call.
- **Stitch Triggers**. Real-time notifications that launch functions in response to changes in the database. The functions can make further database changes, push data to other places, or interact with users – such as through push notifications, text messages, or emails.
- **Stitch Mobile Sync** (coming soon). Automatically synchronizes data between documents held locally in MongoDB Mobile and the backend database. MongoDB Mobile allows mobile developers to use the full power of MongoDB locally. Stitch Mobile Sync ensures that data is kept up to date across phones and all other clients in real time.

Whether building a mobile, IoT, or web app from scratch, adding a new feature to an existing app, safely exposing your data to new users, or adding service integrations, Stitch can take the place of your application server and save you writing thousands of lines of boilerplate code.

MongoDB Mobile

Available in beta, MongoDB Mobile extends your ability to put data where you need it, all the way out to the edge of the network on IoT assets and iOS and Android mobile devices. MongoDB Mobile provides a single database, query language, and the intuitive Stitch SDK that runs consistently for data held on mobile clients, through to the backend server.

MongoDB Mobile provides the power and flexibility of MongoDB in a compact form that is power and performance aware with a low disk and memory footprint. It

supports 64 bit iOS and Android operating systems and is easily embedded into mobile and IoT devices for fast and reliable local storage of JSON documents. With secondary indexing, access to the full MongoDB query language and aggregations, users can query data any way they want. With local reads and writes, MongoDB Mobile lets you build the fastest, most reactive apps. Stitch Mobile Sync (coming soon) will automatically synchronize data changes between data held locally and your backend database, helping resolve any conflicts – even after the device has been offline.

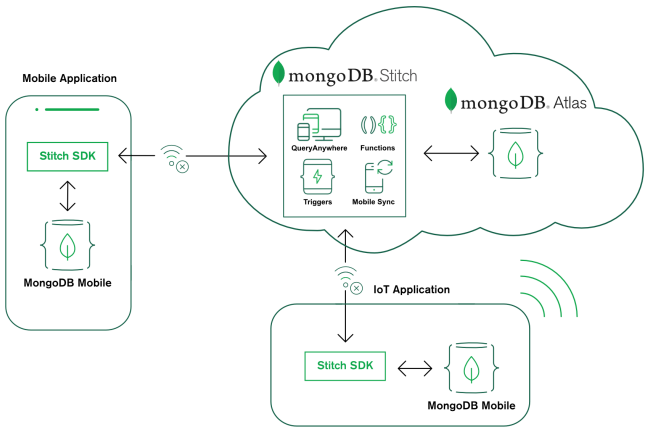

Figure 7: Building end-to-end apps with MongoDB Mobile, Stitch serverless platform, and Atlas database service

The beta program is open now, and you can sign up for access on the MongoDB Mobile product page.

Conclusion

The 4.0 release builds upon MongoDB's core foundations:

- **Best way to work with data**: adding multi-document ACID transactions, data type conversions, native visualizations with MongoDB Charts, the MongoDB Compass visual aggregation pipeline builder, and the MongoDB Stitch serverless platform.
- **Intelligently place data where you need it to be**: with 40% faster shard migrations, non-blocking secondary replica reads, SHA-2 authentication, and the new MongoDB Mobile database.
- **Freedom to run anywhere**: bringing global clusters and enterprise security with HIPAA compliance to the MongoDB Atlas database service, along with the free community monitoring service, and Kubernetes integration.

MongoDB Server 4.0 is available today, ready for production apps.

- You can download it to run on your own infrastructure, or spin it up in the cloud using the on-demand MongoDB Atlas managed database service.
- You can sign-up for free, web-based MongoDB University training on 4.0, where you will see all of the new features in action.
- The Major Version Upgrade service from MongoDB global consulting is designed to accelerate your transition to MongoDB 4.0. You will receive guidance from a consulting engineer on the necessary steps to upgrade, get a walk through of the upgrade process, and help on evaluating the upgrade in a testing environment.

***Safe Harbour Statement**

This paper contains "forward-looking statements" within the meaning of Section 27A of the Securities Act of 1933, as amended, and Section 21E of the Securities Exchange Act of 1934, as amended. Such forward-looking statements are subject to a number of risks, uncertainties, assumptions and other factors that could cause actual results and the timing of certain events to differ materially from future results expressed or implied by the forward-looking statements. Factors that could cause or contribute to such differences include, but are not limited to, those identified our filings with the Securities and Exchange Commission. You should not rely upon forward-looking statements as predictions of future events. Furthermore, such forward-looking statements speak only as of the date of this presentation.

In particular, the development, release, and timing of any features or functionality described for MongoDB products remains at MongoDB's sole discretion. This information is merely intended to outline our general product direction and it should not be relied on in making a purchasing decision nor is this a commitment, promise or legal obligation to deliver any material, code, or functionality. Except as required by law, we undertake no obligation to

update any forward-looking statements to reflect events or circumstances after the date of such statements.

We Can Help

We are the MongoDB experts. Over 6,600 organizations rely on our commercial products. We offer software and services to make your life easier:

MongoDB Enterprise Advanced is the best way to run MongoDB in your data center. It's a finely-tuned package of advanced software, support, certifications, and other services designed for the way you do business.

MongoDB Atlas is a database as a service for MongoDB, letting you focus on apps instead of ops. With MongoDB Atlas, you only pay for what you use with a convenient hourly billing model. With the click of a button, you can scale up and down when you need to, with no downtime, full security, and high performance.

MongoDB Stitch is a serverless platform which accelerates application development with simple, secure access to data and services from the client – getting your apps to market faster while reducing operational costs and effort.

MongoDB Mobile (Beta) MongoDB Mobile lets you store data where you need it, from IoT, iOS, and Android mobile devices to your backend – using a single database and query language.

MongoDB Cloud Manager is a cloud-based tool that helps you manage MongoDB on your own infrastructure. With automated provisioning, fine-grained monitoring, and continuous backups, you get a full management suite that reduces operational overhead, while maintaining full control over your databases.

MongoDB Consulting packages get you to production faster, help you tune performance in production, help you scale, and free you up to focus on your next release.

MongoDB Training helps you become a MongoDB expert, from design to operating mission-critical systems at scale.

Whether you're a developer, DBA, or architect, we can make you better at MongoDB.

Resources

For more information, please visit mongodb.com or contact us at sales@mongodb.com.

Case Studies (mongodb.com/customers)
Presentations (mongodb.com/presentations)
Free Online Training (university.mongodb.com)
Webinars and Events (mongodb.com/events)
Documentation (docs.mongodb.com)
MongoDB Enterprise Download (mongodb.com/download)
MongoDB Atlas database as a service for MongoDB (mongodb.com/cloud)
MongoDB Stitch backend as a service (mongodb.com/cloud/stitch)

US 866-237-8815 ▪ INTL +1-650-440-4474 ▪ info@mongodb.com