Apache Spark and MongoDB

Turning Analytics into Real-Time Action
November 2017

Table of Contents

Introduction

We live in a world of "big data". But it isn't only the data itself that is valuable – it is the insight it can generate. That insight can help designers better predict new products that customers will love. It can help manufacturing companies model failures in critical components, before costly recalls. It can help financial institutions detect fraud. It can help retailers better forecast supply and demand. Marketing departments can deliver more relevant recommendations to its customers. The list goes on.

How quickly an organization can unlock and act on that insight is becoming a major source of competitive advantage. Collecting data in operational systems and then relying on nightly batch ETL (Extract Transform Load) processes to update the Enterprise Data Warehouse (EDW) is no longer sufficient. Speed-to-insight is critical, and so analytics against live operational data to drive real-time action is fast becoming a necessity, enabled by technologies like MongoDB and Apache Spark.

Why Apache Spark and MongoDB?

"Users are already combining Apache Spark and MongoDB to build sophisticated analytics applications. The native MongoDB Connector for Apache Spark provides higher performance, greater ease of use, and access to more advanced Apache Spark functionality than any MongoDB connector available today"

— Reynold Xin, co-founder and chief architect of Databricks.

Apache Spark is one of the fastest-growing big data projects in the history of the Apache Software Foundation. With its memory-oriented architecture, flexible processing libraries and ease-of-use, Spark has emerged as a leading distributed computing framework for real-time analytics.

With over 30 million downloads and 4,300 customers, including more than 50% of the Fortune 100, MongoDB is the most popular non-relational database available today. Its flexible JSON-based document data model, dynamic

schema and automatic scaling on commodity hardware make MongoDB an ideal fit for modern, always-on applications that must manage high volumes of rapidly changing, multi-structured data. Internet of Things (IoT), mobile apps, social engagement, cybersecurity, customer data and content management systems are prime examples of typical MongoDB use cases.

Combining the leading analytics processing engine with the fastest-growing database enables organizations to realize real time analytics and machine learning. Spark jobs can be executed directly against operational data managed by MongoDB without the time and expense of ETL processes. MongoDB can then efficiently index and serve analytics results back into live, operational processes. This approach offers many benefits to teams tasked with delivering modern, data driven applications:

- Developers can build more functional applications faster, using a single database technology.
- Operations teams eliminate the requirement for shuttling data between separate operational and analytics infrastructure, each with its own unique configuration, maintenance and management requirements.
- CIOs deliver faster time-to-insight for the business, with lower cost and risk.

This whitepaper will discuss the analytics capabilities offered by MongoDB and Apache Spark, before providing an overview of how to configure and combine them into a real-time analytics engine. It will conclude with example use cases.

Rich Analytics in MongoDB

Unlike NoSQL databases that offer little more than basic key-value query operations, developers and data scientists can use MongoDB's native query processing and data navigation capabilities to generate many classes of analytics, before needing to adopt dedicated frameworks such as Spark or Hadoop for more specialized tasks.

Leading organizations including Salesforce, McAfee, Buzzfeed, Intuit, Bosch, Amadeus, Expedia, KPMG and many others rely on MongoDB's powerful query functionality, aggregations and indexing to generate analytics in real-time directly against their live, operational data.

In this section of the paper, we provide more detail on MongoDB's native analytics features, before then exploring the additional capabilities Apache Spark can bring.

MongoDB Query Model

MongoDB users have access to a broad array of query, projection and update operators supporting analytical queries against live operational data, supporting tunable consistency to balance performance against data freshness:

- Aggregation and MapReduce queries, discussed in more detail below
- Range queries returning results based on values defined as inequalities (e.g., greater than, less than or equal to, between)
- Geospatial queries returning results based on proximity criteria, intersection and inclusion as specified by a point, line, circle or polygon
- Search queries return results in relevance order and in faceted groups, based on text arguments using Boolean operators (e.g., `AND`, `OR`, `NOT`), and through bucketing, grouping and counting of query results
- JOINs allow documents from separate collections to be conveniently combined in a single operation.
- Graph queries bring native graph processing within MongoDB, enabling efficient traversals across trees, graphs and hierarchical data to uncover patterns and surface previously unidentified connections
- Key-value queries returning results based on any field in the document, often the primary key
- Native BI connectivity for rich data visualisations, and Hadoop integration to expose analytics generated in the data lake to operational applications.

With the combination of MongoDB's dynamic document model and comprehensive query framework, users are able

to store data before knowing all of the questions they will need to ask of it.

Data Aggregation

The MongoDB Aggregation Pipeline is similar in concept to the SQL GROUP BY statement, enabling users to generate aggregations of values returned by the query (e.g., count, min, max, average, standard deviation) that can be used to power analytics dashboards and visualizations.

Using the Aggregation Framework, documents in a MongoDB collection (analogous to a table in a relational database) pass through an aggregation pipeline, where they are processed by operators. Expressions produce output documents based on calculations performed on the input documents. The accumulator expressions used in the $group operator maintain state (e.g., totals, mins, maxs, averages) as documents progress through the pipeline.

Beyond Group_By type queries, the aggregation pipeline allows developers and data engineers to declaratively create sophisticated processing pipelines for data analytics and transformations, powering faceted search, JOINs and graph traversals natively within MongoDB.

Result sets from the aggregation pipeline can be written to a named collection with no limit to the output size (subject to the underlying storage system). Existing collections can be replaced with new results while maintaining previously defined indexes to ensure queries can always be returned efficiently over rapidly changing data.

Incremental MapReduce within MongoDB

MongoDB provides native support for MapReduce, enabling complex data processing that is expressed in JavaScript and executed across data in the database. Multiple MapReduce jobs can be executed concurrently across both single servers and sharded collections.

The incremental MapReduce operation uses a temporary collection during processing so it can be run periodically over the same target collection without affecting intermediate states. This mode is useful when continuously updating statistical output collections used by reporting dashboards.

Optimizing Analytics Queries with MongoDB Indexes

MongoDB provides a number of different index types to optimize performance of real-time and ad-hoc analytics queries across highly variable, fast moving data sets. These same indexes can be used by Apache Spark to filter and extract only relevant subsets of data against which analytics can be run.

Indexes can be created on any field within a document. In addition to supporting single field and compound indexes, MongoDB also supports indexes of arrays with multiple values, short-lived data (i.e., Time To Live), partial, sparse, geospatial and text data, and can enforce constraints with unique indexes. Index intersection allows indexes to be dynamically combined at runtime to efficiently answer explorative questions of the data. Refer to the documentation for the full list of index types.

The MongoDB query optimizer selects the best index to use by running alternate query plans and selecting the index with the best response time for each query type. The results of this empirical test are stored as a cached query plan and are updated periodically.

MongoDB also supports covered queries – where returned results contain only indexed fields, without having to access and read from source documents. With the appropriate indexes, analytics workloads can be optimized to use predominantly covered queries.

Building upon MongoDB, Apache Spark can offer additional analytics capabilities to serve real time, operational processes.

Apache Spark: Extending MongoDB's Analytics Capabilities

Apache Spark is a powerful open source processing engine designed around speed, ease of use, and sophisticated analytics. Originally developed at UC Berkeley in 2009, Spark has seen rapid adoption by enterprises across a range of industries.

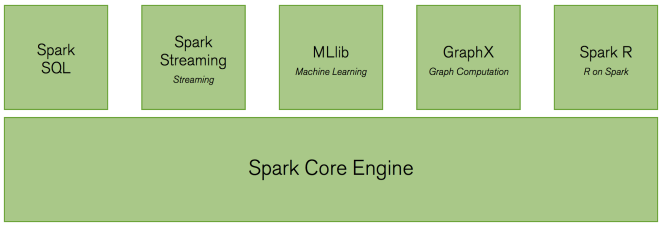

Figure 1: Apache Spark Ecosystem

Spark is a general-purpose framework used for many types of data processing. Spark comes packaged with support for machine learning, interactive queries (SQL), statistical queries with R, graph processing, ETL, and streaming. For loading and storing data, Spark integrates with a number of storage and messaging systems including Amazon S3, Kafka, HDFS, RDBMSs, NoSQL data stores, and MongoDB. Additionally, Spark supports a variety of popular development languages including Python, Scala, Java, R, and SQL.

Apache Spark Benefits

Spark was initially designed for interactive queries and iterative algorithms running mainly in-memory, as these were two major use cases not well served by batch frameworks such as Hadoop's MapReduce. Consequently Spark excels in scenarios that require fast performance, such as iterative processing, interactive querying, large-scale batch operations, streaming, and graph computations.

Developers and data scientists typically deploy Spark for:

- Simplicity. Easy-to-use APIs for operating on large datasets. This includes a collection of sophisticated operators for transforming and manipulating semi-structured data.
- Speed. By exploiting in-memory optimizations, Spark has shown up to 100x higher performance than MapReduce running on Hadoop.
- Unified Framework. Packaged with higher-level libraries, including support for SQL queries, machine learning, stream and graph processing. These standard libraries increase developer productivity and can be combined to create complex workflows.

Spark allows programmers to develop complex, multi-step data pipelines using a directed acyclic graph (DAG) pattern. It also supports in-memory data sharing across DAGs, so that different jobs can work with the same data. Spark takes MapReduce to the next level with less expensive shuffles during data processing. Spark holds intermediate results in memory rather than writing them to disk, which is useful especially when a process needs to iteratively work on the same dataset. Spark is designed as an execution engine that works with data both in-memory and on-disk. Spark operators perform external operations when data does not fit in memory. Spark can be used for processing datasets that are larger than the aggregate memory in a cluster.

The Resilient Distributed Dataset (RDD) is the core data structure used in the Spark framework. RDD is analogous to a table in a relational database or a collection in MongoDB. RDDs are immutable – they can be modified with a transformation, but the transformation returns a new RDD while the original RDD remains unchanged.

RDDs supports two types of operations: Transformations and Actions. Transformations return a new RDD after processing it with a function such as map, filter, flatMap, groupByKey, reduceByKey, aggregateByKey, pipe, or coalesce. The Action operation evaluates the RDD and returns a new value. When an Action function is called on a RDD object, all the processing queries are computed and the result value is returned. Some of the Action operations are reduce, collect, count, first, take, countByKey, and foreach.

As Apache Spark has continued to evolve, DataFrames and Datasets have been developed to provide higher levels of abstraction over RDDs. From Spark 2.0, the DataFrame and Dataset APIs have been merged, allowing greater developer ease of use through the enforcement of structures and type safety.

When to Use Spark with MongoDB

While MongoDB natively offers rich analytics capabilities, there are use-cases where integrating the Spark engine can extend the real-time processing of operational data managed by MongoDB. This allow users to operationalize results generated from Spark within real-time business processes supported by MongoDB. Examples of where it is

useful to combine Spark and MongoDB include the following.

Rich Operators & Algorithms

Spark supports over 100 different operators and algorithms for processing data. Developers can use these to perform advanced computations that would otherwise require more programmatic effort combining the MongoDB aggregation framework with application code. For example, Spark offers native support for advanced machine learning algorithms including k-means clustering and Gaussian mixture models.

Consider a web analytics platform that uses the MongoDB aggregation framework to maintain a real time dashboard displaying the number of clicks on an article by country; how often the article is shared across social media; and the number of shares by platform. With this data, analysts can quickly gain insight on how content is performing, optimizing user's experience for posts that are trending, with the ability to deliver critical feedback to the editors and ad-tech team.

Spark's machine learning algorithms can also be applied to the log, clickstream and user data stored in MongoDB to build precisely targeted content recommendations for its readers. Multi-class classifications are run to divide articles into granular sub-categories, before applying logistic regression and decision tree methods to match readers' interests to specific articles. The recommendations are then served back to users through MongoDB, as they browse the site.

Processing Paradigm

Many programming languages can use their own MongoDB drivers to execute queries against the database, returning results to the application where additional analytics can be run using standard machine learning and statistics libraries. For example, a developer could use the MongoDB Python or R drivers to query the database, loading the result sets into the application tier for additional processing.

However, this starts to become more complex when an analytical job in the application needs to be distributed across multiple threads and nodes. While MongoDB can service thousands of connections in parallel, the application would need to partition the data, distribute the processing across the cluster, and then merge results. Spark makes this kind of distributed processing easier and faster to develop. MongoDB exposes operational data to Spark's distributed processing layer to provide fast, real-time analysis. Combining Spark queries with MongoDB indexes allows data to be filtered, avoiding full collection scans and delivering low-latency responsiveness with minimal hardware and database overhead.

Skills Re-Use

With libraries for SQL, machine learning, graph processing and more – combined with programming in Java, Scala, R, and Python – developers can leverage existing skills and best practices to build sophisticated analytics workflows on top of MongoDB.

Configuring & Running Apache Spark with MongoDB

Written in Spark's native language, the Databricks-certified MongoDB Connector for Apache Spark provides a natural development experience for Spark users as they are quickly able to reuse existing Scala expertise. The connector exposes all of Spark's libraries, enabling MongoDB data to be materialized as DataFrames and Datasets for analysis with SQL, streaming, machine learning, and graph APIs.

The Spark connector can take advantage of MongoDB's aggregation pipeline and rich secondary indexes to extract, filter, and process only the range of data it needs – for example, analyzing all customers located in a specific geography. This is very different from typical NoSQL datastores that do not offer either secondary indexes or in-database aggregations. In these cases, Spark would need to extract all data based on a simple primary key, even if only a subset of that data is required for the Spark process. This means more processing overhead, more hardware, and longer time-to-insight for the analyst.

To maximize performance across large, distributed data sets, the Spark connector is aware of data locality in a

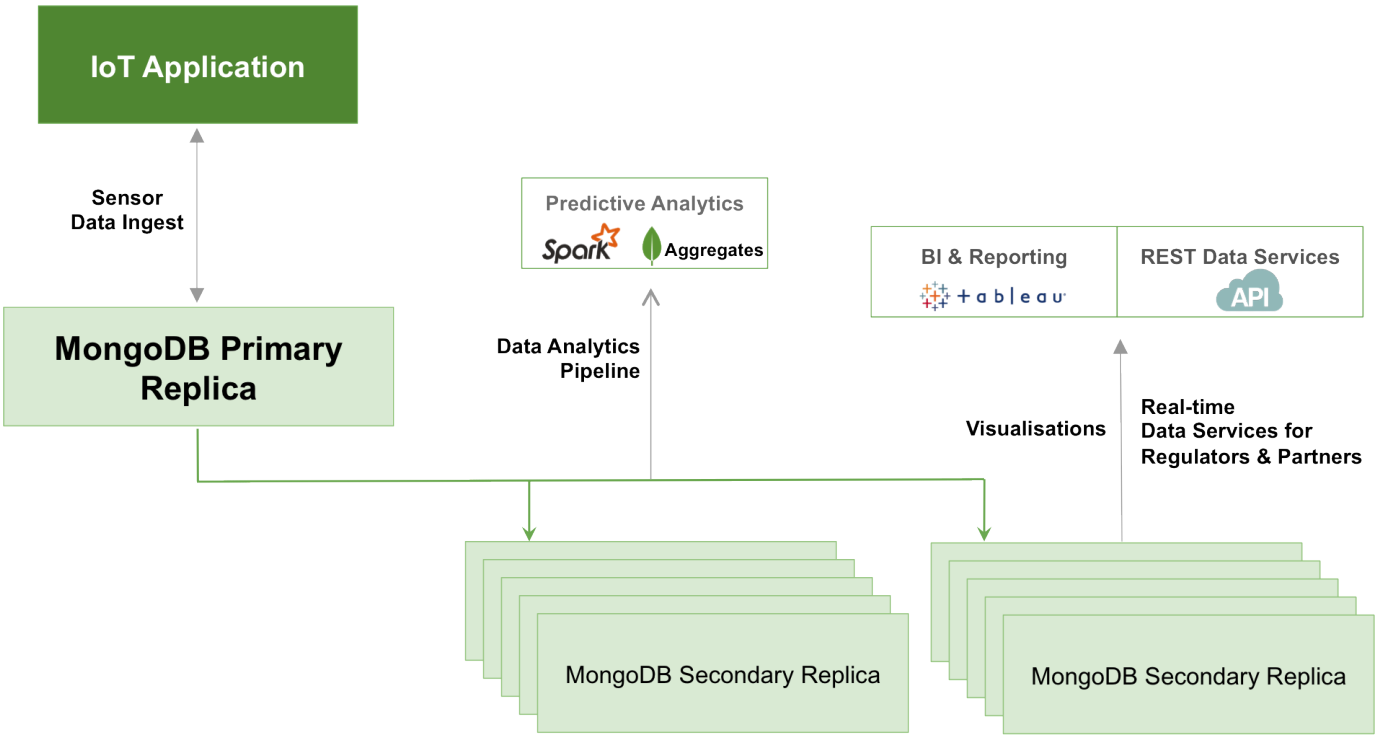

Figure 2: Isolating operational from analytical workloads in a single database cluster

MongoDB cluster. RDDs are automatically co-located with the associated MongoDB shard to minimize data movement across the cluster. The nearest read preference can be used to route Spark queries to the closest physical node in a MongoDB replica set, thus reducing latency.

Review the MongoDB Connector for Spark documentation to learn how to get started with the connector, and view code snippets for different APIs and libraries.

Enabling Analytics Against Live, Operational Workloads

Spark can be run on any physical node within a MongoDB replica set. However, executing heavyweight analytics jobs on a host that is also serving requests from an operational application can cause contention for system resources. Similarly the working set containing the indexes and most frequently accessed data in memory is overwritten by data needed to process analytics queries. Whether running MongoDB aggregations or Spark processes, it is a best practice to isolate competing traffic from the operational and analytics workloads.

Using native replication, MongoDB maintains multiple copies of data called replica sets. Replica sets are primarily designed to provide high availability and disaster recovery, and to isolate different workloads running over the same data set.

By default, all read and write operations are routed to the primary replica set member, thus ensuring strong consistency of data being read and written to the database. Applications can also be configured to read from secondary replica set members, where data is eventually consistent by default. Reads from secondaries can be useful in scenarios where it is acceptable for data to be several seconds (typically milliseconds) behind the live data, which is typical for many analytics and reporting applications. To ensure a secondary node serving analytics queries is never considered for election to the primary replica set member, it should be configured with a priority of 0.

Unlike other databases that have to offload Spark to dedicated analytics nodes that are running an entirely different configuration from the database nodes, all members of a MongoDB replica set share the same availability, scalability and security mechanisms. This approach reduces operational complexity and risk.

Creating Reactive Data Pipelines with Change Streams

MongoDB change streams enable developers and data engineers to build reactive, real-time, web, mobile, and IoT apps that can view, filter, and act on data changes as they occur in the database. Change streams enable seamless data movement across distributed database and analytics engines, making it simple to stream data changes and trigger actions wherever they are needed, using a fully reactive programming style. In addition to updating dashboards, analytics systems, and search engines as operational data changes, other use cases enabled by MongoDB change streams include:

- Powering trading applications that need to be updated in real time as stock prices rise and fall.
- Refreshing scoreboards in multiplayer games.
- Creating powerful IoT data pipelines that can react whenever the state of physical objects change.
- Synchronizing updates across serverless and microservices architectures by triggering an API call when a document is inserted or modified.

Integrating MongoDB & Spark Analytics with BI Tools & Hadoop

BI Integration

Real time analytics generated by MongoDB and Spark can serve both online operational applications and offline reporting systems, where it can be blended with historical data and analytics from other data sources. To power dashboards, reports and visualizations data stored in MongoDB can be easily explored with industry-standard SQL-based BI and analytics platforms. Using the MongoDB Connector for BI, analysts, data scientists and business users can seamlessly visualize semi-structured and unstructured data managed in MongoDB, alongside traditional data in their SQL databases, using the same BI tools deployed within millions of enterprises.

The Connector for BI translates SQL statements issued by the BI tool into equivalent MongoDB queries that are then sent to MongoDB for processing, and then returns results for visualization in the tool's UI.

Hadoop Integration

Many organizations are harnessing the power of Hadoop and MongoDB together to create complete data analytics pipelines:

- MongoDB powers the online, real time operational application, serving business processes and end-users, exposing analytics models created by Hadoop to operational processes
- Hadoop consumes data from MongoDB, blending it with data from other sources to generate sophisticated analytics and machine learning models. Results are loaded back to MongoDB to serve smarter and contextually-aware operational processes – i.e., delivering more relevant offers, faster identification of fraud, better prediction of failure rates from manufacturing processes.

Organizations such as eBay, Orbitz, Pearson, Square Enix and SFR are using MongoDB to operationalize their Hadoop-based data lakes.

The MongoDB Connector for Hadoop presents MongoDB as a Hadoop data source allowing Hadoop jobs to read data from MongoDB directly without first copying it to HDFS, thereby eliminating the need to move TBs of data between systems. Hadoop jobs can pass queries as filters, thereby avoiding the need to scan entire collections and speeding up processing; they can also take advantage of MongoDB's indexing capabilities, including text and geospatial indexes.

As well as reading from MongoDB, the Hadoop connector also allows results of Hadoop jobs to be written back out to MongoDB, to support real-time operational processes and ad-hoc querying by both MongoDB and Spark processes.

The Hadoop connector supports MapReduce, Spark on HDFS, Pig, Hadoop Streaming (with Node.js, Python or Ruby) and Flume jobs. Support is also available for SQL queries from Apache Hive to be run across MongoDB data sets. The connector is certified on the leading Hadoop distributions from Cloudera, Hortonworks, and MapR.

To learn more about creating an operational data lake with Hadoop and MongoDB, download the whitepaper.

MongoDB & Spark in the Wild

Many organizations are combining MongoDB and Spark to unlock real-time analytics from their operational data:

- China Eastern Airlines uses the MongoDB Connector for Apache Spark in it's new fare calculation engine, serving 1.6 billion queries per day. The flight inventory data is loaded into an Apache Spark cluster, which calculates fares by applying business rules stored in MongoDB. These rules compute fares across multiple permutations, including cabin class, route, direct or connecting flight, date, passenger status, and more. In total, 180 million prices are calculated every night. The results are then loaded into MongoDB where they are accessed by the flight search application. Learn more.
- An international banking group has been using Apache Spark and MongoDB to build sophisticated customer analytics. The applications extract and process data from multiple operational systems using Spark machine learning, SQL and streaming APIs, before loading analytics models into MongoDB for consumption by the business groups. The developers were able to reduce over 100 lines of integration code to just a single line after moving from a custom driver to the MongoDB Connector for Apache Spark.
- Qumram exposes user session data stored in MongoDB to Spark's machine learning processes to help global financial institutions detect fraud through behavioral analytics, and to apply deep learning techniques for sentiment analysis with Natural Language Processing.
- Air France has consolidated customer data scattered across dozens of systems into a single view stored in MongoDB. Spark processes are run against the live operational data in MongoDB to improve customer intimacy and personalize interactions in real time, as the customer is live on the web or speaking with the call center. Learn more.
- Stratio implemented its Pure Spark big data platform, combining Apache Spark and MongoDB, at a multinational banking group operating in 31 countries with 51 million clients. The platform enables the bank to create a unified real-time monitoring application to ensure high quality customer service across its online channels. Learn more.
- Artificial intelligence personal assistant company x.ai uses MongoDB and Spark for distributed machine learning problems. Building an artificial intelligence (AI) application requires huge amounts of data to be processed in parallel, both reliably and efficiently. To store the data, x.ai uses MongoDB for its flexible data model and its scaling capabilities. To build machine learning models the company has built robust pipelines in Scala using the distributed data processing capabilities of Spark. The company uses the MongoDB connector to bring Spark and MongoDB together.

Fast Track to Apache Spark: MongoDB University Course

To get the most out of any technology, you need more than documentation and code. Over 725,000 students have registered for developer and operations courses from MongoDB University. Now developers and budding data scientists can get a quick-start introduction to Apache Spark and the MongoDB connector through online MongoDB University course.

Getting Started with Spark and MongoDB provides an introduction to Spark and teaches students how to use the new connector to build data analytics applications. The course provides an overview of the Spark Scala and Java APIs, and includes sample code and demonstrations. Upon completing this course, students will be able to:

- Outline the roles of major components in the Spark framework
- Connect Spark to MongoDB
- Source data from MongoDB for processing in Spark
- Write data from Spark into MongoDB

The course is free. Sign up at MongoDB University.

Conclusion

MongoDB natively provides a rich analytics framework within the database. The Databricks-certified MongoDB Connector for Apache Spark enables data scientists and engineers to apply libraries for machine learning, streaming, graph processing and SQL to MongoDB data. Together MongoDB and Apache Spark are already enabling organizations to turn analytics into real-time action.

We Can Help

We are the MongoDB experts. Over 4,300 organizations rely on our commercial products, including startups and more than half of the Fortune 100. 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 backend as a service (BaaS), giving developers full access to MongoDB, declarative read/write controls, and integration with their choice of services.

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 Professional helps you manage your deployment and keep it running smoothly. It includes support from MongoDB engineers, as well as access to MongoDB Cloud Manager.

Development Support helps you get up and running quickly. It gives you a complete package of software and services for the early stages of your project.

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)

mongoDB

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