MongoDB®

MongoDB Architecture Guide

The foundational concepts that underpin the architecture of MongoDB

Introduction

Data and software are at the heart of every business. But for many organizations, realizing the full potential of the digital economy remains a significant challenge. Since the inception of MongoDB, we've understood that the biggest challenges developers face are related to working with data:

- Demands for higher productivity and faster time to market are being held back by rigid relational data models that are mismatched to modern code and impose complex interdependencies among engineering teams.
- Organizations are unable to work with, or extract insights from, the massive and rapidly growing amount of data generated by modern applications, including time series, geospatial, and polymorphic data.
- Monolithic and fragile legacy databases are inhibiting the wholesale shift to distributed systems and cloud computing that deliver the resilience and scale demanded by digital business and support new regulatory demands for data privacy.
- Previously separate transactional, analytical, search, and mobile workloads are converging to create rich data-driven applications and customer experiences. However, each workload has traditionally been powered by its own database, creating duplicated data silos stitched together with fragile ETL pipelines, accessed by different developer APIs.

To address some of these challenges, non-tabular (sometimes called NoSQL or non-relational) databases have been rapidly adopted over the past decade. But many of these NoSQL databases are simply Band-Aids, offering a niche set of functionality.

The problem is that typical NoSQL databases do one or two things well. They might offer more flexibility in the data model than traditional databases or scale out easily. But to do this, they discard the most valuable features of relational databases. They often sacrifice data integrity and the ability to work with data in the ways needed to build rich and valuable applications – whether these are new digital touchpoints with an organization's customers, or modernized core back-end business processes.

MongoDB

The Document Model

MongoDB was launched in 2009 as a completely new class of general-purpose database and quickly established itself as one of the most popular databases among developers. MongoDB retains the best aspects of relational and NoSQL databases while providing a technology foundation that enables organizations to meet the demands of modern applications. It does this by replacing the rigid tables of relational databases with flexible documents that map to the way developers think and code. Instead of storing data in columns and rows, document databases can store data as JSON (JavaScript Object Notation). A document database can store any type of data, and the structure of documents can be easily modified. This enables developers to be far more productive and build or iterate upon their applications faster. You can add new fields without affecting other documents in the collection, the MongoDB equivalent of a table in a relational database. And you can model data in any way that suits the application, for example, as key-value pairs, as the edges or nodes of a graph, or as nested structures that represent relationships.

Improved support for time series data

Time series data is data that represents how a system, a process, or a behavior changes over time. It may be captured at constant time intervals, like a device measurement per second, or at irregular time intervals, as with alerts and event audits. Time series data is critical for modern applications, in particular for IoT, stock trading, clickstreams, and social media. With the move from batch to real-time systems, the efficient capture and analysis of time series data enables organizations to better detect and respond to events ahead of their competitors, improve operational efficiency, and reduce cost and risk.

Thanks to MongoDB's flexibility, teams have been using the database to store time series data for years. However, correctly modeling the data to achieve a performant solution was not always straightforward. With time series collections – a new collection type introduced with MongoDB 5.0 – and new features such as clustered indexing and window functions, teams can work with and store time series data without having to worry about low-level model optimization. MongoDB will automatically optimize your schema for high-storage efficiency, low-latency queries, and real-time analytics against temporal data.

As developers have experienced the benefits of the document data model for themselves, it has become the most popular alternative to the tabular model used by traditional relational databases.

The three primary advantages of the document data model are:

1. Intuitive: faster and easier for developers

Documents in the database directly map to the objects in your code, so they are much more natural to work with.

The following example of a JSON document in MongoDB demonstrates how a customer object is modeled in a single document structure with related data embedded as subdocuments and arrays. This approach collapses what would otherwise be seven separate parent-child tables linked by foreign keys in a relational database.

```
{
  “_id”:
      ObjectId(“5ad88534e3632e1a35a58d00”),
  “name”: {
    “first”: “John”,
    “last”: “Doe” },
  “address”: [
    { “location”: “work”,
      “address”: {
        “street”: “16 Hatfields”,
        “city”: “London”,
        “postal_code”: “SE1 8DJ”},
      “geo”: { “type”: “Point”, “coord”: [
          51.5065752,-0.109081]}},
  ],
  “phone”: [
    { “location”: “work”,
      “number”: “+44-1234567890”},
    ],
  “dob”: ISODate(“1977-04-01T05:00:00Z”),
  “retirement_fund”:
        NumberDecimal(“1292815.75”)
}
```

With the document data model, there is no need to decompose data across tables, run expensive JOINs, or integrate a separate Object Relational Mapping (ORM) layer. Data that is accessed together is typically stored together, so you have less code to write and your users get higher performance.

> “The most beautiful part is the data model. Everything is a natural JSON document. So for the developers, it is easy – really easy – for them to work quickly. They’re spending time on building business value rather than data modeling.”
>
> – Filip Dadgar, IT manager, Toyota Material Handling Europe

2. Flexible schema: dynamically adapt to change

A document's schema is dynamic and self-describing, so you don't need to predefine it in the database. Fields can vary from document to document, and you can modify the structure at any time, allowing you to continuously integrate new application functionality without dealing with disruptive schema migrations.

When you need to make changes to the data model, the document database continues to store the updated objects without the need to perform costly "ALTER TABLE" operations, update a separate ORM middleware layer, and coordinate all of these changes across multiple developer, DBA, and ops teams. Documents allow multiple versions of the same schema to exist in the same table space. Old and new applications can coexist.

MongoDB also offers schema validation so you can enforce rules governing the structure of your documents. This is useful as your applications move into production because you can govern your schema without having to write controls in the application layer. With schema validation, you can apply data governance standards to a document schema while maintaining the benefits of a flexible data model in development.

3. Universal: JSON documents are everywhere

Lightweight and language-independent, JSON has become an established standard for data communication and storage. Documents allow you to structure data in any way your application needs – rich objects, key-value pairs, tables, geospatial and time series data, and the nodes and edges of a graph. As a result of these properties, you can serve many more classes of application with a single database.

MongoDB stores data as JSON documents in a binary representation called BSON (Binary JSON). Unlike most databases that store JSON data as primitive strings and numbers, the BSON encoding extends the JSON representation to include additional types such as int, long, date, floating point, and decimal128. This makes it much easier for applications using MongoDB to reliably process, sort, and compare data.

Working With Document Data

A few key differences among databases are the expressivity of the query language, the richness of indexing, and the data-integrity controls.

The MongoDB Query API is comprehensive and expressive. Ad hoc queries, indexing, and real-time aggregations provide powerful ways to access, group, transform, and analyze data. You can federate queries across databases, supporting transactional workloads and archived data in your data lake using the same query API and drivers, all with a single connection string.

The MongoDB Aggregation Pipeline allows you to transform and analyze data. Documents enter a multistage pipeline that transforms them into an aggregated result. The most basic pipeline stages provide filters that operate like queries and document transformations that modify the form of the output document. Other pipeline operations provide tools for grouping and sorting documents by specific fields, as well as tools for aggregating the contents of arrays, including arrays of documents. In addition, pipeline stages can use operators for tasks such as calculating an average or concatenating a string. The pipeline provides efficient data aggregation using native operations within MongoDB and is the preferred method for data aggregation in MongoDB.

With ACID transactions, you can maintain the same all-or-nothing and snapshot isolation guarantees as with relational databases. This remains possible whether you're manipulating data in a single document or with MongoDB's scale-out architecture, across multiple documents, and geographically distributed in multiple shards.

With strong data consistency, MongoDB eliminates the application complexity imposed by eventually consistent NoSQL systems. MongoDB's consistency guarantees are fully tunable, enabling you to balance data freshness against performance.

To make it easy for businesses to act on data in real time, many developers are building fully reactive event-driven data pipelines. MongoDB goes beyond many other databases with features such as Change Streams, which automatically detects and notifies consuming applications of any data modifications in the database.

The MongoDB Query API absolves developers from having to research, learn, and stay up-to-date on multiple ways to work with data across different workloads. It's more natural to use than SQL because it feels like an extension of the programming languages developers are already using. To further accelerate developer productivity, MongoDB provides native drivers for popular programming languages and frameworks. Supported drivers include Java, JavaScript, C#/.NET, Go, Python, PHP, Scala, Rust, and more. All supported MongoDB drivers are designed to be idiomatic for the given programming language. This eliminates the need for cumbersome and fragile ORM abstraction layers.

MongoDB.

Distributed Architecture: Scalable, Resilient, and Mission Critical

Through replica sets and native sharding, MongoDB enables you to scale out your applications with always-on availability. You can distribute data for low-latency user access while enforcing data sovereignty controls for data privacy regulations such as GDPR.

Availability and data protection with replica sets

MongoDB replica sets enable you to create up to 50 copies of your data, which can be provisioned across separate nodes, data centers, and geographic regions.

Replica sets are predominantly designed for resilience. If a primary node suffers an outage or is taken down for maintenance, the MongoDB cluster will automatically elect a replacement in a few seconds, switching over client connections and retrying any failed operations for you.

The replica set election process is controlled by sophisticated algorithms based on an extended implementation of the Raft consensus protocol. Before a secondary replica is promoted, the election algorithms evaluate a range of parameters including:

- Analysis of election identifiers, time stamps, and journal persistence to identify those replica set members that have applied the most recent updates from the primary replica
- Heartbeat and connectivity status with the majority of other replica set members
- User-defined priorities assigned to replica set members

By extending data protection, developers can configure replica sets to provide tunable, multinode durability and geographic awareness. Through MongoDB's write concern, you can ensure write operations propagate to a majority of replicas in a cluster. With MongoDB 5.0, the default durability guarantee has been elevated to the majority (w:majority) write concern. Write success will now only be acknowledged in the application once it has been committed and persisted to disk on a majority of replicas.

Choosing the new default versus the former w:1 default allows for a stronger durability guarantee, where acknowledged data can survive replica set elections and complete node failures. The new w:majority default setting is fully tunable, so you can maintain the earlier w:1 default or any custom write concern you had previously configured.

You can also create custom write concerns that target specific members of a replica

set, deployed locally and in remote regions. This ensures writes are only acknowledged once custom policies have been fulfilled, such as writing to at least a primary and replica in one region and at least one replica in a second region. This reduces the risk of data loss in the event of a complete regional failure.

Beyond resilience, replica sets can also be used to scale read operations, intelligently routing queries to a copy of the data that is physically closest to the user. With sophisticated policies such as hedged reads, the cluster will automatically route queries to the two closest nodes (measured by ping distance), returning results from the fastest replica. This helps minimize queries waiting on a node that might otherwise be busy, reducing 95th and 99th percentile read latency. Note that hedged reads are available in shared clusters only.

Scale Up, Out, and Across Storage Tiers

Like most databases, you can scale MongoDB vertically by moving to larger or smaller instance sizes. As a distributed system, MongoDB can perform a rolling restart of the replica set, enabling you to move between different instances without application downtime.

Through native sharding, MongoDB can also scale out your database across multiple nodes to handle write-intensive workloads and growing data sizes. Sharding with MongoDB allows you to seamlessly scale the database as your applications grow beyond the hardware limits of a single server, and it does so without adding complexity to the application.

To respond to evolving workload demands, you can add and remove shards anytime. You also have the flexibility to refine or change your shard key – which determines how data is distributed across a sharded cluster – on demand without impacting system availability. As your shard key is modified or as you change the cluster topology, MongoDB will automatically rebalance data across shards as needed without manual intervention.

By simply hashing a primary key value, many distributed databases randomly spray data across a cluster of nodes, imposing performance penalties when data is queried or adding application complexity when you need to locate data in a specific region. By exposing multiple sharding policies to developers, MongoDB offers a better approach. Data can be distributed according to query patterns or data placement requirements, giving you much

higher scalability across a more diverse set of workloads. MongoDB native sharding gives you the following options:

- **Ranged sharding:** Documents are partitioned across shards according to the shard key value. Documents with shard key values close to one another are likely to be co-located on the same shard. This approach is well suited for applications that need to optimize range-based queries, such as co-locating data for customers in a specific region on a specific set of shards.
- **Hashed sharding:** Documents are distributed according to an MD5 hash of the shard key value. This approach guarantees a uniform distribution of writes across shards, which is often optimal for ingesting streams of time series and event data.
- **Zoned sharding:** This allows developers to define specific rules governing data placement in a sharded cluster.

Beyond vertical and horizontal scaling, MongoDB also offers tiered scaling. When working in the cloud, the MongoDB Atlas Online Archive will automatically tier aged data out of the database and into cloud object storage. Archived data remains fully accessible with federated queries that span both object and database storage in a single connection string. This approach enables you to more economically scale data storage by moving it to a lower-cost storage tier without losing access to the data and without grappling with slow and complex ETL pipelines.

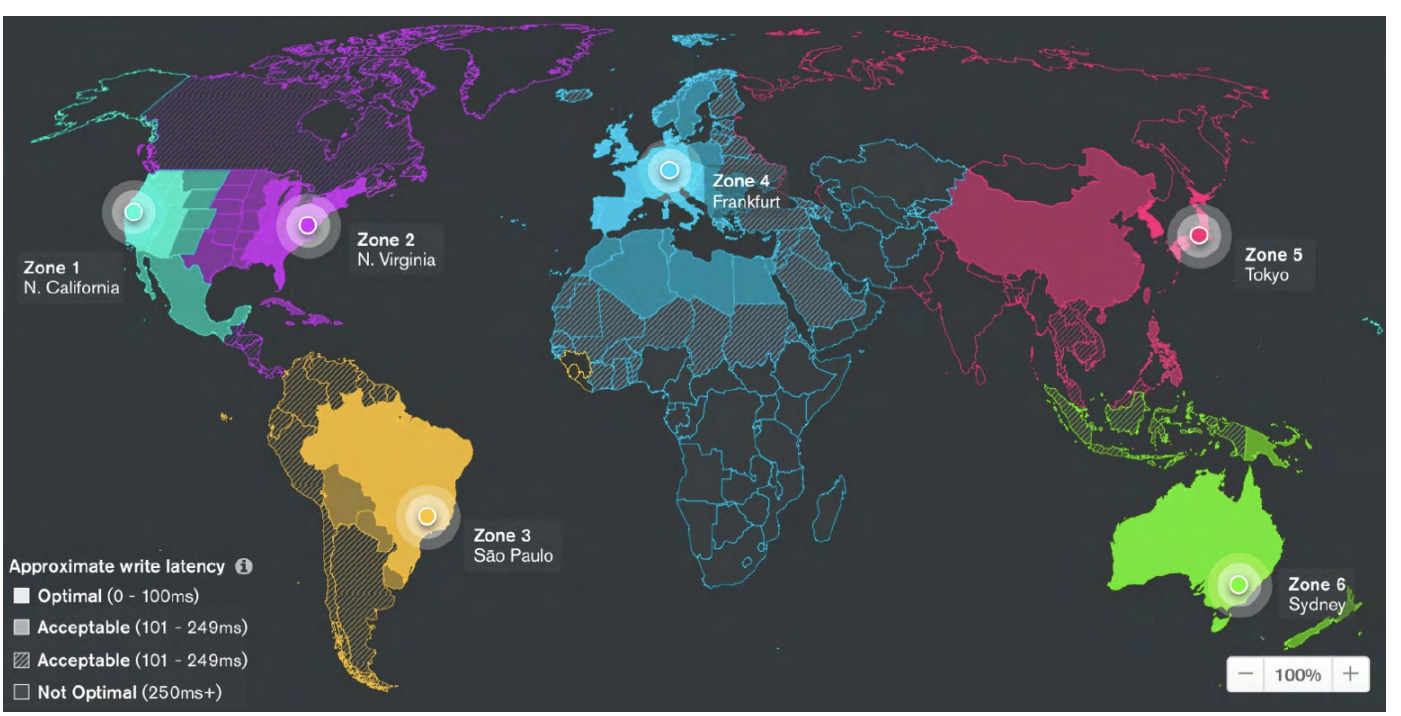

Figure 1: Serving always-on, globally distributed, write-everywhere apps with MongoDB Atlas Global Clusters

Privacy and Security

With the growing digital economy comes an increase in governmental oversight of privacy and data security. MongoDB includes extensive capabilities to defend, detect, and control access to data:

- **Authentication:** MongoDB offers a strong challenge-response mechanism based on SCRAM-256, along with integration to enterprise security infrastructure, including LDAP, Windows Active Directory, Kerberos, x.509 certificates, and AWS IAM.
- **Authorization:** Role-based access control (RBAC) enables you to configure granular permissions for a user or application based on the privileges they need to do their jobs.
- **Auditing:** For regulatory compliance, security administrators can use MongoDB's native audit log to record all database activity and changes.
- **Network isolation:** For users running fully managed databases in MongoDB Atlas, user data and underlying systems are fully isolated from other users. Database resources are associated with a user group, which is contained in its own virtual private cloud (VPC). Access can be granted only by IP whitelisting or VPC peering.
- **Encryption everywhere:** MongoDB data can be encrypted while in motion across the network, while in use in the database, and while at rest, whether on disk or in backups.

With client-side field-level encryption (FLE), you have access to some of the most advanced data protection controls anywhere. FLE makes it even safer to store your most sensitive data in the cloud because it's completely inaccessible to anyone who doesn't have the encryption keys, including those running the database for you.

FLE also makes it easier to comply with "right to be forgotten" conditions in privacy regulations, such as the GDPR and the CCPA. Simply destroy the customer key and the associated personal data is rendered useless.

With FLE, you can selectively encrypt individual document fields, each optionally secured with its own key and decrypted seamlessly on the client. In MongoDB, FLE is totally separated from the database, making it transparent to the server. Instead, it's handled exclusively within the MongoDB drivers on the client. All encrypted fields on the server – stored in memory, in system logs, at rest, and in backups – are rendered as ciphertext, making them unreadable to any party that does not have both client access and the keys necessary to decrypt the data. This is a different and more comprehensive approach than the column encryption used in many relational databases. Most of these databases handle encryption server-side, so data is still accessible to administrators who have access to the database instance itself, even if they have no client access privileges.

MongoDB

One Platform for All Your Workloads

Building on MongoDB's document data model, expressive Query API, and distributed systems DNA, the MongoDB Atlas application data platform delivers a cohesive and integrated set of data and database services. Atlas streamlines how teams work with data, specifically in the context of building software and systems that deliver real-time experiences to both end customers and internal users. Key characteristics of MongoDB's application data platform include:

- A data plane with the ability to support a wide variety of application types that can be independently developed, deployed, and evolved to address a wide variety of application types and use cases
- A unified and consistent experience for developers, data analysts, data scientists, and critical supporting functions such as operations teams, security teams, and data engineers
- Global, multi-cloud data distribution – built on MongoDB's native sharding – to support data residency requirements and provide deployment flexibility
- Transparent data movement between services and automated data life-cycle management

At its core, MongoDB Atlas provides a general-purpose database (Atlas Database) for modern applications. Nearly every application needs a fast database that can deliver single-digit millisecond response times. And with its flexible document data model, transactional guarantees, rich and expressive query API, and native support for both vertical and horizontal scaling, Atlas Database can be employed for practically any use case, reducing the need for specialized databases even as requirements change. Cluster auto-scale adjusts both compute and storage in response to application load, eliminating the need to monitor utilization and react to scaling needs.

Atlas Database is available in more than 80 regions across AWS, Google Cloud Platform, and Azure. Best-in-class infrastructure and database automation ensure continuous availability, elastic scalability, and compliance with the most demanding data security and privacy standards. Uptime is backed by a 99.995% service-level agreement.

Beyond offering fully managed MongoDB databases in the cloud, Atlas provides additional complementary services that allow organizations to support a wide range of application and real-time analytics data workloads.

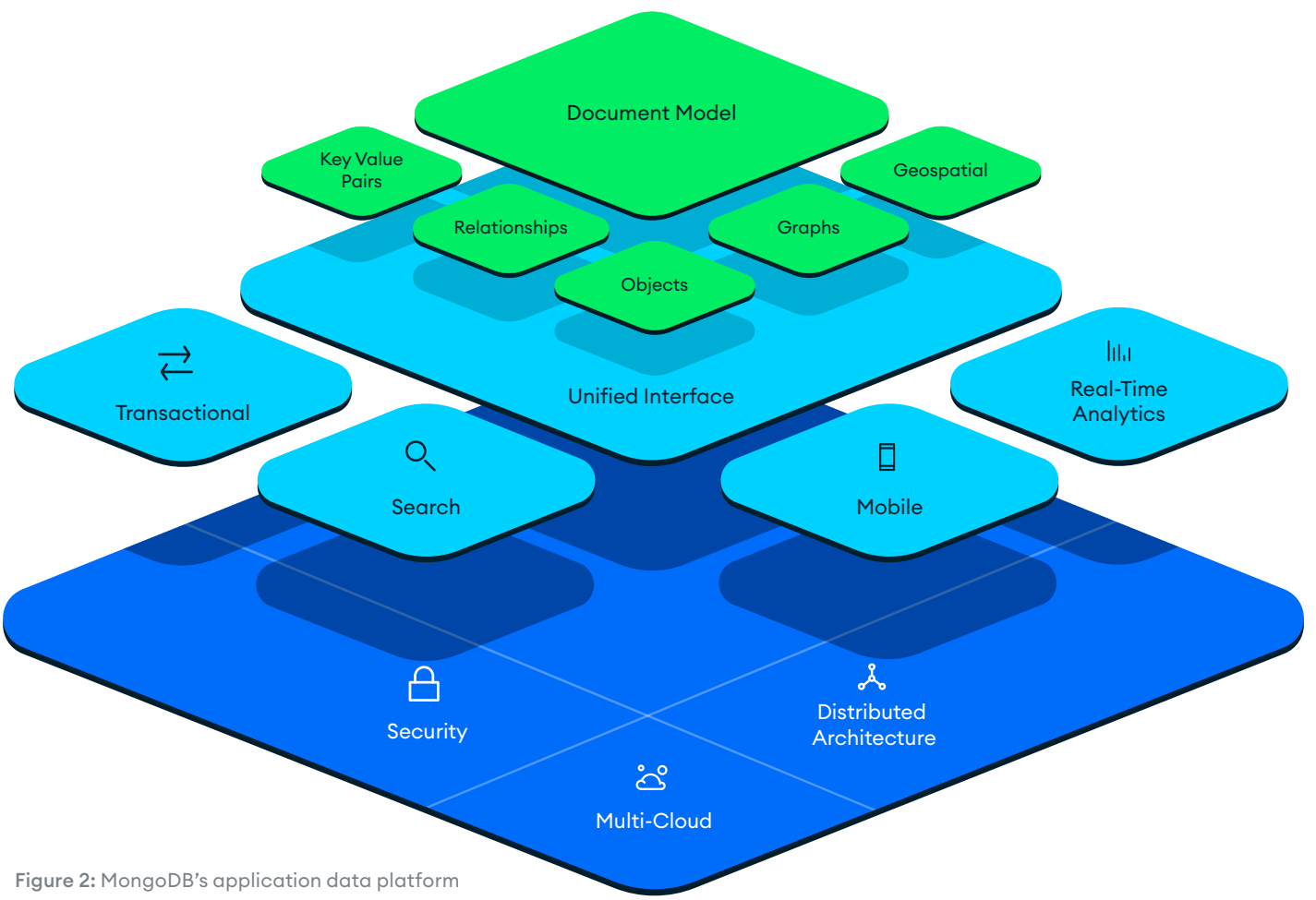

Figure 2: MongoDB's application data platform

Real-Time Analytics

Atlas Database allows you to deploy a read-only analytics node to serve more resource-intensive analytics queries. You can easily target analytics nodes by configuring the read preference, effectively ensuring that analytics queries leveraging MongoDB's built-in aggregation pipeline never contend for database resources with your operational workloads. Analytics nodes, like all read-only nodes within a MongoDB cluster, do not participate in elections and can never be elected to the cluster primary.

Atlas Search

Atlas Search is built into MongoDB Atlas, making it easy to build fast, full-text search capabilities on top of your MongoDB data with no need to learn a different API or deploy a separate search technology. Atlas Search is built on top of Apache Lucene, the industry standard library. Search indexes run alongside the database and are automatically kept in sync. Supported search capabilities include fuzzy search, autocomplete, facets and filters, custom scoring, analyzers for more than 30 languages, and more.

Atlas Data Lake

Atlas Data Lake is an on-demand query service that enables you to analyze data in cloud object storage (Amazon S3) in place using the MongoDB Query API. There is no infrastructure to set up or manage. Atlas Data Lake automatically

The best way to run MongoDB in the cloud

Atlas Database delivers MongoDB as a pay-as-you-go service billed on an hourly basis. To deploy it, you can use a GUI or the admin API to select the public cloud provider, region, instance size, and features you need. Atlas Database provides:

- Automated database and infrastructure provisioning along with auto-scaling, so teams can get the database resources they need, when they need them, and elastically scale in response to application demands.
- Always-on security to protect data, including network isolation, fine-grained access controls, auditing, and end-to-end encryption down to the level of individual fields.
- Certifications with global standards for supporting compliance, including ISO 27001, SOC 2, and more. Atlas Database can be used for workloads subject to HIPAA, PCI-DSS, or GDPR.
- Built in replication both within and across regions for always-on availability, even in the face of complete regional outages.
- Global Clusters for fully managed, globally distributed databases that provide low-

Continued on next page »

parallelizes operations by breaking down queries and dividing the work across multiple compute nodes. Atlas Data Lake can also automatically optimize workloads by utilizing compute in the region closest to your data. This is useful for data residency, granting you the ability to specify the region in which your data should be processed.

Support for federated queries allows you to combine and analyze data across S3 and your Atlas database clusters, together with a single query. In addition, you can easily persist the results of aggregations to either object storage or your cloud database. Supported data formats include JSON, BSON, CSV, TSV, Avro, ORC, and Parquet.

Atlas Charts

Atlas Charts is a data visualization service that natively supports richly structured JSON data. Easily create charts, graphs, and dashboards in a drag-and-drop interface, and share them with other users for collaboration or embed them directly into your applications to create engaging user experiences. Atlas Charts can be configured to read from analytics or secondary nodes, ensuring no impact to operational workloads. Supported data sources include one or more Atlas Database deployments, Atlas Data Lake, or a combination of both.

Realm Sync

Realm Sync provides bidirectional data sync between Atlas Database clusters and Realm, a lightweight, open-source mobile

latency, responsive reads and writes to users anywhere, with strong data sovereignty controls for regulatory compliance. Global Clusters allow you to quickly implement zoned sharding using a visual UI or the Atlas Admin API. Each zone is part of the same cluster, so they can be queried globally, but data is pinned to shards in specific regions based on data localization policies.

- Multi-cloud clusters allow you to distribute the data in a single logical database across multiple cloud providers for cross-cloud redundancy, even wider geographic reach, and seamless migrations across cloud providers. Multi-cloud clusters can also be used to easily leverage the best services from each cloud provider on your live, operational data – e.g., users who run primarily in AWS can quickly spin up a replica on Google Cloud to take advantage of Google's latest AI/ML services.
- Fully managed backups with point-in-time recovery to protect against data corruption, and the ability to query backups in place without full restores.
- Fine-grained monitoring, real-time metrics, query profiler, and customizable alerts for comprehensive performance visibility.

Continued on next page »

database. Realm is a more developer-friendly alternative to embedded data stores such as SQLite or Core Data. This joint solution helps solve the unique challenges of building offline-first applications for mobile, making it simple to store data on-device – allowing data access even when offline – and enabling bidirectional updates when a connection is established. Realm's SDKs give developers the tools needed to access data stored in MongoDB Atlas directly from the client and interact with the platform's broader set of services.

GraphQL

Automatically generate a JSON schema for your MongoDB collections and enable GraphQL for your MongoDB apps with a simple click. By querying against a single endpoint to get exactly the data you need, you can build highly performant applications. When you use GraphQL alongside MongoDB's other app development features – such as built-in authentication and data access control – it's also simple to secure your app.

Event-Driven Architecture

Part of the broader services available to Atlas users, functions allow you to define and execute server-side logic without having to provision or manage servers, making it easy to integrate with cloud services, build APIs, and more. Atlas Triggers allow you to automatically execute functions in real time – in response to changes in the database or user-authentication events, or at preset intervals.

- Intelligent schema and index recommendations with the Performance Advisor, which analyzes slow query logs of your database collections and provides suggestions ranked by impact to your database performance.
- Automated patching and single-click upgrades for new major versions of the database, enabling you to take advantage of the latest MongoDB features.
- Auto-archiving of aged data from your live database clusters to fully managed cloud object storage with Online Archive. Federated query enables you to analyze your data from your operational database and historical data on object storage together and in place with a single query for faster insights. Queries are automatically routed to the appropriate data service without having to think about data movement, replication, or ETL.
- Live migration to move a self-managed MongoDB database into the Atlas service or to move Atlas databases between cloud providers.
- A 512 MB perpetual free tier.

Atlas Database is serving a vast range of workloads for startups, Fortune 500 companies, and government agencies, including mission-critical applications handling highly sensitive data in regulated industries.

MongoDB for Mission-Critical Applications in Your Data Center

If you need to run MongoDB on your own self-managed infrastructure for business or regulatory requirements, MongoDB Enterprise Advanced is a finely tuned package of advanced software, support, certifications, and other services that can help. Enterprise Advanced can be used to power a MongoDB database behind a single application, or to build your own private database service and expose it to your development teams.

As part of the MongoDB Enterprise Advanced subscription, MongoDB Enterprise Server is a version of the database software that includes an in-memory storage engine for high throughput and predictable low latency; advanced security options, such as LDAP and Kerberos access controls; comprehensive auditing; and an encrypted storage engine for protecting data at rest.

MongoDB Ops Manager simplifies the administration tasks associated with running MongoDB on premises or in a private cloud. With Ops Manager, you can automate deployment, monitoring, backup, and scaling of MongoDB. You can also manage the complete life cycle of your MongoDB databases via a powerful GUI, or programmatically with APIs to enable integration with your Infrastructure-as-Code (IaC) tools.

Kubernetes users can use the MongoDB Enterprise Operator for Kubernetes, which integrates with MongoDB Ops Manager to automate and manage MongoDB clusters. It gives you full control over your MongoDB deployment from a single Kubernetes control plane. You can use the operator with upstream Kubernetes, or with any popular distribution such as Red Hat OpenShift or Pivotal Container Service (PKS).

The MongoDB Connector for BI lets you use MongoDB as a data source for your existing SQL-based BI and analytics platforms such as Tableau, Microstrategy, Looker, and more. It is included with MongoDB Enterprise Advanced and available in a pay-as-you-go model for Atlas database clusters.

Run MongoDB for Free With Tools From Us

MongoDB Community Server is the free, source-available version of the database software. It has been downloaded hundreds of millions of times and includes all the core database functionality – flexible document model, expressive query API, replication, and sharding – to support building a wide variety of applications.

MongoDB Compass

You can easily interact with your MongoDB data using MongoDB Compass, the GUI for MongoDB. Through Compass you can explore and manipulate data, create queries and aggregation pipelines visually from the GUI and then export them as code to your app, view and create indexes, build schema validation rules, and more.

Cloud Manager

Cloud Manager is the cloud-based management platform that enables you to deploy, monitor, back up, and scale MongoDB. It enables you to automate administration tasks like deployment, monitoring and alerts, scaling, upgrades, backup, and performance optimization.

It also helps you identify issues before they become emergencies and streamline operations. With Cloud Manager, you can monitor trends, see live workload characteristics, set up alerts, and get performance-optimization suggestions.

Connectors

The MongoDB Connector for Apache Spark exposes all of Spark's libraries, including Scala, Java, Python, and R. MongoDB data is materialized as DataFrames and Datasets for analysis with machine learning, graph, streaming, and SQL APIs.

With the MongoDB Connector for Apache Kafka, you can build robust data pipelines that move events between systems in real time, using MongoDB as both a source and sink for Kafka. The connector is supported by MongoDB and verified by Confluent.

You can use any distribution of Kubernetes to manage the full life cycle of your MongoDB clusters, wherever you choose to run them, from on-premises infrastructure to the public cloud. With MongoDB's Kubernetes integrations, you can run and scale your clusters with ease regardless of your chosen infrastructure topology.

MongoDB.

Getting Started

Every industry is in the midst of digital transformation. Many businesses are unable to realize the full potential of their investments because they fail to modernize their data architecture. As you build or remake your company for a digital world, speed matters – measured by how fast you build applications, scale them, and gain insights from the data they generate. These are the keys to applications that provide better customer experiences; enable deeper, data-driven insights; and make new products or business models possible. MongoDB enables you to meet the demands of modern apps with a complete application data platform that includes all the complementary services developers need.

In this guide we explored the foundational concepts that underpin the architecture of MongoDB. Other guides on topics such as performance, operations, and security best practices can be found at MongoDB.com.

You can get started now with MongoDB by:

1. Reviewing the Use Case Guidance White Paper to identify applicable use cases for MongoDB.
2. Spinning up a fully managed MongoDB cluster on the Atlas free tier or downloading MongoDB for local development.
3. Reviewing the MongoDB manuals and tutorials in our documentation.

Safe Harbor

The development, release, and timing of any features or functionality described for our products remains at our 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.