# MongoDB Multi-Data Center Deployments

A MongoDB Whitepaper
November 2017

# Table of Contents

# Introduction

Business critical applications have always required continuous availability. As more organizations launch services online for consumption by global audiences, availability and scalability across distributed geographic regions become increasingly important considerations in systems design.

There are three principal reasons for the geographic distribution of databases across multiple data centres:

- **Continuous Availability:** Whether the database is deployed on-premise or in a public cloud, the business needs assurance that the service will survive a regional disaster that causes a complete data center outage. Examples include fires, floods or hurricanes. Gartner estimates downtime costs a business an average of $300,000 per hour, with losses much higher for global, internet-based operations.
- **Customer Experience:** Global audiences need consistent, low latency experiences, wherever they are located. Amazon famously concluded that each 100ms in added latency resulted in a 1% loss of sales.
- **Regulatory Compliance:** National governments are placing controls on where customer data is physically located. That data is not allowed to be stored outside of it's country's borders.

With data center-aware replication, scaling and operational automation, MongoDB helps organizations efficiently deploy applications globally with fast time to market, high customer satisfaction and low costs. Customers can choose to deploy MongoDB across their own data center facilities, across those provided by public cloud providers, or use the MongoDB Atlas cloud database service with cross-region replication.

This whitepaper explores the technologies that provide the foundation for MongoDB's data-center awareness; presents multiple deployment scenarios, and then concludes with examples from production users.

# Maintaining Service Continuity

Under normal operating conditions, a MongoDB deployment will perform according to the performance and functional goals of the system. However, from time to time certain inevitable failures or unintended actions can affect a system in adverse ways. Storage devices, network connectivity, power supplies, and other hardware components will fail. These risks can be mitigated with redundant hardware components. Similarly, a MongoDB database provides configurable redundancy throughout both its software components and it's data storage.

## Replica Sets

MongoDB maintains multiple copies of data, called replica sets, using native replication. Users should deploy replica sets to help prevent database downtime. Replica sets are self-healing as failover and recovery is fully automated, so it is not necessary to manually intervene to restore a system in the event of a failure.

Replica sets also provide operational flexibility by providing a way to perform system maintenance (i.e. upgrading hardware and software) while preserving service continuity. This is an important capability as these operations can account for as much as one third of all downtime in traditional systems.

A replica set consists of multiple database replicas. At any given time, one member acts as the primary replica set member and the other members act as secondary replica set members. If the primary member suffers an outage (e.g., a power failure, hardware fault, network partition), one of the secondary members is automatically elected to primary, typically within several seconds, and the client connections automatically failover to that new primary. Read operations can continue to be serviced by secondary replicas while the election of a new primary is in progress. To ensure write availability, the MongoDB driver will automatically retry write operations in the event of a node failure, while the MongoDB server enforces exactly-once processing semantics.

The number of replicas in a MongoDB replica set is configurable, with a larger number of replica members

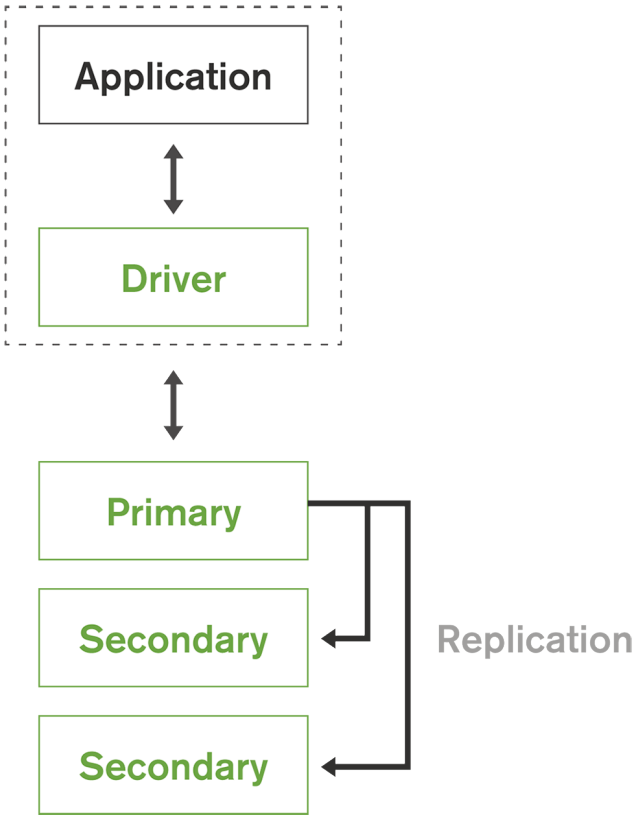


**Figure 1:** Self-Healing MongoDB Replica Sets for Continuous Availability

providing increased data durability and protection against database downtime (e.g., in case of multiple machine failures, rack failures, data center failures, or network partitions). Up to 50 members can be configured per replica set, providing operational flexibility and wide data distribution across multiple geographic sites.

You can learn more about the members of a replica set from the documentation.

## Replica Set Elections

In the event of a primary replica set member failing, the election process is controlled by sophisticated algorithms based on an extended implementation of the Raft consensus protocol. Not only does this allow fast failover to maximize service availability, the algorithms ensure only the most suitable secondary members are evaluated for election to primary, and reduce the risk of unnecessary failovers (also known as "false positives"). Before a secondary replica set member is promoted, the election algorithms evaluate a range of parameters including:

- Analysis of election identifiers, timestamps and journal persistence to identify those replica set members that have applied the most recent updates from the primary member
- Heartbeat and connectivity status with the majority of other replica set members
- User-defined priorities assigned to replica set members. For example, administrators can configure all replicas located in a secondary data center to be candidates for election only if the primary data center fails

Once the election process has determined the new primary, the secondary members automatically start replicating from it. When the original primary comes back online, it will recognize it's change in state and automatically assume the role of a secondary, applying all write operations that have occurred during it's outage.

A tutorial is available providing best practices and guidance on deploying MongoDB replica sets across data centers.

## Scaling MongoDB with Continuous Availability: Shards and Replica Sets

MongoDB provides horizontal scale-out for databases using a technique called sharding, allowing MongoDB deployments to scale beyond the hardware limitations of a single server.

Sharding distributes data across multiple physical partitions called shards. Shards can be located within a single data center or distributed across multiple data centers. As illustrated in Figure 2, each shard is deployed in a replica set, to provide both scalability and high availability to the MongoDB cluster.

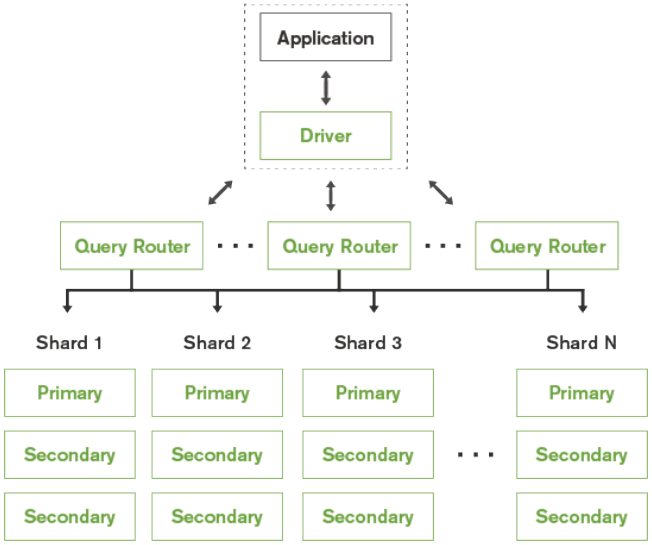


**Figure 2:** Sharding and replica sets - automatic sharding provides horizontal scalability; replica sets help prevent database downtime.

Sharding is transparent to applications; whether there are one or one hundred shards, the application code for querying MongoDB remains the same. Applications issue queries to a query router that dispatches the query to the appropriate shards. The query router abstracts all database partitioning, or sharding, from client applications which "see" a single database, regardless of the number of shards. Adding shards to the database is also transparent to the client applications. The query router is typically deployed on the application servers, thus eliminating unnecessary network hops. Multiple query routers can be used with a MongoDB system, and the appropriate number is determined based on performance and availability requirements of the application.

## MongoDB Data Center Awareness

MongoDB provides a rich set of features to help users deploy always-on, scalable systems. In designing for high availability, administrators must evaluate read and write operations in the context of different failure scenarios. The performance and availability SLAs (Service Level Agreements) of a system play a significant role in determining:

- How the database is sharded (sharding policies are discussed later)
- The number of replicas (copies) of the data
- The physical location of shards and replica sets, both within and across multiple data centers

Administrators can configure the behavior of MongoDB's shards and replica sets to enable data center awareness. Configuration can be based on a number of different dimensions, including awareness of geographical regions in multi-data center deployments, or racks, networks, and power circuits in a single data center.

With MongoDB, administrators can:

- Ensure write operations propagate to specific members of a replica set, deployed locally and in remote data centers. This reduces the risk of data loss in the event of a complete data center outage. Alternatively, replication can be configured to ensure data is only replicated to nodes within a specific region to ensure data never leaves a country's borders.
- Ensure that specific members of a replica set respond to queries -- for example, based on their location. This reduces the effect of geographic latency
- Place specific data partitions on specific shards, each of which could be deployed in different data centers. Again this can be used to reduce geographic latency and maintain data sovereignty.

Read and write behavior can be configured per operation and per collection. Collectively they enable users to precisely control and scale database operations across regions, based on specific application requirements.

## Configuring Write Operations Across Data Centers

MongoDB allows users to specify write availability in the system, using an option called write concern. Each operation can specify the appropriate write concern, ranging from unacknowledged to an acknowledgement that writes have been committed to:

1. A single replica (i.e. the primary replica set member). This is the default write concern;
2. Multiple replicas;
3. A majority of replicas;
4. All replicas.

It is also possible to configure the write concern so that writes are only acknowledged once specific policies have been fulfilled, such as writing to at least two replica set members in one data center and at least one replica in a second data center. In the case of sharded MongoDB clusters, this option enables users to write to multiple data centers in parallel.

## Configuring Location-Aware Reads Across Data Centers

Delivering a low latency experience to customers wherever they are located is a key design consideration for distributed systems. Using MongoDB's native replica sets, copies (replicas) of the database can be deployed to sites physically closer to users, thereby reducing the effects of network latency.

The MongoDB read preference configures how MongoDB clients route read operations to members of a replica set. By default, an application ensures strong consistency by directing all read operations to the primary member in a replica set, but behavior can be controlled in the driver on a per-operation basis. The nearest read option allows the client to read from the lowest-latency members of a replica set, rather than always reading from the primary. This is typically used to route queries to a local data center, reducing the effects of geographic latency. Tags can also be used to ensure that reads are always routed to a specific node or subset of nodes.

## Configuring Shards Across Data Centers

As discussed earlier, sharding is used to scale-out a MongoDB database across multiple nodes. For maximum flexibility, MongoDB supports three sharding policies that can be defined by the user to optimize scaling based on the applications query patterns and deployment requirements:

| | New York City | San Francisco | London | Sydney |
|---|---|---|---|---|
| New York City | -- | 84.42 | 70.64 | 231.46 |
| San Francisco | 85.31 | -- | 171.91 | 158.26 |
| London | 69.39 | 168.20 | -- | 318.77 |
| Sydney | 231.14 | 158.26 | 315.53 | -- |

**Table 1:** Network latencies (milliseconds) between cities[1]

### Range-based 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 retrieving all data for customers in a given user, or data that is accessed in ranges such as time series data.

### Hash-based Sharding

Documents are uniformly distributed using an MD5 hash of the shard key value. Documents with shard key values "close" to one another are unlikely to be co-located on the same shard. This approach guarantees a uniform distribution of writes across shards, but is less optimal for applications that query for groups of documents that are accessed together.

### Zones Sharding

Provides the the ability for DBAs and operations teams to define specific rules governing data placement in a sharded cluster. Zones accommodate a range of deployment scenarios – for example locating data by geographic region, by hardware configuration for tiered storage architectures, or by application feature. Administrators can continuously refine data placement rules by modifying shard key ranges, and MongoDB will automatically migrate the data to its new zone. Refer to the zones documentation for more detail.

### Network Compression for Efficient Cross-Regional Deployments

As a distributed database, MongoDB relies on efficient network transport during query routing and inter-node replication. MongoDB offers compression of the wire protocol from clients to the database, and of intra-cluster traffic. Network traffic can be compressed by up to 80%, bringing major performance gains to busy network environments and reducing connectivity costs, especially in public cloud environments, or when connecting remote assets such as IoT devices and gateways.

## MongoDB Deployment Patterns

The following figures illustrate how data center awareness in MongoDB enables high availability and scalability for both regional and global deployments.

1. https://wondernetwork.com/pings/

## High Availability: Single Data Center

Within a data center the replica set can be distributed across racks so that no single rack contains more than one replica member for a given shard.

In this configuration server and rack failures are tolerated, with the number of replica set members determining the extent of failure that can be sustained by the system, without affecting availability. This deployment would not tolerate failure of the data center itself.

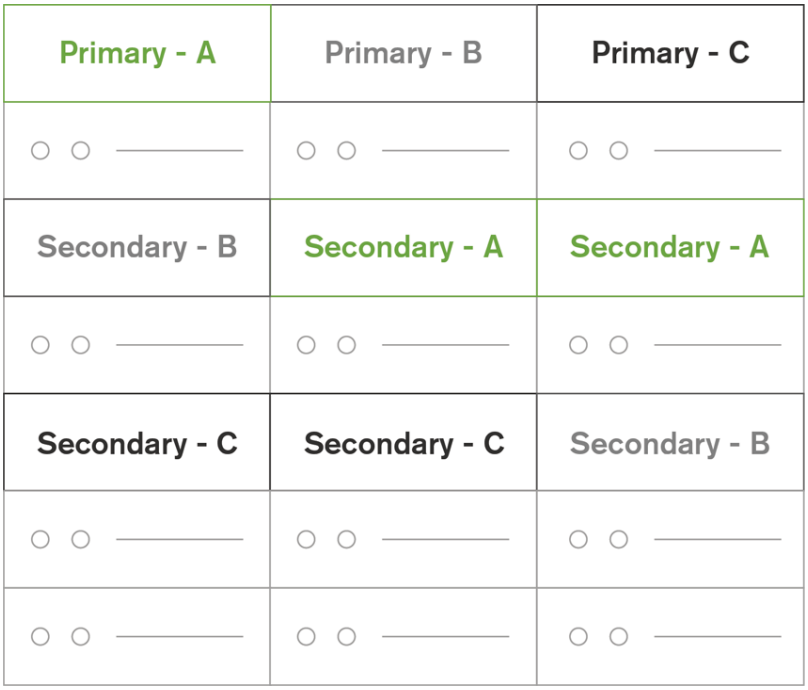


**Figure 3:** Single Data Center - Tolerates Server and Rack Failures

## High Availability: Active/Standby Data Centers

In this typical Disaster Recovery configuration, replica set members are distributed across racks in two data centers. The active data center (Data Center West) contains an even number of replica members including the primary member.

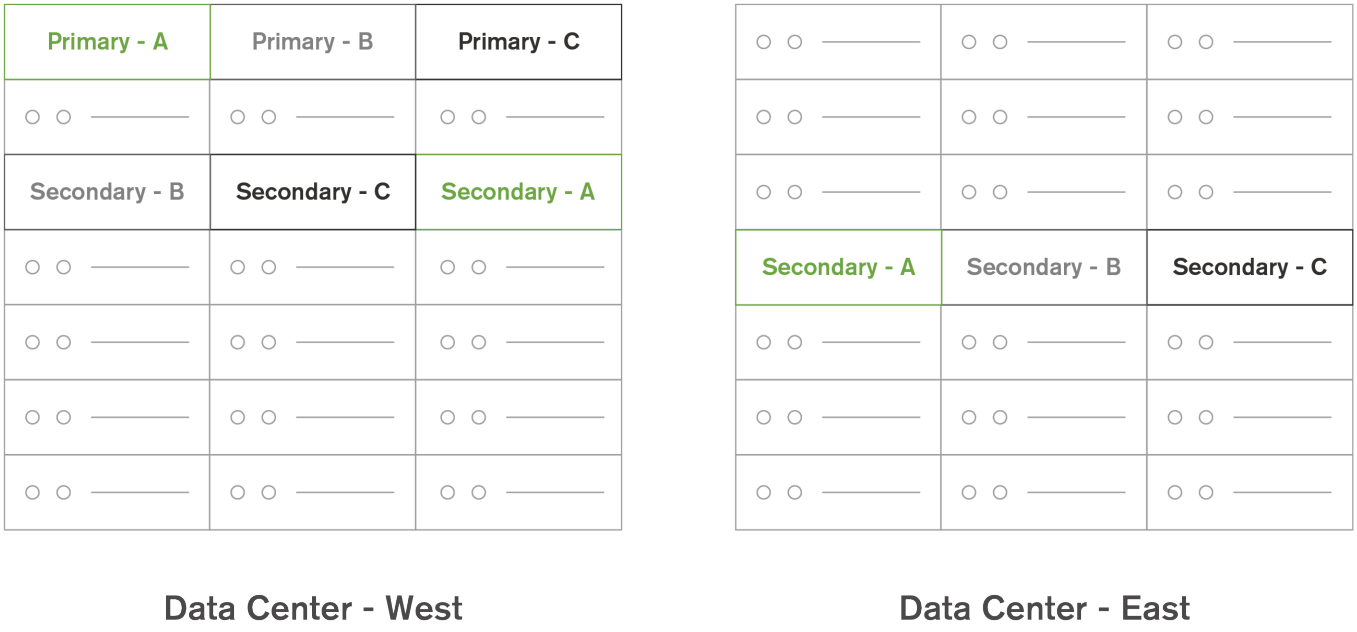


**Figure 4:** Active/Standby Data Centers - Tolerates Data Center Failure

The standby or Disaster Recovery data centre in the East is provisioned with a secondary replica set member, configured with a lower election priority (0.5 is recommended). This ensures that only nodes in the West data center will be Primary, unless they are all unavailable due to a complete data center outage. In this Active / Standby configuration, the user is able to survive a loss of either of the data centres, while the application may use majority write concern to maintain durable write guarantees across any server failure.

In the event of the West data center failing, the operation team's priority is to bring the East into production, while precluding the possibility of a "split brain" should the West recover. To do this, the administrator must ensure that at least one of the MongoDB nodes in West stays down, through a definite measure such as removing power or confirming the permanent physical loss of the West. Next a "forced" reconfiguration of the replica set at the East node removes votes from both members in the West. As a majority of one, the East node immediately elects itself Primary and restores production capability.

If the West is out of service for any length of time, additional replica set members should be added to the East in order to restore redundancy to the MongoDB deployment. In the more likely event that the West recovers, then the following steps should be actioned:

As soon as connectivity is restored, the recovered replica set member in the West automatically resynchronizes with the current primary replica in the East. The remaining voting member in the East then uses its lone vote to elect the higher-priority node in the West as Primary.

Restart the remaining replica set member in the West. It automatically resynchronizes with the rest of the replica set and assumes Secondary role.

Restore election voting rights to the MongoDB nodes in the West, which completes the recovery to initial state.

When coupled with MongoDB's continuous backup and point-in-time recovery tooling, the Active / Standby design pattern demonstrated in this example will provide business continuity through disaster recovery and low RPO (Recovery Point Objective). However, with it's distributed architecture, MongoDB affords much more flexible configuration with support for Active / Active data centre configurations, discussed later.

Note that this deployment is appropriate for a single replica set or for a sharded cluster where the cluster balancer is not used. In those multi-data center environments where the balancer is used to automatically redistribute data around the cluster, it is recommended to deploy MongoDB in an active/active data center configuration.

## Moving Beyond Active / Standby: Global Data Distribution

To reduce the effects of geographic latency, MongoDB can replicate data to local sites where it is accessed by local users. Reads can be issued with a read preference for `nearest`, ensuring the data is served from the closest data center to the user, based on ping distance.

These reads are eventually consistent with the primary, but usually no more than a few milliseconds behind the primary, plus network latency. Writes are issued to the primary.

A typical use case for this type of deployment is distributing data and content assets to geographically remote audiences. Data sets are written to the primary server which then propagates them to replica members in data centers around the world, where they are accessed with low latency by the local audience.

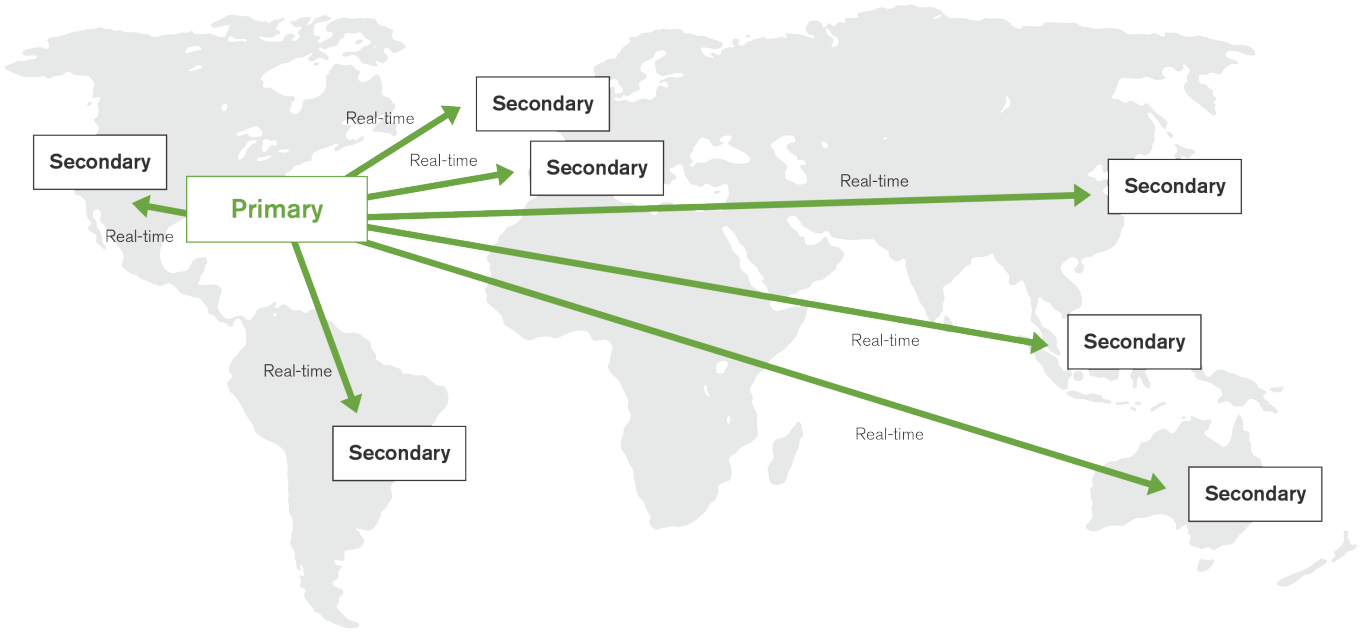


**Figure 5:** Global Data Distribution - Eliminating Geographic Read Latency

## Active/Active Deployments: High Scalability & Availability

Using zone sharding, administrators can pin specific partitions of their database to specific geographic regions, and simultaneously write to the database across multiple regions. Each Zone is part of the same, single cluster and can be queried globally, but data resides in the correct location based on local access requirements and data sovereignty. By associating data to shards within specific data centres, administrators are able to maintain low latency access.

To illustrate further, review figure 6 depicting a database architecture distributed across three datacenters: New York (NYC), London (LON), and Sydney (SYD). The cluster is configured so that each shard has at least one replica in each of the datacenters. The NYC shard has a primary in New York and secondaries in London and Sydney, the LON shard has a primary in London and secondaries in New York and Sydney, and the SYD shard has a primary in Sydney and secondaries in New York and London. In this way, each data center has secondaries from all the shards so the local app servers can read the entire data set and a primary for one shard so that writes can be made locally as well.

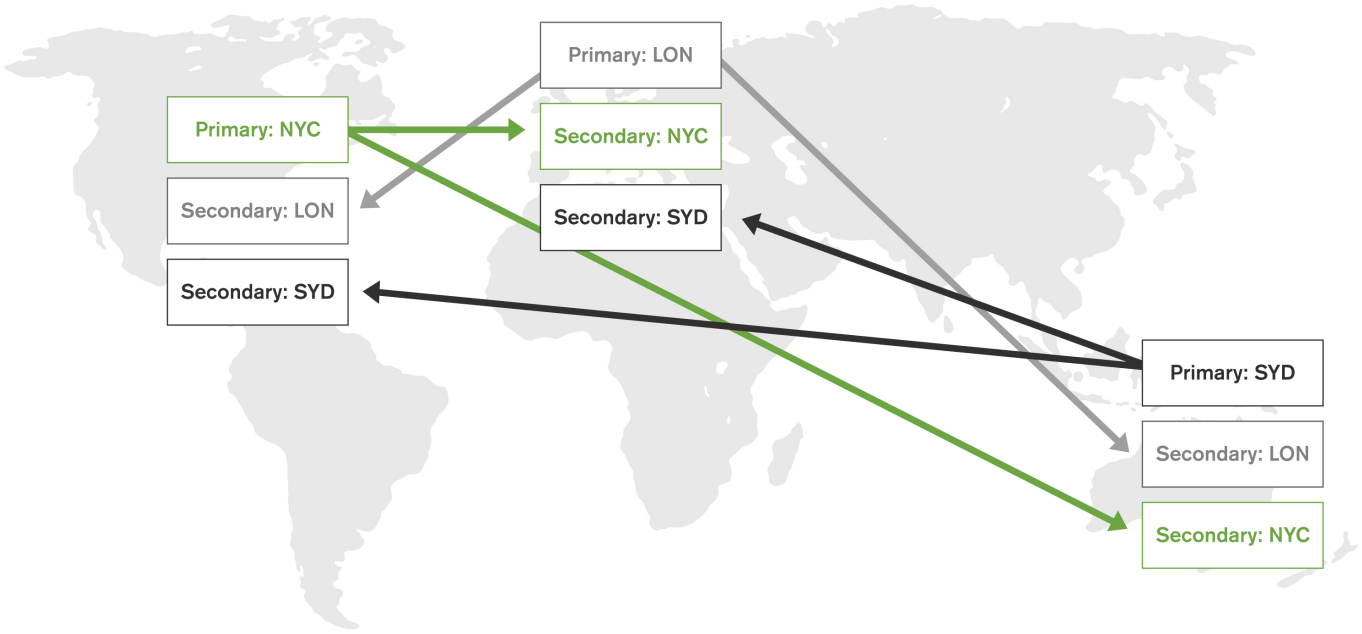


**Figure 6:** Active-active, globally distributed deployment

Learn more by reviewing our tutorial on creating geographically distributed clusters with MongoDB zones:

https://docs.mongodb.com/master/tutorial/sharding-segmenting-data-by-location/).

## Distributed Local Writes for Always-On Insert Only Workloads

MongoDB zones provide a solution for continuous availability of insert-only workloads such as the ingestion of sensor data in IoT applications. Zones can be used to create configurations specifically for localized writes in a distributed cluster, ensuring there is always a node available to accept inserts, even during a data center failure. As demonstrated in Figure 7, the topology for distributed local writes will maintain nodes from both shards in each data center. In case a datacenter is unavailable, application side logic can automatically modify the shard key to redirect the write to the alternative data center.

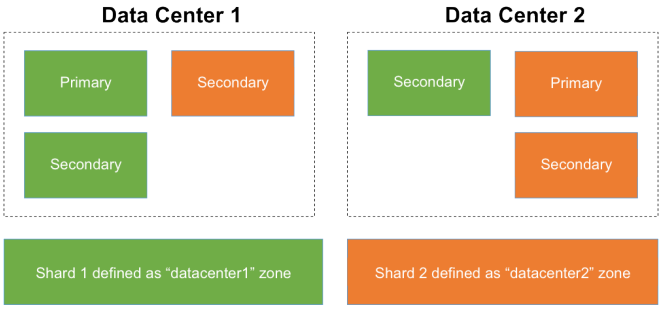


**Figure 7:** Maintaining continuous availability and local writes with MongoDB zones

Learn more by reviewing our tutorial on configuring localized writes with MongoDB zones.

# Managing Multi-Data Center Deployments

Running a database at scale without functional management tools will create unsustainable operational overhead and introduce business risk. These issues are amplified when the database is deployed across multiple data centers. At the same time, organizations want the flexibility to run applications anywhere. MongoDB provides complete platform independence: on-premises, hybrid deployments, or as a fully managed service in the cloud, with the freedom to move between each platform as business requirements change. Users have multiple deployment options for creating globally distributed MongoDB clusters.

## MongoDB Atlas: Database as a Service For MongoDB

MongoDB Atlas is a cloud database service that makes it easy to deploy, operate, and scale MongoDB in the cloud by automating time-consuming administration tasks such as database setup, security implementation, scaling, patching, and more.

MongoDB Atlas is available on-demand through a pay-as-you-go model and billed on an hourly basis.

It's easy to get started – use a simple GUI to select the public cloud provider, region, instance size, and features you need. MongoDB Atlas provides:

- Built in replication for always-on availability. Cross-region replication within a public cloud can be enabled to help tolerate the failure of an entire cloud region – more details follow below.
- Security features to protect your data, with fine-grained access control and end-to-end encryption
- Fully managed, continuous and consistent 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 and customizable alerts for comprehensive performance visibility
- One-click scale up, out, or down on demand. MongoDB Atlas can provision additional storage capacity as needed without manual intervention.
- Automated patching and single-click upgrades for new major versions of the database, enabling you to take advantage of the latest and greatest MongoDB features
- Live migration to move your self-managed MongoDB clusters into the Atlas service with minimal downtime

### Cross-Region Replication of MongoDB Atlas Clusters

When users create a cluster, the Atlas provisioning engine automatically distributes replica set members across the

cloud provider's availability zones within the selected region, thus providing tolerance to the loss of any one data center.

MongoDB Atlas clusters can also span multiple regions offered by a cloud provider. This enables developers to build apps that maintain continuous availability in the event of geographic outages, and improve customer experience by locating data closer to users.

When creating a cluster or modifying its configuration, two options are available:

- Teams can deploy a single MongoDB database across multiple regions supported by a cloud provider for improved availability guarantees. Reads and writes will default to a "preferred region" assuming that there are no active failure or failover conditions. Replica set members in additional regions will participate in the automated election and failover process if the primary member is affected by a local outage, and can become a primary in the unlikely event that the preferred region is offline.
- Read-only replica set members can be deployed in multiple regions, allowing teams to optimize their deployments to achieve reduced read latency for a global audience. As the name suggests, read-only replica set members will not participate in the automated election and failover process, and can never be become a primary.

Teams can activate both of the options outlined above in a single database deployment to more easily provide continuous availability and an optimal experience for their users.

### Next Steps with MongoDB Atlas

MongoDB Atlas can be used for everything from a quick Proof of Concept, to test/QA environments, to powering production applications. The user experience across MongoDB Atlas, Cloud Manager, and Ops Manager is consistent, ensuring that disruption is minimal if you decide to manage MongoDB yourself and migrate to your own infrastructure.

Built and run by the same team that engineers the database, MongoDB Atlas is the best way to run MongoDB in the cloud. Learn more or deploy a free cluster now.

## Managing MongoDB on You Own Infrastructure

Created by the engineers who develop the database, MongoDB Ops Manager is the simplest way to run MongoDB on your own infrastructure, making it easy for operations teams to deploy, monitor, backup and scale MongoDB. Many of the capabilities of Ops Manager are also available in the MongoDB Cloud Manager tool hosted in the cloud. MongoDB Enterprise Advanced customers can choose between Ops Manager and Cloud Manager for their deployments.

Ops Manager and Cloud Manager incorporate best practices to help keep managed databases healthy and optimized. They ensure operational continuity by converting complex manual tasks into reliable, automated procedures with the click of a button.

- **Deploy.** Any topology, at any scale
- **Upgrade.** In minutes, with no downtime
- **Scale.** Add capacity, without taking the application offline
- **Point-in-time, Scheduled Backups.** Restore complete running clusters to any point in time with just a few clicks, because disasters aren't predictable
- **Performance Alerts.** Monitor 100+ system metrics and get custom alerts before the system degrades

## Deployments and Upgrades

Ops Manager coordinates critical operational tasks across the servers in a MongoDB system. It communicates with the infrastructure through agents installed on each server. The servers can reside in the public cloud or a private data center. Ops Manager reliably orchestrates the tasks that administrators have traditionally performed manually – deploying a new cluster, upgrades, creating point-in-time backups, and many other operational activities.

In addition to initial deployment, Ops Manager makes it possible to dynamically resize capacity by adding shards

and replica set members. Other maintenance tasks such as upgrading MongoDB, rolling out new indexes or resizing the oplog can be reduced from dozens or hundreds of manual steps to the click of a button, all with zero downtime.

Administrators can use the Ops Manager interface directly, or invoke the Ops Manager RESTful API from existing enterprise tools, including popular monitoring and orchestration frameworks.

## Monitoring

High-performance distributed systems benefit from comprehensive monitoring. Ops Manager and Cloud Manager have been developed to give administrators the insights needed to ensure smooth operations and a great experience for end users.

Featuring charts, custom dashboards, and automated alerting, Ops Manager tracks 100+ key database and systems health metrics including operations counters, memory and CPU utilization, replication status, open connections, queues and any node status.

The metrics are securely reported to Ops Manager where they are processed, aggregated, and visualized in a browser, letting administrators easily determine the health of MongoDB in real-time. Historic performance can be reviewed in order to create operational baselines and to support capacity planning.

Integration with existing monitoring tools is also straightforward via the Ops Manager RESTful API, and with packaged integrations to leading Application Performance Management (APM) platforms such as New Relic. This integration allows MongoDB status to be consolidated and monitored alongside the rest of your application infrastructure, all from a single pane of glass.

Ops Manager allows administrators to set custom alerts when key metrics are out of range. Alerts can be configured for a range of parameters affecting individual hosts, replica sets, agents and backup. Alerts can be sent via SMS and email or integrated into existing incident management systems such as PagerDuty, Slack, HipChat and others to proactively warn of potential issues, before they escalate to costly outages.

Organizations using Cloud Manager can share access to their real-time monitoring data with MongoDB support engineers, providing fast issue resolution by eliminating the need to ship logs between different teams.

## Disaster Recovery: Backups & Point-in-Time Recovery

A backup and recovery strategy is necessary to protect your mission-critical data against catastrophic failure, such as a fire or flood in a data center, or human error, such as code errors or accidentally dropping collections. With a backup and recovery strategy in place, administrators can restore business operations without data loss, and the organization can meet regulatory and compliance requirements. Taking regular backups offers other advantages, as well. The backups can be used to create new environments for development, staging, or QA without impacting production.

Ops Manager and Cloud Manager backups are maintained continuously, just a few seconds behind the operational system. If the MongoDB cluster experiences a failure, the most recent backup is only moments behind, minimizing exposure to data loss. Ops Manager and Cloud Manager offer point-in-time backup of replica sets and cluster-wide snapshots of sharded clusters. You can restore to precisely the moment you need, quickly and safely. Ops teams can automate their database restores using Ops Manager and Cloud Manager. Complete development, test, and recovery clusters can be built in a few simple clicks. Queryable Backups allow partial restores of selected data, and the

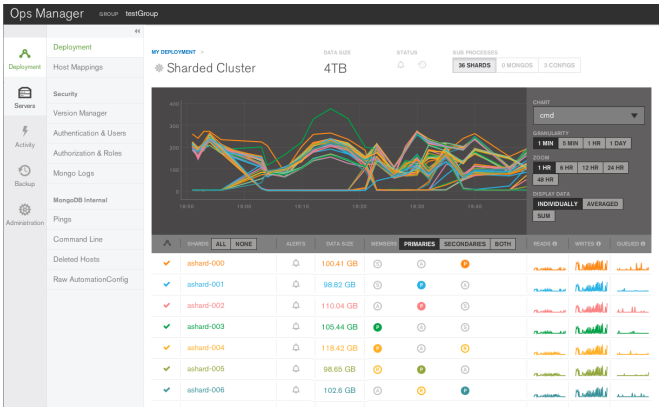


**Figure 8:** Ops Manager provides real time & historic visibility into the MongoDB deployment.

ability to query a backup file in-place, without having to restore it. Now users can query the historical state of the database to track data and schema modifications – often a demand of regulatory reporting. Directly querying backups also enables administrators to identify the best point in time to restore a system by comparing data from multiple snapshots, thereby improving both RTO and RPO.

Because Ops Manager only reads the oplog, the ongoing performance impact is minimal – similar to that of adding an additional replica to a replica set.

By using MongoDB Enterprise Advanced, you can deploy Ops Manager to control backups in your local data center, or use the Cloud Manager service which offers a fully managed backup solution with a pay-as-you-go model. Dedicated MongoDB engineers monitor user backups on a 24x365 basis, alerting operations teams if problems arise.

### Integrating MongoDB with External Monitoring Solutions

In addition to Ops Manager and Cloud Manager, MongoDB Enterprise Advanced can report system information to SNMP traps, supporting centralized data collection and aggregation via external monitoring solutions. Review the documentation to learn more about SNMP integration.

## MongoDB Multi-Data Center Production Deployments

MongoDB is deployed by thousands of organizations around the world, and serves over a half of all Fortune 100 companies. As illustrated below, users from many different industry sectors deploy MongoDB across multiple data centers to scale globally and maintain continuous availability.

### McAfee

McAfee uses MongoDB for its Global Threat Intelligence (GTI) cloud-based intelligence service that correlates data from millions of sensors around the globe. A critical element of McAfee's ability to protect customers from cyberthreats, GTI "connects the dots" between malicious web sites and associated malware, viruses and more, and delivers real-time threat information to McAfee end client products.

McAfee optimizes delivery of content to end users by leveraging MongoDB's GridFS as a homegrown Content Delivery Network (CDN). Analytics and incremental updates are packaged and stored in GridFS, then sent to endpoint security systems. McAfee benefits from high availability since GridFS files are available in all of their data centers across the country without any additional work. Plus, with zones sharding, they can ensure data is geographically close to the systems which use it, making it faster for end users to pull down software updates.

### eBay

As a top 10 global retail brand with close to 200m million active buyers and 1 billion live listings across 190 markets around the world, eBay cannot afford systems downtime. This is why the company relies on MongoDB as one of its core enterprise data platform standards, powering multiple, customer-facing applications that run ebay.com. For example, the company's product catalog is deployed on a 50-node MongoDB replica set distributed across eBay's three North American data centers.

eBay uses MongoDB for a broad variety of mission-critical applications, supporting distributed read and write intensive services. Learn more.

### Tier 1 Bank

The bank estimates savings of $40 million over 5 years following the migration of its reference data management application from a proprietary relational database to MongoDB. Not only did the existing database carry high license costs and expensive hardware requirements, but it was also unable to handle the availability and replication requirements that the business demanded. It would take 24 to 36 hours for the data to replicate across 12 global data centers, which meant international locations were using out-of-date information. This cost the bank a number of fines for failing to meet regulatory requirements. With MongoDB, that data is now replicated globally in minutes.

Through the project, this bank decreased development time, increased availability and realized substantial cost savings. Additionally, by migrating away from a license-heavy software model, the bank was able to shift a major portion of expenses from CapEx to OpEx.

### Under Armour

Under Armour powers its sports clothing and accessories ecommerce catalog with MongoDB. The company wanted to more rapidly update its store in order to meet the shifting demands of its business and branding, and identified the need for more flexible data infrastructure. After evaluating multiple non-relational technologies, Under Armour made the decision to replace their legacy Microsoft SQL Server database with MongoDB.

While a dynamic schema was particularly attractive to the development team, it was MongoDB's native support of multi-data center replication and sharding that addressed Under Armour's disaster recovery and scalability needs as they embarked on global expansion.

### EMEA Multi-National Banking and Financial Services Group

MongoDB powers a derivatives application deployed across a 110-node MongoDB cluster spanning three continents and managed by Cloud Manager.

### YouGov

YouGov is one of the world's leading market research organizations, used by governments, corporations and causes around the globe to track public opinion on a range of issues. YouGov has a global MongoDB cluster comprised of five shards to manage it's survey responses. Two shards are in North America and two in EMEA, and another with replicas spanning both regions. MongoDB enables content to be allocated to specific shards, while providing transparent access through a single interface.

With data distributed by MongoDB's replica sets, YouGov ensures specific data is globally available, but can also be served regionally at low latency. Shards in the MongoDB cluster are configured in a "write-local, read-global" pattern. Users are directed to a primary replica set member in their region, so the database delivers very low latency writes as the user responds to each survey question. That data is then replicated globally for rapid retrieval anywhere in the world. With this architecture, MongoDB presents an abstract, unified, global interface to the data.

## Conclusion

Geographically distributed replica sets, zones sharding and configurable read and write operations enable developers and operations teams to optimize MongoDB deployments for a global audience.

## 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