

Scalability With MongoDB Atlas

Introduction

Scalability is the ability of a database to constantly adjust its resources to meet application demands. As an application grows or traffic increases, the original server resources, such as RAM, CPU, storage, and I/O, might not suffice. This is when you will need to scale your database. You can scale temporarily if you expect a sudden burst of traffic (e.g., for a new ad campaign) or more permanently when you see a steady increase in usage.

MongoDB has a full range of vertical, horizontal, and elastic scaling options available, and they are built into [MongoDB Atlas](#), MongoDB's multi-cloud developer data platform. In this document, we will discuss these scaling options.

[MongoDB's documentation](#) provides more detailed information on specific topics. We have provided links throughout this document to help guide you to other valuable resources.

Understanding Vertical and Horizontal Scaling

Vertical scaling

Vertical scaling (or *scale-up*) is the action of increasing the processing power and capacity of a single server (or VM, or instance) by adding more CPU, RAM, or disk space. Vertical scaling has been available in the majority of general-purpose databases for decades.

In many cases, vertical scaling is very useful and easy to set up, but eventually you will reach limits in maximum processing power and throughput, or going to the next machine size will be prohibitively expensive. The limitations of available technology may restrict a single machine from being sufficiently powerful for a given database workload. Additionally, cloud-based providers have hard ceilings based on available hardware configurations.

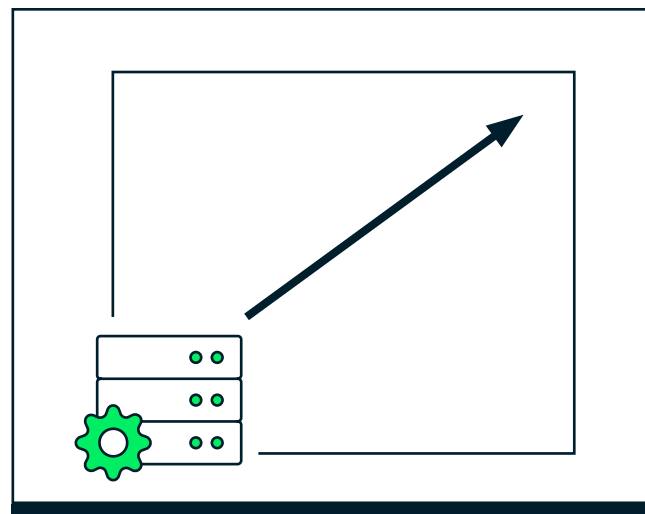


Figure 1: Vertical scaling

Horizontal scaling

Horizontal scaling (or *scale-out*) involves dividing your application data and workload over multiple servers and adding servers to increase capacity. Each machine handles a subset of the overall workload, giving you unlimited scalability. Expanding the capacity of the deployment only requires adding additional servers as needed, which gives you cost linearity compared to vertical scaling.

MongoDB natively supports horizontal scaling through a method called *sharding*, which we will describe later in this document.



Figure 2: Horizontal scaling

Horizontal vs. vertical scaling: which one to choose?

There are several factors to consider when deciding whether to scale vertically vs. horizontally, including read/write throughput, IOPS, RAM, and disk limitations.

Other basic considerations when deciding whether you should scale vertically or horizontally include:

- **Workload permanence:** If you only need to scale temporarily, vertical scaling is likely all you will need. This way, you can quickly scale back down afterward and save some money.
- **Current architecture:** If you use a more traditional relational database, you might need some significant investments in additional software before horizontal scaling. In contrast, if you use a solution that natively supports horizontal scaling, the process of sharding will be easier.
- **Pricing:** As you scale vertically, you will inevitably hit a point where adding more resources becomes prohibitively expensive. At that point, it may be cheaper to start scaling horizontally.

Scaling Proactively Vs. Reactively

Proactive scaling refers to scaling your database in advance of known load or high-traffic events that will occur in the future. This could be based upon a regular pattern (e.g., day of the week or certain times of the year), or it could be done before specific events, such as launching a marketing campaign.

In contrast, reactive scaling refers to scaling in response to application metrics. These could be

warning signs, such as throughput and query response times, or they could be alerts coming from your database monitoring. In the worst-case scenario, this could be in response to an outage caused by the excess load.

Naturally, when possible, proactive scaling is preferable. MongoDB Atlas makes scaling as easy as setting the right configuration.

MongoDB Atlas Vertical Scaling with Auto-Scaling

Vertical scaling in Atlas is as simple as configuring a [cluster tier](#) or leveraging the power of [cluster auto-scaling](#), both described below.

Atlas cluster sizing and tier selection

Choosing the correct Atlas [cluster tier](#) and configuration is an important step in setting up a successful MongoDB production deployment. You can always modify a cluster at a later time, but getting started with the right tier is possible with a few calculations based on your data size, working set, and network requirements.

You can also configure your cluster to automatically scale its cluster tier, storage capacity, or both in response to cluster usage, thereby reducing the manual maintenance required for your cluster. This is done with [zero downtime](#).

Cluster Tier

M30 (8 GB RAM, 40 GB Storage) 
3,000 IOPS, Encrypted, Auto-expand Storage

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

Dedicated Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
M10	2 GB	10 GB	2 vCPUs	from \$0.08/hr
M20	4 GB	20 GB	2 vCPUs	from \$0.20/hr

Dedicated Clusters for high-traffic applications and large datasets

- Additional hardware configurations available for specialized workloads










Tier	RAM	Storage	vCPU	Base Price
 M30	8 GB	40 GB	2 vCPUs	from \$0.54/hr
Class	General			
Storage	40 GB is included in the base price			
	10 GB  512 GB			<input type="text" value="40"/> GB
Auto-scale	<input checked="" type="checkbox"/> Cluster Tier Scaling View docs			
	Minimum cluster size <input type="text" value="M10"/>		Maximum cluster size <input type="text" value="M40"/>	
	<input checked="" type="checkbox"/> Allow cluster to be scaled down			
	<input checked="" type="checkbox"/> Storage Scaling 			
IOPS	<input type="checkbox"/> Provision IOPS 			
	3000 IOPS			
Additional Info	3000 max connections Up to 10 Gigabit network performance			
M40 	16 GB	80 GB	4 vCPUs	from \$1.04/hr
M50 	32 GB	160 GB	8 vCPUs	from \$2.00/hr
M60 	64 GB	320 GB	16 vCPUs	from \$3.95/hr
M80 	128 GB	750 GB	32 vCPUs	from \$7.30/hr
M140	192 GB	1000 GB	48 vCPUs	from \$10.99/hr
M200 	256 GB	1500 GB	64 vCPUs	from \$14.59/hr

Figure 3: MongoDB Atlas cluster tier

Atlas cluster auto-scaling

MongoDB cluster tiers M10 and greater support [cluster auto-scaling](#). Cluster auto-scaling is an intelligent and fully automated capacity management service in MongoDB Atlas. It allows your cluster resources to auto-scale in response to workload changes. Atlas monitors key metrics in real-time and adjusts cluster compute and storage based on predictive modeling and proven practices from managing tens of thousands of MongoDB deployments.

Cluster auto-scaling removes the need to write complex scripts or use consulting services to make scaling decisions. You can easily specify a range of minimum and maximum cluster sizes in the API or Atlas user interface (UI) to handle your workload's variability. You can also maintain visibility into auto-scaling events with automated alerts and detailed activity feed updates.

Cluster auto-scaling offers other significant benefits. With intelligent automation, key metrics are tracked against thresholds in real-time, and auto-scaling events are applied in a rolling fashion to ensure no downtime to your cluster. Automation is based on heuristics from managing tens of thousands of MongoDB deployments. Flexible capacity management allows users to set upper and/or lower cluster tier limits for cluster tier auto-scaling and to save on costs for periods of reduced workload with automated cluster tier downscaling. And cluster auto-scaling offers real-time visibility and control. Users receive real-time alerts when cluster tier auto-scaling events occur. They can review all cluster tier auto-scaling events in the Atlas activity feed, and they maintain the ability to manually scale and make other cluster configuration changes.

Cluster auto-scaling is enabled by default when you create new clusters in the Atlas UI, but not when you create new clusters via the [Atlas Admin API](#). With auto-scaling enabled, Atlas automatically scales your cluster tier, storage capacity, or both in response to cluster usage. Auto-scaling allows your cluster to adapt to your current workload and reduces the need for manual intervention.

Cluster tier scaling automatically scales your cluster tier up or down in response to various cluster metrics. To opt-out of cluster tier auto-scaling, uncheck the "Cluster Tier Scaling" box in the Auto-scale section.

To control how Atlas should auto-scale your cluster, you set:

- The maximum cluster tier to which your cluster can automatically scale up. By default, this setting is set to the next higher cluster tier.
- The minimum cluster tier to which your cluster can scale down. By default, this setting is set to the current cluster tier.

Auto-scaling reacts quickly but not instantly; if you expect your application to have a spiky workload, it's a good idea to set some extra headroom into the minimum cluster tier to accommodate temporary increases in usage.

Storage scaling automatically increases your cluster storage capacity when 90% of disk capacity is used. This setting is enabled by default to help ensure that your cluster can always support sudden influxes of data. To opt-out of cluster storage scaling, uncheck the Storage Scaling box in the Auto-scale section.



Dedicated Clusters for high-traffic applications and large datasets

- Additional hardware configurations available for specialized workloads

Tier	RAM	Storage	vCPU	Base Price
M30	8 GB	40 GB	2 vCPUs	from \$0.54/hr
<input checked="" type="checkbox"/> M40	16 GB	80 GB	2 vCPUs	from \$0.77/hr

Class Low-CPU General Local NVMe SSD

Storage *80 GB is included in the base price*

10 GB 1 TB 23 GB

Auto-scale **Cluster Tier Scaling** [View docs](#)

Minimum cluster size R40 Maximum cluster size R50

Allow cluster to be scaled down

Storage Scaling ⓘ

Figure 4: MongoDB Atlas cluster auto-scaling

Proactive scaling

Atlas auto-scales by default, but you can also control cluster scaling directly through the UI, API, or integrations like Kubernetes, Terraform, and CloudFormation. For example, if you expect a change in workload characteristics – e.g., seasonal variation, product launch, or usage growth due to a new marketing campaign – it’s

simple to scale your cluster in advance. Scaling a live cluster in Atlas is as easy as setting it up in the first place. By editing the configuration in the Atlas UI and selecting the new specifications, you kick off the scaling event. Atlas applies the new configuration, scaling up each node in a rolling fashion so there is zero downtime.

Scheduled scaling

If you have regular workload variations, you can configure Atlas to [scale on a schedule](#) using Atlas Scheduled Triggers. For example, if an application sees the most usage during known business hours, you could scale down a cluster for cost savings on weekends or overnight, and scale it back up before the load returns.

Scheduled scaling is a good alternative to (or companion to) auto-scaling when workloads

follow known patterns. Auto-scaling responds to fluctuations in workload automatically but is more conservative about scaling down in order to avoid false positives (e.g., scaling down during a transient lull, only to have to scale back up quickly). When you are confident that workload will decrease for a given time period, scheduled scaling can reduce the size of your cluster more quickly than auto-scaling will.

MongoDB Atlas Horizontal Scaling With Sharding

As mentioned above, MongoDB Atlas provides horizontal scaling for databases using [sharding](#). Sharding is fully transparent to your application: The client only has to send queries to the cluster, and MongoDB automatically routes them appropriately. MongoDB distributes data across multiple [replica sets](#) called shards. Sharding allows MongoDB deployments to scale

beyond the limitations of a single server, such as bottlenecks in RAM or disk I/O, without adding complexity to the application. With automatic balancing, MongoDB ensures data is always equally distributed across shards as data volume increases or decreases.

Sharding is only available for Atlas clusters with tiers M30 and larger.



Sharded cluster

A MongoDB sharded cluster consists of the following components:

- **Shards:** Each shard contains a subset of the data and is deployed as a replica set.
- **Mongos:** The mongos acts as a query router, providing an interface between client applications and the sharded cluster.

- **Config servers:** Config servers store metadata and configuration settings for the cluster.

The graphic below describes the interaction of components within a sharded cluster.

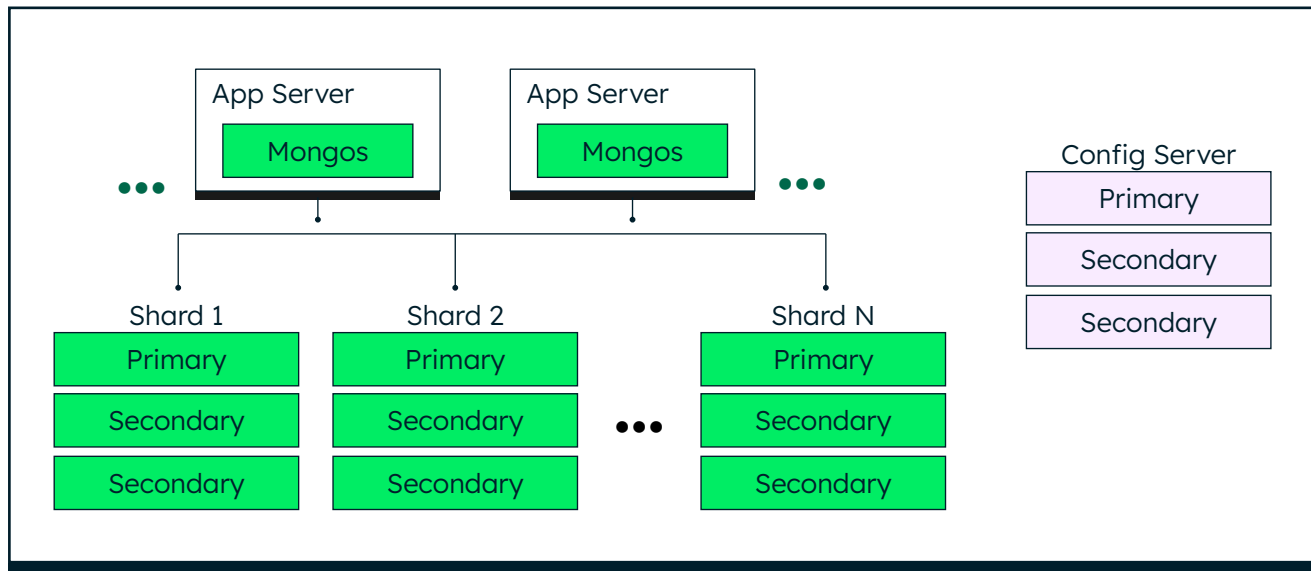


Figure 5: Sharding architecture

Atlas automates the deployment and management of a sharded cluster and all its components. After the user configures the desired number of shards, Atlas automatically sets them up and deploys the mongos and config servers.

Users should consider deploying a sharded cluster in the following situations:

- **RAM limitation:** The size of the system's active working set (frequently accessed documents plus indexes) is expected to exceed the capacity of the maximum amount of RAM available in your maximum Atlas tier.
- **Disk I/O limitation:** The system will have a large amount of write activity, and the operating system will not be able to write data fast enough to meet demand, or I/O bandwidth will limit how fast the writes can be flushed to disk.

- **Storage limitation:** The dataset will grow to exceed the storage capacity of a single replica set. Atlas has a 4 TB (compressed) storage limit for a replica set.
- **Location-aware requirements:** The dataset needs to be assigned to a specific data center to support low-latency local reads and writes.

Applications that meet these criteria or are likely to do so in the future should be sharded in advance rather than waiting until they have consumed available capacity. When designing their data models, applications that will eventually benefit from sharding should consider which [collections](#) they will need to shard and the corresponding [shard keys](#).

Sharding strategies

MongoDB supports three types of sharding strategies, enabling administrators to accommodate diverse query patterns:

- **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.
- **Hash-Based Sharding:** Documents are uniformly distributed according to 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 uniform distribution of writes across shards – provided that the shard key has high cardinality – making it optimal for certain write-intensive workloads.

- **Zone-Based Sharding:** MongoDB Atlas zones allow developers to define specific rules governing data placement in a sharded cluster. You can easily create distributed databases to support geographically distributed apps, with policies enforcing data locality within specific cloud regions. Each zone can have one or more shards. Zones are exposed in Atlas through Atlas Global Clusters, discussed below.

The importance of selecting a good shard key

MongoDB sharding has always been highly flexible – you can shard on any document field or combination of fields. This means you can select a *shard key* that is best suited to your application’s requirements.

The choice of shard key is important, as it defines how data is distributed across the shards. Ideally, you want to select a shard key that:

- Gives you low-latency and high-throughput reads and writes by matching data distribution to your application’s data access patterns.

- Evenly distributes data across the cluster so you avoid any one shard taking too much load (e.g., a “hot shard”).
- Provides linear scalability as you add more shards in the future.

Not only do you have the flexibility to select any field(s) of your documents as your shard key, but it’s also possible to seamlessly *change* the shard key (reshard) on a live cluster. If you chose a shard key that didn’t work well, or if your application’s requirements changed such that the original shard key was no longer the best one, the impact on performance would be significant.



Live resharding: horizontal scaling without fear or friction

Prior to MongoDB 5.0, resharding was a manual process that required either downtime or custom logic. Starting with 5.0, MongoDB made complex manual resharding a thing of the past with live resharding. You just need to run the

`reshardCollection` command from the shell, point at the database and collection you want to reshard, specify the new shard key, and let MongoDB take care of the rest.

```
reshardCollection: "<database>.<collection>", key: <shardkey>
```

When you invoke the `reshardCollection` command, MongoDB clones your existing collection into a new collection with the new shard key, then starts applying all new `oplog` updates from the existing collection to the new collection. This enables the database to keep pace with incoming application writes. When all `oplog` updates have been applied, MongoDB will automatically cut over to the new collection and remove the old collection in the background.

Let's walk through an example where live resharding would really help a user:

- The user has an orders collection. In the past, they needed to scale out and choose the `order_id` field as the shard key.

- They realize that they have to regularly query each customer's orders to quickly display order history. This query does not use the `order_id` field. To return the results for such a query, all shards need to provide data for the query. This is called a scatter-gather query.
- It would have been more performant and scalable to have orders for each customer localized to a shard, avoiding scatter-gather, cross-shard queries.
- They realize that the optimal shard key would be "customer_id: 1, order_id: 1" rather than just the `order_id`.
- With live resharding, the user can just run the reshard command, and MongoDB will reshard the orders collection for them using the new shard key, without having to bring down the database and the application.

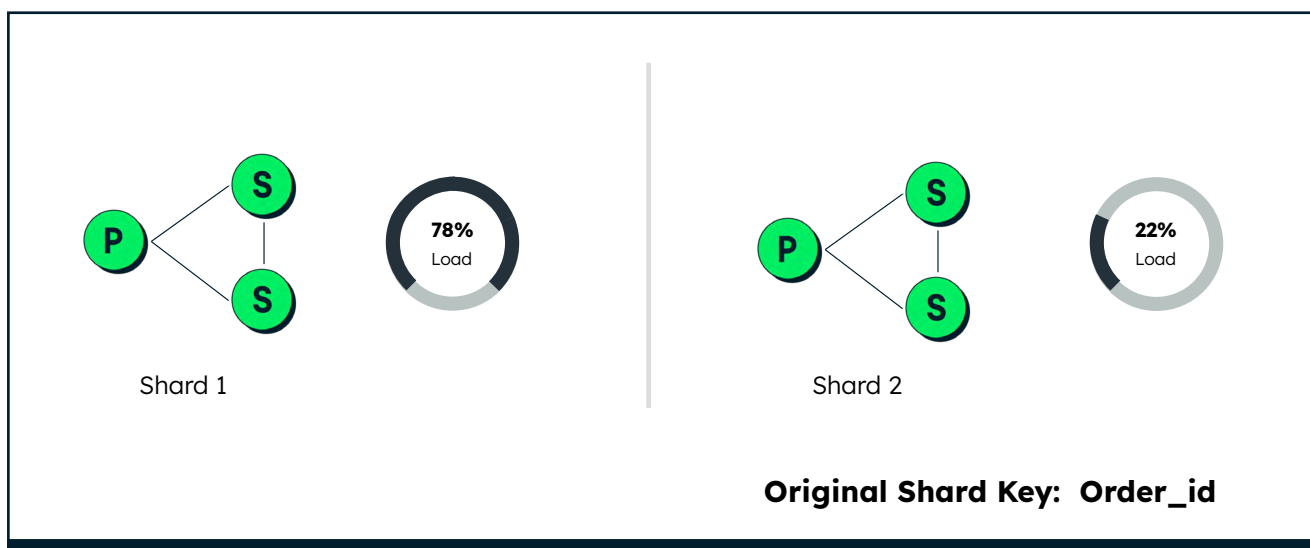


Figure 6: Before live resharding



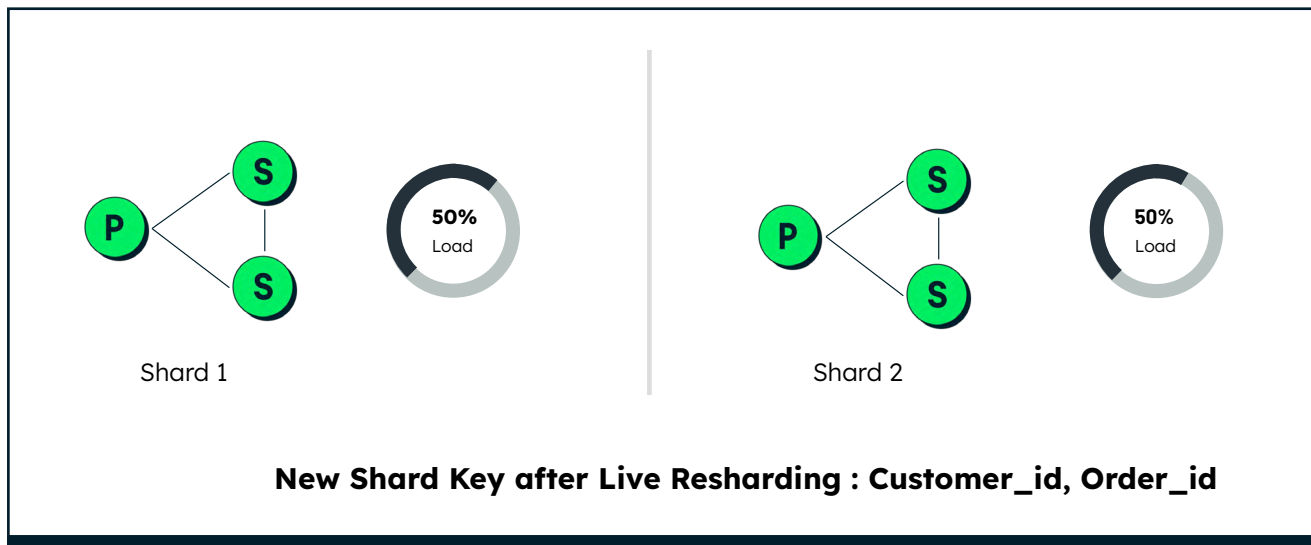


Figure 7: After live resharding

With live resharding, not only can you change the field(s) for a shard key, you can also review your sharding strategy, changing between range-based or hash-based sharding, or modifying your geographic distribution strategy with zones.

At this point in time, no other mainstream distributed database provides a capability like live resharding.

Deploying a sharded cluster in MongoDB Atlas

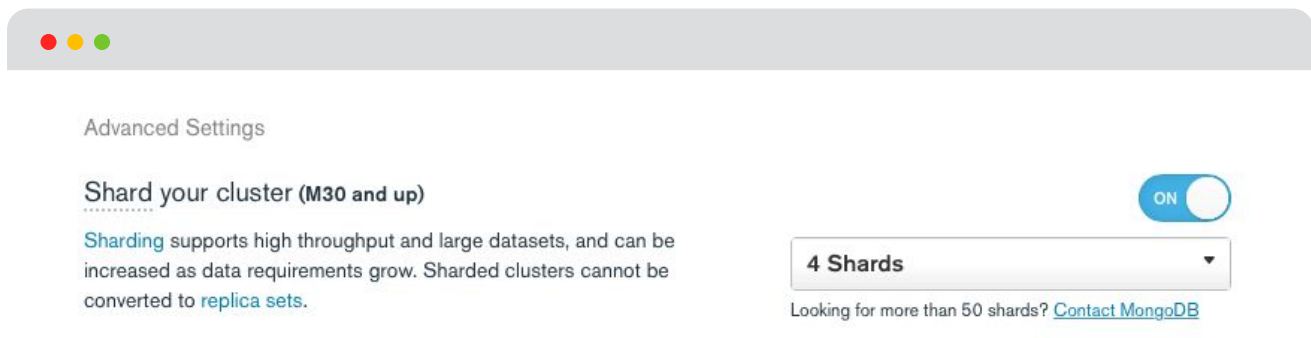


Figure 8: Deploying a sharded cluster in MongoDB Atlas

Deploying a sharded cluster in Atlas is as simple as a setting in the cluster configuration. When you turn on sharding and set the number of shards, Atlas automates the setup of the sharded cluster infrastructure.

There is also, however, a more advanced deployment option that distributes shards in different cloud regions around the world: Global Clusters.



Global sharding with MongoDB global clusters

Today's users expect high-performing, responsive applications regardless of where they are located, but distributing your data globally can be challenging. MongoDB Atlas [global clusters](#) remove the complexity of globally distributing your database to keep relevant data close to end-users for regulatory compliance and low-latency reads and writes.

MongoDB Atlas global clusters simplify data residency requirements by automating the process of isolating data that's subject to regulations within specified geographic boundaries, so you can confidently and easily meet local privacy and compliance needs in any region.

Global clusters use a highly curated implementation of MongoDB zones that allows you to place

database instances capable of accepting reads and writes in distinct, geographically distributed zones made up of one or more cloud regions. You can do this on AWS, Google Cloud, or Azure, or on a multi-cloud cluster, and it is all on-demand and fully managed by Atlas. Global clusters enable you to read and write your data locally, providing single-digit millisecond latency for distributed applications and allowing you to have strong controls over where the data lives.

As your application and user base grow, global clusters allow you to seamlessly adjust coverage with support for more than 80 cloud regions, making it simple to add or remove zones at any time. You can quickly deploy using prebuilt zone templates or build your own custom zones by choosing cloud regions in an easy-to-use, visual interface.

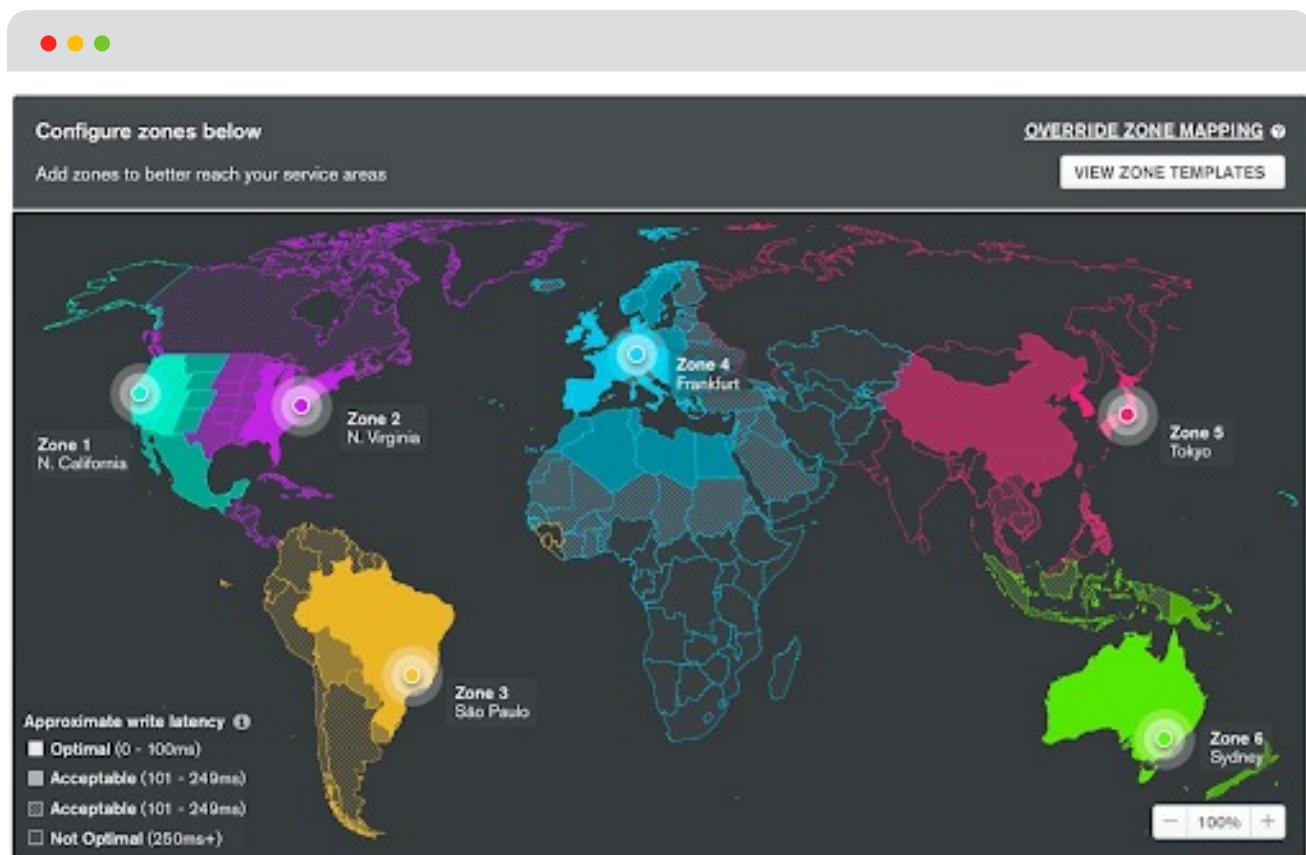


Figure 9: Serving always-on, globally distributed, write-everywhere apps with MongoDB Atlas global clusters

MongoDB Atlas Elastic Scaling With Serverless Instances

[Serverless computing](#) is quickly gaining popularity among developers as a preferred way of building and running modern applications. Cloud functions are commonly used to power business logic in applications, and many teams rely on completely automated IT operations.

The appeal of serverless technology is hard to deny: Elastic scaling eliminates the need for upfront resource provisioning and ongoing maintenance, and consumption-based pricing means paying only for resources that are used. It abstracts and automates away many of the lower-level infrastructure controls that developers don't want to have to learn or manage; instead, they can focus on building differentiated features.

When it comes to databases, compute and storage resources have traditionally been tightly coupled. Applying a serverless model

to databases means decoupling them and changing the way engineering teams think about infrastructure. Rather than asking a developer to predict an application's future workload patterns, break them down into individual resource requirements, and then map them to arbitrary units of database instance sizes, serverless databases offer a much simpler experience: Define where your data lives, and get a database endpoint you can use.

This not only streamlines the database deployment process, it also eliminates the need to monitor and adjust capacity on an ongoing basis. Developers are free to focus on thinking about their data rather than their databases, and leave the lower-level infrastructure decisions to intelligent, behind-the-scenes automation.

Understanding serverless databases

Serverless databases share many of the same characteristics as serverless application platforms:

- **Elastic scaling:** The ability to automatically scale a database based on workload, including the ability to scale down to zero compute resources when there is no workload. Unlike serverless application platforms, which only manage compute, serverless database platforms have both a compute layer and a storage layer that can scale elastically.

- **Consumption-based pricing:** A pricing model that only charges for the data stored in the database and the resources used to service the database workload.

These attributes confer advantages similar to the serverless application model:

- You don't need to think about scaling up to meet increasing workloads or storage needs.
- You don't need to worry about paying for database resources you are not using.

MongoDB Atlas serverless instances

[Serverless instances](#) provide you with the flexible, expressive power of MongoDB with the ease of the serverless model, so you can get an on-demand database endpoint for your application without having to think about resource provisioning. You simply choose a cloud region to get started, then start building. MongoDB Atlas serverless instances

will seamlessly provide the database resources your application needs at any given time and come with the same built-in security defaults as Atlas, keeping your data secure and available. This removes the need to manually scale up and down between different cluster tiers.



Configuration, automation, and pricing

- Serverless instances require minimal configuration: All you need to do is choose a cloud provider and region to create a serverless instance on MongoDB Atlas.
- Serverless instances scale automatically to meet your workload: With serverless instances, MongoDB Atlas will instantly provide the database resources your application needs at any given time, with the ability to fully scale down to zero and back up when needed without cold starts, allowing you to better address sudden spikes in usage or longer idle periods when the database is not needed.
- Serverless instances are billed on usage: Our consumption-based, pay-as-you-grow model only charges for the operations you run (reads, writes, data transfer, and storage), so you never pay for excess resources. Daily tiering of read operations even provides you with automatic discounts as your usage scales without any up-front commitments required.

You can view and manage serverless instances using the same UI and API as your existing database clusters. They also support the latest MongoDB capabilities, so you never have to worry about backward compatibility or upgrades.

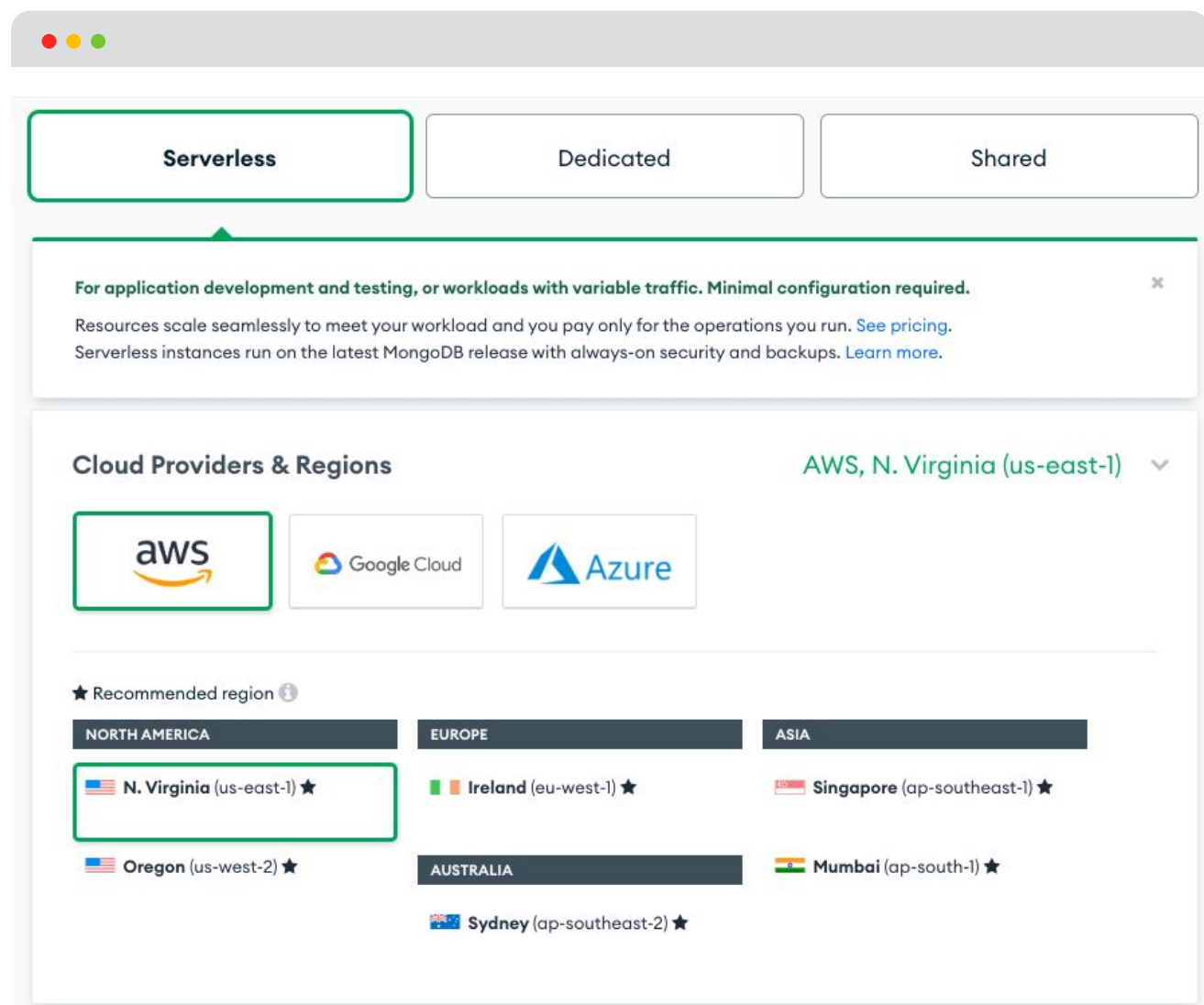


Figure 10: Creating a serverless instance



Conclusion

Scalability is a key requirement of modern, distributed applications and database systems. MongoDB Atlas has been built for scalability from the ground up, with innovative options to automatically scale vertically, horizontally, and elastically. Users have the option of using Atlas's several scaling options, including auto-scaling,

sharding, and serverless instances. MongoDB Atlas also makes it easy to combine scalability with global data distribution using Global Clusters.

You can test Atlas's scalability by deploying a [free MongoDB Atlas cluster](#)

MongoDB Scalability Resources

For a deeper dive into scalability with MongoDB, sign up for the free, on-demand [M103 Basic Cluster Administration](#) MongoDB University course. MongoDB can also provide [instructor-led training for operations](#) professionals and teams, covering a broad set of skills to manage mission-critical Atlas environments.

MongoDB [Professional Services](#) connect your team with our consulting engineers to support a variety of operations challenges, from deployment architecture and preparing for production to optimizing performance and costs on a live deployment.

About MongoDB

MongoDB's developer data platform allows developers to move fast and simplify how they build with data for any application. The platform provides developers with a unified interface to serve transactional, search, real-time, and data lake application needs. The result: less time spent rationalizing data infrastructures, and more focus on applications that users love and their organizations need.

We are the MongoDB experts. More than 35,000 customers rely on us to drive their businesses. We offer software and services to make your life easier. For more information, please visit [mongodb.com](#) or [contact us](#)

Learn More

[Documentation](#)

[Resources](#)

[Customer success stories](#)

[Webinars and events](#)