ObjectRocket
a rackspace company

MongoDB Jetpack

What’s Inside

Introduction 2
What is MongoDB? 3
Top 4 Use Cases for MongoDB 5
When to Scale MongoDB Instances 8
How to Properly Scale and Maintain MongoDB 11
Intro to Massive Scaling with MongoDB 18
Upgrading MongoDB Instances 20
Why Choose ObjectRocket? 24
About ObjectRocket 25

Intro

It’s no secret that NoSQL databases have gained tremendous ground over relational databases in the last decade. A primary driver for this is the huge amount of data generated today from sources we never would have dreamed of two decades ago. Cell phones, mobile applications, and social media alone created the need to handle massive amounts of data from many different sources and with varying structures. This data works best with “schemaless” databases, like MongoDB.

Read on to learn more about what MongoDB is and how businesses are using DBaaS for MongoDB to get ahead of the competition.

What is MongoDB?

Introduced in 2009, MongoDB is the top NoSQL document database. A document store database is a database that stores data as documents. It is used for various workloads by a wide range of companies including Google, Facebook, TheChive, UPS, Braze, Cisco, Electronic Arts, and many more.

According to DB-Engines ranking, there are now 45 document datastores out there.

MongoDB has been leading that pack of NoSQL document store databases for most of the last 4 years.

It's currently the

#1 MOST WANTED DATABASE

according to a recent Stack Overflow survey.[1]

Read on to learn about why you should use MongoDB

ObjectRocket

Why Use MongoDB?

MongoDB and similar NoSQL databases allow for faster, more flexible development cycles that help lower software and deployment costs. Additionally, developers love the schemaless aspect where they are not forced into more rigid structures like they are with relational databases. MongoDB meets the need for flexible data models that serve unstructured, semi-structured, and structured data types.

There are several additional reasons to use MongoDB instead of other document store DBs:

- MongoDB meets the need for flexible and scalable architectures that are built from the ground up as fault tolerant clusters with redundancy. It provides this high availability via multi-member replica sets that use a PRIMARY/SECONDARY relationship where writes and reads go to the PRIMARY by default.
- MongoDB with its native BSON format and JSON extensions supports many more programming languages (27) versus some of its competitors like Cassandra (13) and Couchbase (14).
- MongoDB provides robust indexing capabilities including secondary indexes and index intersection to cover a wide variety of queries.
- MongoDB includes a powerful Aggregation Pipeline that allows for data transformations along the way.
- MongoDB has implemented retryable writes and change streams to be ACID compliant for Multi-Dcoument Transactions.

[1]Stack Overflow Developer Survey Results 2018. Accessed 8/4/18. https://insights.stackoverflow.com/survey/2018/

Top 4 Use Cases for MongoDB

Let’s review some of the use cases our clients face, and we’ll explore how MongoDB addresses each one.

1 Customer Analytics

Data aggregation is one of the keys to creating amazing customer experiences. Companies are collecting massive amounts of data about their existing and potential customers and aggregating it with publicly available data. From all of this disparate data, companies can build customer profiles and nurture paths with the goal of getting the customer to buy more products.

With all of this data coming from different sources with different schemas, tying it all together at such a massive scale is a huge challenge. The flexibility and scalability of MongoDB provides a solution. MongoDB allows for the aggregation of this data and building analytical tools in order to create amazing customer experiences. MongoDB's speed allows for dynamic experiences that can evolve based upon the customer behavior in real time.

2 Product Catalog

Product catalogs are not new to the evolving digital experience. What is new is the volume and richness of the data that feeds the interactions in product catalogs that we use today. MongoDB provides a great tool to store many different types of objects with different sets of attributes. MongoDB's dynamic schema capability allows for product documents to only contain attributes that are relevant to that product. Gone are the days of needing every product record to contain every possible attribute. MongoDB users can very quickly and easily make changes to their catalogs, providing a better experience for developers and customers.

3 Real Time Data Integration

Companies have vast amounts of data spread across their organization. Data provides value if it's aggregated in one "single view". Previously, energy and resources were spent on data ingestion, transformation, and schema changes in order to obtain a single source of data. MongoDB's flexibility and query capabilities make it easy to aggregate this data and create the tools that make organizations more efficient. This aggregation can be achieved to provide a "single view" of their data in real time. With the addition of change streams in MongoDB 3.6, developers can now monitor and take action on specific events quickly.

4 Mobility and Scaling

With most mobile application development, companies are dealing with varying data structures coming from multiple sources and potentially highly dynamic growth. The flexibility and scalability of MongoDB provides a great database solution for dealing with this type of environment. With schemas that can evolve over time, mobile application developers don't have to spend time adjusting the database. Instead, developers can focus on developing the customer experience.

Today, modern businesses are thinking about better ways to store and manage their data, gain better customer insights, adapt to changing user expectations, and beat competitors to market with new applications.

Other Use Cases

Specific Use Cases for MongoDB

- Mobile and mobile analytics
- Catalog data and inventory
- Content management
- Social media user data
- Game user data
- Internet of Things (IoT)
- Augmented reality (AR), Virtual reality (VR), and MR related applications
- Core and off-chain datastores that support blockchain initiatives

Good General Uses for MongoDB

- Highly unstructured data (no need for a rigid, defined schema)
- Lots of data
- High write activity
- Balancing writes and reads/ queries
- Desire for duplicate or denormalization

MongoDB is a great tool that many companies find useful, but managing MongoDB doesn't fit into everyone's business model. It's hard to find the right expertise and many companies can't afford to hire the headcount.

ObjectRocket for MongoDB can help.
No matter where your app is hosted, we can help you get the most from your data.

When to Scale MongoDB Instances

Scaling tends to be reactive in nature leading to issues such as degraded application performance or, even worse, total application downtime. These issues ultimately lead to a negative customer experiences that impact your business. So how do you know when it’s the right time to scale your Mongo database so you don’t impact your bottom line?

Find the right threshold for your app

To ensure your application weathers the storm of growth in application usage and surges in traffic, it is imperative to **load test** a system to determine the threshold that your application can handle before you apply changes like scaling to your production system. More often than not, this process is overlooked and results in fire drills when you encounter issues with an increase in traffic.

- The road to a successful MongoDB load test should include the following considerations:
- Which metrics need to be monitored so you can find bottlenecks?
- What threshold should be used in terms of when to scale?
- Which tools should be used to analyze the metrics?
- Where should you run the load test?

Which metrics should I monitor to find bottlenecks?

First, you'll need to determine which criteria to use to find the limiting factors to your particular application. Every application is different. Knowing these requirements will allow you to determine the appropriate metrics to monitor from the database side.

Here are some examples:

- Your application requires X number of inserts per second when there is an increase in traffic.
- Data needs to return to the user in less than 100 milliseconds.
- There is a limit in the number of concurrent connections.
- There is a limit in server resources, such as high CPU or load average.

Once you are able to determine the application-side metrics, you can decide what to monitor on the database side.

For example, if your requirements are to have a certain number of insert requests, focus on write metrics such as:

- Server I/O
- Database locking
- opcounters, in terms of active writes
- Available write tickets, if you are running WiredTiger as your storage engine

Tools used to analyze the metrics

Once you determine which metrics to monitor, you need to gather the data from the database side. Here are some utilities utilizing you can use to analyze the load factor:

- mongostat utility shows real-time database metrics on locking, read, and write queues.
- db.currentOp() utility determines the current active operations.
- mongotop allows you to view the top active collections.
- Server utilities such as top, sar, iostat show metrics such as CPU utilization or disk throughput.

In addition to real time analysis, you can also mine the MongoDB logs for additional details about database activity as well as any error conditions. By default, MongoDB logs queries that take longer than 100ms to execute so it is a good tool to identify poor performing queries.

Determine where to run the load test

Once you determine what you will use to gather the data, you are ready to load test. Common sense dictates that you should never load test in a production environment. Sometimes this becomes a show stopper because it can be difficult to spin up an environment fast enough. If you are not already on the ObjectRocket platform, you can spin up a MongoDB test cluster very quickly.

Need some help?

Knowing when to scale your MongoDB environment can get complicated. ObjectRocket customers enjoy the best scaling and sharding support hands-down.

Contact us today to get started.

How to Properly Scale and Maintain MongoDB

The ease of installing and running MongoDB can lead to neglect of basic database management. Then one day you're getting frantic calls from a client or coworker about an application that's crashed. Your current MongoDB setup couldn't handle the increased usage and simply gave up.

Not accounting for growth and increased data needs is a failure of scaling. Here are some things you can do to make sure your MongoDB setup can handle your workload at any point.

Apply good DevOps practices

The convenience of MongoDB doesn't free you from the responsibility of applying a good DevOps strategy. Your database still needs a sound schema, careful monitoring of data loads, a good indexing strategy, and enough hardware resources to support large data clusters. Poor design or implementation of any of these factors can lead to failure.

Large enterprises looking for a more feature-filled version of MongoDB should know that ObjectRocket offers a Percona solution to support enterprise-level capabilities like enhanced security and added agility for tracking and managing your database.

Keep an eye on everything

Are you prepared for any usage ramp-up for applications your MongoDB instances support? You'll suddenly find yourself at the end of your capacity if you're not keeping track of the way your clusters are handling different workload levels.

Keep an eye out for replication lag in particular. Your MongoDB needs to handle system capacity at times when your oplog window is at its shortest. Too much lag could cause data to be overwritten and not be recoverable, or cause problems with syncing up your old and new data.

If you don't have the time or resources to keep a close eye on your MongoDB instances, a fully hosted and managed MongoDB service may be a great fit for you.

Leave the internals alone

There should be no reason for you to go in and tamper with your internal databases or the system collections. You should be restricting yourself to using shell helpers and administrative commands for any needed performance operations.

Any issues with functionality are most likely tied to problems with design, installation, or maintenance. The path forward should NOT be trying to force the core structures to do what you want.

Scale wisely

Work with your MongoDB management company to define which scaling strategy is right for your workload. There are a couple of different methods for scaling MongoDB that we'll cover in-depth in the next chapter.

Have a security strategy

Attacks in recent years on MongoDB instances makes it doubly important that your database have some type of security protocols in place. This means doing more than running processes on different ports. MongoDB provides you with five different ways to secure your database. Failing to do so leaves you vulnerable to a ransomware attack that cripples your processes.

Call in an outside company experienced with establishing security parameters within MongoDB if you're unfamiliar with how to properly set them up. They can walk you through how each of the different security areas works and help you find the right configurations. Contact ObjectRocket to talk about MongoDB security.

ObjectRocket can help you set up a new MongoDB instance or help you configure your current one for the support your applications need.

Intro to Massive Scaling with MongoDB

With databases, scaling refers to having the ability to expand to meet additional needs around storage/disk, RAM/memory, CPUs/compute cycles, networking, or other resources.

Two Approaches to Scaling

Scaling proactively

When you proactively plan to scale, there are at least two general patterns:

- You know that you have a big marketing push coming up where you think that you will be adding a significant number of customers and/or amount of data.
- Your application or business tend to be cyclical in nature (e.g., Christmas buying, New Years' resolutions, etc.) where there will be a lot of activity or events that you will want to capture and keep a high volume of data.

Scaling reactively

Are you seeing small warning signs? When you start hitting bottlenecks and you expect to continue growth, you *already* need to be thinking about scaling.

Trouble signs to watch for include things like...

- increased query times for end users
- increased login times
- requests and servers freezing up
- the dreaded "database slow" cries from developers
- slower server response times
- increased load on hosts
- out-of-memory errors
- unintended elections
- errors in the logs

When you start seeing these signs, it's time to start scaling so you can keep up with demand and make sure you aren't losing customers.

There are two ways to scale: Up (vertically) or out (horizontally).

Scaling vertically

This is the proverbial Big Iron method: One big machine with lots of resources (CPU cores, higher CPU speeds, lots of RAM, storage).

The main benefits of vertical scaling include reduced architectural complexity and fewer hosts to maintain. This is helpful if you don't have anyone that can handle the maintenance for you.

Today, there are many ways to scale up vertically. There are better options for commodity hardware, cheaper disks and storage, better storage options, cheaper memory, better software, and networking so you can more gracefully handle failovers and interruptions.

Scaling up works well for many applications and needs. For those we would recommend the Replica Sets discussed below. One thing to keep in mind with using larger replica sets is that there can sometimes be hidden costs to scaling vertically. If your environment continues to grow rapidly, you may have to constantly be moving to larger and larger machines or have additional resources added to your hosts until you reach a point where that is no longer an option. You should also consider that upgrade cycles are less efficient on a single larger host versus a horizontally scaled environment. With continued growth, you would have to decide whether to continue to scale up or if you feel that you may benefit from scaling horizontally.

Scaling horizontally

Sharding is horizontal scaling. Sharding stores data across multiple nodes, distributing the load and the processes across the hosts. Replication is handled via Master-Slave with the ability to add additional nodes as needed.

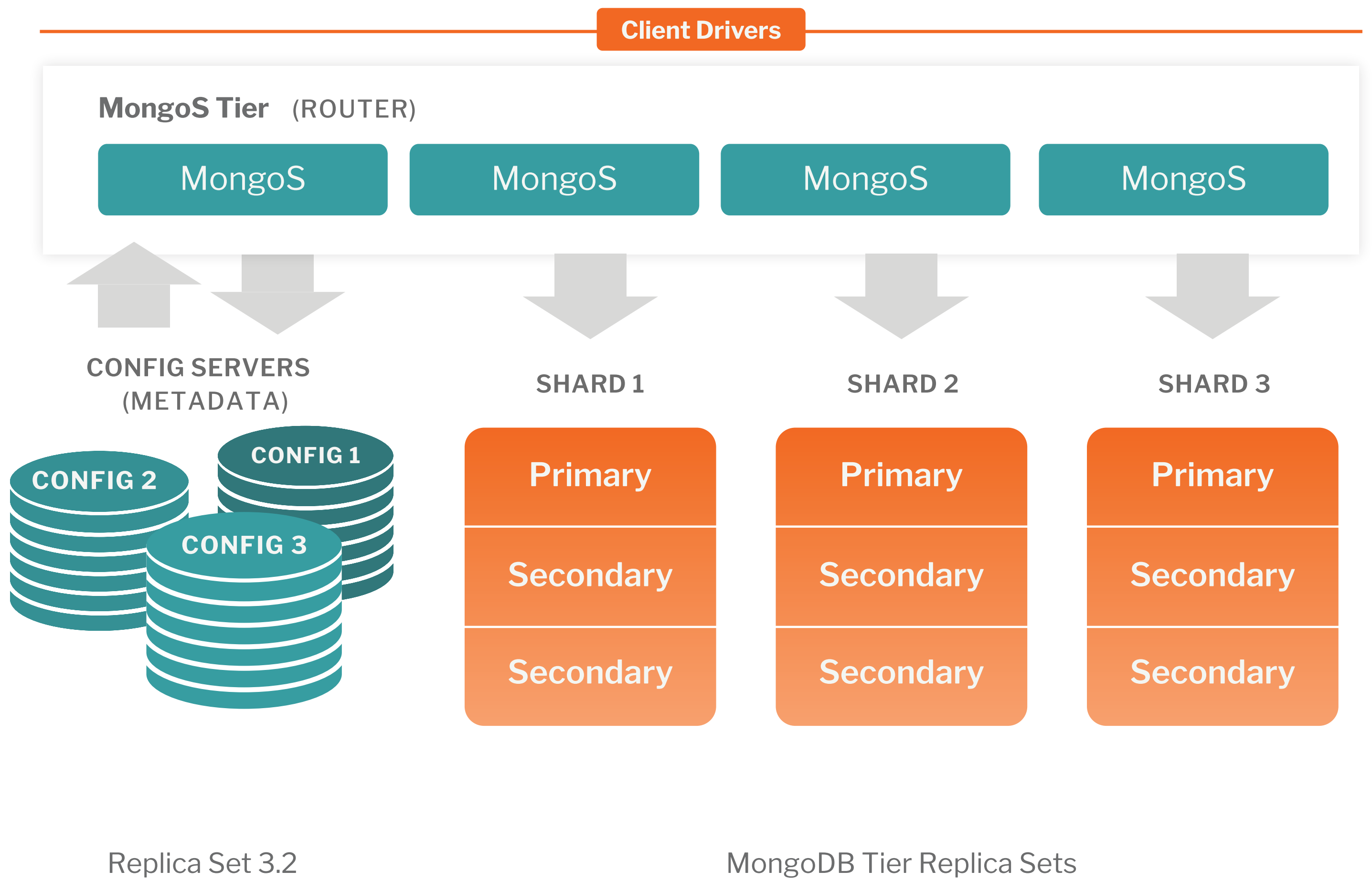
The 'chunks' of data are distributed by the Balancer across the disks on the nodes.

This increases read and write capacity by distributing read and write operations across a group of machines, instead of hammering one machine with writes or with reads. Luckily, there have been great improvements in balancer function over the last few releases.

Scaling horizontally takes advantage of MongoDB's built in sharding ability and also benefits from the ability to use cheaper commodity hardware.

When you scale horizontally, you add additional resources with physical or virtual hosts.

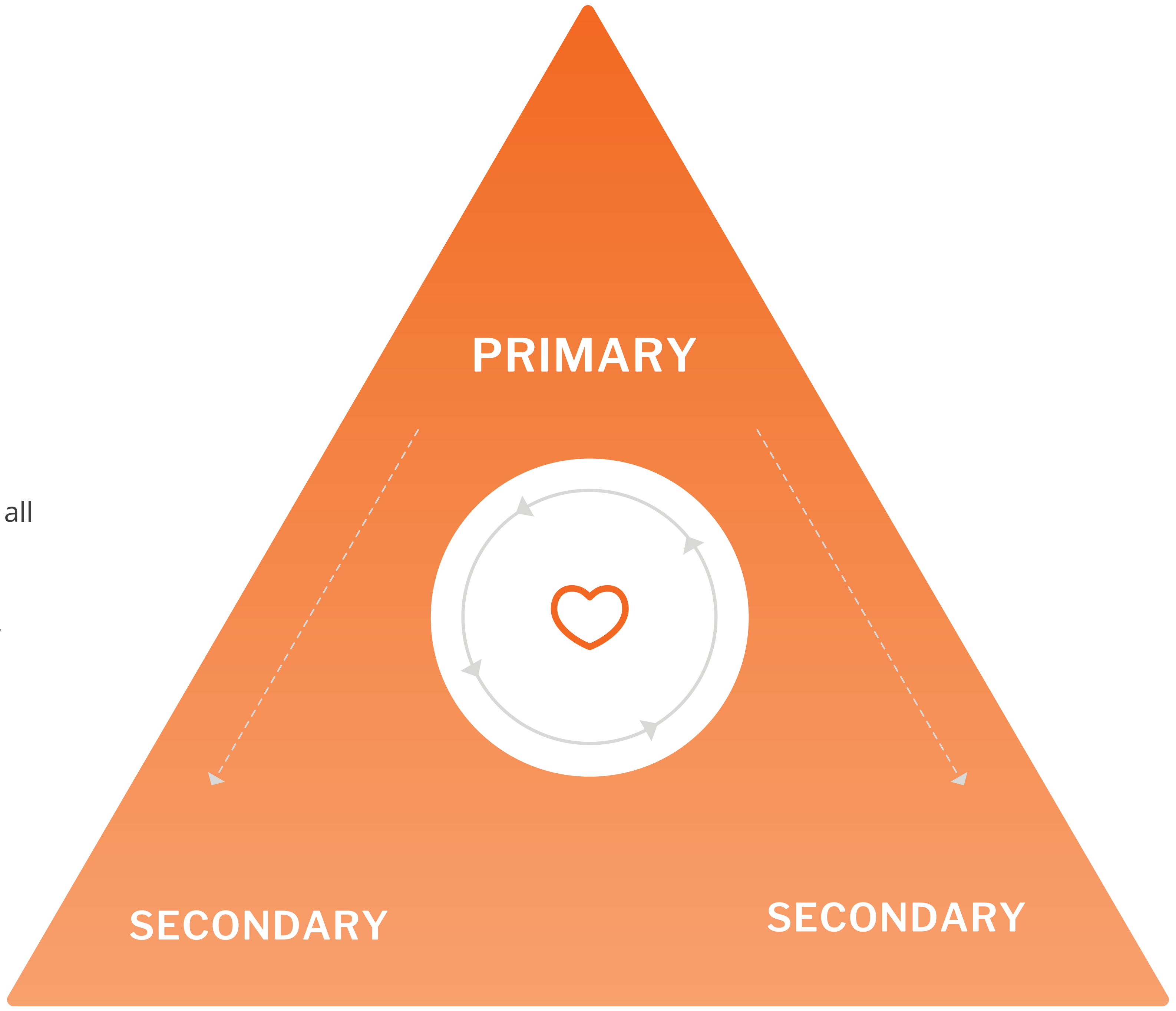
- Physical – lots of lower cost commodity hardware
- Virtual – add additional CPU cores or nodes via VMs or cloud
- Networking – add load balancers, additional mongoS processes, etc.
- Utilizing improved load balancing technologies (hardware and software) to shuttle traffic to where it needs to go via load balancers, etc.



What is a replica set?

A replica set in MongoDB is a group of processes that maintain the same data set. Replica sets provide redundancy and high availability, and are the basis for all production deployments.

The secondaries replicate the primary's oplog and apply the operations to their data sets.



Replica sets or shards?

The trade-off you make in sharding comes with some increased overall complexity. But sharding also provides the benefit of simplifying maintenance by allowing for rolling upgrades and the ability to perform certain operations such as index builds in parallel at the same time across your shard/nodes.

Here are some comparisons between using larger replica sets vs. sharded clusters:

Replica Set	Sharding
Simple	Expertise needed
Lots of Reads across a wide data set (Don't want to scatter gathers)	Lots of Writes/Updates (Want to go directly to exact shards for results)
Lots of data, lower activity rates	Lots of data, lots of activity
Need more "normal" resources – ex. just disks	Need more of all resources – Disks, RAM, CPU, write scopes

ObjectRocket

WHY MANAGED MONGODB FROM OBJECTROCKET?

The ObjectRocket Difference, in a word, expertise.

1 We have been managing MongoDB from the get-go.

2 We support larger replica sets and sharded clusters.

3 Get the best proactive hands-on support, hands-down.
The right response 24 × 7 × 365

LEARN MORE

"ObjectRocket has done so many things over the yearsto help us scale from an environment that was only around 10 instances and about 150 shards to where we are today.

Jon Hyman
CTO and Cofounder, Braze
Read This Case Study

Upgrading MongoDB Instances

Upgrading from an older version of MongoDB can be painful. We get it. Your app is working, you don't have the time or resources to upgrade (even though you know you need to). It stays at the bottom of your backlog because new features and enhancement to your application are always more critical to the business. You keep putting it off, knowing that you are getting farther and farther behind.

We talk to many IT and DevOps leaders that find themselves in this situation. They use MongoDB 2.4 or 2.6 and they've fallen really far behind. And they feel stuck.

Read on to learn about the path forward.

Reap the benefits of upgraded MongoDB instances

Each release brings new features, major bug fixes, and performance enhancements. When evaluating an upgrade to a newer version, you may have some features you'd like to add and there may be some nagging bugs you'd like to squash. We have upgraded thousands of instances over the years and we've learned a lot.

Here are some reasons it can be painful to upgrade your MongoDB instances:

Downtime

Both minor and major version upgrades can usually be completed without any downtime, if your driver is set up to be resilient during a step down and mongos restart. Timelines for upgrades depend on many factors, such as which version you're currently running, your applications' needs, and which features will offer the most benefit for your app. Most customers can expect a 30-minute maintenance window with a 15 second period of downtime. Contact support for an estimate based on your set up.

Rewriting code

It's not uncommon to have to change your code in order to upgrade. Each new major version may or may not deprecate query operators and tools. If you're using a lot of features, there's a good chance some of your queries will need to changed. Features may also be shelved but in most cases there is a replacement feature or a workaround that can be implemented.

Updating drivers

Since you are using an older version of MongoDB, your driver might already be out of date. New drivers offer new features and can improve your application and user experience. You might have to update your driver in advance. Sometimes, this can be painful because a driver upgrade involves dependencies so other components/ frameworks might need an upgrade as well.

For all the reasons listed above, we always recommend testing a new major version before upgrading.

Upgrade considerations

Here are some things we look at when figuring out how long an upgrade to an instance will take:

- **Code language**
- **Driver version**
- **Code compatibility, like deprecated operators**
- **MongoDB version constraints** (dataset compatibility checks)

Why upgrading is important

There are several reasons to keep upgrading MongoDB. Deprecated versions won't get backported when comes to bugs. Each version involves hundreds of bugs. For example, there have been more than 150 bug fixes related to the database engine and more than 280 other enhancements and improvements for version 3.4 alone. If you're still not convinced, check out the new features/improvements section for select improvements.

New features/improvements

MongoDB version 3.0

- Introduction of MMAPv1 – MMAP storage engine
- Introduction of WiredTiger

MongoDB version 3.2

- General improvements for the WiredTiger storage engine.
- Read Concern provides isolation level for reads.
- Partial Indexes provides more flexible indexes and requires less storage while covering the queries better. Read our blog: MongoDB Partial Indexes – Is It Time To Rethink Your Indexing Strategy?
- Document validation provides the capability to validate documents during updates and insertions.
- Aggregation runs faster on sharded clusters
- Geo Indexes v3 for faster Geo Lookups

MongoDB version 3.4

- General improvements for the WiredTiger storage engine.
- Linearizable read concern – Critical to financial/ bank applications for read accuracy.
- Faster balancing – Important for scaling faster. Move more chunks in parallel.
- Improved initial sync – Makes weekly compaction to run faster.
- Decimal support – Makes working with decimals much easier. No workarounds necessary.
- Aggregation stage for recursive search – Important for commerce applications. Allows faster search.
- Views – Adds another layer of security. Read our blog: Enhance Your Organization's Security with MongoDB Views
- Improved performance and security. In 3.4, there have been more than 150 bug fixes related to the database engine and more than 280 other enhancements and improvements.

MongoDB version 3.6

You can get the full list of what is available in the MongoDB 3.6 release notes. Here are some of the updates and changes we wanted to highlight:

Change streams
Tailing MongoDB's oplog, is a popular feature with many uses, such as real-time notifications of all the changes to your database.

WiredTiger
MongoDB 3.6 comes with WiredTiger storage engine version 3.0.

Client sessions
Client sessions created for applications that require causal consistency are now better supported.

Retryable writes
Retryable writes allow MongoDB drivers to automatically retry certain write operations a single time if they encounter network errors, or if they can't find a healthy primary in the replica sets or sharded cluster.

Bug fixes

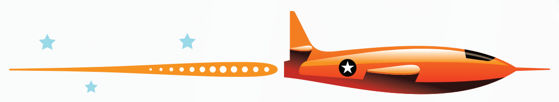
Aggregation enhancement
The $lookup operator adds support for specifying multiple join conditions as well as uncorrelated subqueries by allowing variable specification and pipeline execution on the joined collection.

JSON schema operator ($jsonSchema) enhancement
This operator matches documents that validate against the given JSON schema.

Wire protocol and compression
The new wire protocol opcode called OP_MSG allows network compression for communication among mongod and mongos drivers.

Want to spin up a MongoDB 3.6 instance today?

Head to the app and get started.



Why choose ObjectRocket?

So, you want to get your payload of data into orbit without crashing your ROI?
Here are some of the top reasons why we think you should meet us on the ObjectRocket launchpad.

✓ Open Source Innovators

We're a leader in open source database management and are well known for our deep knowledge of NoSQL databases (especially MongoDB, Elasticsearch, and Redis).

✓ Polyglot Persistence

Using aDBaaS that offers multiple types of databases is critical. That way, youcan use the right database for your use case, saving time and money.ObjectRocket manages several types of open source databases so thatyou can use one vendor for all your needs.

✓ Fast and Secure

We know how to get the most out of host machines to power demanding workloads. Our platform provides high security while performing millions of operations per second.

✓ We Grow With You

ObjectRocket was built on the core premise of enabling simple and reliable scalability for all of our databases. RocketScale™ is an ObjectRocket technology that automatically adds data nodes as you need them.

✓ Cost Conscious

In June 2017, Crimson Consulting Group released an analysis showing that contracting with a fully managed DBaaS is a much better value than managing databases in-house. Crimson noted that ObjectRocket lowers data management costs by "orders of magnitude." That's the kind of language scientists use to describe the distance between stars.

Contact us today so we can help you travel light years ahead of your competition.

SCHEDULE A CONSULTATION

About ObjectRocket

ObjectRocket's technology and expertise helps businesses build better apps, faster so developers can concentrate on creating applications and features without having to worry about managing databases. We'll migrate your data at no cost and with little-to-no downtime. Our DBAs do all the heavy lifting for you so you can focus on your builds. We provide 24x7x365 expert support and architecture services for MongoDB, Elasticsearch, Redis, and Hadoop instances in data centers across the globe.