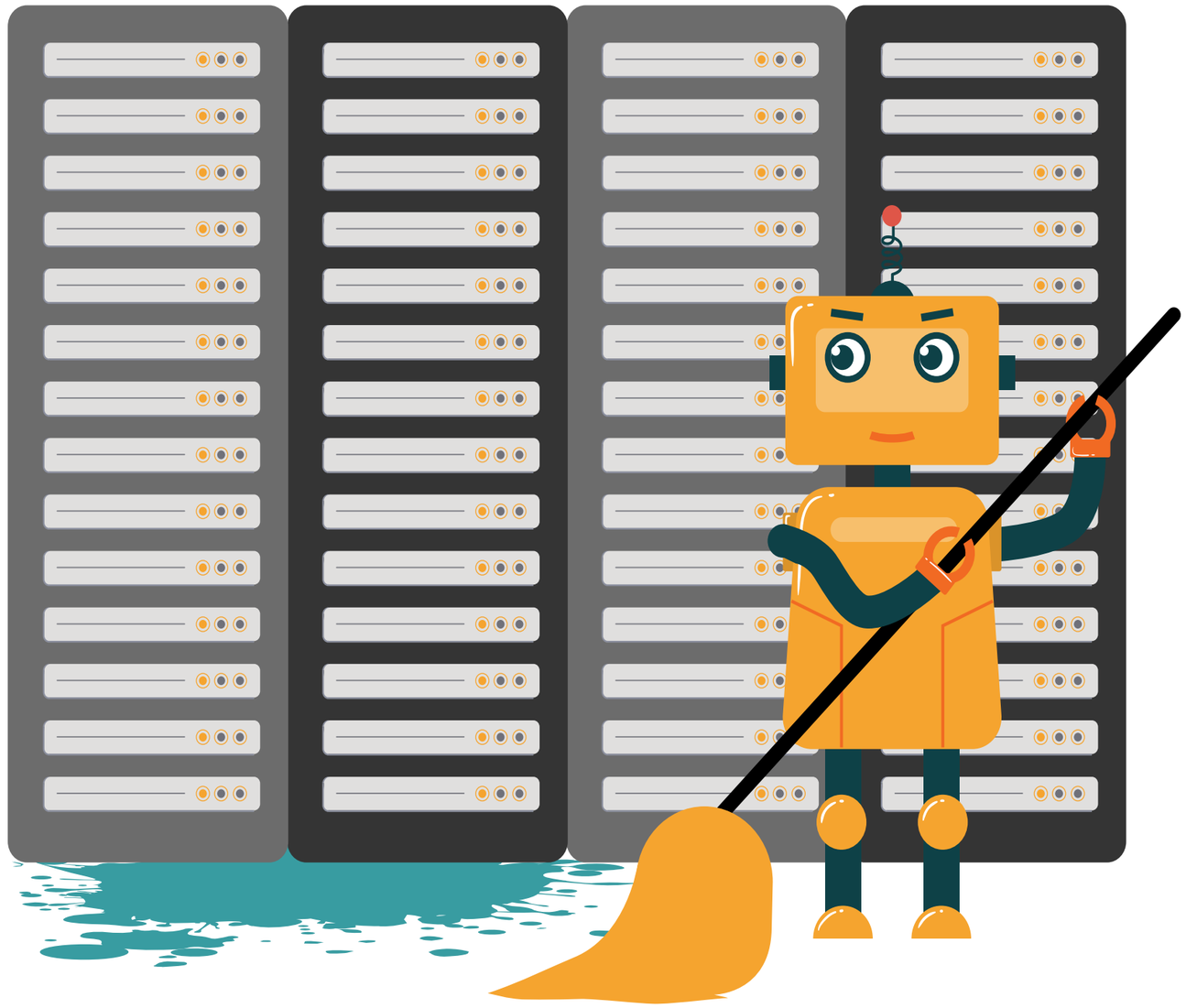
ObjectRocket
a rackspace company

Dirty Jobs: Database Edition



What's Inside

Intro

After 6+ years of supporting thousands of production instances, ObjectRocket knows quite a bit about managing databases. We automate many common tasks but we really shine when it comes to scaling and troubleshooting our customers' NoSQL databases.

Our team has seen just about everything. Zombie servers coming back to life, developers creating infinite loops, customers deleting the wrong collection, etc. We learn from our customers' and our own mistakes and keep making our database services that much better. We have a lot of knowledge to share and a lot of tips and tricks that greatly improve performance and save our customers time and money.

Here are some of the common database management tasks ObjectRocket performs so our customers don't have to.

#1 Data Migrations

#2 New Deployments

#3 Scaling and Sharding

#4 Upgrades

#5 Compactions

#6 Parsing Logs

#7 Draining Shards

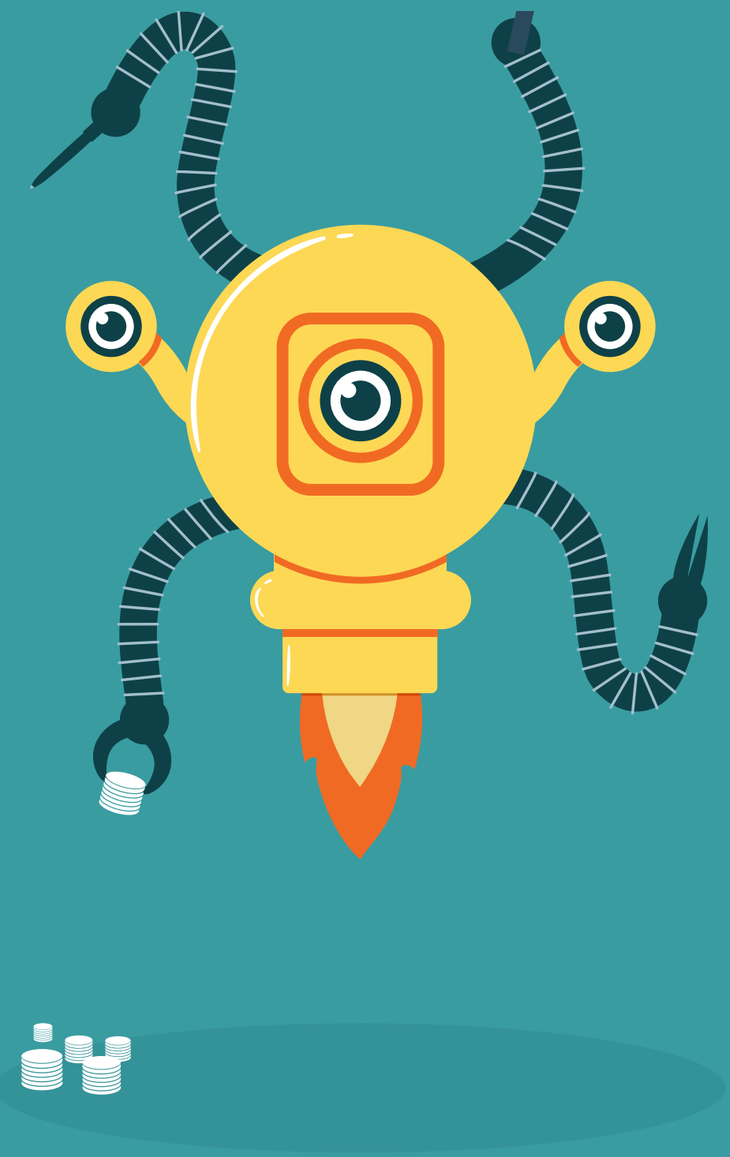
#8 Monitoring

#9 Query Analysis and Index Optimization

#10 Health Reports

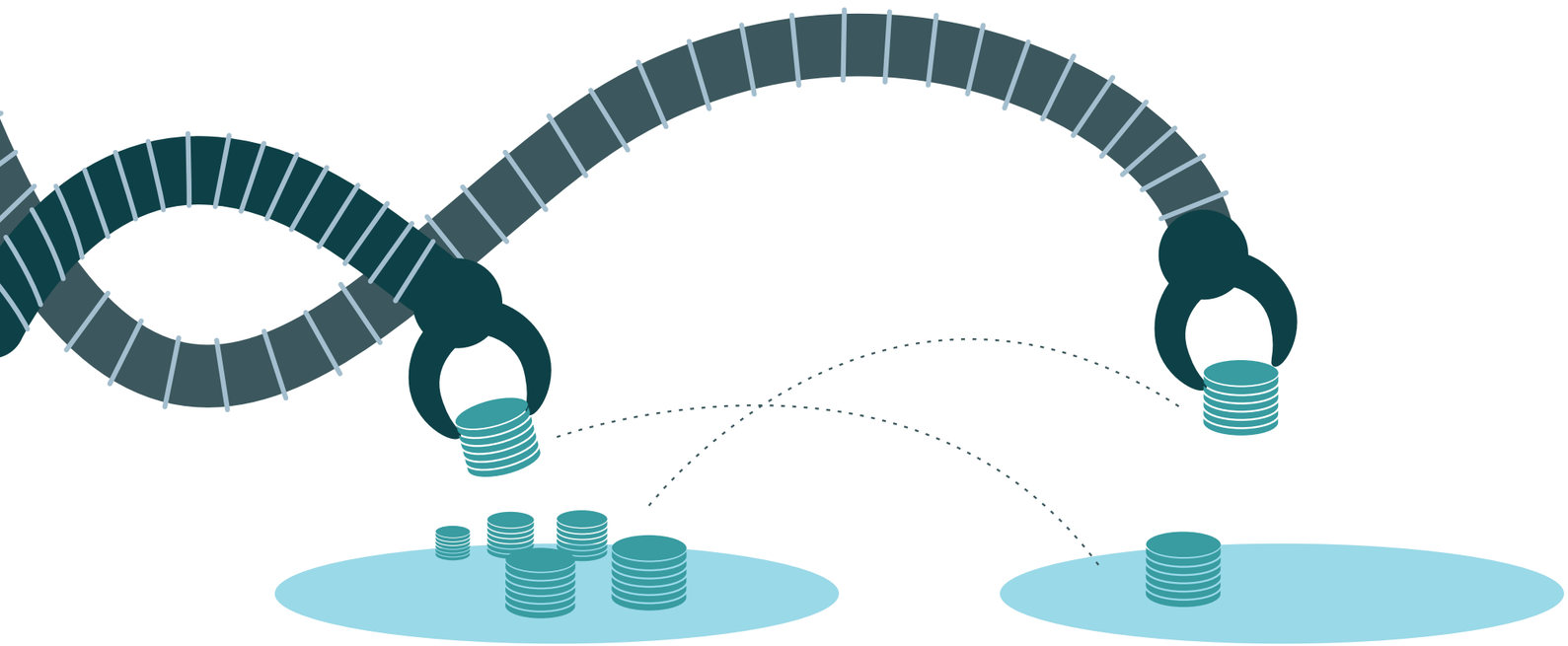
Dirty Job #1
Data Migrations

Migrating data is rarely easy. There's a lot at stake, especially when migrating production data. When it comes to out-of-the-box migration tools — you are all alone. With ObjectRocket, data migration is easy.



Every ObjectRocket plan includes complimentary migration services by our Data Migration Team. Our team is experienced in migrating workloads across a variety of technologies, providers, and architectures.

Our Data Migration Team is composed of technical account managers, database architects, and customer data engineers. They work closely with your team to help identify and assist you with the best path forward for getting your data up and running. Whether it's a simple dump and restore, or a live cut-over requiring minimal downtime, we'll work together to get the right plan in place for your workload. Whether your organization has a simple database or a large, complex application with distributed databases, we make it easy to migrate to the ObjectRocket platform by working with you one-on-one.



ObjectRocket's data migration service includes:

- Preparing connectivity from your current environment to ours
- *For MongoDB*: Adding replica members on ObjectRocket to your existing instance
- *For Elasticsearch*: Either loading from a backup or a "reindex from remote", where the new cluster connects to the old cluster and reads data into the new database.
- *For Redis*: Either loading from a backup or migration via replication.
- Migrating your existing configuration files to our infrastructure
- One-on-one guidance as you update your application
- A short maintenance window and clean data migration
- Assurance that your application has been appropriately migrated
- Removing old replica members, if no longer needed

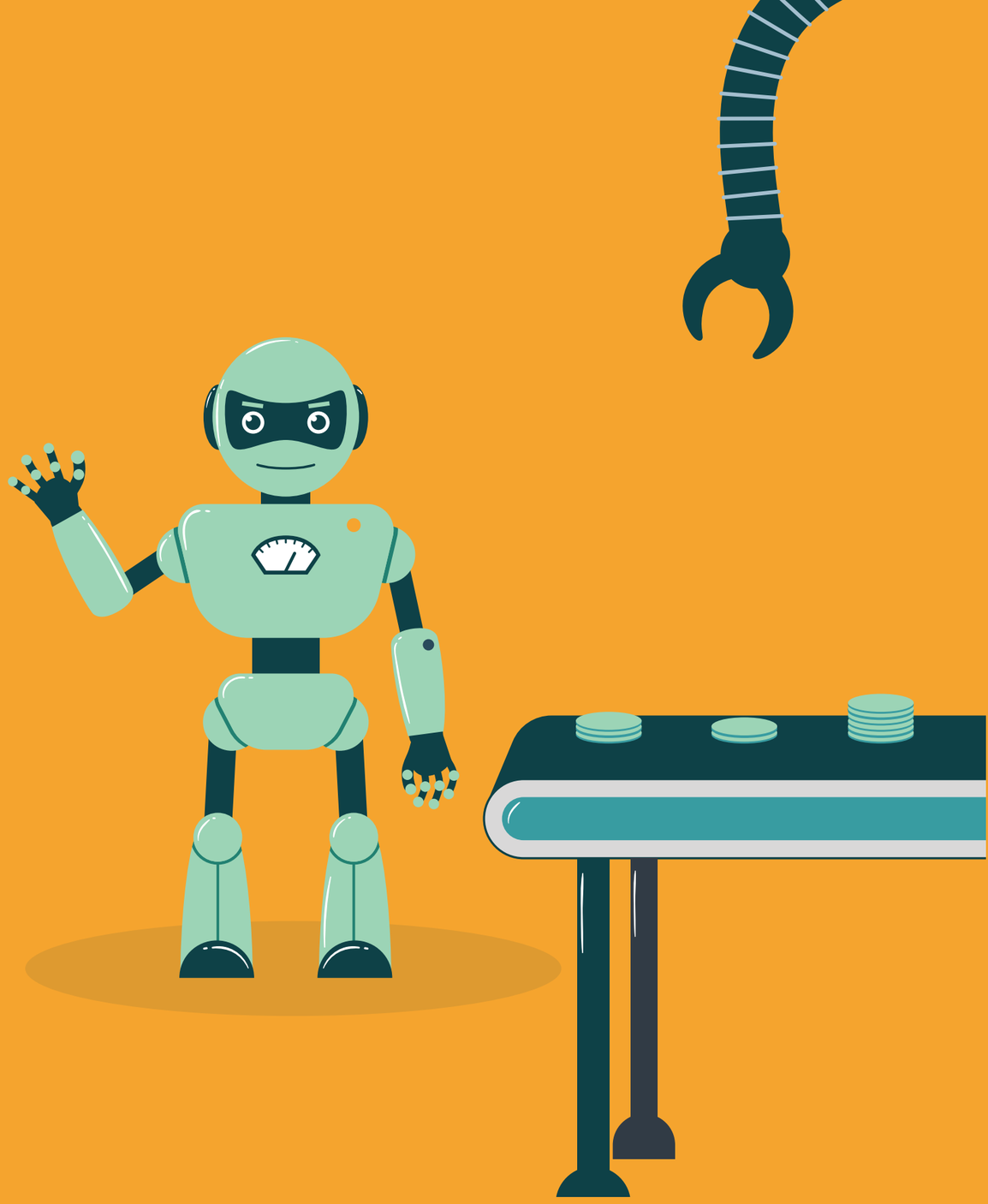
> ***We really appreciate how amazing the ObjectRocket team was quickly migrating us. Timing was of the essence and their team did an absolutely amazing job.***

Bryan Welfel

Co-Founder, Smooch Labs (creators of JSwipe)

Dirty Job #2
New Deployments

One of the biggest values ObjectRocket provides for customers is planning for new deployments.



When preparing a database deployment, it's important to understand how the application is going to hold up in production. ObjectRocket has developed a consistent, repeatable approach to managing a deployment environment so surprises are minimal once the database is in production.

The best approach incorporates prototyping a set up, conducting load testing, monitoring key metrics, and using that information to scale. The key part of the approach is to proactively monitor the entire system — which helps us understand how the production system will hold up before deploying and determine where you may need to add capacity. Having insight into potential spikes in memory usage, for example, could help put out a write-lock fire before it starts.

> ***"Working with ObjectRocket made our deployment a breeze. From the very first meeting through production deployment they have been responsive, professional, and consultative. We'll keep coming back, not only because they have a great product, but because they are great people to work with."***

Jeff Hansen

Technical Director, Digital Engagement, SingleStone

ObjectRocket's deployment planning includes:

MongoDB

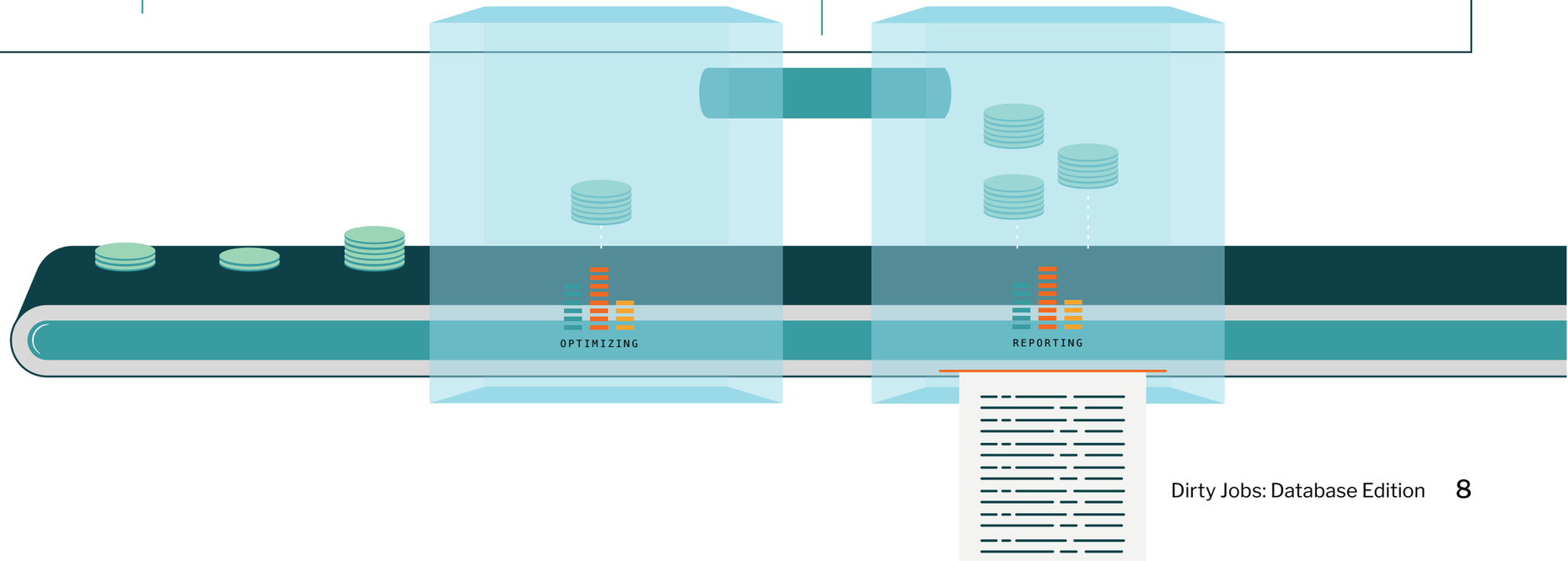
- Sizing clusters (number of data nodes, size of data nodes, disk to storage ratio)
- Consulting on query patterns
- Recommendations for indexing
- Schema optimization

Elasticsearch

- Sizing clusters (number of data nodes, size of data nodes, disk to storage ratio)
- Optimizing cluster settings for indexing or search
- Installing any custom plugins, scripts, or dictionaries needed
- Consulting on mapping templates and indexing strategies
- Configuring Curator
- Assistance configuring other components of the Elastic Stack (Beats, Logstash, etc.)

Redis

- Assessing the use case and finding the best data types.
- Research and contribute to common libraries and advise on best practices and any potential pitfalls to provide better performance.
- Profile customer activity and find any potential bottlenecks from a physical perspective to find the appropriate plan size(s) and possible separation of data.



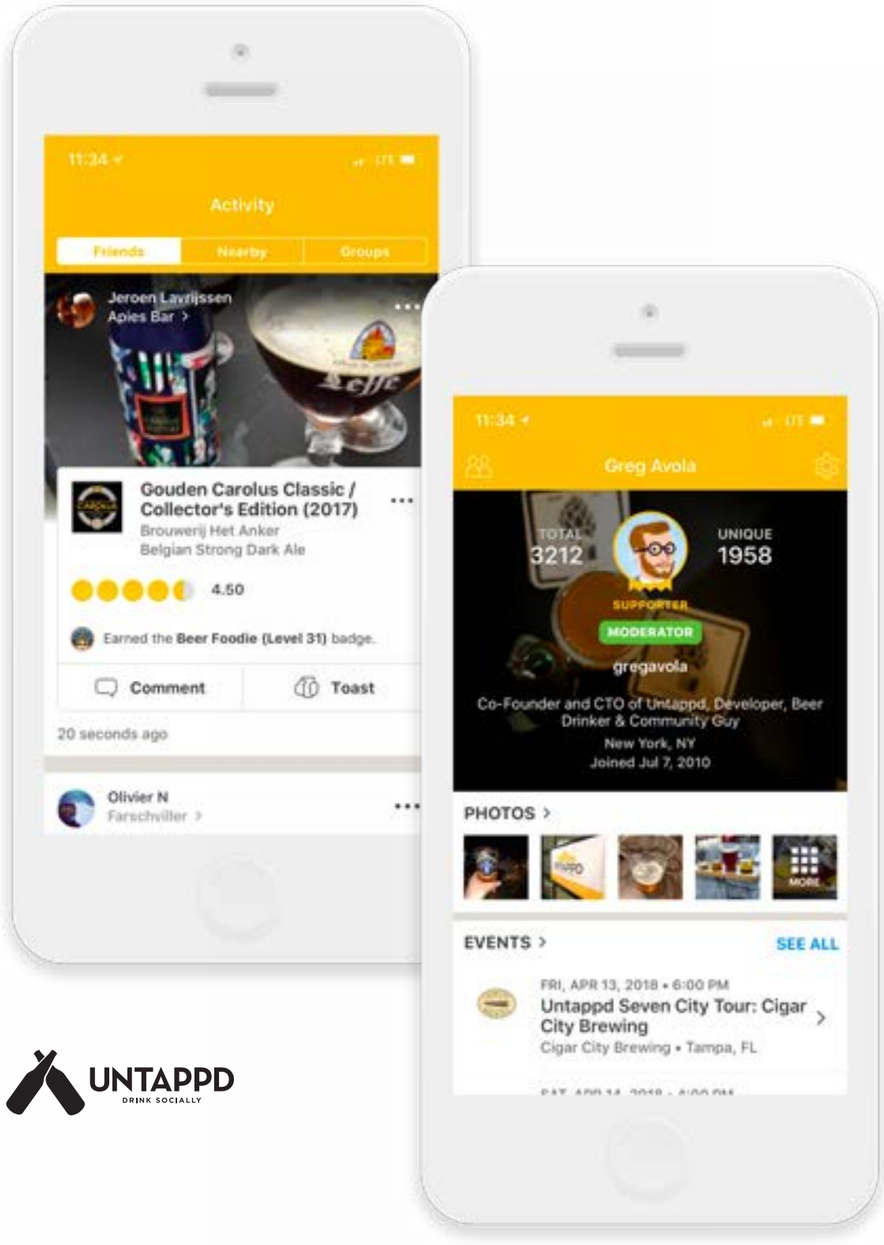
CASE STUDY

Story time with Greg Avola, CTO of Untappd

One time we were looking at our API logs for a server. We do a lot of requests, sometimes up to 38,000 requests per minute at certain peak hours. We have an internal service that works for a business product, but we saw some spikes in activity every 30 minutes. What was happening?

I investigated. “What requests are coming through?” I wondered. I wanted to see all the logs, but it’s very difficult to parse patchy logs at that volume. So we spun up an Elasticsearch instance with ObjectRocket, and we wrote all the requests, the index for a particular day. And then I used Kibana to actually search for the data, and I immediately found what I was looking for. Done.

That kind of thing is very easy to accomplish with ObjectRocket. I don’t have to write any complex queries. I don’t have to do any CSVs. I just set it up, do it, experiment, and take it down when I don’t need to have it anymore.

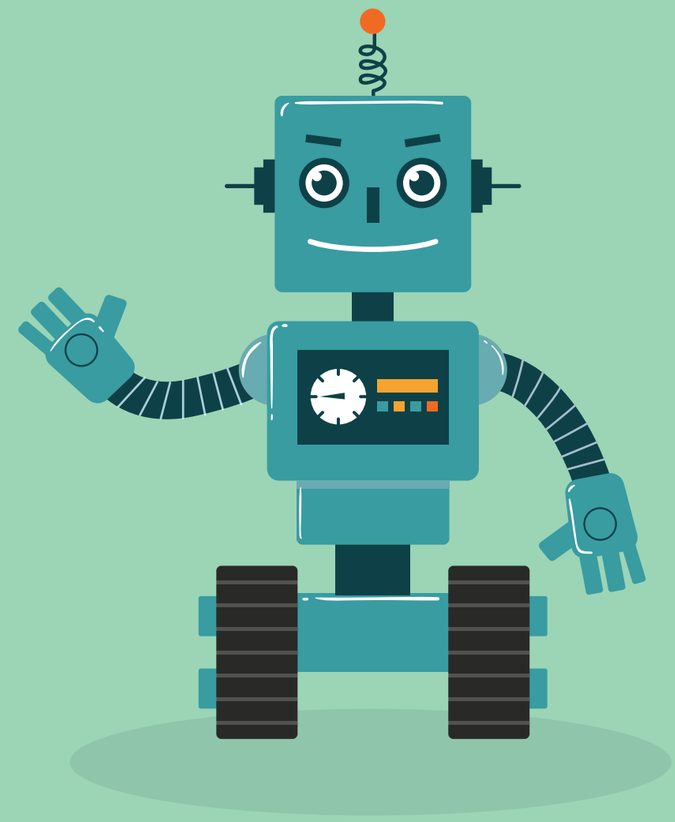


Dirty Job #3
Scaling & Sharding

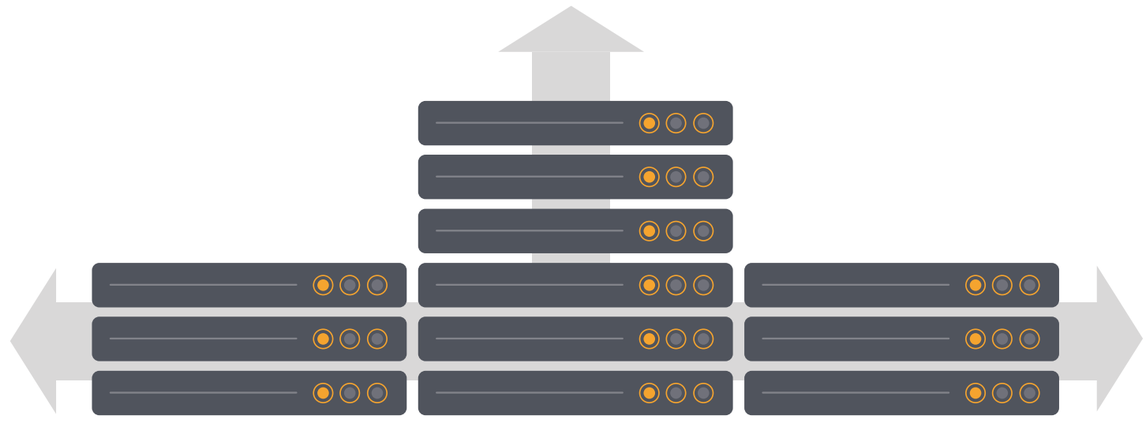
It's incredibly easy to spin up an instance of MongoDB, Elasticsearch, or Redis. Many of our customers come to us once they begin to feel the pain of scaling.

A lot of decisions need to be made at this point:

- Which way should we scale?
- How do we estimate how much space we'll need?
- Are there any strategies we can employ to save space?
- How should we approach usage spikes?



There are two methods to scale a database: vertical or horizontal.



Vertical scaling involves increasing the capacity of a single server, such as using a more powerful CPU, adding more RAM, or increasing the amount of storage space. Limitations in available technology may restrict a single machine from being sufficiently powerful for a given workload. Additionally, cloud-based providers have hard ceilings based on available hardware configurations. As a result, there is a practical maximum for vertical scaling.

Horizontal scaling (or sharding) involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required. While the overall speed or capacity of a single machine may not be high, each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server. Expanding the capacity of the deployment only requires adding additional servers as needed, which can be a lower overall cost than high-end hardware for a single machine. The trade off is increased complexity in infrastructure and maintenance for the deployment.

MongoDB

ObjectRocket was built on the core premise of enabling simple and reliable scalability for MongoDB. At its core, ObjectRocket utilizes MongoDB's native scaling architecture called sharding.

It's a cloud world. In the cloud, there are interesting things we can do when we have lots of compute at our disposal. Customers can start off small and grow automatically. When users sign up, they instantly get a fully provisioned sharded cluster in whatever plan size they choose. All of the complicated sharding components are automatically set up and there is zero configuration for a customer to complete. That means scaling on ObjectRocket is seamless.

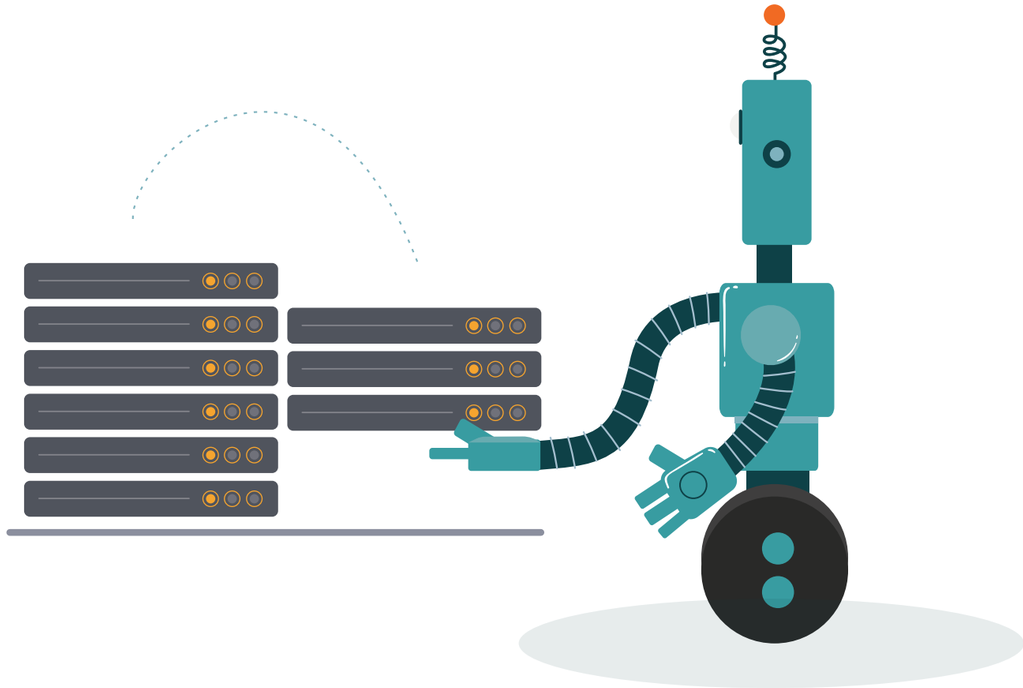
> ***ObjectRocket has done so many things over the years to help us scale from an environment that was only around 10 instances and about 150 shards to where we are today with over 44 instances and 1000 shards.***

Jon Hyman
CTO and Cofounder, Braze

Elasticsearch

ObjectRocket for Elasticsearch offers various data node sizes (for vertical scaling) and can support anywhere from 2 to 30+ data nodes (for horizontal scaling).

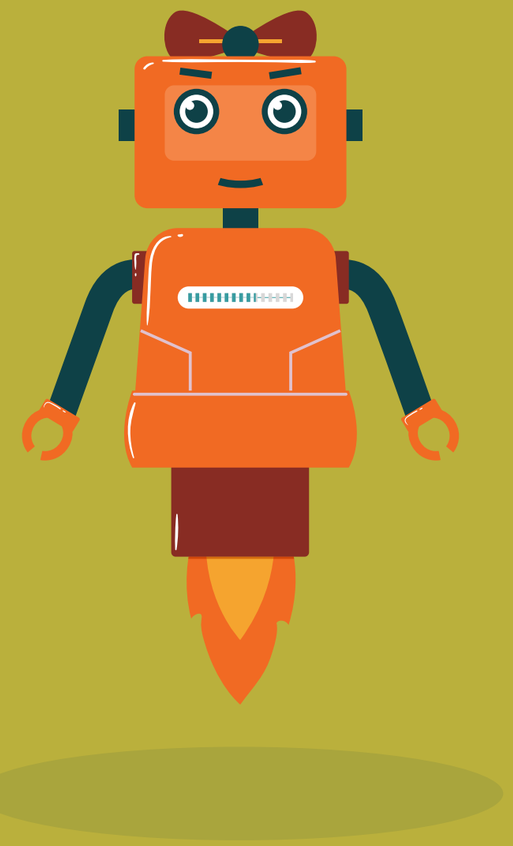
Elasticsearch is great at spreading data across your cluster with the default settings, but once your cluster begins to grow, the defaults can get you in trouble. Selecting the right shard and indexing settings can be a moving target. By planning ahead, making some good decisions up front, and tuning as you go, you can keep your cluster healthy and running optimally. We help businesses refine their Elasticsearch instances all the time.



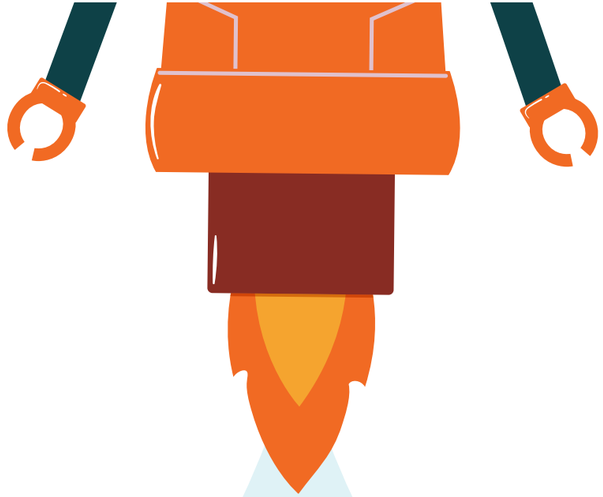
RocketScale™

By default, all Elasticsearch clusters on the ObjectRocket platform are created with RocketScale enabled and set to scale at 85% usage. This means that if any data node hovers above 85% utilization for 10 minutes or more (which is configurable), a new data node is added and the customer is notified.

Dirty Job #4
Upgrading



Upgrading databases can be painful. We talk to many CTOs and other technical leaders that find themselves in this situation. They've fallen really far behind and they feel stuck. ObjectRocket can help.



Upgrading databases isn't at the top of your list. 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 enhancements to your application are always more critical to the business. You keep putting it off, knowing that you are getting farther and farther behind.

ObjectRocket has performed thousands of upgrades for our customers with little to no downtime involved.

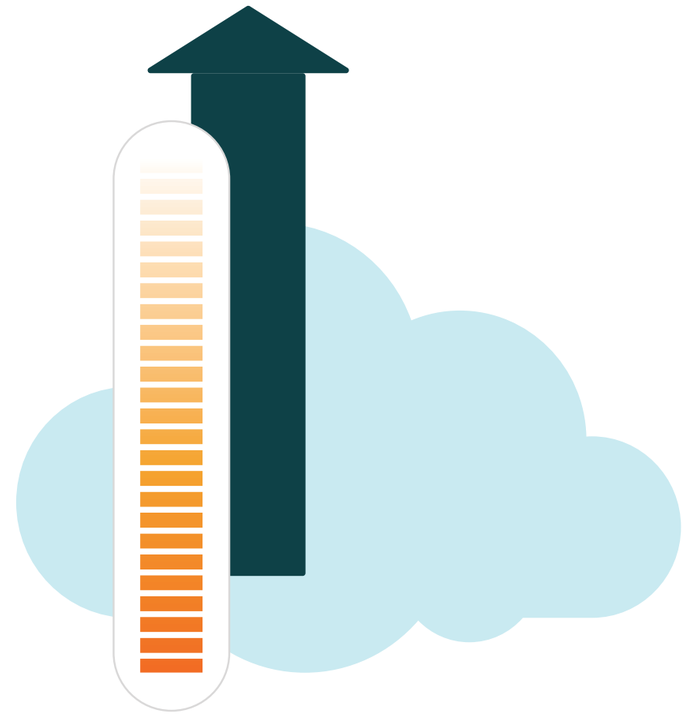
Upgrades Made Easy

ObjectRocket handles all of the heavy lifting of upgrades for you.

MongoDB

From possible downtime, to rewriting code, to updating drivers — it can be painful to upgrade MongoDB instances.

MongoDB Inc. releases updates about once a year. In February 2018, MongoDB deprecated version 3.0. MongoDB has also announced that they are deprecating version 3.2 in September 2018. Each release brings new features, major bug fixes, and performance enhancements. When evaluating an upgrade to a newer version, customers may have some features they'd like to add and there may be some nagging bugs they'd like to squash.



Elasticsearch

Upgrading to major versions of Elasticsearch is almost never simple. The velocity of Elasticsearch version releases has truly become a double-edged sword. On one hand, the number of new features and capabilities in each new version is staggering, but major version upgrades are always painful. When you're running Elasticsearch in production, downtime coupled with a potential reindexing is a non-starter for most companies. This is why we see so many users still running Elasticsearch version 1.x, despite the fact that it's been out of support for over a year.

Redis

Upgrades to Redis on ObjectRocket is simple. There's a one-touch button upgrade that only incurs a few seconds of interruption, as everything is handled on the backend.

Upgrades with ObjectRocket

ObjectRocket makes upgrades much easier. We handle upgrades all the time. We have upgraded thousands of instances over the years and we've learned a lot about making the process simple with little to no downtime. When you work with ObjectRocket, we ensure your upgrades won't fail and we can help ensure that there aren't inconsistencies in your data. While some companies offers tools, we can handle all of the upgrade details for you.

ObjectRocket's upgrade service includes:

Setting up a timeframe and acceptable amount of downtime during the upgrade

Moving the data to a free temporary cluster using backups and remote reindexing

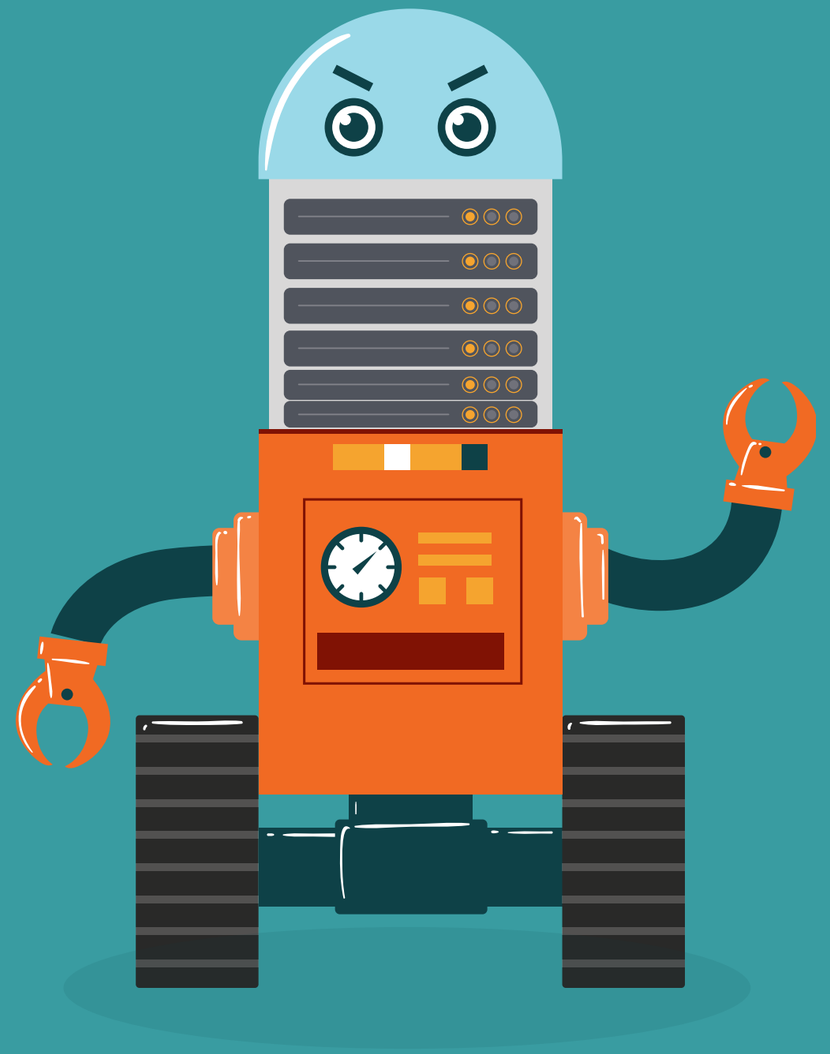
Consulting on mapping/feature changes that will be needed

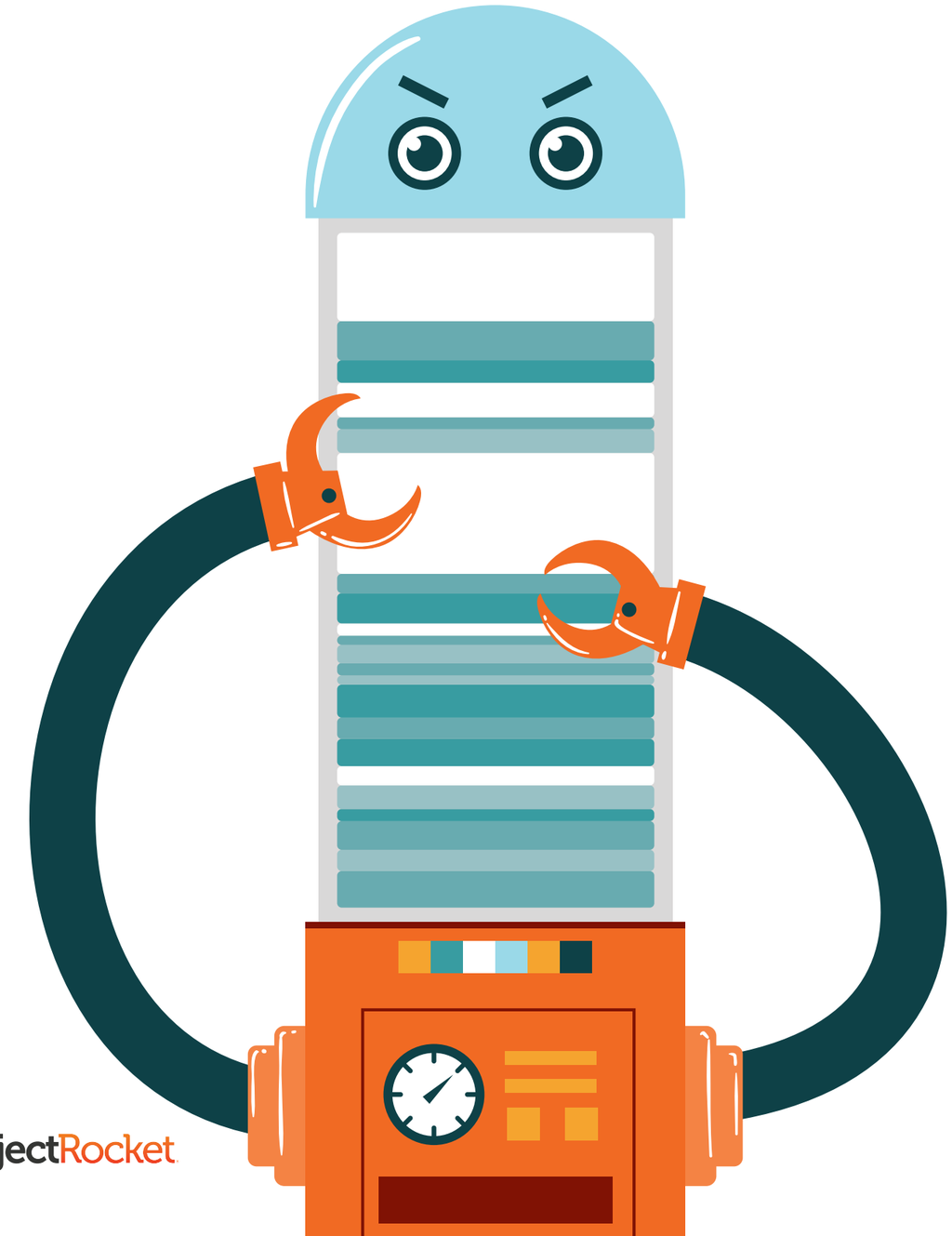
Preparing the new cluster for testing

Assisting in a cutover to the new cluster

Dirty Job #5
MongoDB Compactions

Running a big MongoDB installation necessitates a certain amount of periodic maintenance. One of these routine tasks is compaction.





When documents are deleted from a collection, empty spaces are left behind. Over time, the collection becomes fragmented. MongoDB tries to reuse this space where it can, but the space is never returned to the file system. Consequently the file size never decreases despite documents being deleted from the collection.

This can be a more serious problem where data usage patterns are fairly unstructured. As time goes by, your database ends up taking more space on disk and in RAM to hold the same amount of data – because in practice it's actually data plus empty spaces – which slows the server down and reduces overall query capacity.

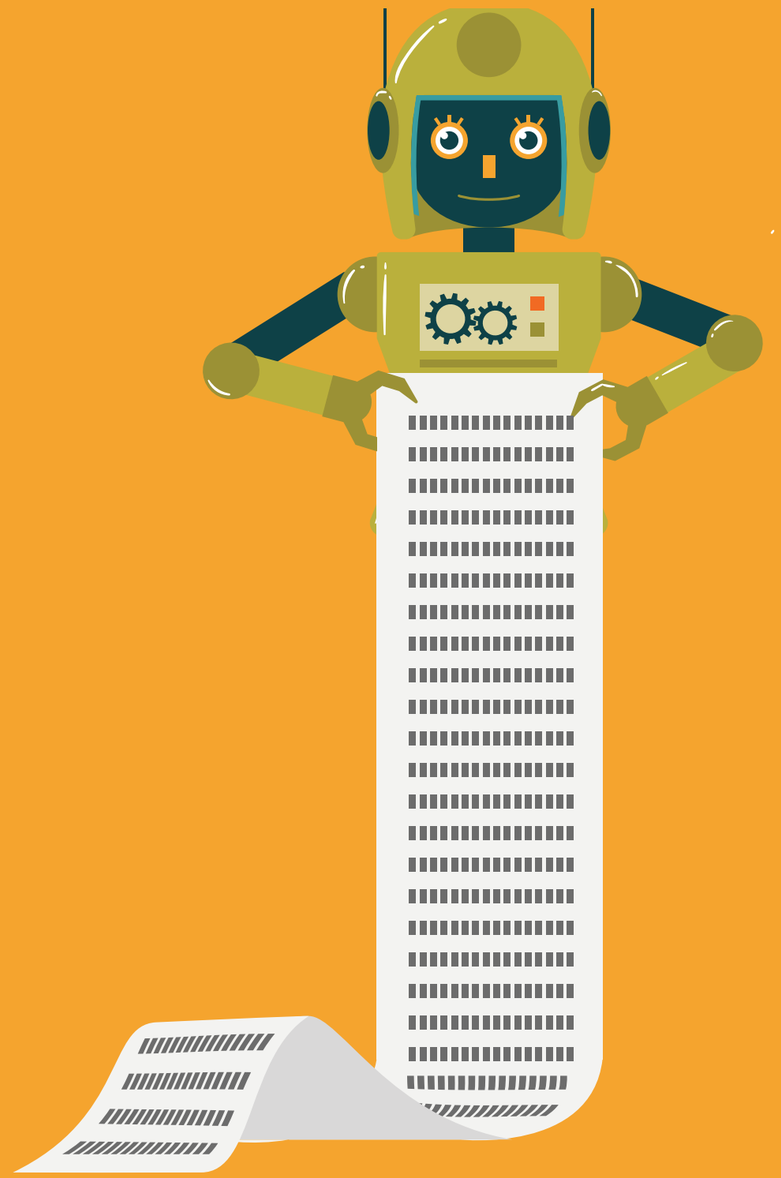
The aim of compaction is to get the empty space back into use. It rewrites and defragments all data and indexes in a collection. It does this by coalescing the documents – i.e., it moves all of the documents at the "beginning" of a collection, leaving the empty space at the end of the collection.

If you're using the MMAPv1 storage engine, compaction will not return the recovered space to the file system, but if you're using the WiredTiger storage engine, it will.

> ***You would have to use other database management providers to understand just how good ObjectRocket is. I'm not just buying dollars per gigabyte. I am buying a deluxe team that is there for us 24/7.***

BJ Fox
VP of Engineering at Thunder Industries

Dirty Job #6
Parsing Logs



As part of the normalization process, log messages are passed through a parser to determine if the details match specific parameters. If the parameters are met, then the log messages are tokenized to allow for better searching, alerting, and reporting.

Log Analysis and Troubleshooting

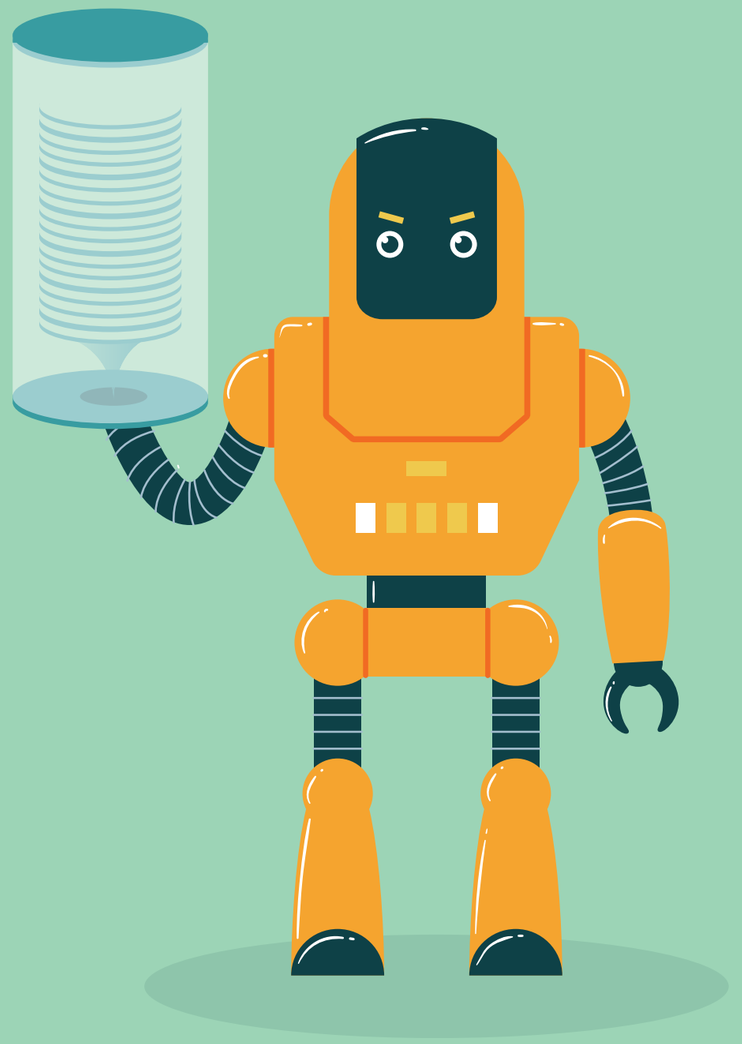
ObjectRocket analyzes logs to increase troubleshooting efficiency. A parsed log helps answer many types of questions because you can slice and dice an application's behavior with very fine granularity.

Often, organizations don't pay much attention to their logs because there are just too many of them. It's not unusual to generate thousands of log messages per second. From process behavior (e.g., tracking what the Balancer and websvr processes are doing right now) to overall health metrics of your whole database (e.g., monitoring the performance impact of the new code that you just pushed), log messages give ObjectRocket the knowledge to keep applications in a healthy state.

> ***"There are a few different ways we parse logs for customers. We have some logs aggregated and put into one place, which is nice for some troubleshooting. For other logs, I come up with custom regular expressions all the time to get really granular. Sometimes I just open up 150 terminals and watch logs stream across my screen until I narrow things down to find what I'm looking for.***
>
> ***ObjectRocket Customer Data Engineer***

Dirty Job #7
Draining Shards

Most database hosting services base their price on capacity.
So, removing capacity can save money. Most database-as-a-service providers either charge extra or force you to do it yourself.
That's not the case with ObjectRocket.



Balancing

When you delete old data, it can throw your shards off balance. Issues can occur when deleting because it can leave empty chunks lying around in your shards. This sounds innocuous enough, but the balancing algorithm adjusts the number of chunks across all your shards and ignores the size of each chunk. If some of those chunks are empty and others are 64MB, you can quickly have one full shard and one half empty shard. This is common in customers that started small and have grown considerably without reindexing and/or modifying shard count.

If there's one data node that's very full and the rest are not, that could be a sign that there's an issue with the size of the shards, cluster settings, or the cluster plan size. RocketScale™ detects most of these scenarios and alerts us internally rather than adding a new data node, so one of our engineers can investigate the large imbalance in node utilization. Unless we can completely fix it on our end, we'll usually reach out to customers when we get these alerts to propose solutions specific to the cluster.

Here are some important points:

Good shard keys prevent unneeded balancing

Give adequate time to balancing existing and newly sharded collections

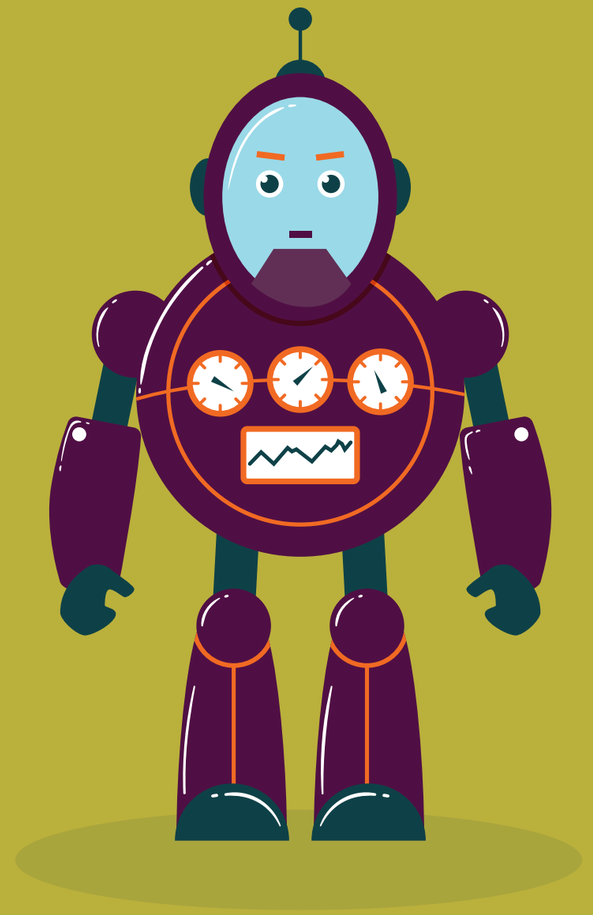
Set a window to prevent balancing during peak times

> ***I love ObjectRocket so much because of the level of support they provide for us. We have a Slack channel; we can talk with them. If there's an issue, they're always very on top of it.***

Greg Avola
CTO at Untappd

Dirty Job #8
Alerts & Monitoring

Sleep better at night knowing that ObjectRocket automation and database engineers have your back with specific monitoring and alerts set up to ensure your data is always available.



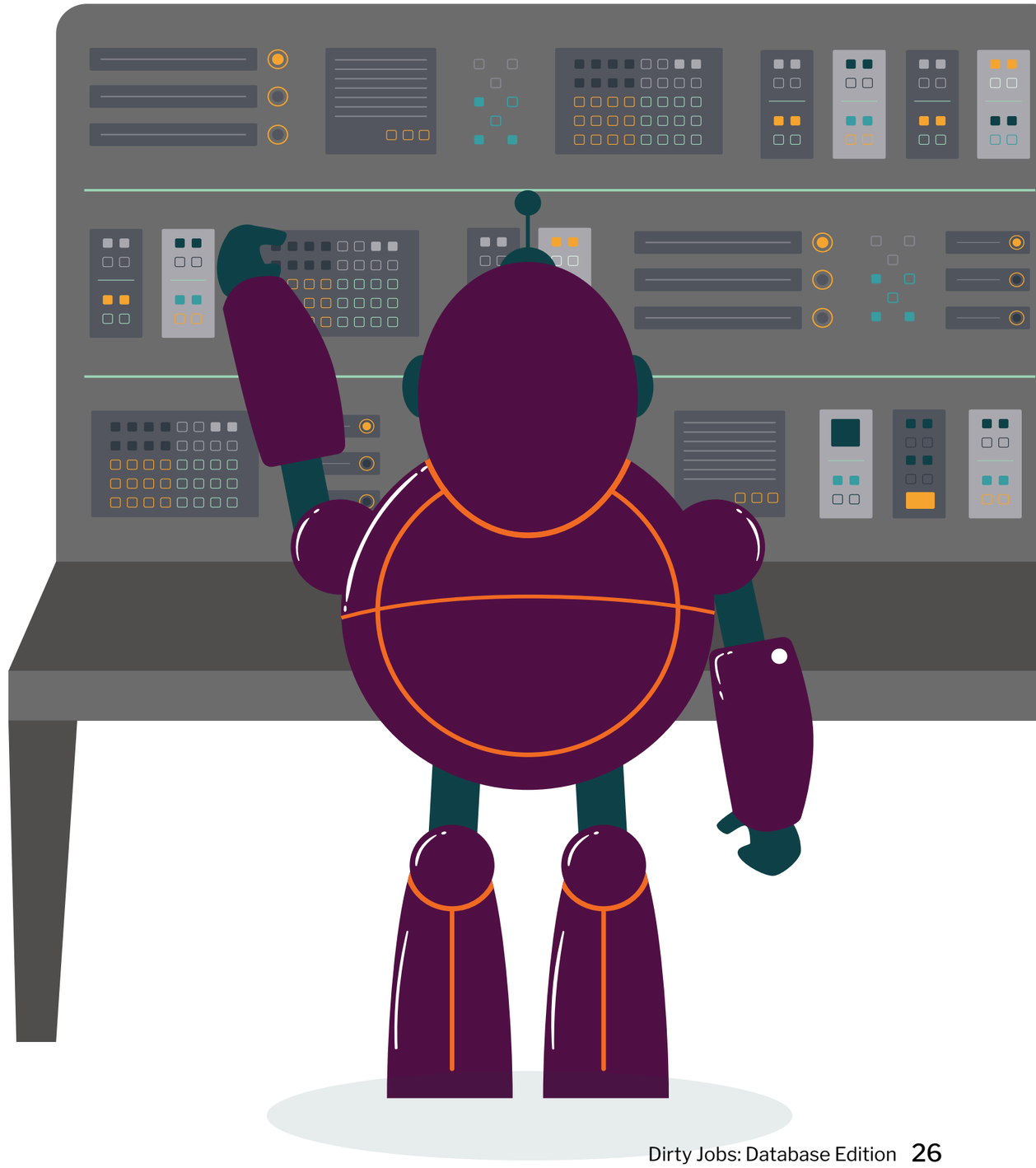
ObjectRocket's robot friends proactively monitor metrics such as RAM and disk usage, IO, number of operations, and much more. When one of the monitored metrics is off, the ObjectRocket team is alerted immediately. If there's a failover or disk usage issue, we take care of it.

ObjectRocket has alerts set up for each type of database:

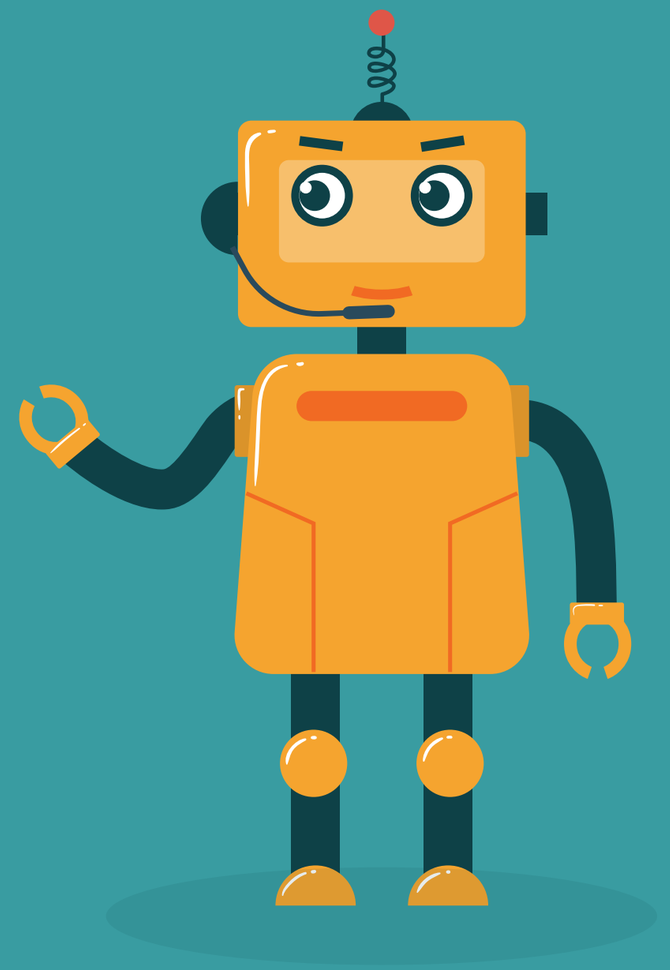
254
metrics/minute
on every MongoDB instance

254
metrics/minute
on every Elasticsearch instance

50
metrics/minute
on every Redis instance



Dirty Job #9
Query Analysis & Index Optimization



Have questions about why your queries are slow? ObjectRocket DBAs can provide answers. Our experts optimize MongoDB and Elasticsearch queries for customers all the time.

Poor performing queries are usually the result of not having an index at all or not setting up the appropriate indexes for your queries.

> ***What's really helpful to me as CTO is we don't really have to have that non-relational database expertise in-house. We rely on ObjectRocket to help us optimize our queries at what is sometimes an absolutely insane scale.***

Alen Durbuzovic
CTO at Chive Media Group

Index Management

If you've ever used physical cookbooks or large reference manuals, you know how handy it is to have a good index at the back of the book. If there's no index that tells you what page the "chicken parmesan" recipe is on, it can take forever to flip through the book to find it. The same holds true for database indexes. The larger your cookbook or database instance, the more important it is to have a good index to help find things quickly.

MongoDB

Are snapshots of your MongoDB instances trending in the wrong direction?

Troubleshooting MongoDB isn't always easy because there are often a combination of issues causing slow database response. One of the most common issues we see slowing down MongoDB stem from index issues.

- **Missing an index entirely.** If there's no index at all, it can slow your MongoDB instances down to a halt.
- **Indexing too much**

If every "the" and every "and" is indexed, it can slow down your MongoDB instances, also. Each time data is inserted into results, the indexes are updated. While it's better than having no index at all, it can take longer for MongoDB to find what you're looking for and then everything slows down.

Elasticsearch

Have you noticed that your app is slow?
Having an Elasticsearch DBA review your index strategy may help.

Massive Indexes and Massive Shards

The most common Elasticsearch index issue we run into are massive indexes with massive shards. This causes is poor efficiency in cluster use. As the shards get larger and larger, they get harder to place on a data node since it will require a large block of free space on a data node. This leads to nodes with a lot of unused space. For example, if there's two 8GB data nodes but each shard is 6GB, 2GB are left stranded on each data node.

ObjectRocket philosophy

Don't be stingy with indexes

Spreading data across multiple indexes increases the number of shards in the cluster and help spread the data a little more evenly. In addition to an easier game of Tetris® when Elasticsearch places shards, multiple indexes are easier to curate. Also, the alias capabilities in Elasticsearch can still make multiple instances look like a single index to your app.

Increase shard count as your index size increases

If we see shard sizes starting to get a little too large, ObjectRocket can update your index template to use more shards for each index. Our rule of thumb: if a shard is larger than 40% of the size of a data node, that shard is too big. In this case, we recommend reindexing to an index with more shards, or moving up to a larger plan size (more capacity per data node).

Too Many Indexes and Too Many Shards

Indexes and shards have overhead. That overhead manifests itself in storage/memory resources as well as in processing performance. Since the cluster must maintain the state of all shards and where they're located, a massive number of shards becomes a larger bookkeeping operation which will have an impact on memory usage. Also, since queries will need to be split more ways, there will be a lot more time spent in scatter/gather for queries.

Solutions for this are highly dependent on the size of the cluster, use case, and a few other factors, but in general we can mitigate this with a few recommendations.

Make sure shards aren't too large

In general, 25GB is what we target. However, 50GB is where we have the conversation with our customers about reindexing. This has as much to do with the performance of the shard itself as it does with the process of moving that shard when you need to.

Keep shard size less than 40% of node size

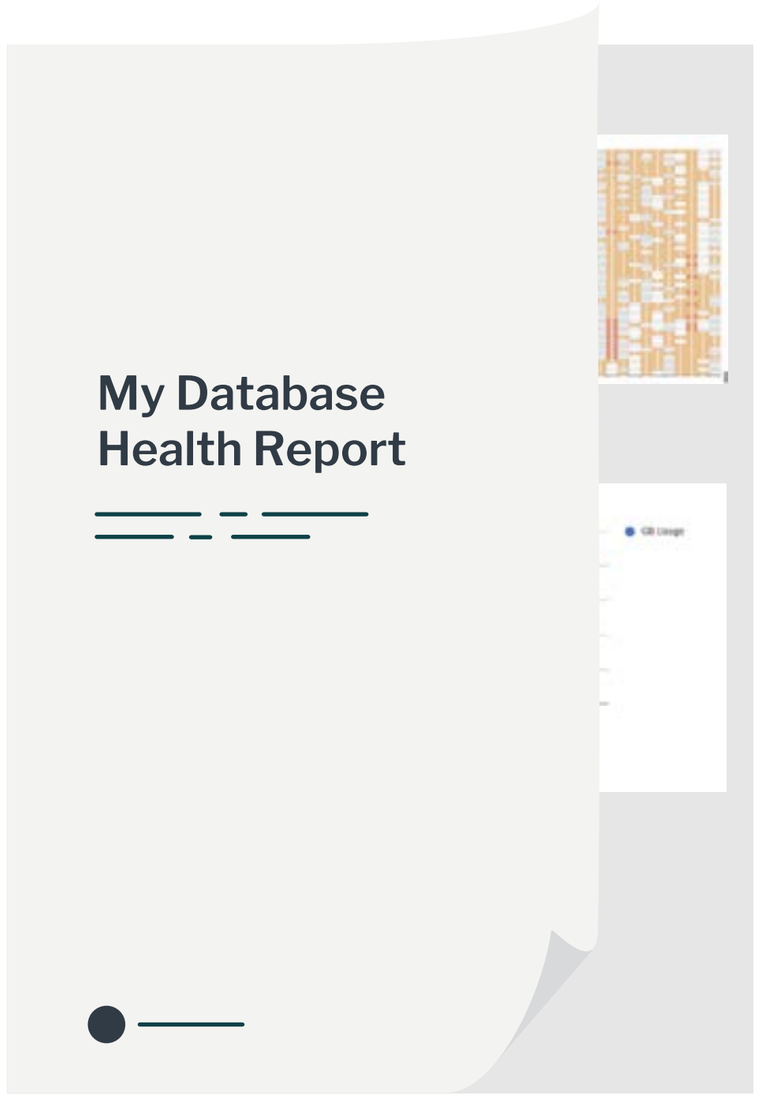
We try to size the cluster and the shards to ensure that each of the largest shards don't take up more than 40% of a data node's capacity. In a cluster with a number of indexes at a mix of sizes, this is fairly effective, but in a cluster with a single or very few indexes that are very large, we are even more aggressive and try to keep this below 30%.

Selecting the right shard and indexing settings can be a moving target, but having ObjectRocket help you by planning ahead, making some good decisions up front, and tuning as we go, **you can keep your cluster healthy and running optimally.**

Dirty Job #10
Health Reports

ObjectRocket uses the right people, processes, and technologies to automate and enhance performance and ensure your future-readiness.

ObjectRocket provides full health reviews of customer instances. Based on capacity, usage, and network metrics, DBAs put together a set of actionable recommendations to maintain the overall health and reliability of the customer's environment. Each report is unique based on the customer's workload and use case.



Why choose ObjectRocket?

Our team of database engineers, DBAs, and robot friends will get your payload of data into orbit without crashing your ROI. **Aside from all the dirty database jobs we take care of for you, here are some other reasons 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.

© 2018 Rackspace, US Inc. | All trademarks, service marks, images, products and brands remain the sole property of their respective holders. Date Modified: 10/10/2018