Rivery

The Data Practitioner's Guide to

Cloud Database Migration

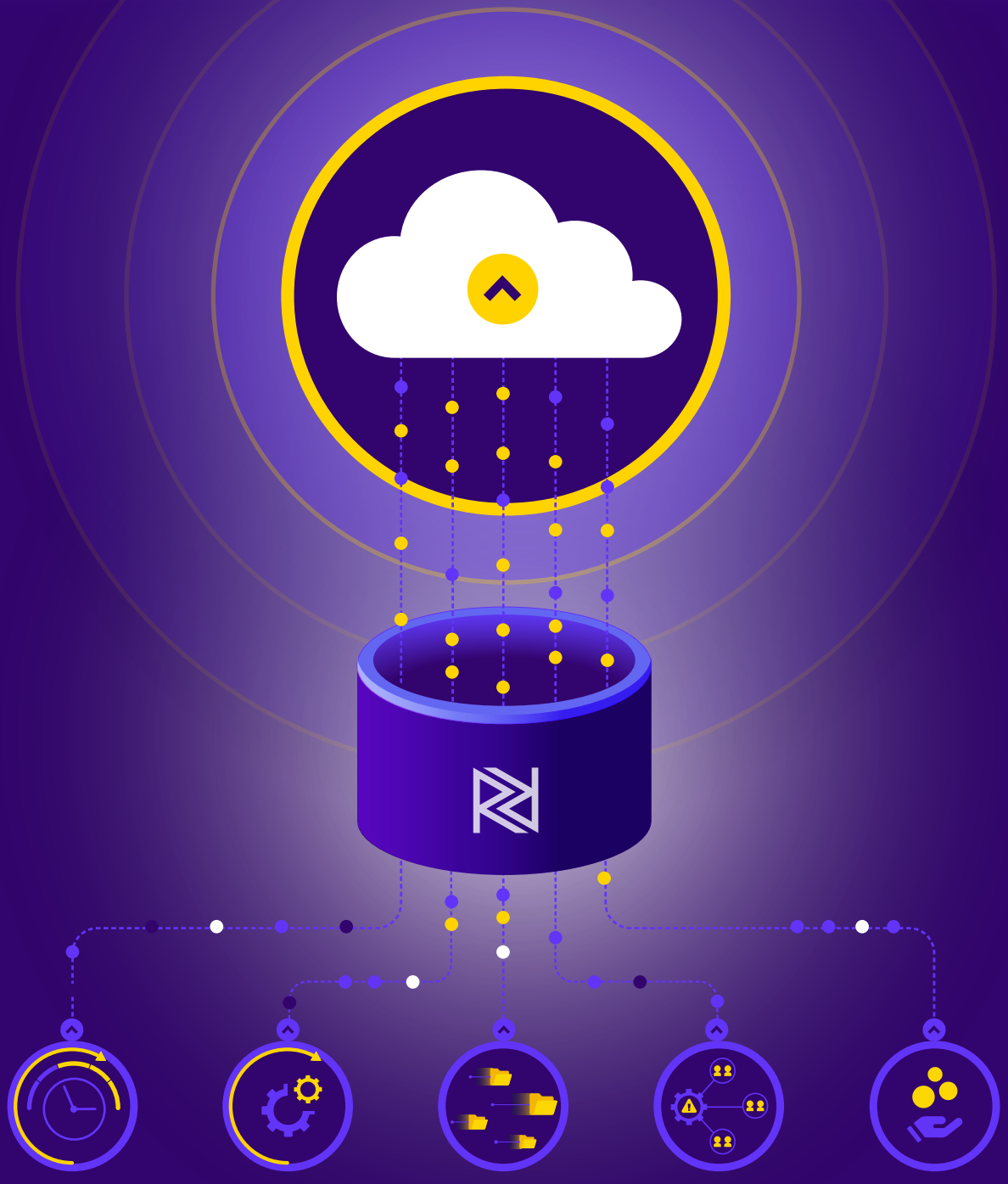
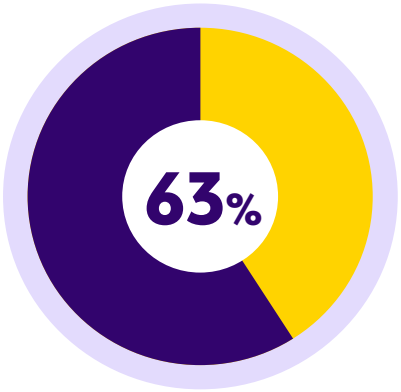


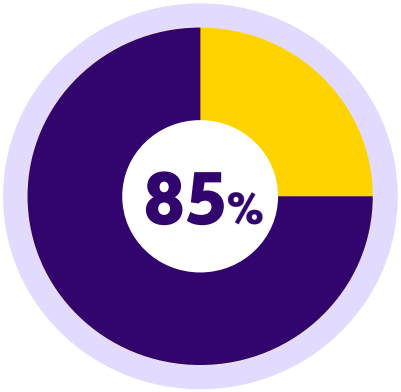
Table of contents

Introduction

The benefits of moving your self-hosted or self-managed database to the cloud outstrip the challenges by far. The facts can't be ignored:



According to IDC over 63% of companies are actively migrating mission-critical databases to the cloud



85% of companies will adopt a cloud-firstapproach by 2025, according to Gartner

Moving your databases to the cloud is not a question of if, it's a question of when. The sooner, the better.

Many organizations learn the hard way that migrating data to the cloud is not a simple plug-and-play undertaking. If not done properly, it can be a rollercoaster of pitfalls and headaches - all too commonly these projects miss deadlines and go over budget.

To help navigate the potential pitfalls, this guide includes real-world best practices and extensive tried-and-tested methods of migrating databases to the cloud.

One last thing to keep in mind, is that every cloud database migration is unique, so be sure to consult with people who understand your specific needs before actually pulling the trigger.

What is cloud database migration?

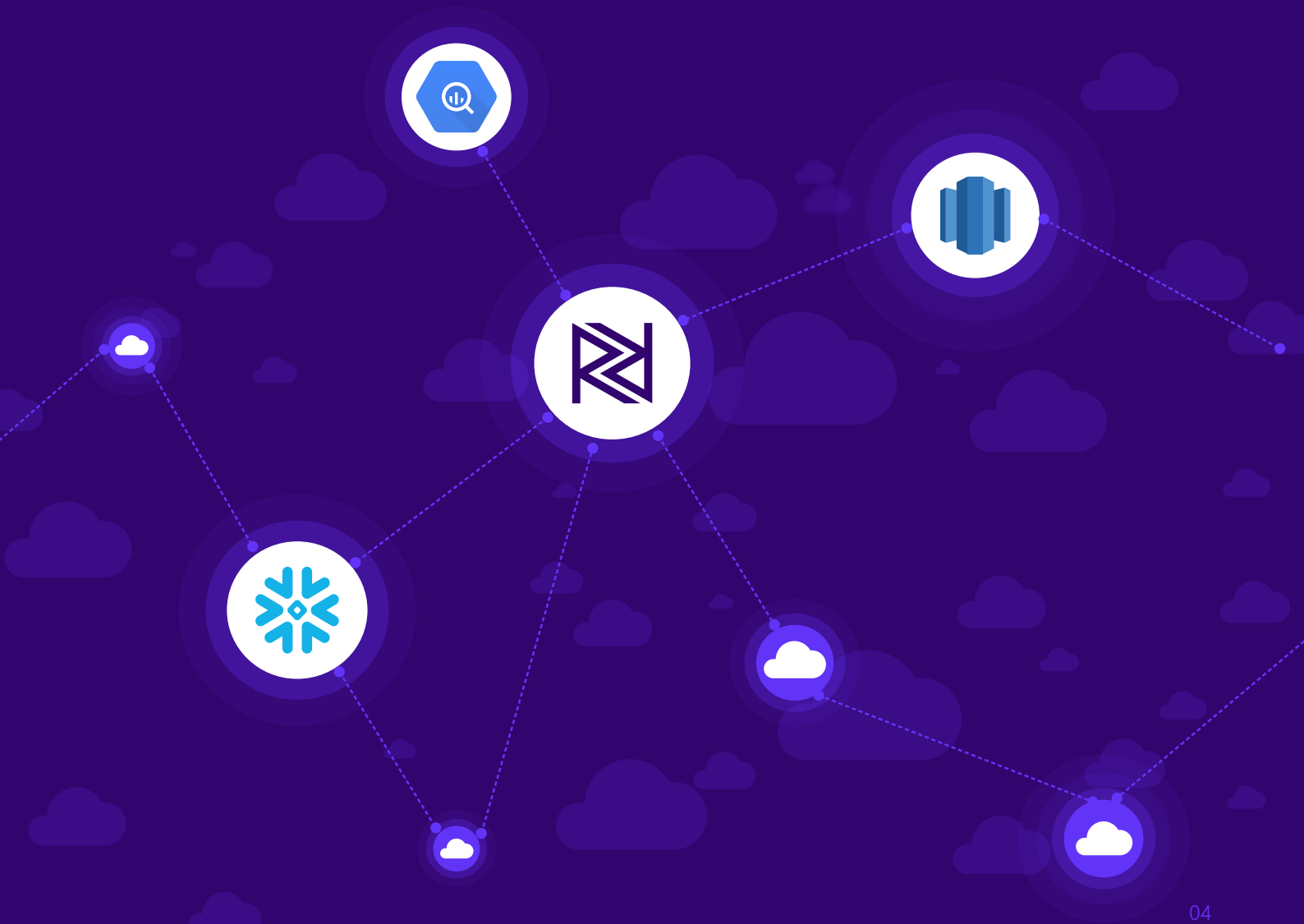
Simply put, cloud database migration means the ability to take a legacy database or warehouse, which is currently stored on-premise / or self-managed, and move its contents and usage to the cloud, either to a relational database (RDBMS) or more commonly to a data warehouse (using columnar storage).

For example, a company might be moving from a MySQL or PostgreSQL on premise database to Snowflake, Redshift or BigQuery - three common cloud data warehouses.

EMAAR

Companies like Emaar use Rivery for seamless database data migration, pulling in data 46 different MS SQL serves, processing 300GB+ of data every week.

Read case study →



Why are companies migrating their data to the cloud - what are the advantages?

For most organizations, managing and hosting database servers on their own is simply not worth the headache. The hassle of maintaining the hardware and/or software for a database server requires a database administrator and comes with an endless stream of troubleshooting technical issues, applying patches, software updates, performing backups, etc.

In order to remain competitive, companies are turning to cloud databases and cloud data warehouses to reap the benefits of the following advantages:

Scalability

Both in terms of performance and data size, cloud vendors can easily ramp up database size and/or increase performance with zero downtime. Some solutions, such as Snowflake and Firebolt also separate compute power from storage.

This means providing one group of users with less powerful (but much cheaper) compute engines to perform analysis, while providing another use case, like a daily report process, with more powerful (albeit more expensive) data engines.

This also addresses the issue of peak usage patterns. For example, a company that needs to run monthly reports on the first of every month will often need an extra boost of computing power to run the reports, but does not need all that horsepower during most of the month.

Cost Savings

As mentioned before, with cloud databases there is no need for any internal DBA to manage or maintain database servers, or internal IT professionals to maintain infrastructure. There is also no costs for software licensing fees or hardware - providing zero upfront or fixed costs.

With most cloud solutions, pricing is usage based and is relative to the compute power consumed. While storage costs are also a factor with cloud databases, storage costs are usually smaller than compute costs.

For example, Snowflake storage costs are about $24 a month per terabyte of stored data, so for 50TB of data storage, the cost would be less than $15k a year.

For some use cases, moving to the cloud could potentially cost more in the long run. For example, if the monthly costs are higher than the fixed costs of having two dedicated DBAs on staff + software + hardware costs.

It can be easy to rack up costs with usage based pricing if you're not careful. Be sure to keep an eye on costs and avoid becoming a cautionary tale.

Separating operational and reporting / analytical database

It's not uncommon for companies to use the same database solution for operational needs and for reporting/analytics needs, especially smaller organizations. As data sizes and connectivity needs increase, it makes sense to separate these two needs into two distinct systems. One common differentiator between a system being used for operations vs reporting is storing data in rows vs columns.

"Legacy" databases, such as Oracle or MS SQL server, have been around for decades and were initially created to be relational in nature. This means they were built and optimized for operations and transactions - frequent inserts, updates and deletes. For example the back end of an ecommerce store, SaaS application, or bank. While databases shine in transactional performance, they were not built for reporting, and do not scale or perform well for analytics purposes.

Whereas data warehouses are usually columnar, and were built and optimized to perform well at scale when used for analytics purposes. On the flip side, most data warehouses do not perform well if they constantly insert new data, or update / delete existing data.

The terms operational database and analytical database are often used to describe the two main use cases they are being used for - back end operations and reporting.

Reliability

Vendors in the cloud database or data warehouse space need to be reliable 100% of the time - their business literally depends on it.

That's why cloud vendors have comprehensive redundancies, failsafes, and backups in place. Most of the top cloud vendors provide solutions that are far more resilient than most in-house solutions (unless it's NASA or the likes).

Reduced risk of lost data

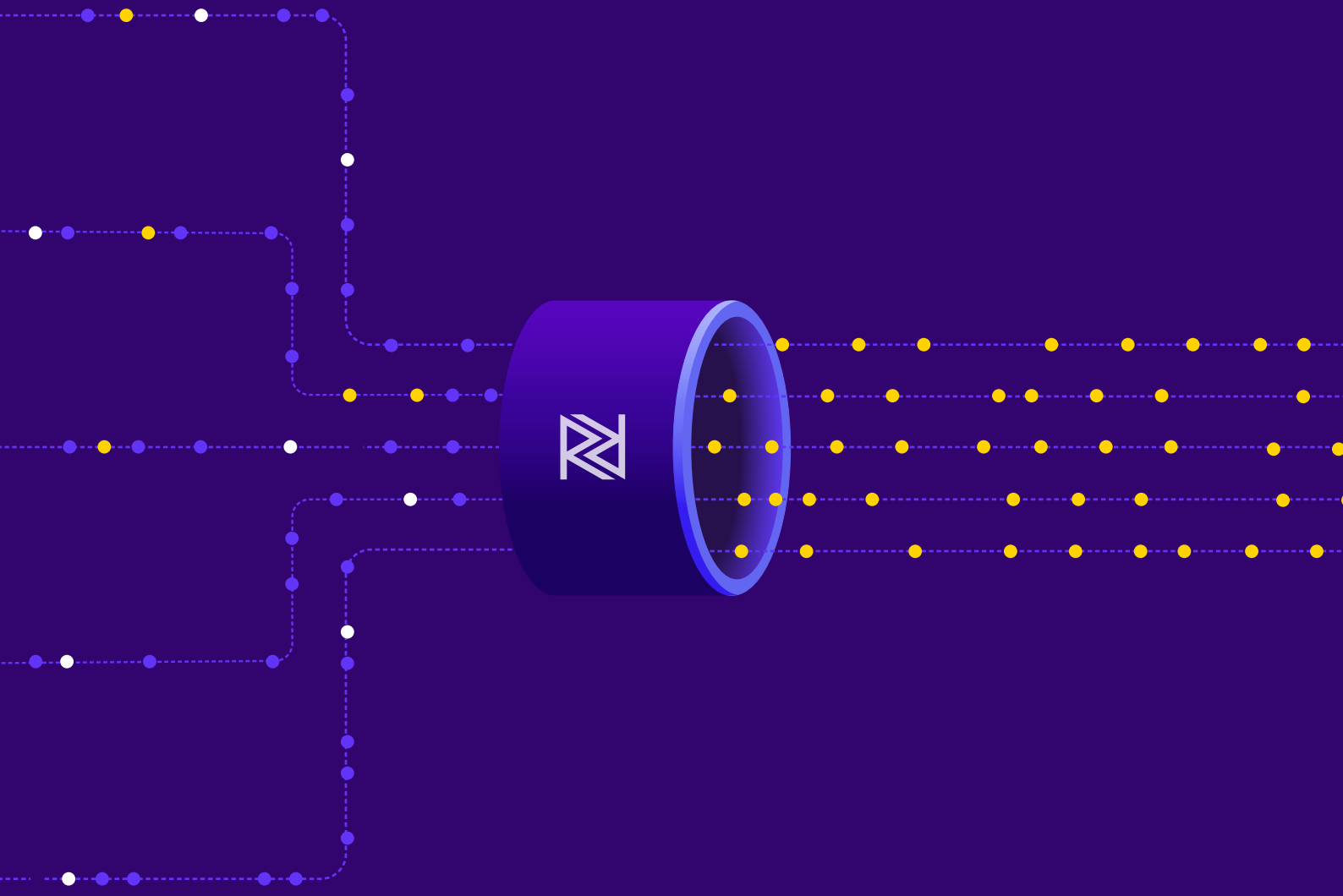
Availability aside, a hardware failure or natural disaster can be catastrophic in terms of lost data. Most cloud solutions provide various levels of backups, redundancies and even geographically distributed servers, assuring a random strike of lightning won't wipe out your data.

Improved Security

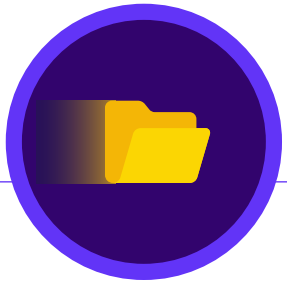
While many online articles actually say that moving to a cloud database reduces security, this is no longer the case, though it might have been true a few years ago. Reduced security is only a perceived issue, not a real one.

While with an on-premise database, you physically are in control of the hardware, cloud database providers have dedicated security teams and monitoring systems working 24/7 to keep unauthorized users out. Microsoft alone has committed to spend $20 billion on cybersecurity over five years.

Database Migration Strategies

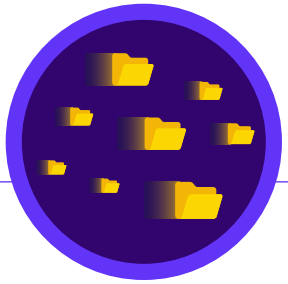


Here are three common strategies:



Big Bang Database Migration

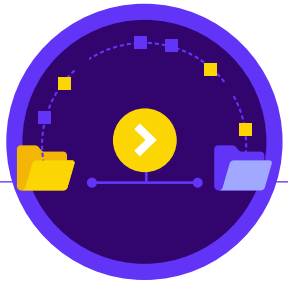
This is a hard cutoff from the old system to the new one, done at a single point in time (like over the weekend). This is a fairly out-of-date idea. With this method, it can be difficult to make sure the new and old systems are in sync, and when something goes wrong, rolling back the migration is a nightmare.



Piecemeal Database Migration

(also called trickle migration)

The idea here is to break down the overall migration into smaller individual projects that can be migrated individually of each other. This is a good idea in terms of being able to start slow and small, spreading the migration over time, though it introduces an added layer of complexity. Some systems will be using the old database and some systems using the new database, which might be a deal breaker in terms of having all of the data needed in one place.



Parallel Live Migration

(also called zero-downtime migration or live-live migration)

The goal of this migration strategy is to have both the old and new databases "live" in parallel with production data.

Parallel live migration, if possible, is preferred. Here's why:

There is going to be a point where either something goes wrong, or some data is missing, or the data in the new system doesn't match the data in the old system.

Either way, there are three requirements that only parallel live migration provides:

A

Ongoing validation that the data in the old and new systems match

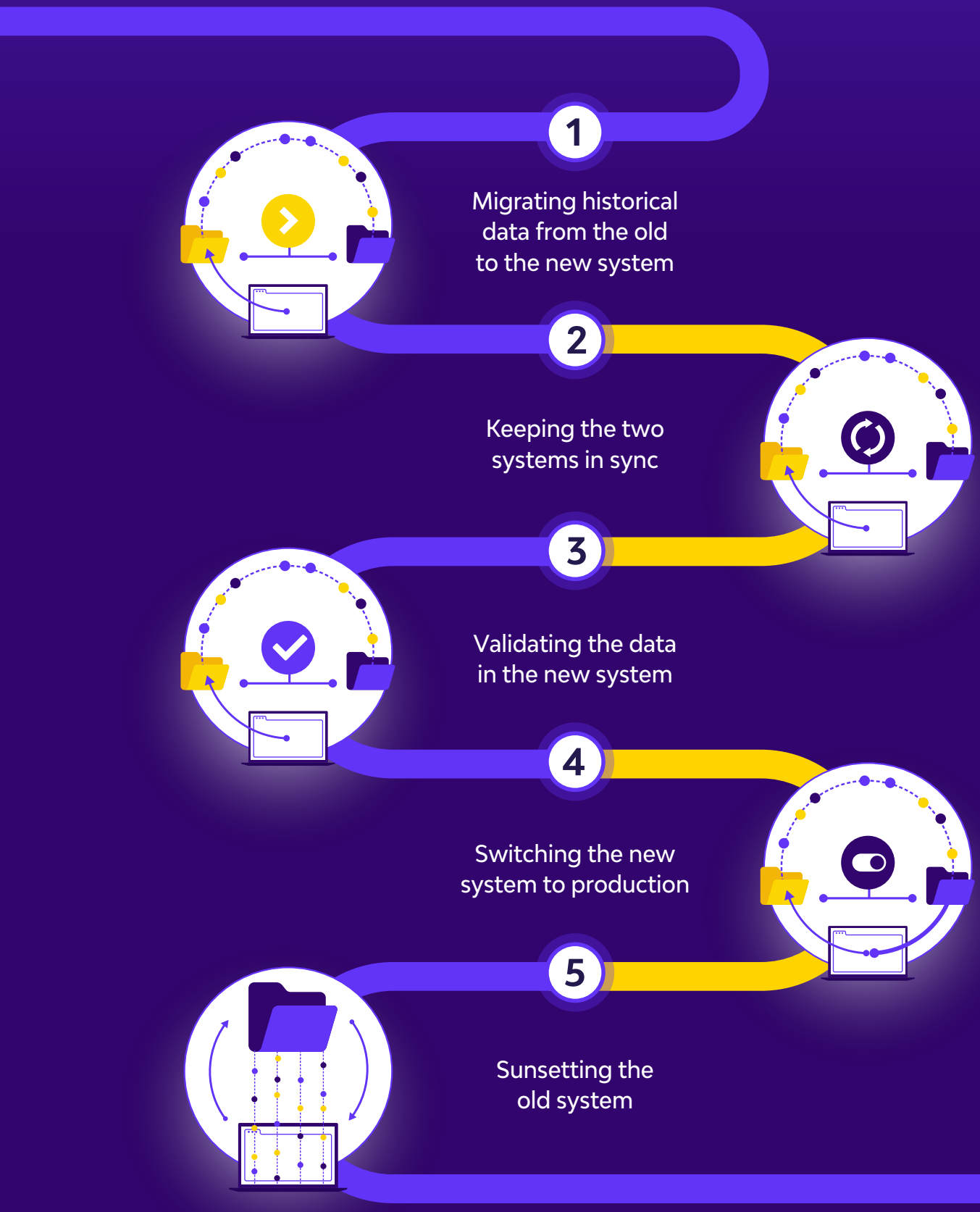
B

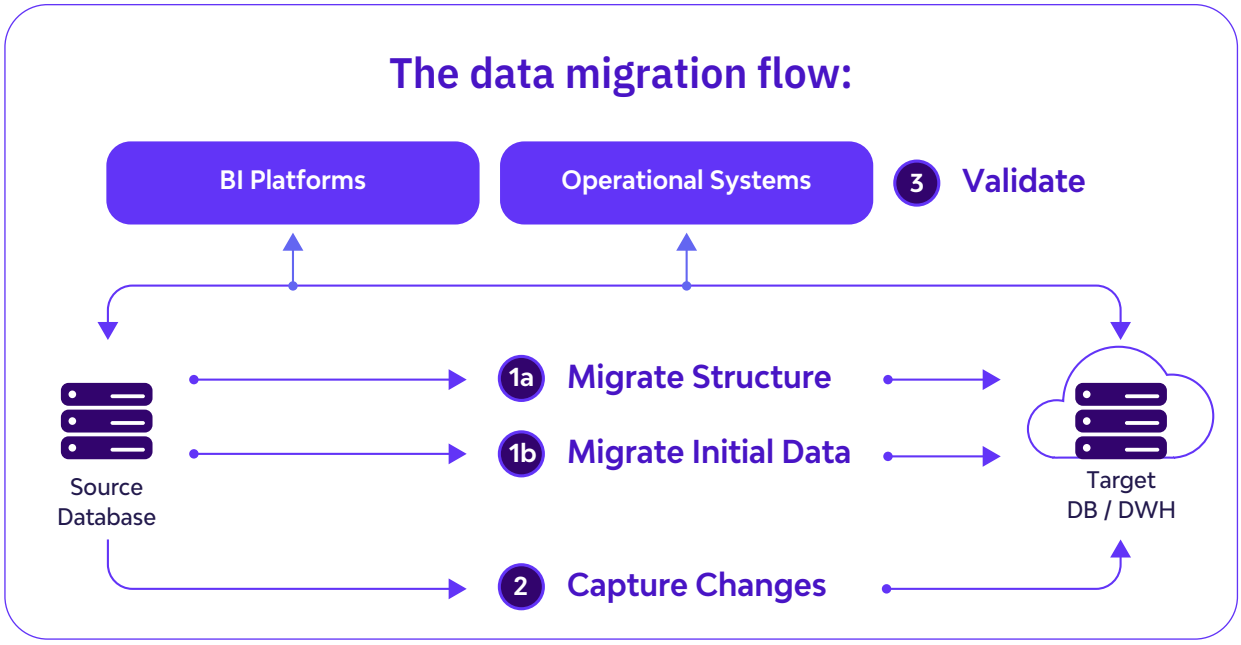
An easy way to test the new system with live data and live reports

C

Being able to reliably roll back if needed

The 5 data stages of a parallel live migration





Migrating data from the old to the new system

In some ways this is the easiest and in other ways, sometimes the hardest of the stages. Before migrating any of the data, figure out how to deal with differences between the two systems. While most databases and data warehouses support the same basic SQL commands at their core, every SQL based solution has its own specific "special sauce" and features that go beyond ANSI SQL standards. For example, how databases deal with JSON data, or arrays of data in a single cell. What they support in terms of data types, indexes, stored procedures, views, triggers, foreign keys, and functions are just a few of the common differences between databases.

This is also an opportunity to take advantage of new features or functionality provided by the cloud database. For instance, if moving from a relational to a columnar database it's time to rethink data structures. Also, replicating existing structures may also replicate existing issues / blockers into the new data warehouse. The actual migration of historic data up to a specific date in the recent past, is usually not the hard part. Try to use a specific end date when loading data, like the end of last month for the initial data dump and load. This is because it's rarely 100% perfect on the first load. If doing a data dump with an end date of "today" it will be much harder to do a diff between data loads done on different days.

When migrating a lot of data, here are two other tips for the initial migration.

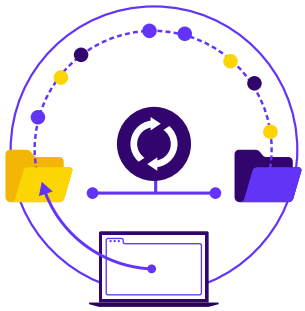
Start with a relatively small, but well defined subset of data, let's say orders from Jan 1st to Jan 31st of 2022, or users with IDs 0 to 1,000,000. Start doing data validation on that subset of data and only after everything is validated, move on to the rest of the data.

Depending on data size (are we talking "Big data" or XXXL data?) migrate it in chunks. Very often imports fail mid-stream and need to be restarted, which can be very annoying at the least if it takes several hours per import. Also consider potential bottlenecks or systems choking on large files.

2

Keeping the two systems in sync

The whole point of moving to the cloud is to not deal with managing infrastructure. There are plenty of excellent tools and SaaS platforms available which deal with this very problem, so no need to reinvent the wheel. Figuring out which tool is best for your specific needs is beyond the scope of this guide.



One of the biggest decisions to make is how often syncing data to the new cloud solution is needed. Is it every 5 minutes? Or is it enough to sync once every 15 minutes, every hour or maybe just once a day? Make sure to ask stakeholders what they really need, vs just what would be nice to have.

It's always possible to go back and change this, for example going from once a day to once every 15 minutes - just be aware of possible implications such as added costs or API limits, etc.

Also make sure the solution can deal gracefully with outages and failures. If for some reason a sync fails, or a source system is offline, make sure it can pick up from where it left off, ideally for up to 48 hours so it can handle those "being offline all weekend and just realized it Monday morning" situations.

Many tools (like Rivery) can take care of both the initial historic data load and keeping the two systems in sync - providing a single solution for your data migration needs.

In terms of sync mechanics, there are two fundamentally different approaches to reading data from the source database:
1. Connecting to the database and getting data using a Select statement.
2. Reading from a log file that captures changes in the database.

The latter methodology has many advantages and is called Change Data Capture, or CDC for short.

CDC is a methodology of capturing data changes in a source database in order to pass these updates to a target database or data warehouse in near real time. Most CDC solutions incorporate some type of logging system for the source database, and then read the data from the logs in order to make the changes to the target system. Example types of changes that are being tracked are inserts, updates, deletes, and schema changes.

Learn more about the pros & cons of CDC Learn more

3

Validating the data in the new system

Compare the data in system A (the on-prem database) with the data in system B (the cloud solution). Start with high level aggregated data, usually in the form of reports, and then drill down into smaller chunks of data, or even line items as needed.

For example, let's look at ecommerce data. Start with comparing top level metrics for the entire year of 2021. Total sales in dollars, total number of orders, etc. There is probably an existing reporting solution that provides all of this information, so ideally it's simply comparing two sets of reports - one that is using the data from the old system and one that is using the data from the new system.

If there are any discrepancies, looking at more granular data and different metrics will help find what's causing the issue. For example, are there the same number of orders? Same number of customers, etc. Once everything looks good at the top level, validate and compare the data at a more granular level in any case, as averages to totals can hide discrepancies that cancel each other out.

Trying to compare the two systems manually by simply looking at reports usually isn't enough. It's best to create new reports and queries that actually compare the data from the two systems. For example, don't just report the daily sales for both systems, also report the delta in daily sales between system A and system B.

Aside from looking at the data values, it's also important to run some metadata checks (i.e. check that all tables/columns were created in the destination database as well) and run some reports at the SQL level as well. Here are some examples:

General Tests

1 Table Comparison

Goal: Validate the same table names in each schema related to the chosen Data Sources.

Compare type: Metadata.
Query example (MySQL): select TABLE_NAME from INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA = 'XXX'

2 Total Rows

Goal: Validate the same number of rows in each table related to the chosen Data Sources.

Compare type: Raw Data.
Query example (MySQL): select TABLE_NAME, table_rows as TOTAL_ROWS from INFORMATION_SCHEMA.TABLES where TABLE_SCHEMA = 'XXX'

3

Specific Tests

1 Sum Checks

Goal:
Comparison of all the columns of Numeric data type.

Compare type:
Aggregated Data.

2 String Checks

Goal:
Compare String length values for all the columns of String data type.

Compare type:
Aggregated Data.

3 Table Structure

Goal:
Compare column structure for each column in the table.

Compare type:
Metadata.

How long should the two systems be running in parallel?

The short answer is - it depends.
The longer answer is anywhere between two to thirteen months (yes, over a year).

Let's start with thirteen months.
On one end of the spectrum, there are instances where risk tolerance is zero, and the cost of running two databases in parallel is much lower than the downside of having bad data in the new system.

With cloud database migrations for banks, this is often the case. Thirteen months was the time frame chosen to ensure that year end reports, and anything else that was happening during the year was tested.

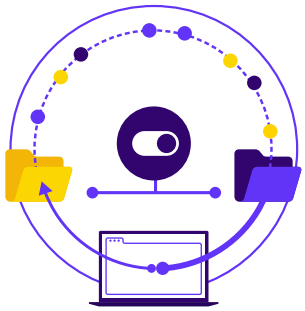
On the other end of the spectrum, if the downside of bad data is "woops", then two months should be enough to test and spot inconsistencies between the two systems.

Another step to take during the validation stage is to measure performance and costs. Having at least two months of both systems running in parallel should provide a good estimate of how performance and monthly costs compare between the two systems, and provide ample time to optimize both if needed.

4

Switching the new system to production

Once the data in the new database has been verified and validated, and that performance and costs are in-line with what's expected, it's time to "flip the switch".



How this is done depends heavily on the specifics of each individual use case, but the two important things are:

1. The switch can be done with zero downtime.
2. It's fairly easy to revert back if needed.

After switching to the new system, it's very important to run yet another round of validation and performance testing, especially if there are now more people using systems that are connected to the new database.

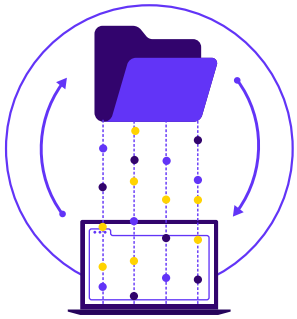
Now is also the time to validate that the old system is still being updated and staying in sync with the new system, in case there is a need to revert.

5

Sunsetting the old system

Congratulations!

While turning off the old database itself should be a fairly simple procedure, now is the time to make sure it's not connected to anywhere in your infrastructure. A good place to check for this is in error logs and monitoring systems.



Once the old system has been shut down, make sure to notify everyone internally, and then pour yourself a drink.

What are some common mistakes to avoid for cloud database migration?

Lack of system specific experience

Don't assume that a MS SQL Server DBA who knows all of the ins and outs of SQL Server, will automatically be able to transfer those skills and knowledge to a cloud data warehouse like Snowflake. This has a knock on effect to performance, cost, et cetera, as well.

Make sure that your DBA or consulting partner fully understands how to optimize the new data warehouse and take advantage of all of its features - this is often called database refactoring.

Refactoring involves completely rebuilding a resource to make more effective use of a cloud-native environment. Here are examples of refactoring different types of data:

Structured data — when migrating to the cloud, many organizations change their data format from traditional SQL tables to NoSQL key-value pairs. This lets them leverage massively scalable, high-performance databases.

Unstructured data — cloud providers offer value-added services that can be used to prepare, process, and extract insight from unstructured data. By applying these services while ingesting the data, organizations can derive additional value from existing assets. For example, instead of just copying video assets to the cloud, an organization can use AI services or video APIs to extract tags, concepts, transcripts, and other data from video content.

Sensitive data — sensitive data can be restructured to make it easier to manage and secure in the cloud. For example, a large table containing both sensitive and non-sensitive data can be broken up into separate tables, or even completely separate databases, to enable stronger control over access and authorization.

Please note: Depending on the complexity of the changes required, it may be easier to lift and shift as is, and then work on refactoring at a later stage.

Not having a dedicated project manager

Using a dedicated project manager to manage the project and everyone involved is always preferred, opposed to a DBA being entirely in charge of the migration project.

The technical aspect of cloud database migration is just one part of the puzzle. There is a need for someone to be on top of communication, documentation, risk management, QA, adherence to timelines, etc.

Doing it on your own

Depending on the scope and risk tolerance of the migration project, this might not be a mistake per-se, but keep in mind that the first time doing something is when most mistakes are made.

Hiring an outside consulting firm or consultant to manage the project and provide guidance is usually a good idea, and if nothing else, get a good second opinion on the existing plan.

Not using the right tool(s) for the job

"If all you know how to use is a hammer, everything looks like a nail"
Doing a Google search for "Cloud Database Migration tools" provides a plethora of results. There is probably more than one tool suitable for your specific needs (like Rivery of course), just make sure not to choose a solution based purely on the tools your internal team are familiar with.

Not setting up clear communications, expectations and responsibilities in advance

Cloud database migrations are complex projects with many moving parts and people involved. Scope creep and missed deadlines are all too common.

Make sure to have a clear plan of what the project entails, which is distributed to all relevant people, and have a good channel of communication. For example, an internal slack channel for Q&A and project updates.

Summary

Cloud database migrations can be long and complex projects, but don't have to be stressful.

Rivery's solutions engineering team has helped hundreds of companies make the jump to a cloud database. We've worked with both small startups and fortune 500 companies, so chances are we can help you as well.

Ready to see Rivery's cloud database migration in action?

Learn more