GlobalLogic®
A Hitachi Group Company

# How to ease the Heebie Jeebies of Migration?

Tags: Platform | Cloud | Migration

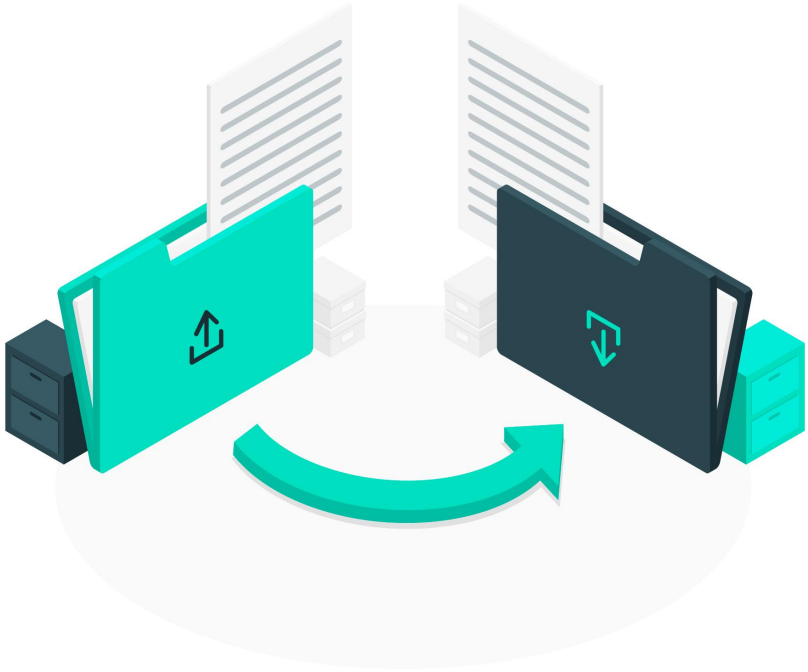

Image Credit

**Author: Sujatha Malik, Principal Architect**

# Table of contents

# Context

The migration in this article is about the final state in which a new system has to reach. Enterprises envision a spanky new system as their dream to-be state and hence begins our journey of Digital Transformation with them. Migration is when the old system has been deprecated and the new system hosts the data of the old system, supports onboarding of new data and is positively accepted by the length and breadth of the targeted audience. Most importantly, migration in this article's context is not limited to data migration, it is the complete system migration.

Many enterprises start with a big bang approach on Digital Transformation and a massive 70% of digital transformations are failures according to Mckinsey. This is a grim figure as one observes and a lot of it is contributed by unsuccessful migration attempts to the new system. Migrations are notorious to have not been planned correctly or mostly overshoot their time targets.

And what are Heebie Jeebies? This means a state of nervous fear or anxiety. As in the movie Madagascar, Maurice had Heebie Jeebies for Alex, the lion. On a similar note, because of inherent agility and planning required for making a new system live, people have Heebie Jeebies for the migration process. Let us read further to know how to ease these Heebie Jeebies. First we have to know what we are dealing with. So, let us understand the different types of migration.

*Fig 1: Maurice with expression of Heebie Jeebies*

# Types of Migration

## The Conventional data migration

The conventional data migration is when data is exported from source systems into flat files and a program is written to read this file and load the data into the target system. A very siloed kind of approach and can be applied when there is not much difference between source and target data schema. And also when the load of data to be migrated is relatively small.

An online pet pharmacy enterprise was moving away from its existing pharmacy vendor to a new one. All the roads were laid for the new system with a switch in place to be activated once the new system is hydrated with data. We had around 90,000 prescriptions to be migrated from the old pharmacy vendor to a new vendor database during the go live of the new pharmacy vendor. This was not a big packet to be migrated. Hence we decided to use the conventional data migration.

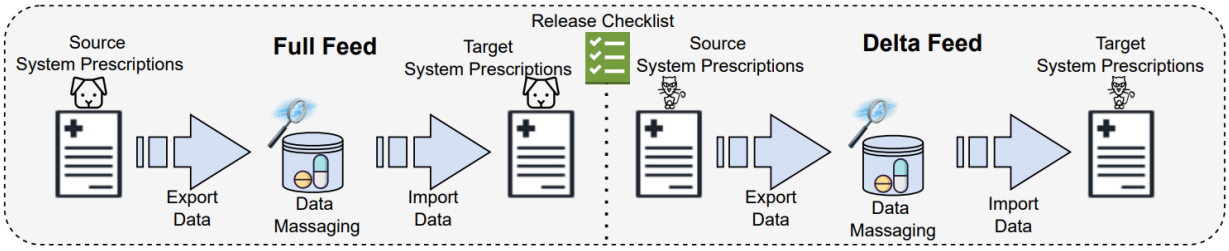


*Fig 2: A Conventional data migration of prescriptions*

The data was exported from the old vendor and passed on to our team. We massaged the information and transformed it into a file which can be easily imported into the new vendor's system. The entire process was rehearsed and prepared for a couple of months ahead. This was planned with precision in a way that full feed and delta feed timings were curated. A release checklist was created which helped us track each and every task in this migration process. And this entire process was so smooth without any downtime for the end users of the online pet pharmacy store.

## The Custom migration

There may be a very unique scenario where the migration process is so complex that it needs its own dedicated system. This software system which is incarnated for migration has its own life cycle and will

be sunset once it attains its glory. This is an extreme measure one needs to take if the problem is so big that to break it down, we need such a bespoke solution.

Our esteemed client in the online travel industry is prepping for one such mega migration. The problem is so unique and the load of data is humongous that a decision was made to migrate using a very bespoke service built with one single purpose - to stage the data and finally push it to the new system when the users have to be activated. The number of records to be migrated are in the range of 250K Millions.

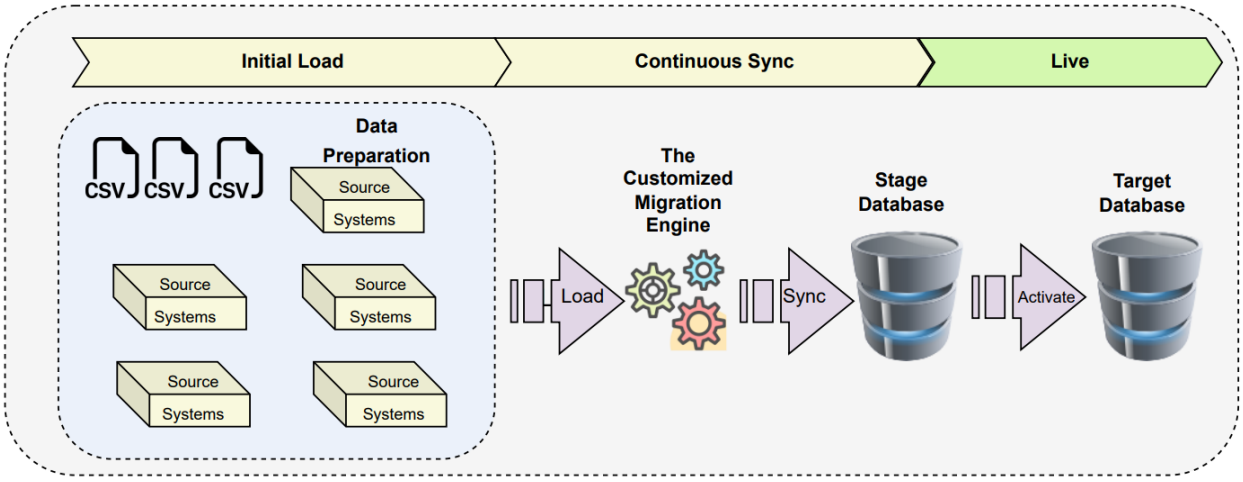


*Fig 3: A Customised Migration Process*

The disparate source systems are one of the primary reasons for such a unique migration solution to be applied. This bespoke service runs like an engine and ingests all the different sources of data and prepares it for the stage database. Then onwards the step to the new system is a smooth transition which can be executed on the runtime when activation is requested. This custom well “oiled” engine will pave the path for moving to a better world for our client.

## Data migration aided with technology

We have seen the Conventional data migration. Now imagine if you were to execute this conventional data migration with all automation aided using modern tech stack, what can be better? We let tools take care of our errors, retries and deployments and much more. You may think migration will be the easiest with this. But there is a catch. The planning and agility needs a hawk eye which tools may help in monitoring but the action has to be executed by the artist.

There are few cloud services that help us in automating these steps of migration. I am being partial towards AWS PaaS first approach here but there are equivalent tools in other primary cloud providers which are equally competitive.

The components in such a migration system are -

- **AWS Glue** - AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data.
- **AWS S3** - AWS Simple storage service can be used to store all ETL script and log storage.
- **AWS Secret Manager** - AWS Secret manager securely encrypts and manages the secrets. It shall be used to manage database credentials.
- **AWS Cloudwatch** - CloudWatch Events Rule shall be used to trigger the ETL script in a scheduled manner. CloudWatch Logs shall be used for monitoring Glue logs.
- **AWS DMS** - AWS Database Migration Service (AWS DMS) is a managed migration and replication service that helps move your database and analytics workloads to AWS quickly, securely, and with minimal downtime and zero data loss.

Using these services, let us see how can we achieve the migration process -

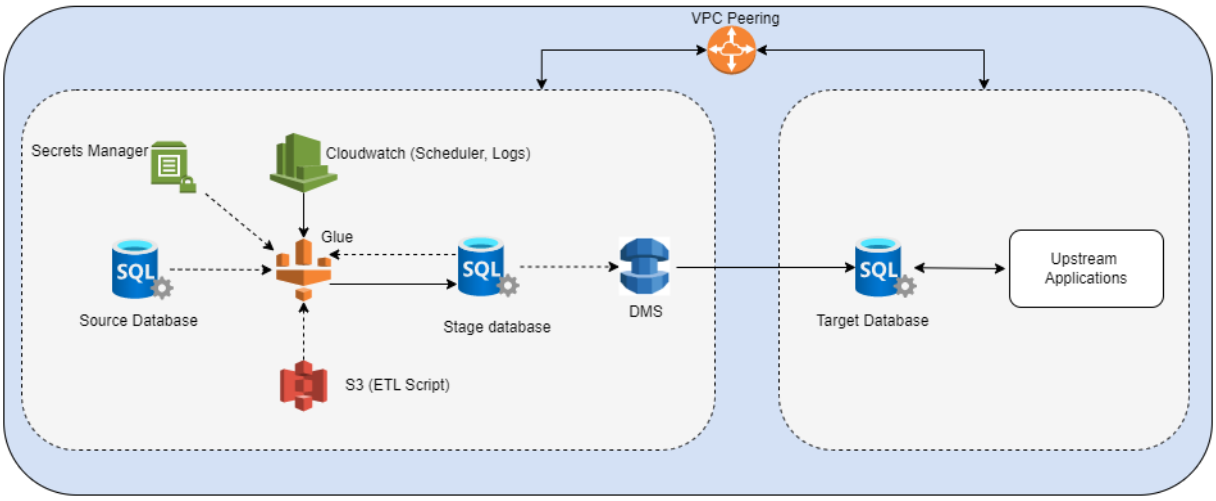


*Fig 4: Data migration aided with technology*

This is a very simple workflow for source to target systems using AWS Glue. The two AWS accounts have to be in VPC peering for this execution. Please note, there may be cases when the client infrastructure team does allow such access and hence it is a smart move to run this through the infrastructure team. The data is transformed and placed in the stage database. Once data is ready to be activated it is moved to the target system using DMS.

All these tools do definitely help us cut down our development efforts but remember we should know how to make maximum use of the tools. And this is the easy part of the story, the difficult part is when we validate the data which is migrated.

## On Prem to Cloud migration

This is the mother of all migrations. The classic vanilla case of enterprise lifting from on premise servers and shifting to cloud servers. The entire execution itself is taken care of by many out of the box

solutions provided by cloud vendors. One such example is the AWS Migration Acceleration Program (MAP) which is a comprehensive and proven cloud migration program based on our experience migrating thousands of enterprise customers to the cloud. MAP provides tools that reduce costs, automate execution, and accelerate results.

We partnered with one of the leaders in screening and compliance management solutions for their transformation effort. One of the projects executed for this partner was Data Migration and 2-Way Sync. The objective of this solution was to create a performant two way synchronisation strategy that is required to support both the current On-Premises solution features and the migrated ones to a new service-oriented solution on Azure. Additionally, this solution has to be able to handle copious amounts of binary content.

Firstly, let us take a look at the tech stack used for this migration.

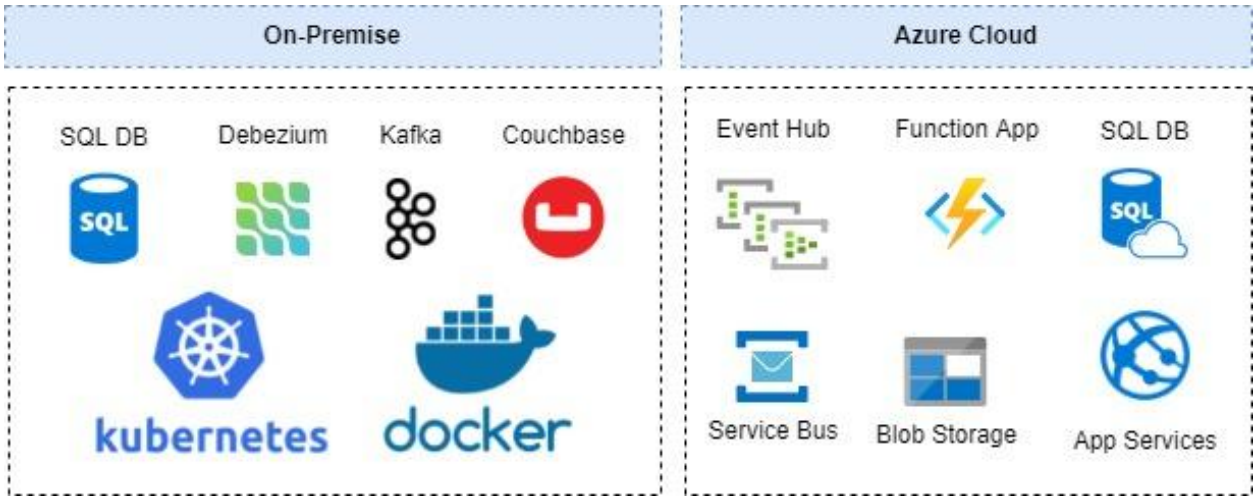


*Fig 5: Tech stack used for On-Prem to Cloud Migration*

Our solution consisted on these components -

- **ACL** - Legacy component which figures out the changes in the on-prem database and raises events for them to the cloud.
- **Upstream Components** - Cloud-based components that filter, transform and persist the changes to the entity's home domain in the cloud and raise replication events for others to react to, if required.
- **Replication Components** - Cloud-based components that receive the replication events to save the data or perform certain actions on the same, if required.
- **MassTransit** - In case, cloud-based changes have to be synced back to the on-prem database, then this tool is used to read all the events and forward it to the downstream components to synchronise the changes.

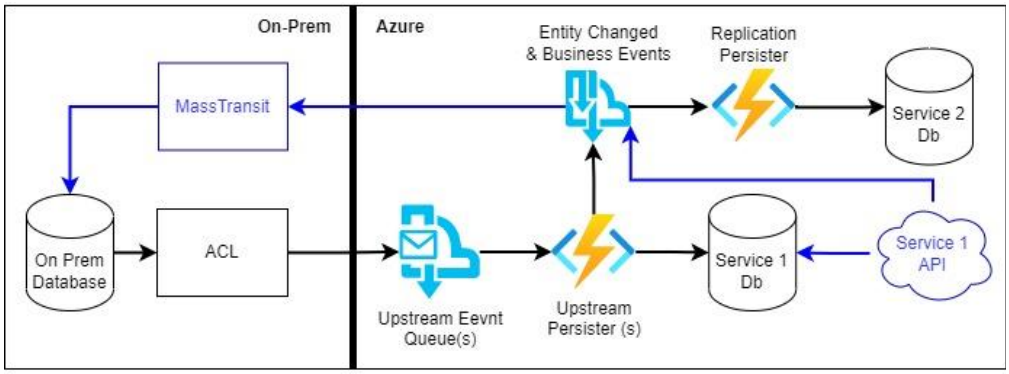


*Fig 6: On-Prem to Azure Cloud Migration*

The two way sync was achieved using the following key features of our product -

- A table-to-table data synchronisation solution from on-premise database to a cloud database or vice versa via event-driven architecture.
- If changes happened on-prem, a change capture service, figures the changes and raises events, which get synced to home domain and raise notifications for other domains to synchronise the same, if required.
- If changes happen on the cloud, then they are pushed to the on-prem's data replication service via events.

Much can be covered more in this type of migration - on premise to cloud.

And as we read this, think tanks are adding in more different ways, for example, using CodeGPT, to perform migrations. But in the interest of moving towards the actual subject matter, let us understand what are some different ways which can ease your Heebie Jeebies.

# How To Tame the Migration Dragon

This section is your lifesaver - the go to cheat sheet for how to be successful during the migration process.

## Plan shorter and early test cycles

Similar to how it is important to integrate and start testing early in microservices architecture, get the data to the target system using the migration process early in the test cycle. Include as many testing cycles as possible, which will give better results to improve the migration process. Our recommendation is to have five or more testing cycles.

And it is very crucial that these test cycles are executed in near real time production-like environments with data which is very similar to the production environment. Use morphing tools to place production data into a stage environment once the data is cleaned of sensitive information.

## Plan for Validation Strategy

The validation of the migrated data needs to be exhaustive. Leave no man behind as this may lead to financial losses or even worse losing customer base due to bad experience post migration. Here is one exemplary set of validation steps for the post migration scenario.

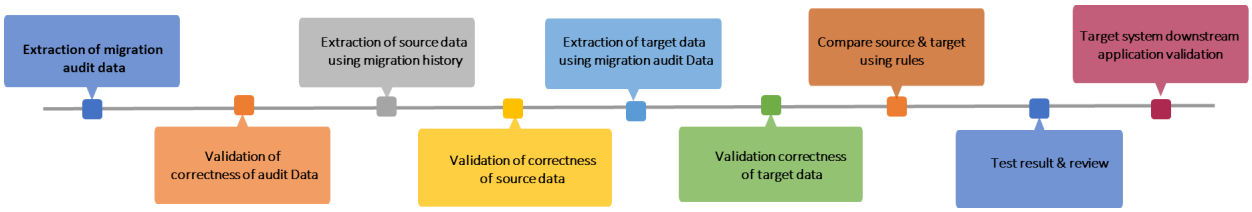


*Fig 7: Exemplary validation strategy*

## Start with Beta users

Select a cohort of Alpha and Beta users who will pilot with the migrated data. In the production systems, this step always helps in minimising the risks involved. Choose the hand picked users of Alpha and Beta cohort which will give a levy to relax more during the live data migration. Alpha users will be a very small subset - maybe a hundred or so. Beta will be a slightly larger set - can be a few thousands of users and finally we are good to go with a complete data set of live users.

## Prepare for poison pills

Poison pills have to be planned for right from the beginning. A poison pill (in the context of Kafka) is a record that has been produced to a Kafka topic and always fails when consumed, no matter how many times it is attempted. There is a reason for message schemas to evolve and if this upgrade is not backward compatible, there are chances of poison pills to occur. Checking if topics have poison pills in production is a prudent step to preempt any last minute show stoppers. Here is one workflow which helps you tackle the poison pills.

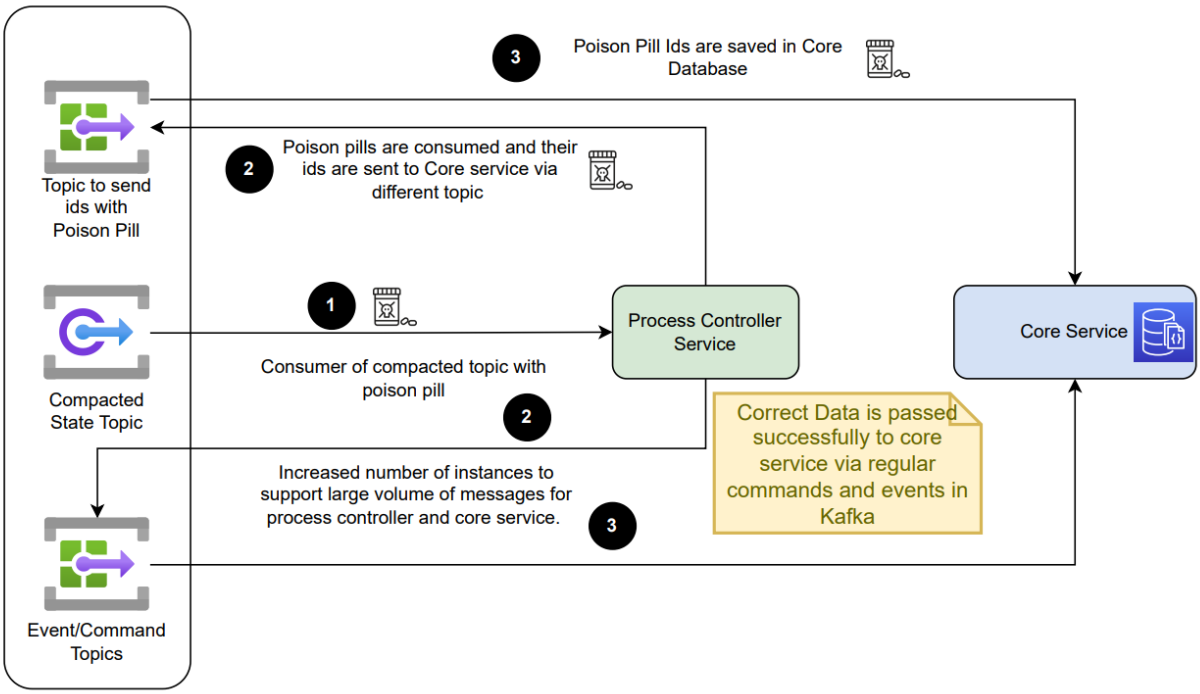


*Fig 8:A Sample Workflow to address poison pills*

## Every migration is unique and solved separately

We have discussed a plethora of migration types. All of them have their own unique ways of solving them. So, it is advisable to visit the drawing board at the earliest to analyse the type of migration which needs to be applied for the use case at hand.

## Plan for the downtime

If the target system allows for downtime, go for it. It is always good to have time on your side when we are at the mission critical stage of transition.

## The rollback strategy

Discuss the rollback strategy with clients to set the right expectations. This brings in the right perspective what is needed for the rollback. Mock-run the rollback strategy and do not leave it for the unknown as you may never know this may be the last twig to save the situation.

## Ask for Paid Help, if Available

It is always prudent to use paid support help if you have the levy to avail it. For instance, our client had a licensed version of MongoDB which included a few support hours. We utilised this to the maximum for improving the performance of our system and also the migration scripts.
Many times, such support brings in a totally different dimension with respect to technology since it is their home ground. They are well aware of the Easter eggs and the Rabbit holes which will help during the migration process.

## Early Reviews

Do not feel shy to reach out and make sure the migration architecture is reviewed from the client as well as internal review boards. We have run into challenges when the actual rubber has hit the road and the client's infra team put their foot down saying that such an migration arrangement is not allowed due to their policies. That was a real eye opener and we had to go back to the drawing boards to make the changes needed.

## And Finally, No guts, no glory!

Having talked about all the gotchas and potholes, migration is not a feat for the weak hearted. Be ready to face the unprecedented and think on the foot all the while. Use the hard facts to identify the MVP of data which must absolutely be migrated and target this cohort of data migrated first. This will buy you more time for fixing the road blocks.

# References


- https://www.mckinsey.com/capabilities/transformation/our-insights/perspectives-on-transformation
- Octane - https://sites.google.com/globallogic.com/octane/about