developers.mews.com

Are we migrating to microservices and should you?

Josef Starýchfojtů

6-8 minutes

“Should we migrate from monolith to microservices?” That’s the inevitable question most scaling companies ask these days, including Mews. Looking at the state of the art, one might rephrase Dan Ariely’s quote about Big Data to microservices.

Microservices are like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it.

So… should we?

Context

Our engineering team currently consists of 60+ people, out of which 25 are backend engineers. Our plan is to grow this team to about 90 people this year and continue that pace going forward.

This team is made up of 10+ cross-functional teams that maintain a monolithic application written in C# and running on Azure app services with Azure SQL for main data storage. Mews is used all over the world, by almost 2000 hotels, with usual request throughput around 10k rpm.

The monolith contains 4 main products: a property management system for core hotel operations, public API used by 500+

companies, a guest portal for self-service, and a booking engine.

Why?

So, why ask such a question, apart from the fact that everybody else is talking about it?

Technical indicators

From a technical point of view, hosting multiple products on the same machine and database causes issues sometimes, as a problem in one of them might result in problems in all. Our database is at 25% of the maximal size offered by Azure and thus brings up natural questions about how to scale it. Also, due to its size, the build of our solution is not the fastest.

There are various solutions to these problems and microservices are often one of them.

Organizational indicators

A lot of people often state that the biggest (and sometimes the only) reason to go for microservices is to solve organizational problems, not technical. The first one we saw at Mews is ownership, especially infrastructure ownership. What if the database is down or struggling? There must be a single team which finds the root cause and then notifies the team which caused it. Thus, Product teams are not the end-to-end owner of their scope.

Another problematic aspect is innovation and consistency. Since it is a single codebase, it is desirable, but also difficult, to keep it consistent and drive innovation at the same time. Meaning, when a new concept is introduced, it should be rolled out everywhere. However, the innovations of one team might not match the priorities of another.

Research

Migrating to microservices is not only a demanding task on its own, but also brings many new problems. With lack of experience in this area, it's hard to even estimate the value and effort of such a change, let alone plan the migration or maintain it afterwards. Therefore, as one of the main goals for the previous quarter, we decided to dig deep into it.

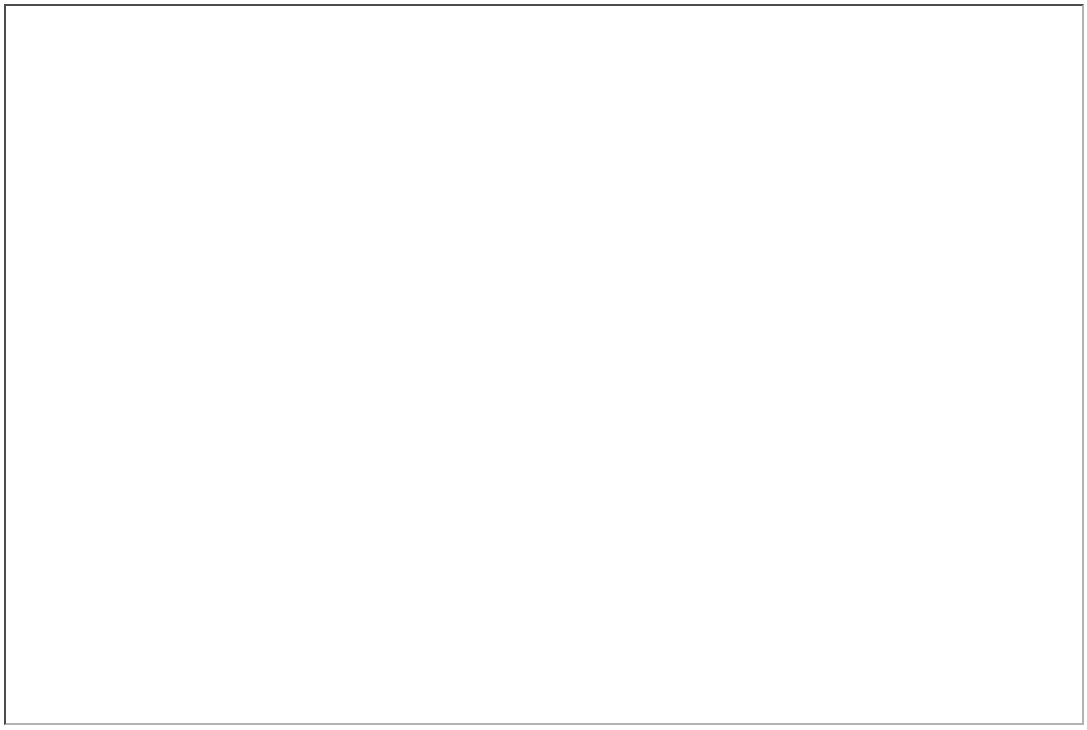
We felt that the best way to do that was simply to ask people who had already done it. We use PlatoHQ for mentoring our leaders, where you can book mentors for 30 minutes from known companies like Facebook, Spotify, Netflix, Airbnb etc. So, I booked 10 sessions and asked about their experience.

Interviews

It is hard to sum up all the interviews, but let me present the most important and repeated themes.

- The biggest indicator should be a **problem with delivery speed**. Does your team struggle to ship features frequently? Do you see a lot of cross-team dependencies?

- The vast majority of them did **not recommend going with microservices, unless you really have to**.
- **Distributed systems are very complex**, do not underestimate that. Its problems often appear much sooner than one might expect.
- **Getting the boundaries right is very hard, but necessary**. Start with a Modulith (modular monolith) instead—it's a first step anyway.
- Airbnb migrated when they had around 700+ engineers.
- **Monoliths are great for product teams;** microservices are good for heavy backend components. This is the case in Facebook and many other companies.
- You should be very advanced in tooling, CI/CD, and testing.
- **Macroservices over microservices**. More than one service, but well-sized ones. This is recommended by many teams from Airbnb and Uber to GitHub.
- Microservices on their own do not solve ownership issues, but they might help.
- Can't you solve your problems within a monolith? Is it worth migrating?
- Architecture of the system gets easier for the team members, but much harder for cross-team architects.
- Some companies who rewrote to microservices benefited more from the rewrite itself rather than the benefits of microservices.



Summary: Tackling problems within monoliths

So, are we migrating to microservices or not? No… and yes. We are definitely not planning to migrate in the near future. Our delivery process and overall development is just fine and we don't have any major struggles. We're just not there yet. But that doesn't mean we won't tackle the problems we have or head in that direction.

First, we are moving towards a more modular monolith. Building the system over the last eight years has given us good knowledge about the boundaries of several domains. Now it's time we put it in practice. That itself should solve a lot of problems and prepare us for a possible transition to more services.

Regarding ownership, there are several improvements we can make. For just one example, we annotated each transaction in our system with an owning team, enabling us to route incidents or build dashboards for the right group of people.

Technically, we started to use Cosmos DB in production for semi-structured and big data, like event logs. That buys us a lot of time before we hit any limits or major problems in our SQL database.

Moreover, we can always shard the data. Solving infrastructure ownership is hard, but doable. We are simply horizontally scaling a monolith. That means using an Application Gateway to route requests to specific instances based on the target endpoints, reflecting the product boundaries. This might be inefficient at large scale, but, again, we're not there yet.

With our current pace of growth, we know the day may come when we really move towards more services—and we already have some good candidates. Even though that day is not today for us, it might be for you! Let us know about your journey. 🙂