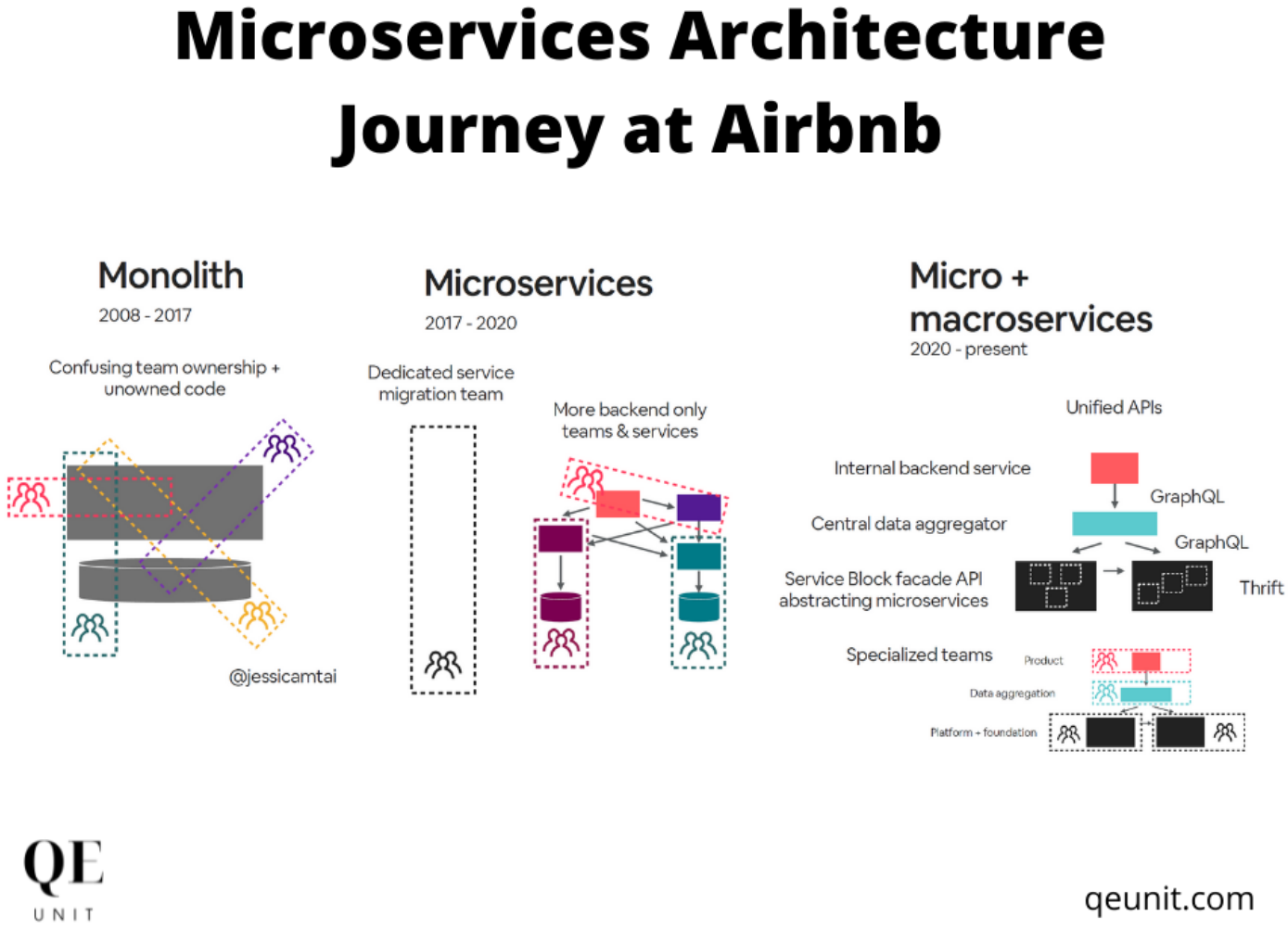
Title: Airbnb's Microservices Architecture Journey To Quality Engineering | QE Unit URL: https://medium.com/qe-unit/airbnbs-microservices-architecture-journey-to-quality-engineering-d5a490e6ba4f Author: Antoine Craske Tags: cloud



Achieving balance is an endless beginning.

An equilibrium even harder to maintain when the business depends on software quality and speed to survive.

Companies have the challenge of continuously delivering Quality at Speed software , constraining the software lifecycle with Quality Engineering.

Airbnb faced numerous challenges while accelerating and scaling its value proposition,

especially in the evolution of its information systems.

This article shares Airbnb's architecture iterative journey to Quality Engineering with practical takeaways. References are available at the end of the article.

Follow the QE Unit for more Quality Engineering from the community.

From Monolith to Microservices, nothing changes

Airbnb initiated like many software companies with a few motivated engineers to build a minimum viable product to reach their market.

The product started with a Monolith architecture inside a Monorepo (access the full Airbnb's Monorepo Journey here). Airbnb grew from 2008 in this model, with features completed by small teams with limited dependencies.

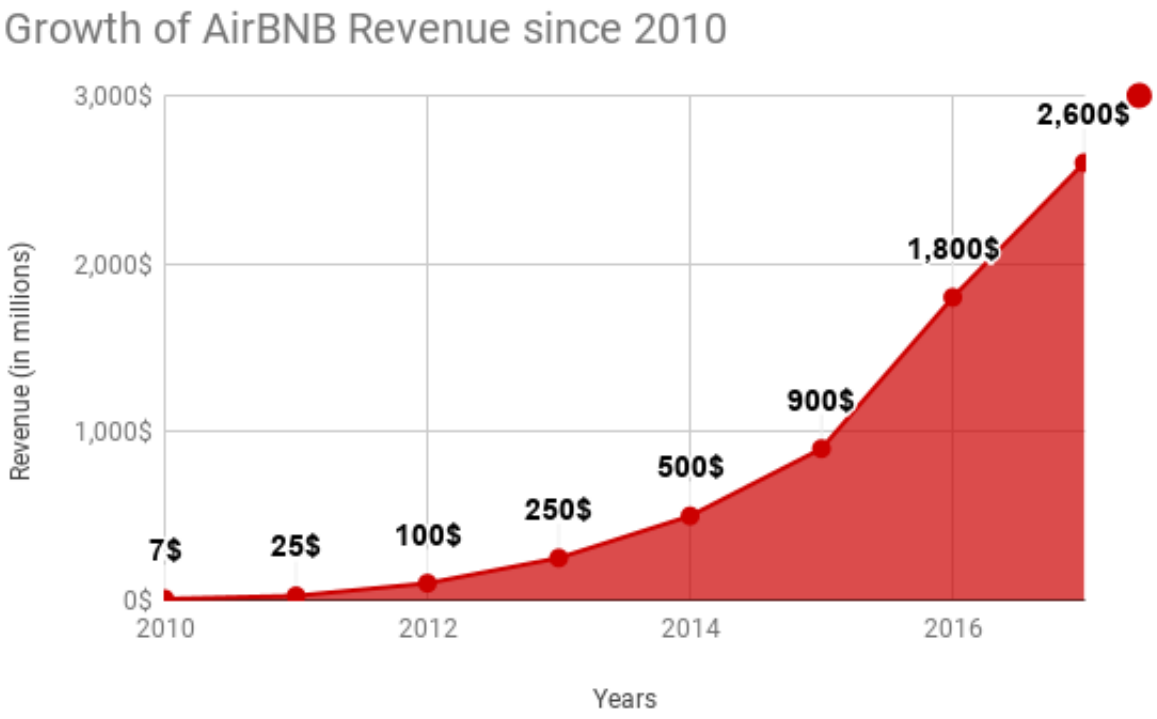


Figure 1: Airbnb revenues were growing up to $2.6B in 2017.

But in 2017, the centralized architecture reached its limits:

- **Velocity of software changes decrease**
- **Parallel evolution faces limiting factors**
- **Component ownership is confused.**

Airbnb decided to embrace Microservices.

The existing Monorepo is split between front and back-office (respectively _Hyperloop _for Speed and *Treehouse* for stability). Microservices appear in each repo, while a dedicated service migration team is responsible for the transition of components.

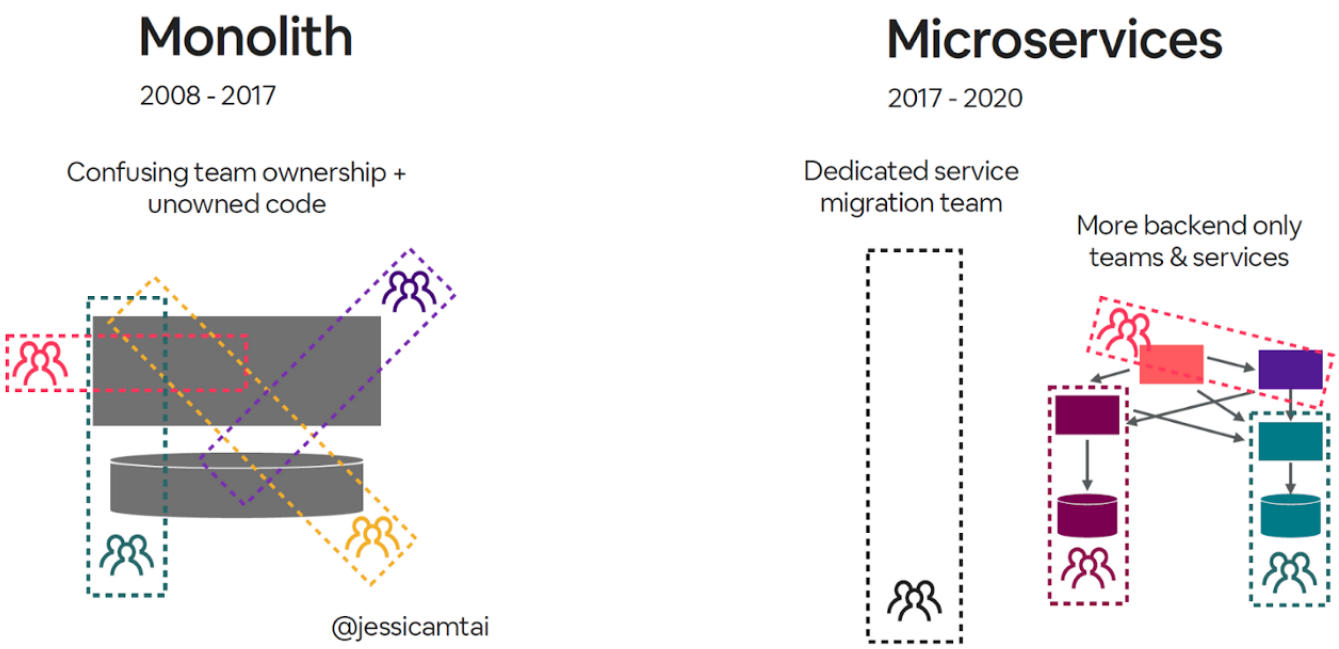


Figure 2: The architecture evolution from Monolith to Microservices at Airbnb.

3 years later in 2020, the business is going to reach $5B revenues (until the COVID hit). Teams grew and continue to migrate services. But still, the original issue is back again: features require changes on multiple services and different teams.

Driven by their architecture mantra “we cannot impact business growth”, engineering teams step back to find sustainable solutions to these recurring problems.

The challenges of evolving Airbnb’s architecture

Airbnb has the challenge to implement architecture solutions solving today’s problems that also support tomorrow’s scale; that’s Quality Engineering problem-solving.

The company relies on an incremental and iterative process:

1. Define the problem(s)
2. Find solution(s) to improve

3. Make the solution used
4. Increase the solution adoption
5. Solve the solution's scaling challenge

This methodology enables to separate the concerns for each problem, find structuring solutions and scale them later on, only if working on a small scope and needed.

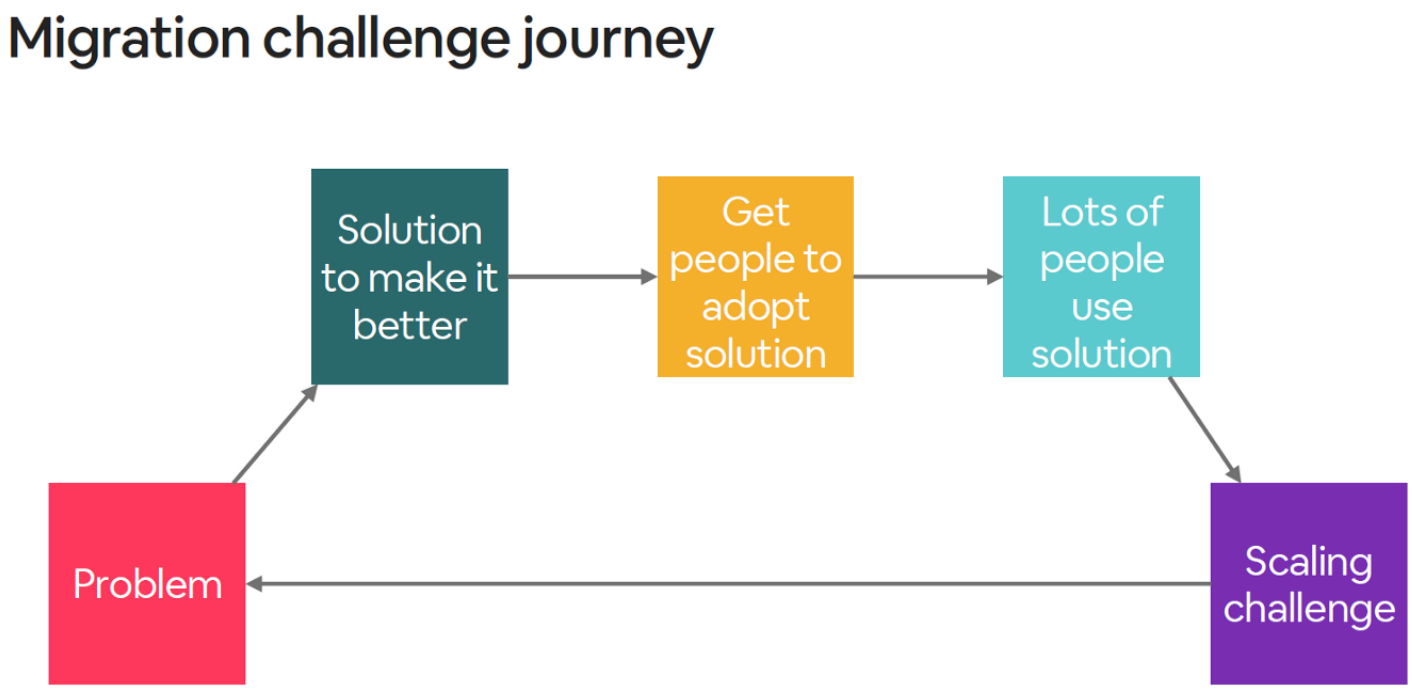


Figure 3: The summary of the architecture migration process at Airbnb.

Airbnb went through this process implementing the following practices:

1. **Provide infrastructure as code to improve developer's productivity**
2. **Clarify ownership and improve with tooling & observability**
3. **Define a new architecture supported by organization & methods**
4. **Lead a deprecation working group to accelerate the migration**

Let's see how each of these problems were solved.

1. Provide infrastructure as code to improve developer's productivity

When business change capability depends on software, slow and faulty software development

directly impacts the company's competitiveness.

With experience in the Microservices architecture, Airbnb had invested in automation and tooling. But faster cycles were required in an evolving set of technologies.

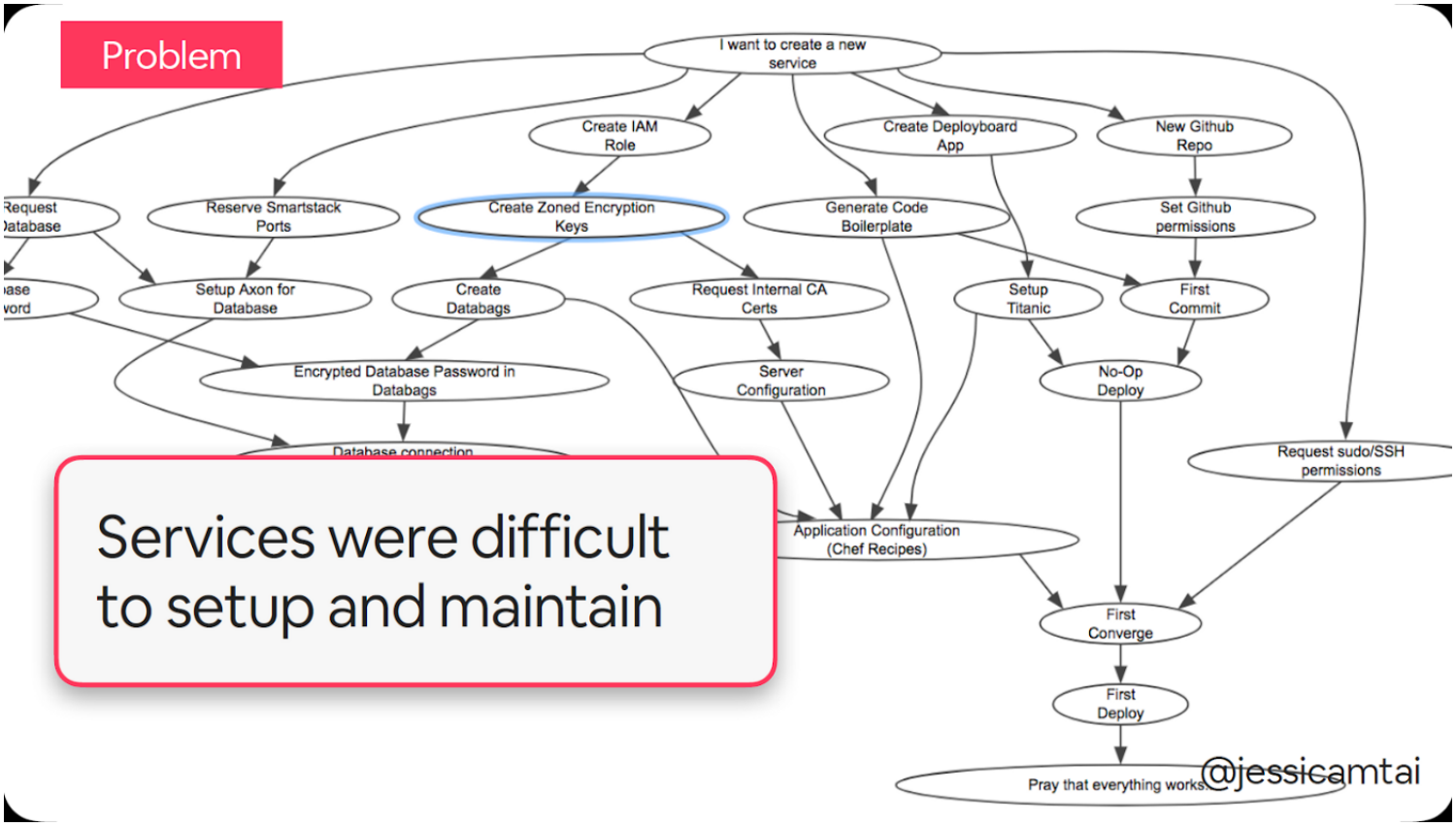


Figure 4: A number of interdependent actions were required for services at Airbnb.

The pragmatic solution is to invest in infrastructure-as-code in a single repository, enabling to gradually increase the adoption of services in parallel.

When more services are operated, the scaling challenge of understanding such complexity arises.

2. Clarify ownership and improve with tooling & observability

There were so many Microservices tied together; even small changes led to interdependent changes and impacts, complex to master.

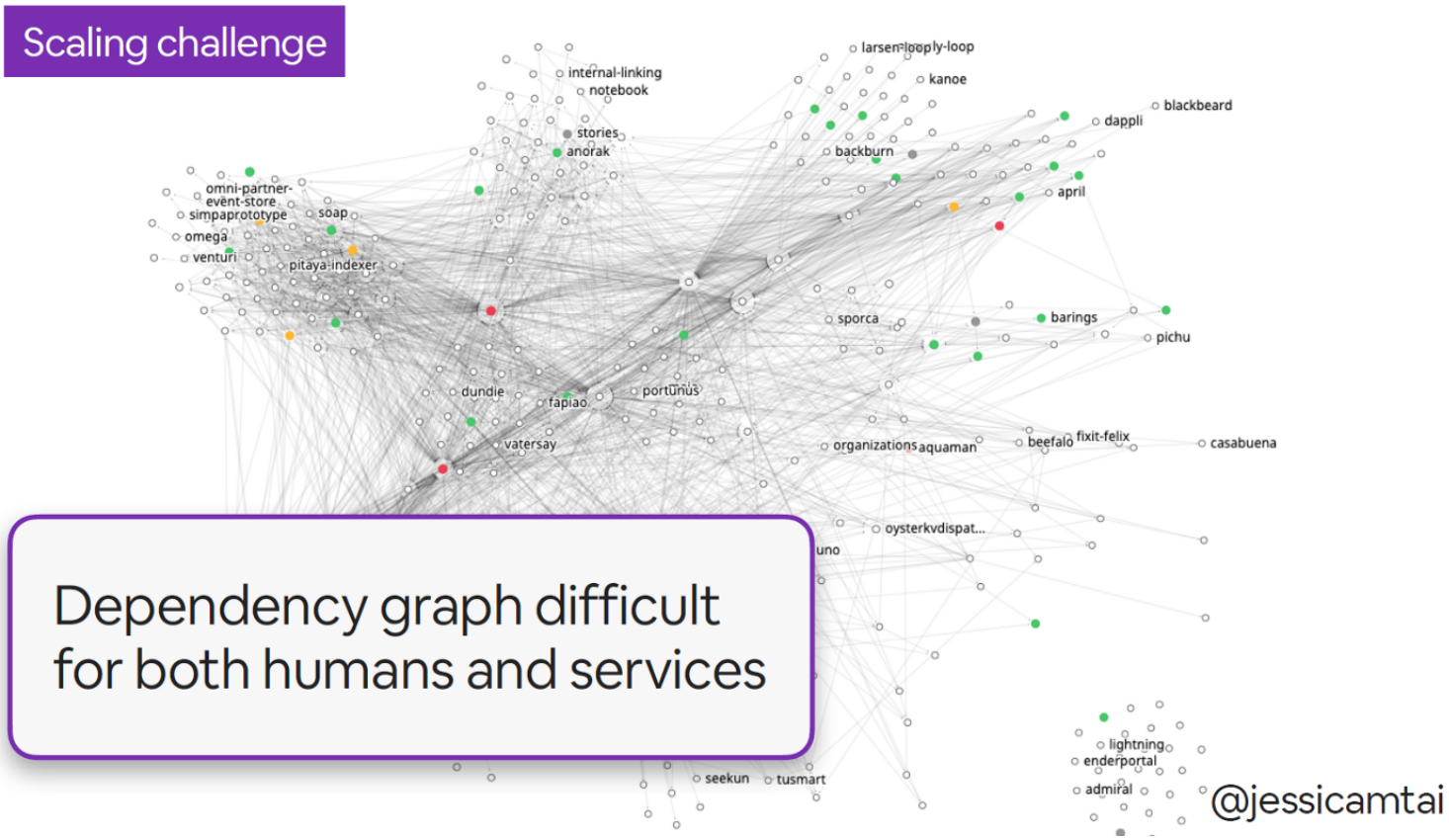


Figure 5: The dependency graph shows the complexity of services.

Airbnb invest in productivity improvements focusing on supporting the new architecture on 3 areas:

- **Ownership**
- **Observability**
- **Tooling**.

Ownership

Airbnb deploys “Scry Ownership”, an application master of the technical component’s ownership data such as owners, maintainers, communication channels.

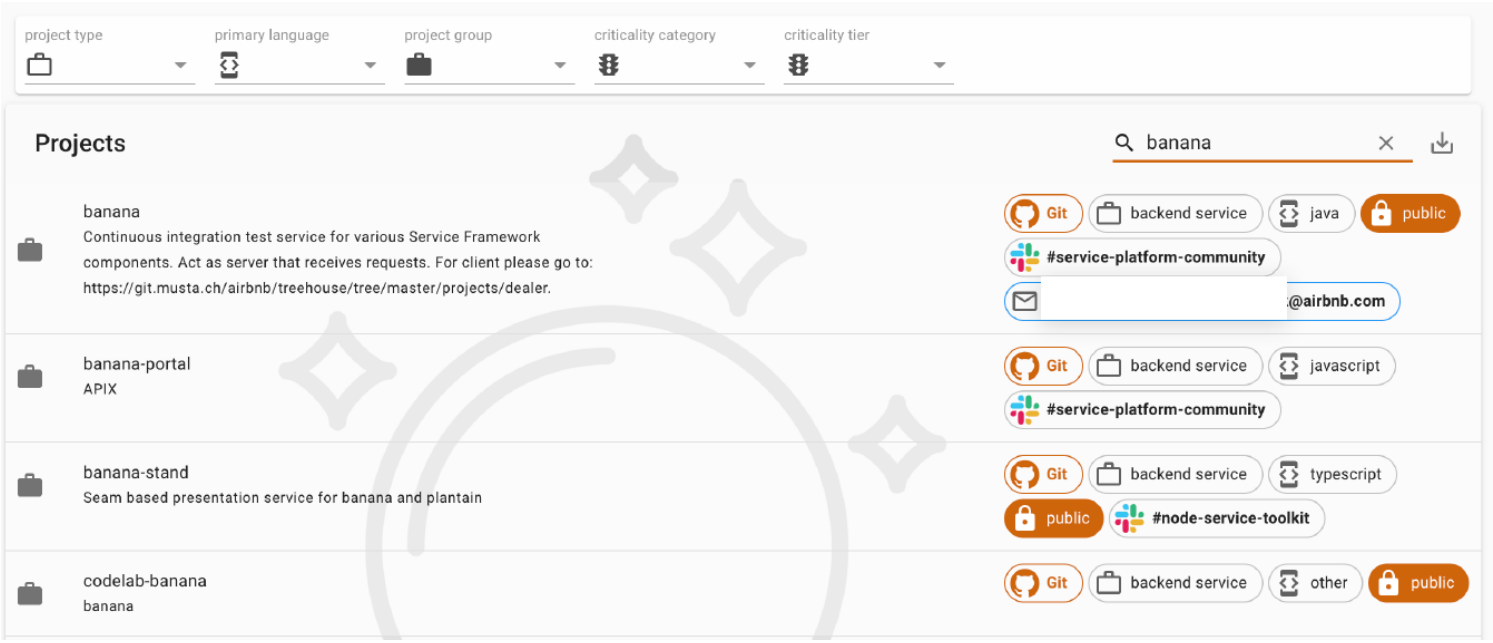


Figure 6: Airbnb ownership application for ownership metadata.

Observability

A series of observability dashboards are set up to systematically review practices implemented. Here's an example of the dashboard to track ownership being set:

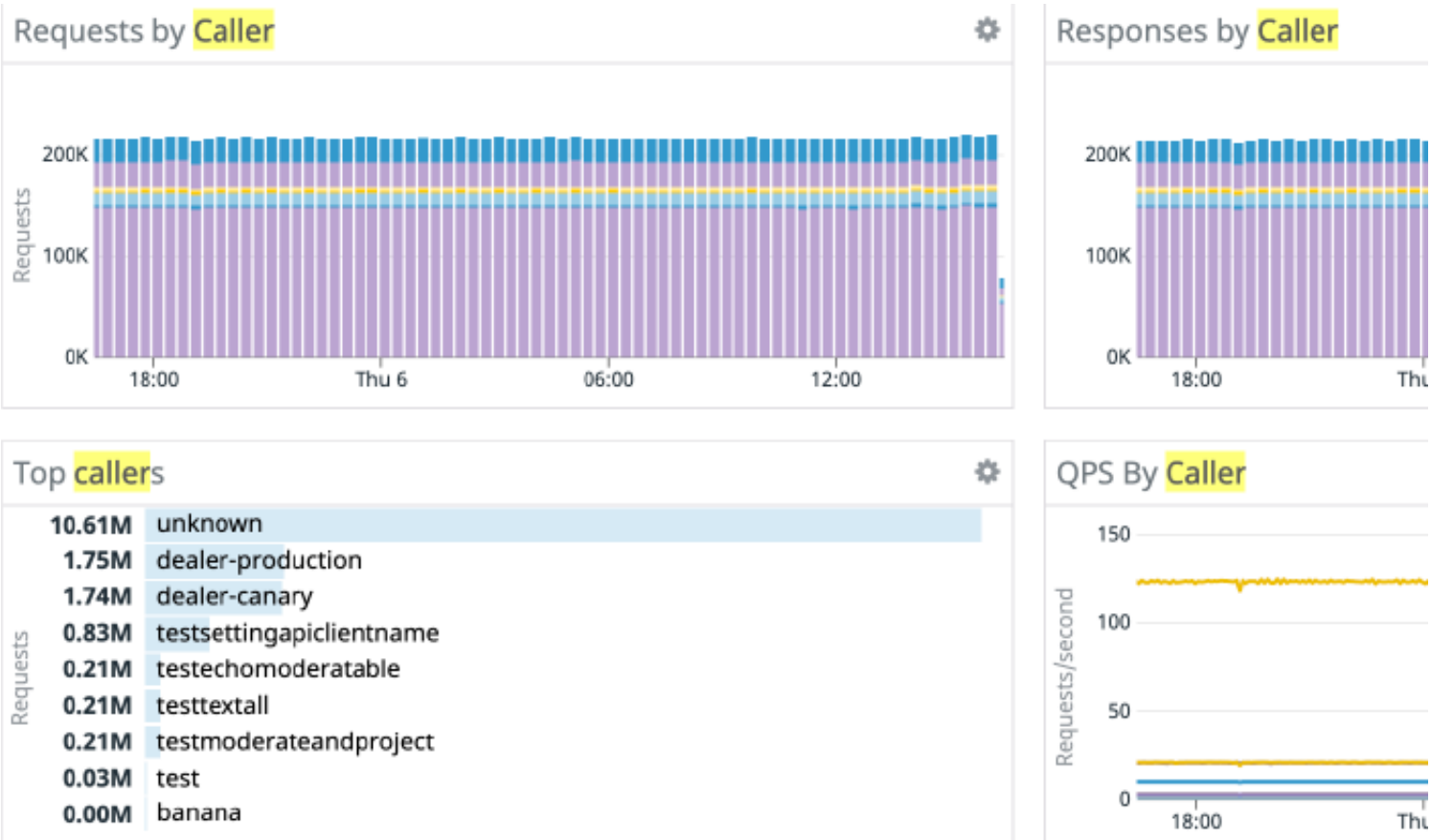


Figure 7: Airbnb observability dashboard for services ownership.

Tooling

A challenge remains in the data space. Be-spoke Thrift schemas are useful serializers and data descriptors, but they require other components to retrieve data in products.

Airbnb's take the advantage of GraphQL to build a unified data access layer that directly provides querying capabilities.

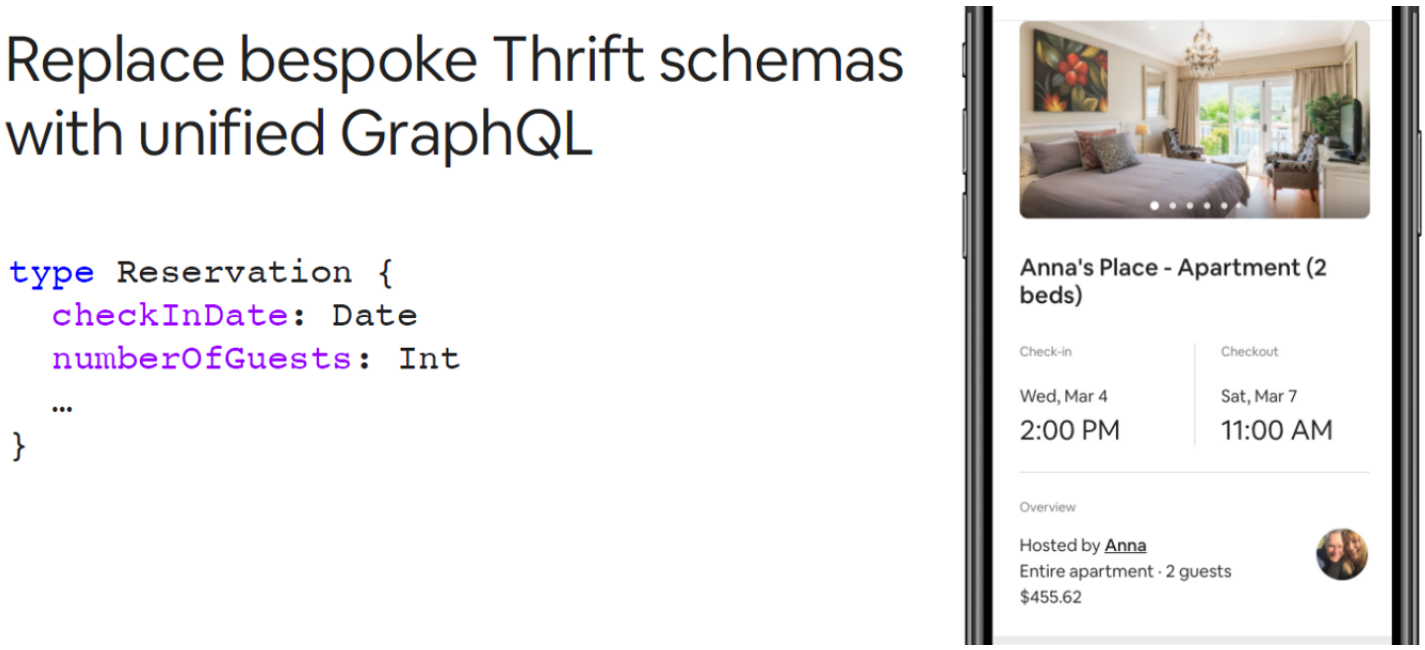


Figure 8: Airbnb unified access layers with GraphQL.

Code generation is the last piece to improve developer productivity. With more technologies, templates are provided for standard components of each layer.

Data access is even embedded by design with code annotation clarifying the data type and access classification directly in the code.

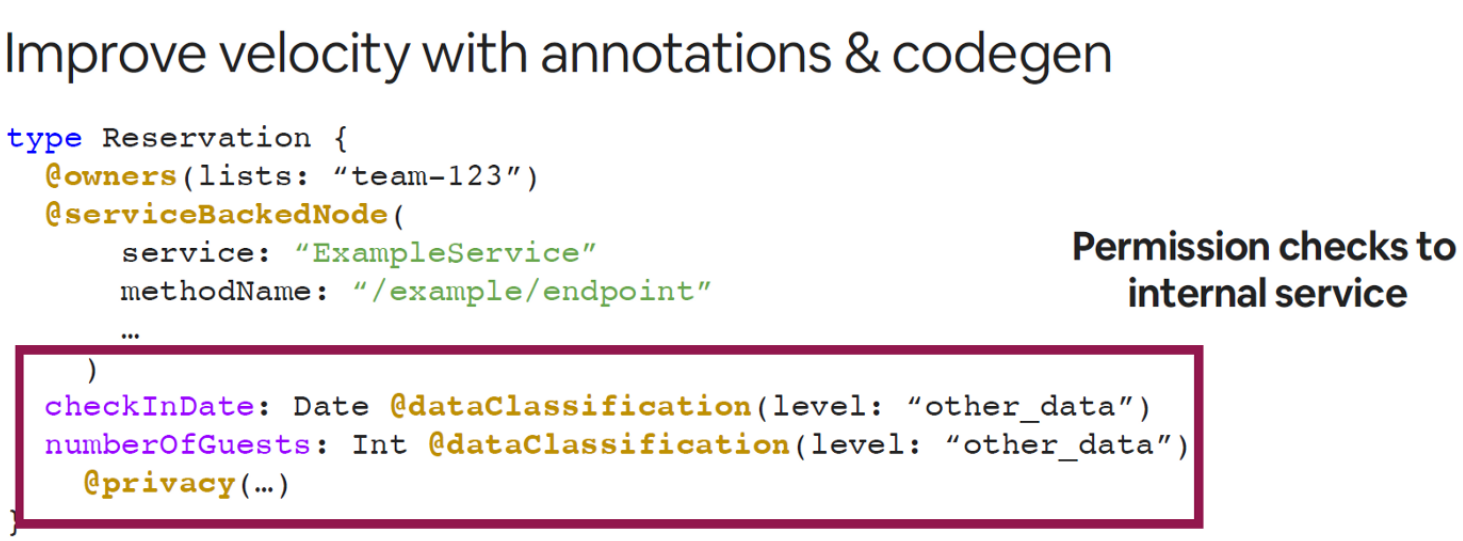


Figure 9: Airbnb code generation tooling embedding data access.

When the stars seem aligned, another problem of velocity appears. This time, the central data aggregator components become a limiting factor.

3. Define a new architecture supported by organization & methods

The central data aggregator has too slow build and deploy times for the teams to iterate on time.

Even with productivity improvements, the structural complexity accumulated is too high to be handled in central components. A new organization is required.

Entropy is the natural tendency of a system to become more complex over-time, requiring supporting and counter-forces to equilibrate the ecosystem.

Quality Engineering forces enter in action with the domains of *Architecture*, _Organization _and *Methods*.

Architecture to support the growth

It was too complex; teams had to perform slow and costly impacts analysis over different domains, coordinate multiple teams, and correct side-effects bugs.

Let's analyze the *"Microservices-only architecture"* cascading issue tree:

- The complexity is overly distributed in fine-grained services

- These fine-grained Microservices lack stable points of collaboration
- The missing glue ends up distributed among components and teams
- Even small changes tend to result in mixed impacts.

The core problem is the lack of urbanization, resulting in a lack of modularity and separation of concerns for both Monolith or Microservices architectures.

Airbnb embraces a new architecture style *Micro+macroservices*:

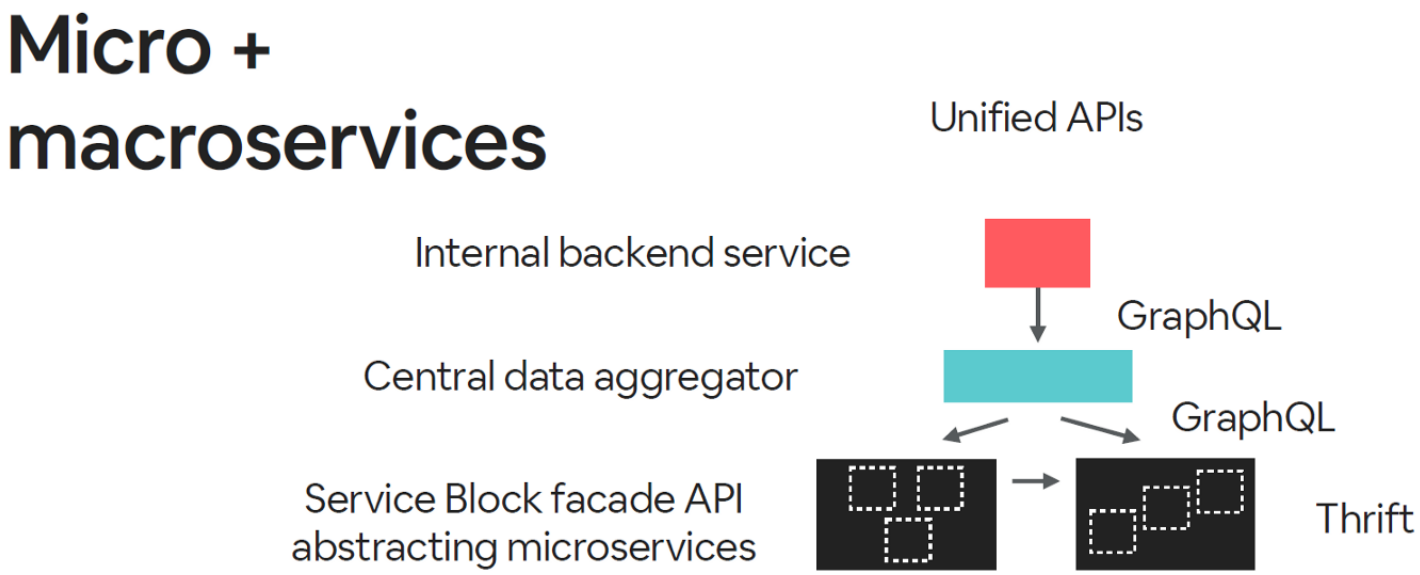


Figure 10: Airbnb architecture applies separation of concerns per service.

In such architecture, each type of complexity is distributed is a clear layer:

- **Unified APIs** are Microservices supporting quick product iterations
- **Central data aggregator** are stable monolith's glue with tight coupling
- **Service block facade APIs** abstract Microservices providing entity blocks.

Read this article for more information about the value of Architecture for Quality at Speed.

Organization alignment support the new architecture

An organization designed to support the business objectives can be a game-changer in transformation. The alignment was crucial for Airbnb to accelerate.

The organization is aligned on the evolving architecture with product teams working on top of Unified APIs, data aggregation teams on the central data layer (*i.e.* "the glue"*)*, and domain

platform teams (reservation, user) for each data facet.

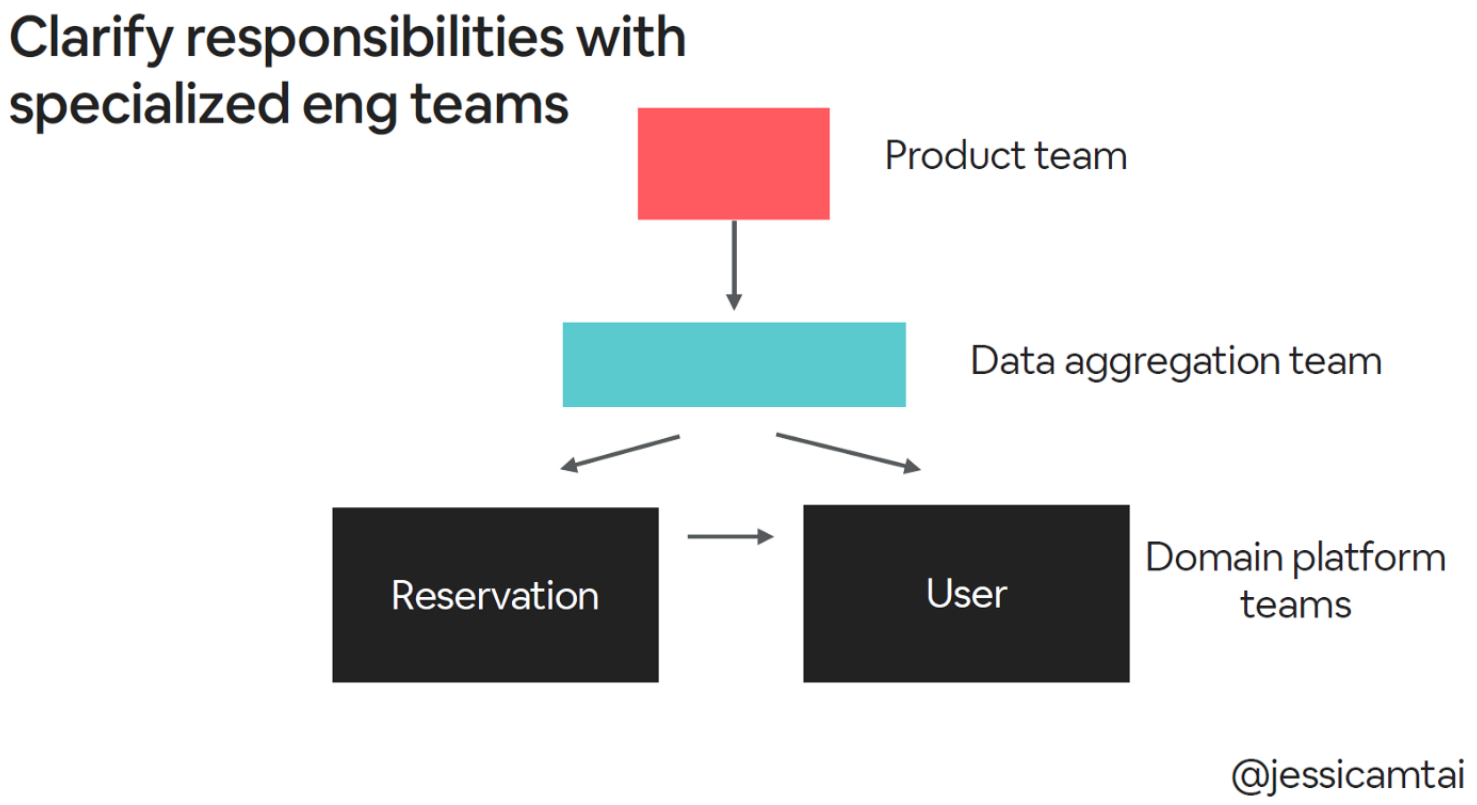


Figure 11: Airbnb organization supports the new architecture responsibilities.

Methods enforce paved-path to new architecture

Architecture review, IT committee, craft governance — these are all methods used to review proposed solutions against the current context and future architecture.

The value of such methodology is to act as a counter-force, external from a project context driven by other objectives, to equilibrate choices.

Airbnb enforced a paved-path to the *Micro+macroservices architecture* using these methods.

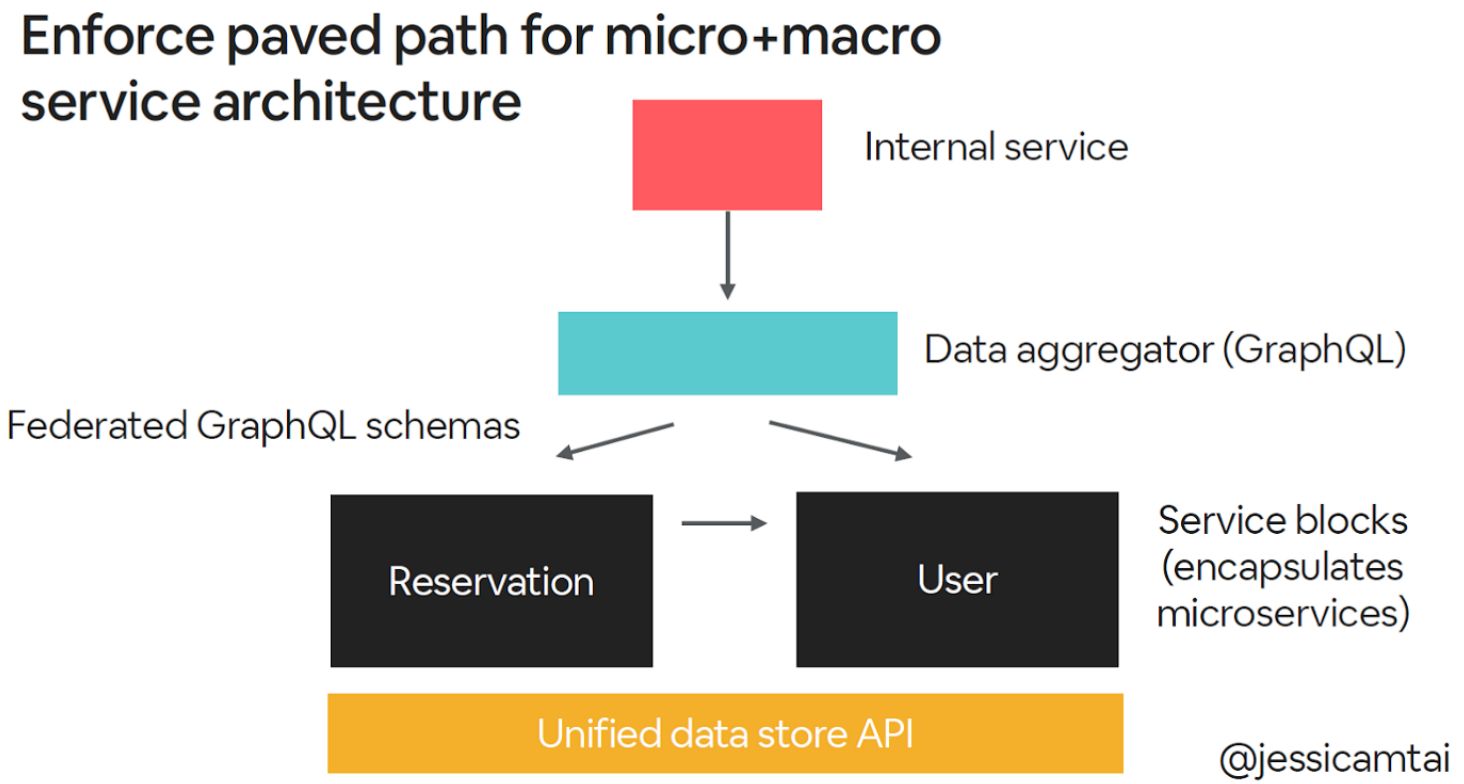


Figure 12: The review process ensures solutions fit in the defined architecture.

Even if not covered, the *Management* of Airbnb surely had a great effect on leading the transition and evolving the culture on the new model, as well as developing the *Skills*, completing the scope of **MAMOS**.

Equipped with these changes, Airbnb is able to lead parallel tracks of migration. But deprecating the Monolith still takes too much time.

4. Lead a deprecation working group to accelerate the migration

Changing our bank account alone can already be a nightmare. The task is even more complex when it takes years to accomplish with multiple teams to coordinate.

Airbnb is like a lot of other organizations with legacy systems and on-going business: they cannot dynamite the house they are living in while building the new one.

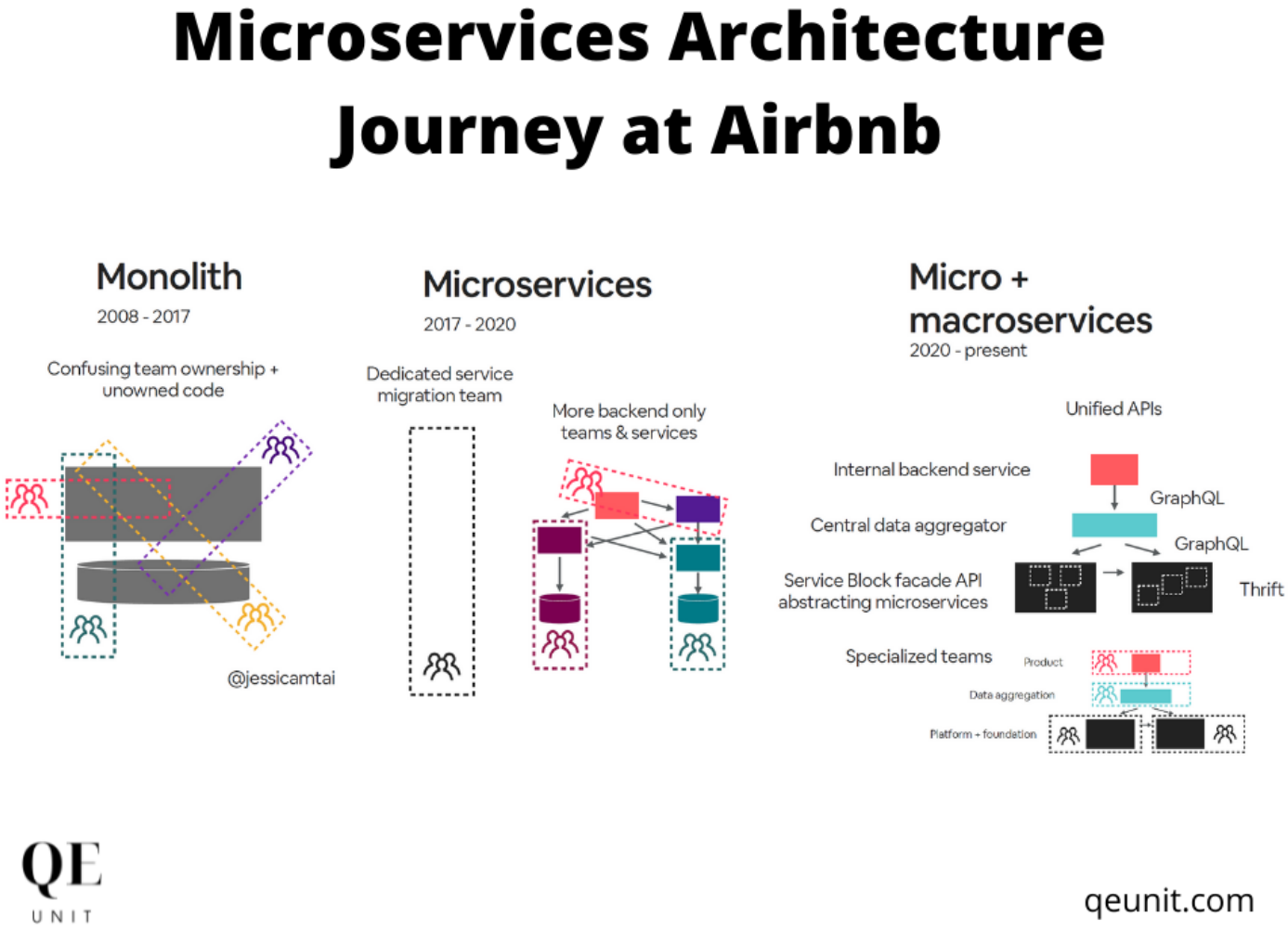
A specific organizational unit is dedicated to accelerate the migration out of the monolith, leading the following:

- **Mobile app deprecation > 12 months** for results
- **Elevate & track monolith debt** for visibility

- **Sunset low usage endpoints** to accelerate removal
- **Long-term ownership** of the migration
- **Recognize depreciation as impactful** for valorization.

This group has a lobbying counter-force but is essential to achieve the objective of migration. Deprecating the monolith cannot be “everyone’s responsibility”.

When this task will be accomplished, other challenges would have been faced.



Airbnb’s Architecture Journey to Quality Engineering

This perspective of Airbnb’s architecture evolution has concrete Quality Engineering takeaways for Quality at Speed software.

First, the differentiation between a localized or a scaling problem enables a better identification

of the underlying causes, and so of incremental solutions to adopt.

In a Lean approach, Airbnb's teams worked on problems when they faced them, avoiding waste like unnecessary pre-optimizations.

One actionable takeaway lies in the architecture from my perspective, the other elements of MAMOS being an alignment of this structuring block.

Starting with a Monolith in a Monorepo is the way to start, to then divide it based on the business evolution, like they did with two repositories.

Their story then exemplifies issues of a *"Microservices-only architecture"*, replaced by the *Micro+macroservices architecture*, with a delayed feedback loop.

Which architecture-style are you gonna use?

Follow me for more Quality Engineering.

Clap this article if you liked it 👏

_Author: _*Antoine Craske*

References

The Human Side of Airbnb's Microservice Architecture. InfoQ, Youtube.

The Great Migration: from Monolith to Service-Oriented, Youtube.

Airbnb's Great Migration: Building Services at Scale, Youtube.

Adam Miskiewicz, *Taming Service-Oriented Architecture Using A Data-Oriented Service Mesh*, Airbnb Tech Blog.

Liang Guo, *Building Services at Airbnb, Part 1*, Airbnb Tech Blog.

Joab Jackson (2021), *How Airbnb and Twitter Cut Back on Microservice Complexities*. TheNewStack.

Nitesh Gupta, *Microservice, Miniservice, and Macroservice*, DZone.

Martin Fowler, James Lewis, *Microservices*. MartiFowler.com.

Zhamak Dehghani, *How to break a Monolith into Microservices*, MartiFowler.com.

Chris Richardson (2018), *Microservices Pattern*. Manning.

Antoine Craske, *Business Agility Requires More Than Organizational Stability*. QE Unit.

Antoine Craske, *How To Make The Right Choice Between A Monorepo Or Multirepo*. QE Unit.

Antoine Craske, *More Technology, More Problems*. QE Unit.