Milan Jovanović • Following
Practical .NET and Software Architecture Tips | Microsoft MVP
1d •

How do you migrate to Microservices?

Let's say you're starting from a monolith.

You'll probably use some form of the Strangler Fig pattern.

What you should've done is build a Modular Monolith.

Modules are perfect candidates for extraction.

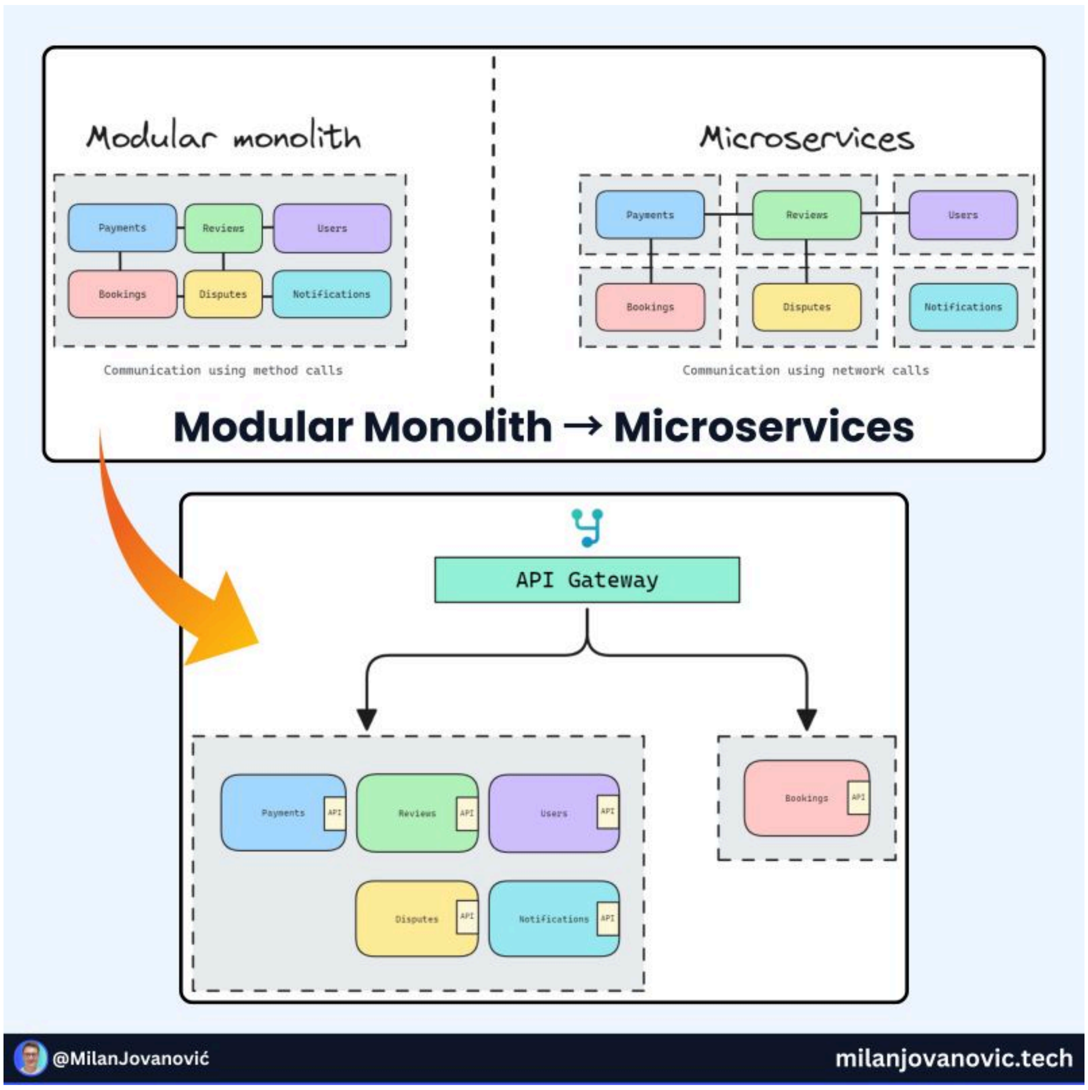
When I'm evaluating modules, I look for these characteristics:

- Low coupling with other modules
- High cohesion within the module
- A distinct business function
- Potential performance or scalability gains from separation

The migration process should be straightforward.

Extract the module, fix communication, and migrate data.

I explained the entire process here: https://lnkd.in/efnqeFbu



268 16 comments 30 reposts

Like Comment Repost Send

Interesting I agree Great advice Insightful Very infor

Most relevant

Ahad Chowdhury • 2nd
Full-Stack Web Developer • AI and Big Data Enthusiast
1d

Very insightful 👌

Like 👍 1 | Reply 1 reply

Milan Jovanović in Author
Practical .NET and Software Architecture Tips | Microsoft MVP
1d

Thanks!

Like | Reply

Pavle Davitković • 2nd
Software Engineer @TraceOne

1d

What are mandatory steps to perform when you start the migration?

Like 1 | Reply 1 reply

Milan Jovanović Author
Practical .NET and Software Architecture Tips | Microsoft MVP

1d

Well first you gotta decide why you're doing it, and have a good reason for it

Like 1 | Reply

System Design Playbook
70 followers

1d

Milan Jovanović Strong post. But I think the real challenge isn't how to migrate — it's whether it's worth it at all.

Before migrating from monolith to microservices, a few hard questions need to be asked:

1. Business Needs:

• Are different modules evolving at different speeds?

• Do teams need independent release cycles?

• Is scale hurting user experience or operational costs?

2. Technical Bottlenecks:

• Are parts of the system suffering from latency, tight coupling, or frequent deployment clashes?

• Can we isolate hot paths that require scale (e.g. payments, notifications)?

3. Post-Migration Trade-offs:

• Operational overhead skyrockets — infra, observability, CI/CD pipelines

• Data consistency shifts from simple ACID to eventual consistency, sagas, retries

• Cross-team coordination gets trickier — contracts, APIs, versioning all come into play

Also, performance isn’t always better. Internal service calls over network are slower than in-process calls — unless you genuinely need that separation.

Sometimes a well-structured modular monolith, backed by clear boundaries and team ownership, can get you 80% of the benefits with 20% of the pain.

Would love to hear: In your experience, when does the migration really pay off?

Like | Reply

Philip Jacob • 3rd+ 1d
Technical Architect @ Banerasoft Inc. | MCA in Computer Science

Very useful in Visualizing the migration