Monolith vs microservices: which should you choose?

1 November 2026 · 6 min read · blogs.iictp.org/monolith-vs-microservices

A monolith is one deployable application, and microservices split a system into many small services that deploy independently. Neither is better in general. Microservices trade simpler code boundaries for harder operations, so they pay off mainly when separate teams need to change and scale parts of a system independently.

TL;DR The short version

- Neither a monolith nor microservices is better in general.
- Microservices let teams deploy and scale independently, but every call becomes a network call that can be slow or fail.
- A modular monolith is often a good middle ground; move gradually, and only if shared releases slow independent teams.

Read online, listen to it, or change the text size: https://blogs.iictp.org/monolith-vs-microservices

What does a monolith offer?

Calls between parts are ordinary function calls, so they are fast, easy to debug and consistent within a transaction. There is one thing to deploy and monitor. For a small team or a young product, that simplicity is a real advantage.

What do microservices offer?

Teams can deploy their service without coordinating a release of everything, and each service can scale and use the technology that suits it. A failure can be contained to one service if the boundaries and fallbacks are designed well.

What do microservices cost?

Every call becomes a network call that can be slow or fail, so you need timeouts, retries and monitoring. Data is split, so consistency becomes harder. You also need strong deployment, observability and ownership practices, or the system becomes a distributed monolith.

What is a modular monolith?

It is one deployable application whose code is divided into modules with clear interfaces and separate data ownership. It keeps most of the simplicity of a monolith and makes later extraction of a service much easier, so it is a sensible default for many teams.

How do you decide?

Ask whether independent teams are being slowed by shared releases, whether parts of the system have very different scaling needs, and whether you have the operational maturity to run many services. If the answer to the first two is no, stay modular and simple.

How do you move from one to the other?

Do it gradually. Identify a boundary that matches a business capability, put an interface in front of it, and move it out behind that interface, as in the strangler approach. Avoid a rewrite that must match every feature before it ships.

> Related credential: Certified Technical Practitioner. Explore the exam facts and syllabus.

Sources and further reading

1. Newman, S. Building Microservices. O’Reilly, 2nd ed. 2021.
2. Fowler, M. Strangler Fig Application. martinfowler.com.

Published by IICTP, an independent technical credential body, on 1 November 2026. The current version of this post is at https://blogs.iictp.org/monolith-vs-microservices. More guides: blogs.iictp.org.