What is event-driven architecture?

1 November 2026 · 5 min read · blogs.iictp.org/what-is-event-driven-architecture

In event-driven architecture, services communicate by publishing events, which are records that something happened, and other services react to them. The publisher does not know who is listening, so parts of the system can change and scale without being wired directly together.

TL;DR The short version

- In event-driven architecture, services publish facts about what happened and others react, without being wired directly together.
- It loosely couples services, so new consumers can be added without changing the publisher.
- Delivery is usually at least once, so consumers must be idempotent; use an outbox and dead-letter queues, and plain requests when callers need an immediate answer.

Read online, listen to it, or change the text size: https://blogs.iictp.org/what-is-event-driven-architecture

What is an event?

An event is a fact about the past, such as “order placed”, usually with the data needed to understand it. It is different from a command, which asks another service to do something and expects a particular one to respond.

What is a broker for?

A message broker or event stream carries events from publishers to subscribers. It stores them, delivers them to every interested consumer and lets slow consumers catch up, so a delay in one service does not block the others.

What are the benefits?

Services are loosely coupled, so a new consumer can be added without changing the publisher. Spikes are absorbed by queues, and the log of events can be replayed to rebuild a read model or onboard a new service.

What are the pitfalls?

Delivery is usually at least once, so consumers must be idempotent. Ordering is guaranteed only within a partition, and tracing a flow across services is harder. Events also become a public contract, so changing them needs care.

How do you avoid losing events?

Use a transactional outbox so that saving data and recording the event happen together, and add dead-letter queues for messages that keep failing. Monitor consumer lag so slowness is visible.

When is it the wrong choice?

When a caller needs an immediate answer, a plain request and response is simpler. Use events where decoupling and resilience matter, not as a default for every interaction.

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

Sources and further reading

1. Hohpe and Woolf. Enterprise Integration Patterns. Addison-Wesley, 2003.
2. Kleppmann, M. Designing Data-Intensive Applications. O'Reilly, 2017.

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