**Microservices**

In microservices, services are inter-dependent. If Service A continuously calls Service B while B happens to be slow or down, failure can cascade and bring A down or make it hang as well. The Circuit Breaker pattern helps by behaving as an automated switch: it prevents calls to a volatile service after a failure threshold is crossed, and only resumes calls after recovery is sensed.

* **The breaker has three states**:
* **Closed**: all calls are allowed (normal).
* **Open**: calls are immediately blocked (service considered down).
* **Half-Open**: after a timeout, a few test calls are allowed. If successful, the circuit closes; if failures continue, it reopens.
* This reduces latency (fail fast instead of waiting for timeouts), prevents resource exhaustion, and isolates failures. It also allows controlled recovery, ensuring that one bad service doesn’t take down the whole system.
* **In DevOps practices**, it combines with alerts and monitoring. Alarms can be triggered by degraded services due to logs of the breaker. Along with retries, fallbacks, and auto-scaling, it enhances resilience, user experience, and uptime of a system. It is a prominent pattern in distributed resilient systems.