






Resiliency patterns

06/23/2017 • 2 minutes to read • Contributors     

Resiliency is the ability of a system to gracefully handle and recover from failures. The nature of cloud hosting, where applications are often multi-tenant, use shared platform services, compete for resources and bandwidth, communicate over the Internet, and run on commodity hardware means there is an increased likelihood that both transient and more permanent faults will arise. Detecting failures, and recovering quickly and efficiently, is necessary to maintain resiliency.

Pattern	Summary
Bulkhead	Isolate elements of an application into pools so that if one fails, the others will continue to function.
Circuit Breaker	Handle faults that might take a variable amount of time to fix when connecting to a remote service or resource.
Compensating Transaction	Undo the work performed by a series of steps, which together define an eventually consistent operation.
Health Endpoint Monitoring	Implement functional checks in an application that external tools can access through exposed endpoints at regular intervals.
Leader Election	Coordinate the actions performed by a collection of collaborating task instances in a distributed application by electing one instance as the leader that assumes responsibility for managing the other instances.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Retry	Enable an application to handle anticipated, temporary failures when it tries to connect to a service or network resource by transparently retrying an operation that's previously failed.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.