What is Circuit Breaker?

It is an architecture pattern in which an intermediary service is placed between a caller and a target. Meaning that you wrap a protected function call within a circuit breaker object, whose purpose is to observe conditions in the target, specifically monitor failures. Once failures reach a certain threshold, the circuit breaker trips. All further calls to it will return with an error, without the protected call being made. If a hazard occurs, the circuit breaker process reroutes traffic to another service that has the necessary logic to migrate the hazardous conditions.

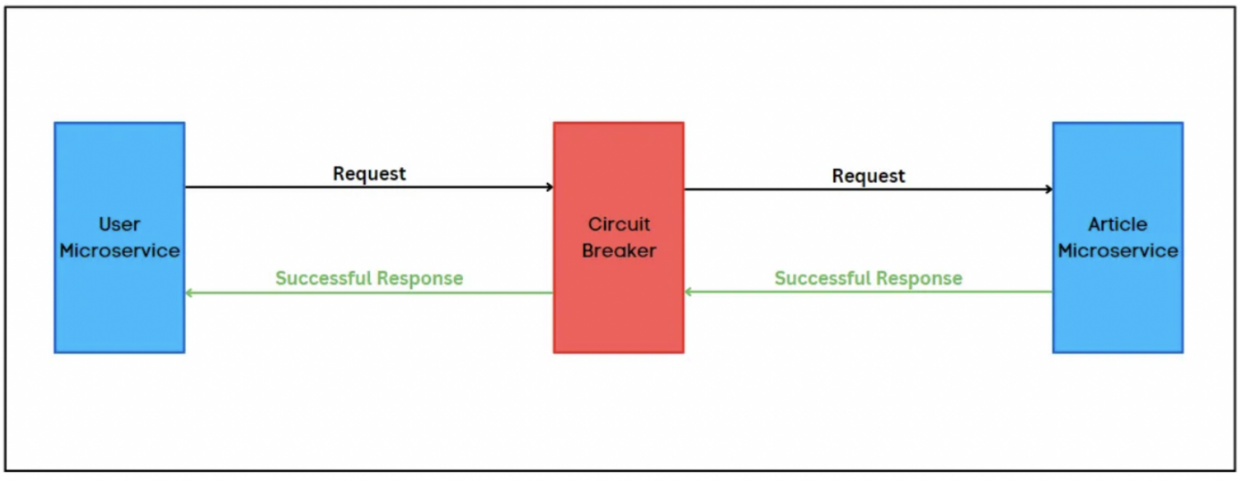
- It allows developers to prevent cascading failures in microservices architecture, this by invoking remote services with the help of a proxy.
- Usually used for synchronous calls, but it can also be useful for asynchronous communications. Can be used in any scenario where the goal is to protect parts of a system from failures in other parts.

How does it work?

Circuit breaker pattern has 3 states: Closed, Open and HalfOpen.

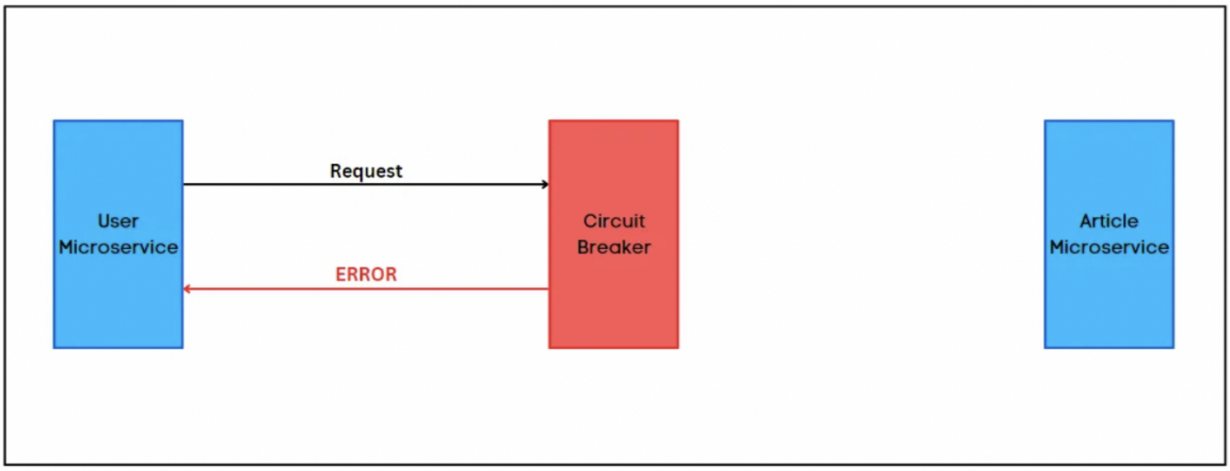
Closed State

It is the initial state of the circuit breaker or the proxy. It allows microservices to communicate as usual and monitor the number of failures occurring within a define time period. If the specified threshold value is exceeded, the circuit breaker will pass from a Closed state to an Open state. Otherwise, the failure count is reseted and timeout period.



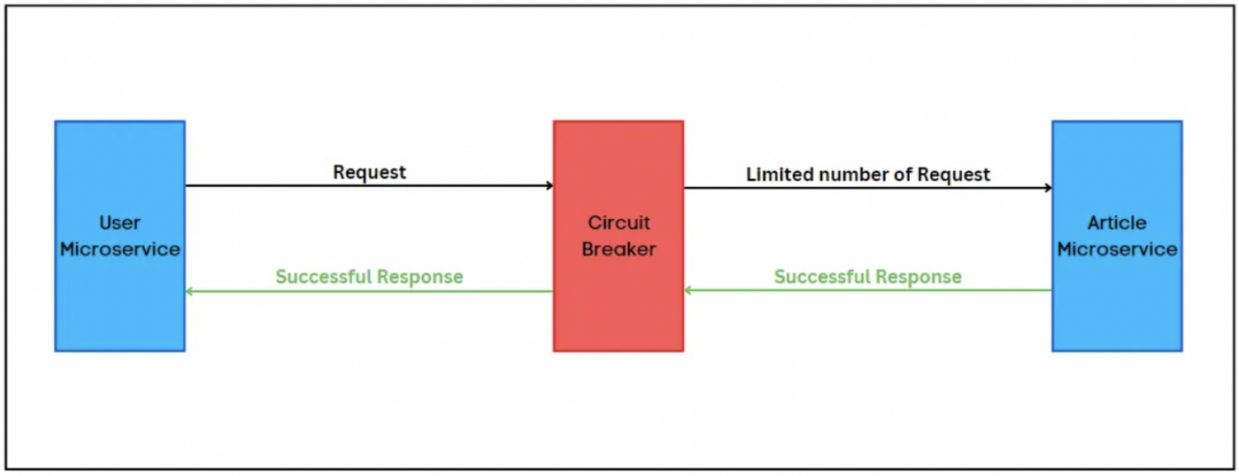
Open State

In this state, communications between microservices is completely blocked. The article service will not receive any requests and the user service will receive an error from circuit breaker. It remains in this states, until the timeout period ends. Then it moves to next Halp-Open state.



Half-Open State

The circuit breaker allows a limited number of requests to reach article service. In the case that those requests are successful, it will switch to the Closed state and allow normal operations. Otherwise, it will block all requests for the defined timeout period.



Pros

- Using circuit breaker patterns to respond to the onset of a hazardous condition, is a great option to prevent accidents before they happen.
- It is also a great alternative to make systems fault-tolerant.
- Helps reduce resources tied up in operations which are most likely to fail.
- Helps avoid waiting on timeouts for the client and broken circuit avoids blasting on a struggling server.
    - Reduces application downtimes.

Cons

- Requires an infrastructure management tech, such a service that can manage switching it on and off.
- Testing can become a little tricky and harder. A large amount of responses need to be in force and tested.
- If it is not properly configured, further on there can be some issues presented in the services.

Michelle Muñiz #28241