Kresil
Kotlin Resilience

Kotlin Multiplatform library for fault-tolerance with Ktor Integration

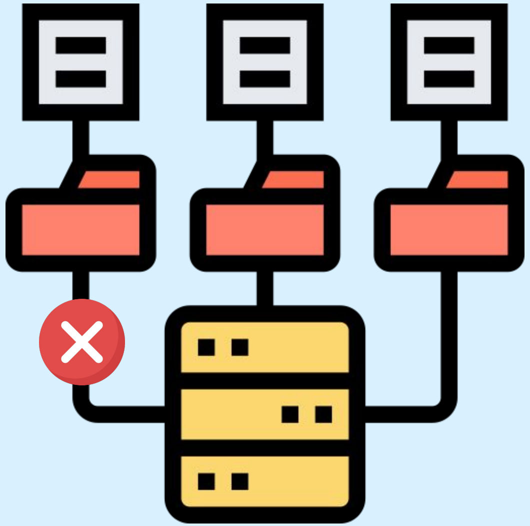
ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Supervisor: Prof. Pedro Félix
Author: Francisco Engenheiro - 49428
Project and Seminary
BSc in Computer Science and Engineering
Summer 2023/2024

01

Introduction

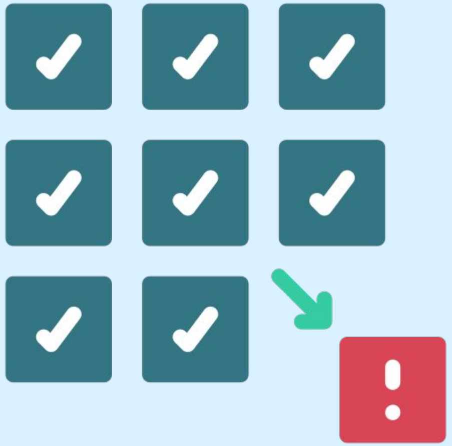
Resilience in Distributed Systems



If failure is inevitable, what can be done?

Fault-Tolerance Service

A fault-tolerant service is a service that is able to maintain all or part of its functionality, or provide an alternative, when one or more of its associated components fail.



Resilience Mechanisms

Retry - Repeats failed executions

Rate Limiter - Limits executions/period

Circuit Breaker - Temporarily blocks possible failures

Time Limiter - Limits duration of execution

Cache - Memorizes a successful result

Fallback - Defines an action to fallback on failure

And many more...

Resilience Types

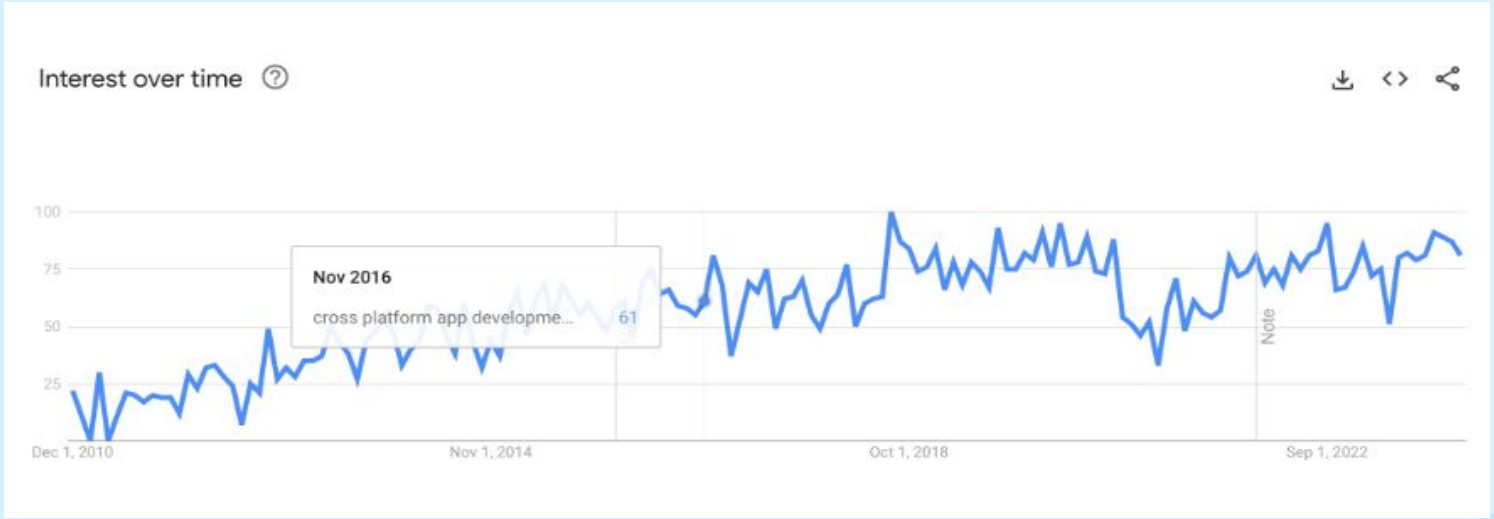
Reactive - Mitigates impact from failures

Proactive - Prevents failures from happening

02

Problem

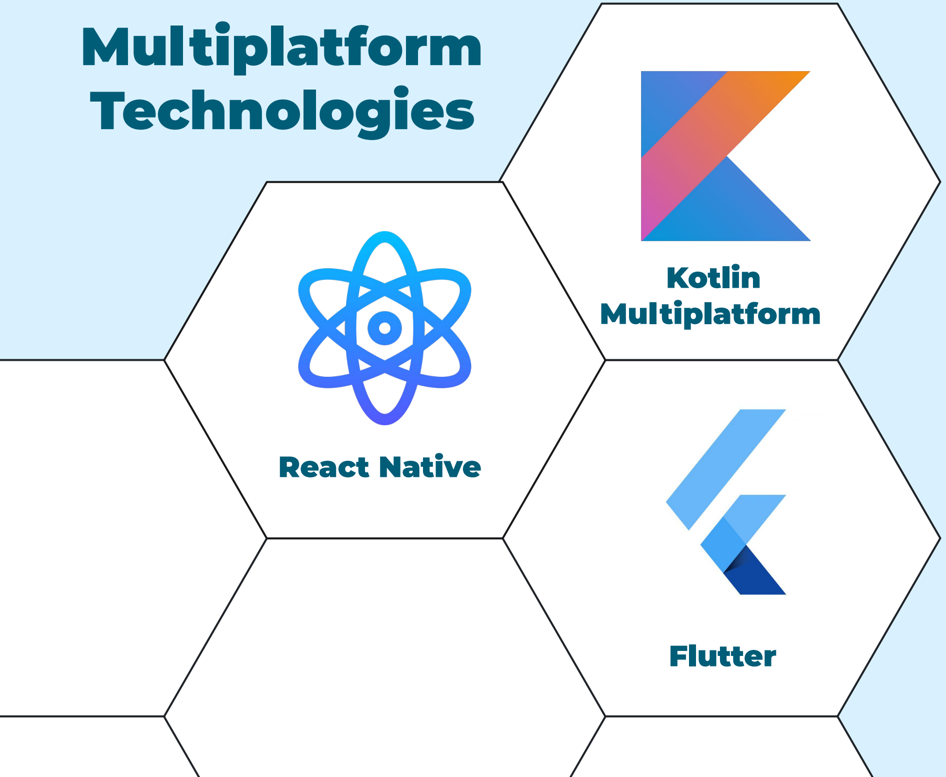
Multiplatform Development Interest



Font: https://www.jetbrains.com/help/kotlin-multiplatform-dev/cross-platform-frameworks.html

Problem

Multiplatform Technologies



No libraries that provide resilience mechanisms in a multiplatform context

Multiplatform Considerations

Concurrency Model

Time Management

Synchronous vs Asynchronous

Mocking

Logging

Kotlin Multiplatform

Kotlin/JS

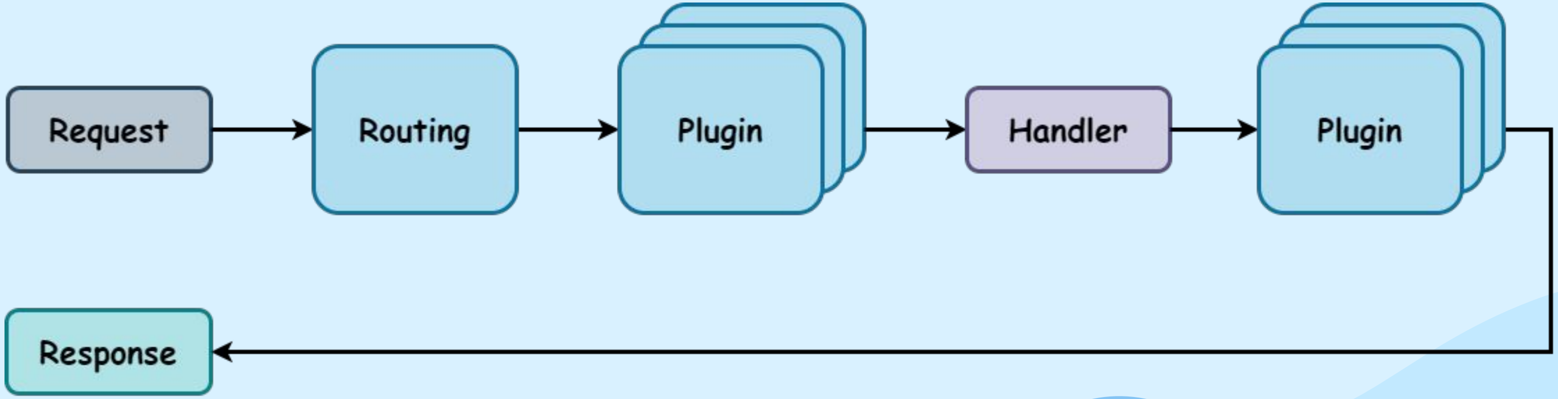
Kotlin/Native

Kotlin/Android

Kotlin/JVM

Ktor Framework

- Built with Kotlin Multiplatform;
- Enables asynchronous server and client development;
- Based on the coroutines concurrency model;
- Modular



Existing Solutions

Platform-Specific

Resilience4j

POLLY

Netflix's Hystrix

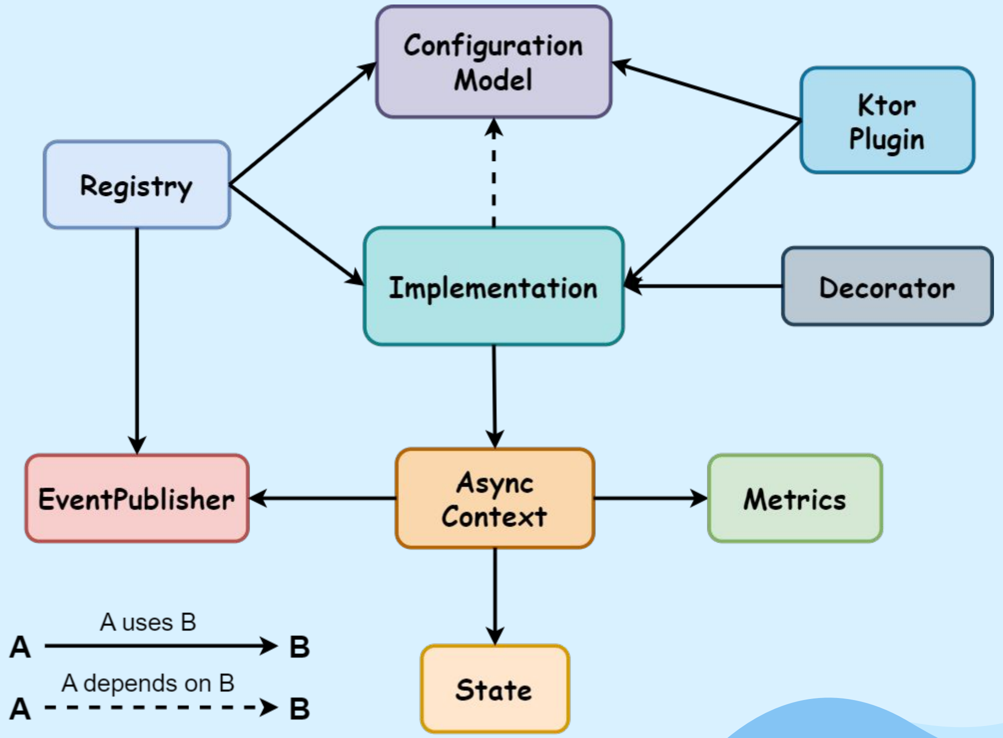
Multiplatform

Arrow

03

Common Design and Implementation Strategy

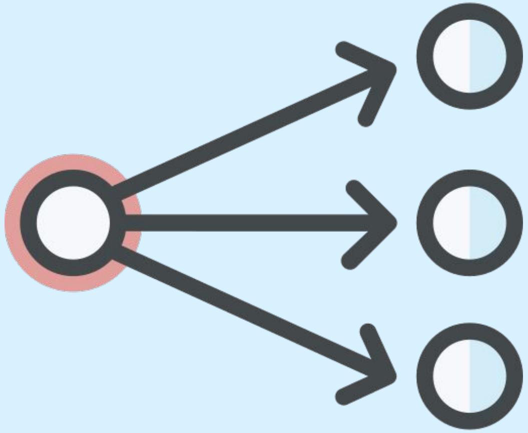
Mechanism Model



Mechanism Configuration

Policies - Define the mechanism behaviour

Event Listeners



Each mechanism:

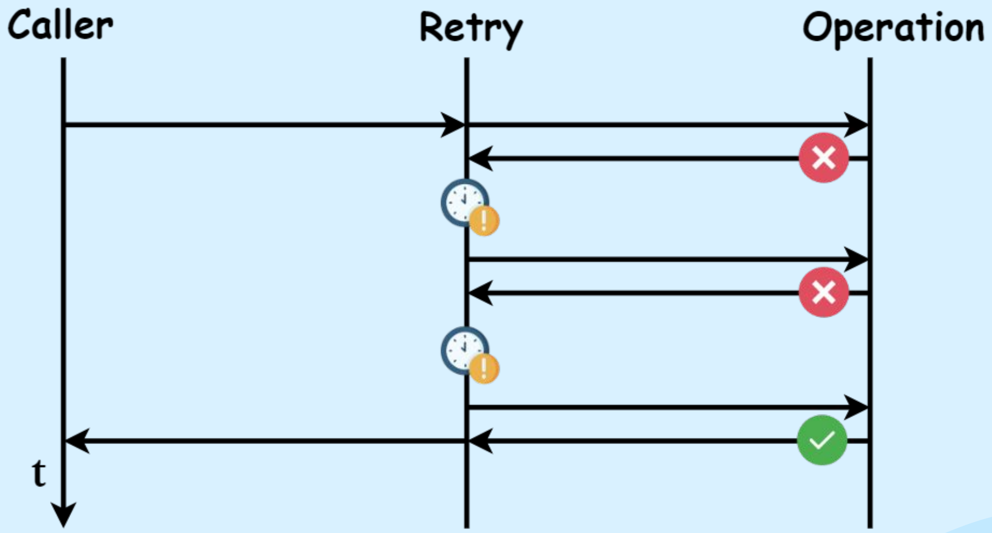
- **Provides listeners for specific and undiscriminated events**
- **Uses asynchronous coroutine primitive Flow**
- **Supports cancellation of registered events**

04

Mechanisms

Retry Mechanism

Retry is a reactive resilience mechanism that can be used to retry an operation when it fails and the failure is a transient (temporary) fault



Delay Handling

Delay Strategy

Defines the amount of time the application should wait before retrying the operation

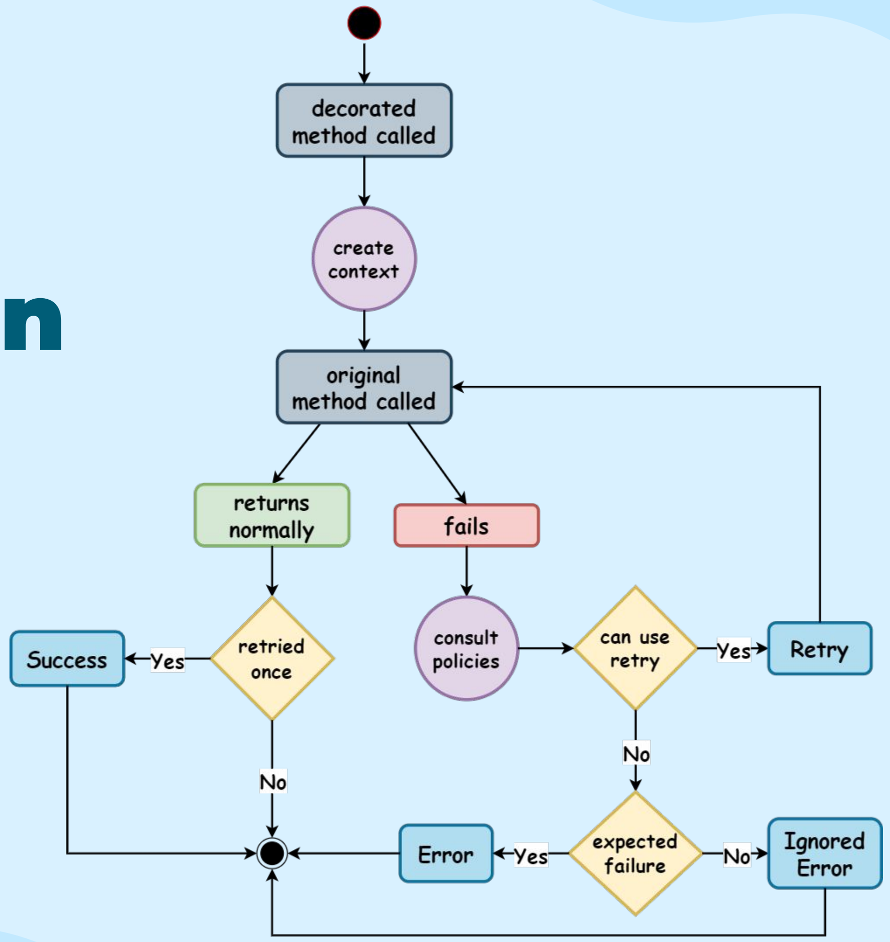
Options

- No delay
- Constant delay
- Linear delay
- Exponential delay
- Custom Delay

Delay Provider

Executes the actual waiting period by pausing or blocking the process (depends on the implementation)
for the specified duration.

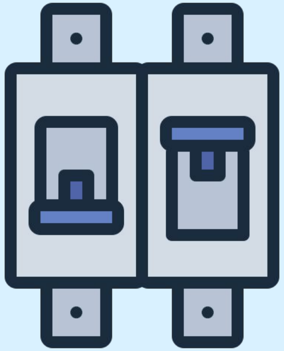
Retry Execution Flow



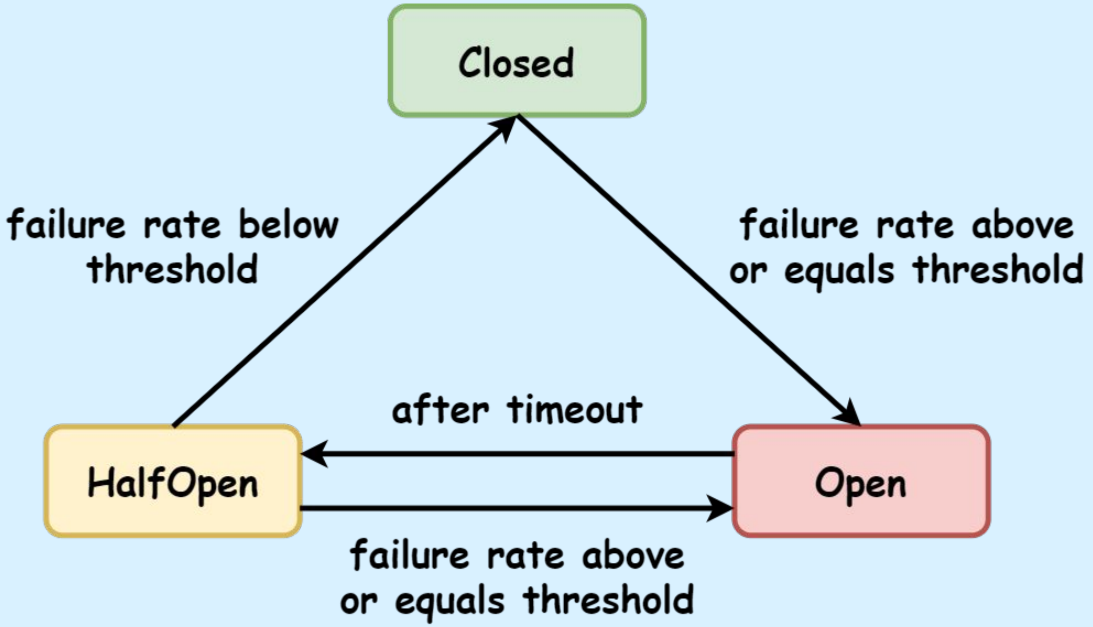
Circuit Breaker Mechanism

Circuit Breaker is a **reactive** resilience mechanism that can be used to **protect** a system component from **overloading or failing**

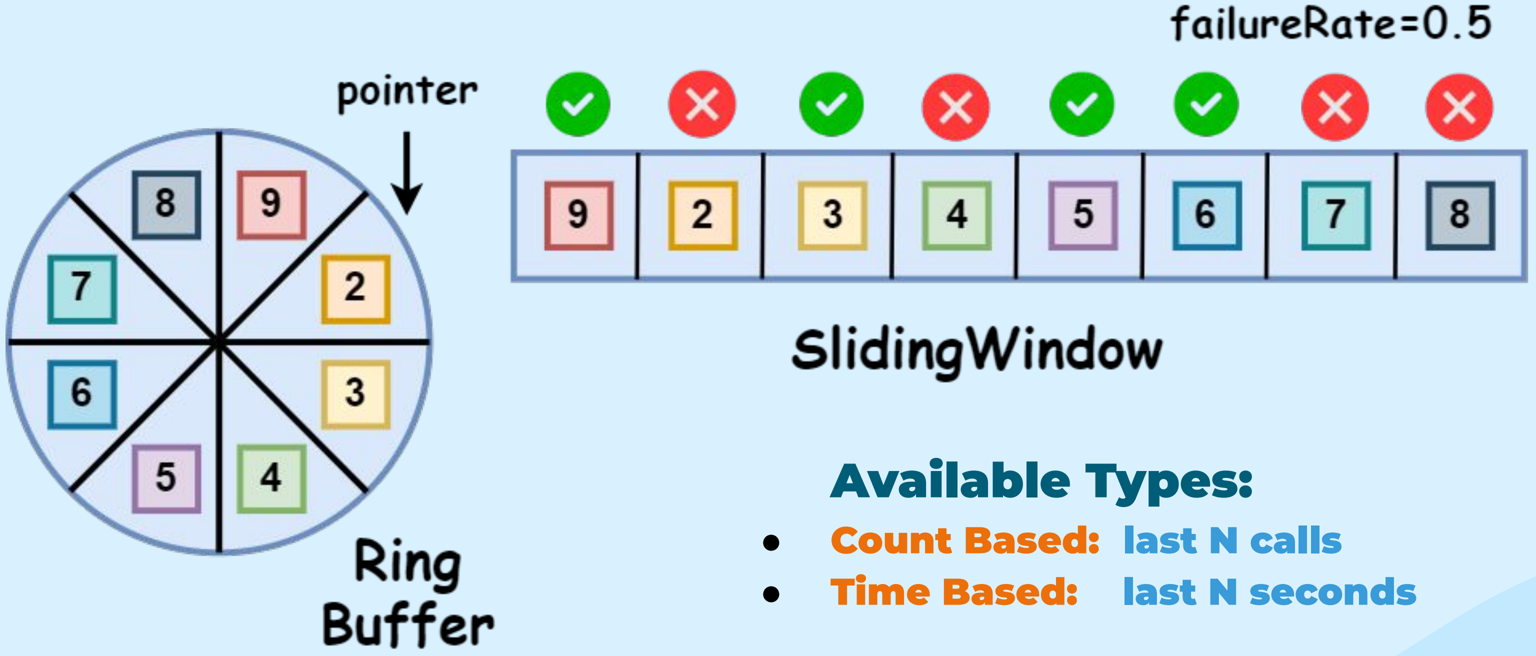
By monitoring the health of the system, it can short-circuit execution requests when it detects that the protected system component is not behaving as expected



Circuit Breaker States


Closed
failure rate below threshold
failure rate above or equals threshold
after timeout
HalfOpen
Open
failure rate above or equals threshold

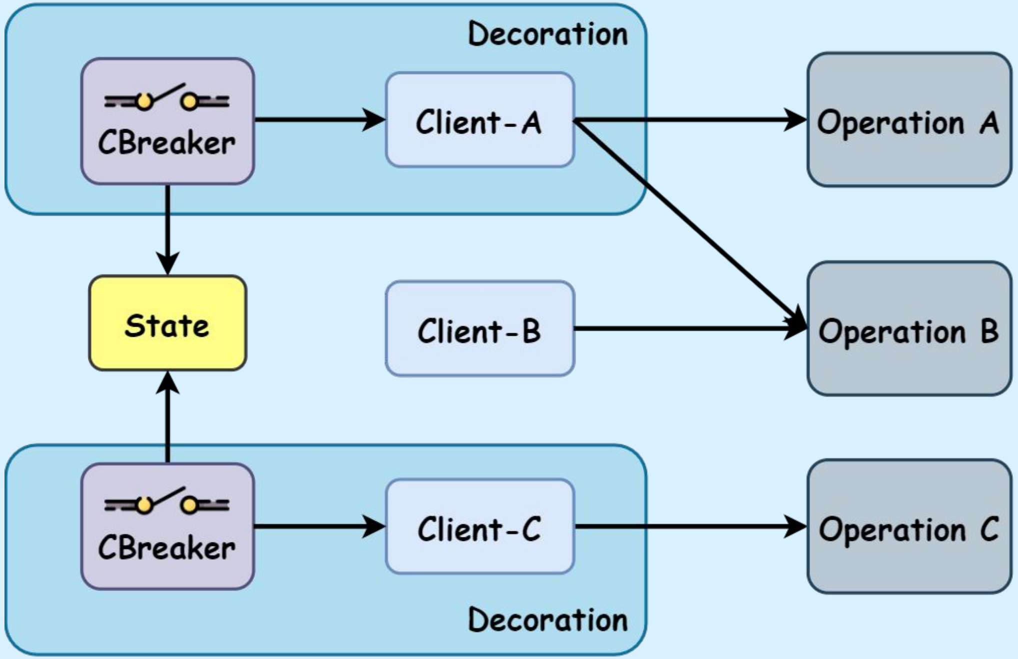
Circuit Breaker Sliding Window



Available Types:

- **Count Based:** **last N calls**
- **Time Based:** **last N seconds**

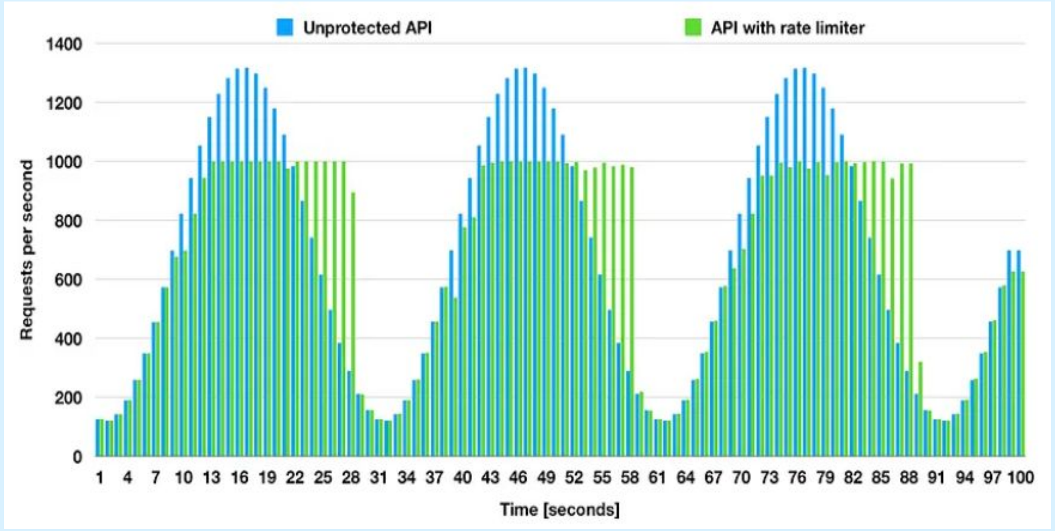
Circuit Breaker Decoration



Rate Limiter Mechanism

Rate Limiter is a **proactive** resilience mechanism that can be used to **limit** the number of **requests** that can be made to a system component, which could be **bound to a time unit**

Therefore controlling the consumption of resources and protecting the system from overloading



Types of Rate Limiting

Total Requests

Key-based

User-based

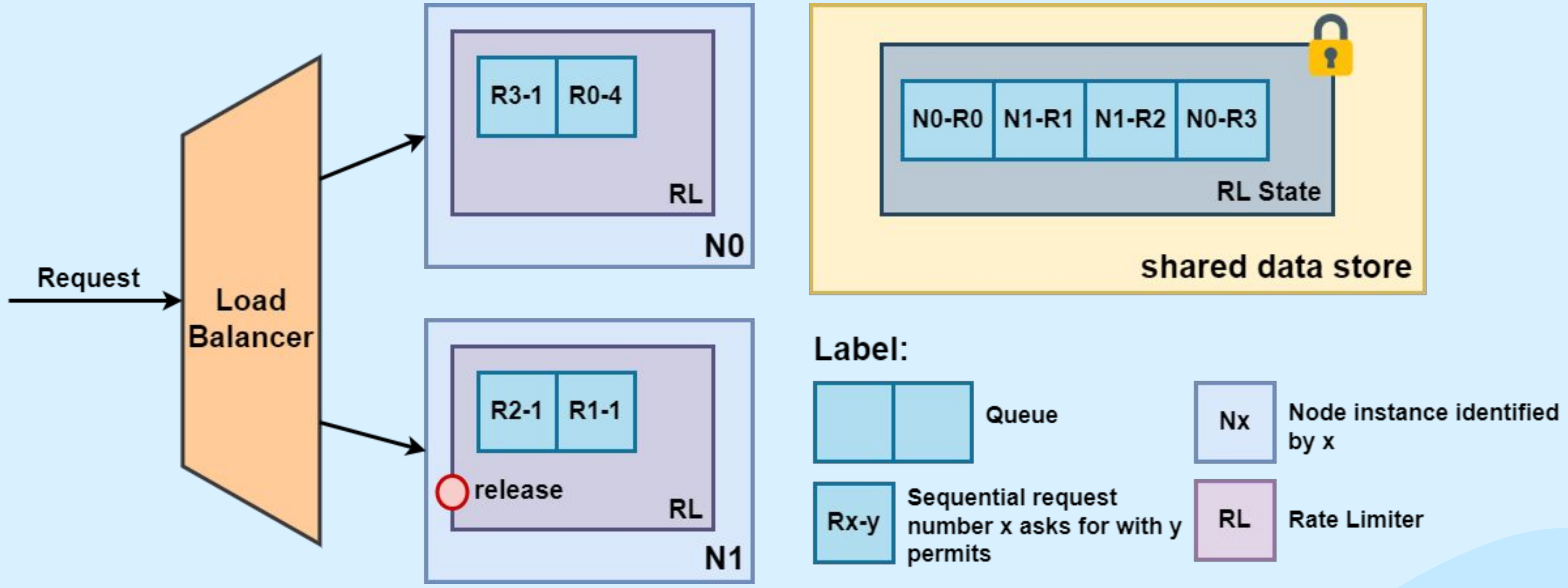
Rate Limiting Exceeded

Reject: Immediately deny the request and return an error response message

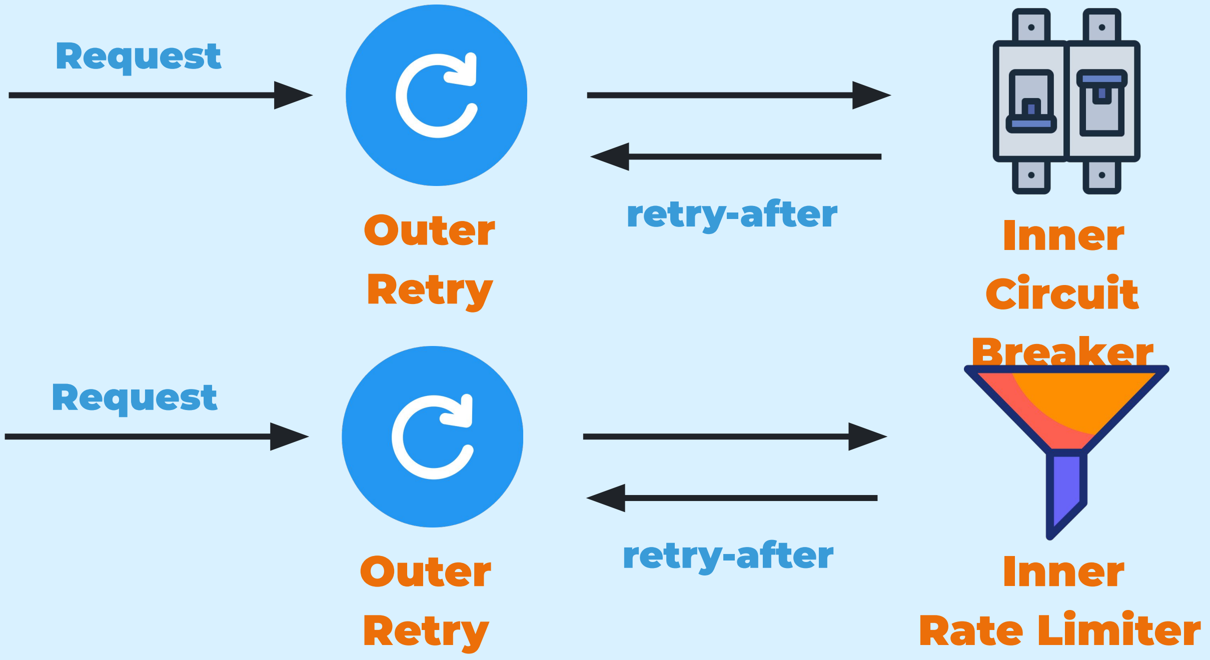
Wait: Place the request in a queue to be processed later when the rate limit allows

Both: Place request in the queue and reject after timeout expires

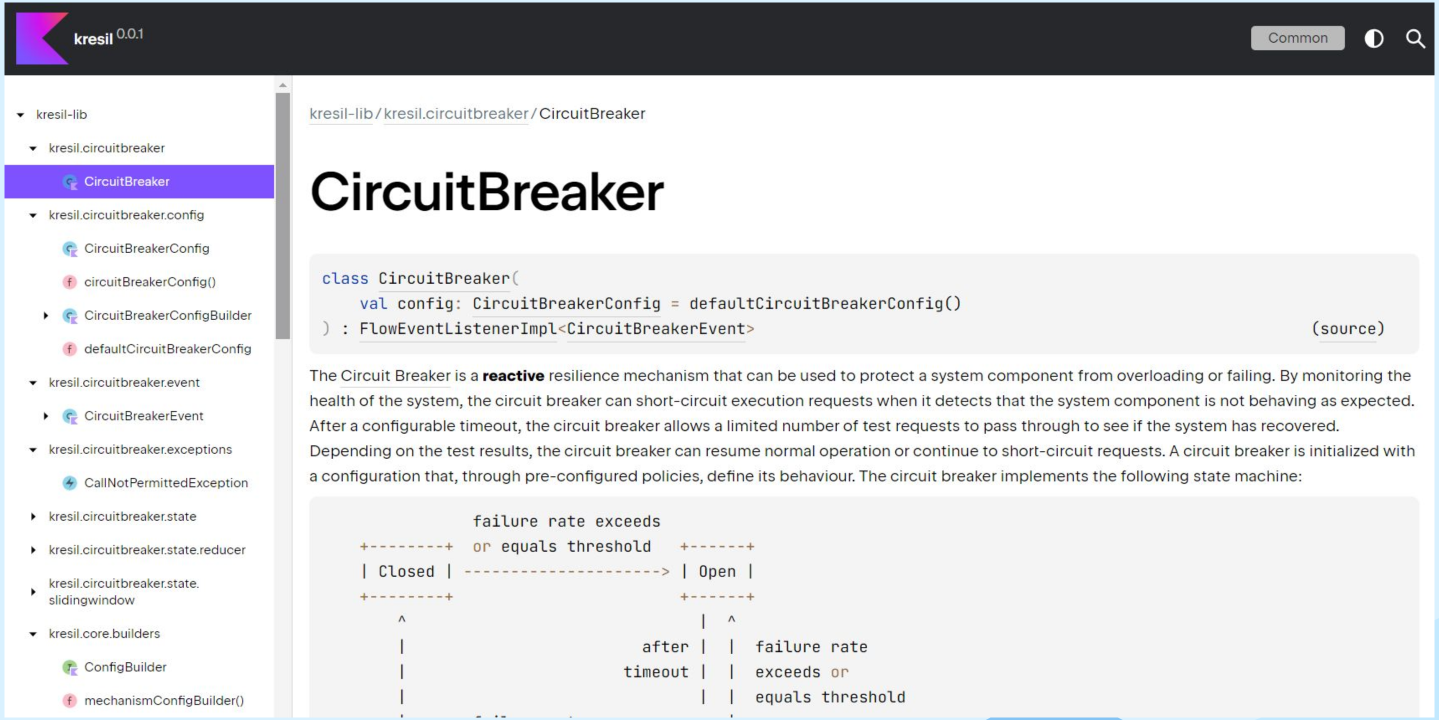
Distributed Rate Limiting



Mechanism Combination



API Documentation



- kresil-lib
  - kresil.circuitbreaker
    - CircuitBreaker
  - kresil.circuitbreaker.config
    - CircuitBreakerConfig
    - circuitBreakerConfig()
    - CircuitBreakerConfigBuilder
    - defaultCircuitBreakerConfig
  - kresil.circuitbreaker.event
    - CircuitBreakerEvent
  - kresil.circuitbreaker.exceptions
    - CallNotPermittedException
  - kresil.circuitbreaker.state
  - kresil.circuitbreaker.state.reducer
  - kresil.circuitbreaker.state.slidingwindow
  - kresil.core.builders
    - ConfigBuilder
    - mechanismConfigBuilder()

kresil-lib / kresil.circuitbreaker / CircuitBreaker

CircuitBreaker

```
class CircuitBreaker(
    val config: CircuitBreakerConfig = defaultCircuitBreakerConfig()
) : FlowEventListenerImpl<CircuitBreakerEvent>                          (source)
```

The Circuit Breaker is a **reactive** resilience mechanism that can be used to protect a system component from overloading or failing. By monitoring the health of the system, the circuit breaker can short-circuit execution requests when it detects that the system component is not behaving as expected. After a configurable timeout, the circuit breaker allows a limited number of test requests to pass through to see if the system has recovered. Depending on the test results, the circuit breaker can resume normal operation or continue to short-circuit requests. A circuit breaker is initialized with a configuration that, through pre-configured policies, define its behaviour. The circuit breaker implements the following state machine:

```
                failure rate exceeds
    +---------+ or equals threshold  +------+
    | Closed | ---------------------> | Open |
    +---------+                      +------+
        ^                              |  ^
        |                        after |  |  failure rate
        |                      timeout |  |  exceeds or
        |                              |  |  equals threshold
```

Future Work

Javascript
Adapter

Pipeline
Builders

Metrics

Selective
Dependency Import

Other
Mechanism
s

Demo