

# Обеспечение отказоустойчивости

## Технические инструменты:

Уже обсуждали:

- **Репликация данных:** Дублирование данных на нескольких узлах.
- **Избыточность компонентов:** Резервные компоненты для восстановления.
- **Паттерн Circuit Breaker:** Предотвращение распространения сбоев.

Есть еще инструменты:

- **Аварийное переключение (Failover)**

Это процесс автоматического переключения с одной системы на другую в случае сбоя или планового отключения.

## Как работает:

Системы мониторят состояние друг друга. При обнаружении сбоя, трафик автоматически перенаправляется на резервную систему.

## Варианты реализации:

1. **Active-Passive:** Один активный узел, один или несколько пассивных.
2. **Active-Active:** Все узлы активны и могут обрабатывать трафик.

## Пример:

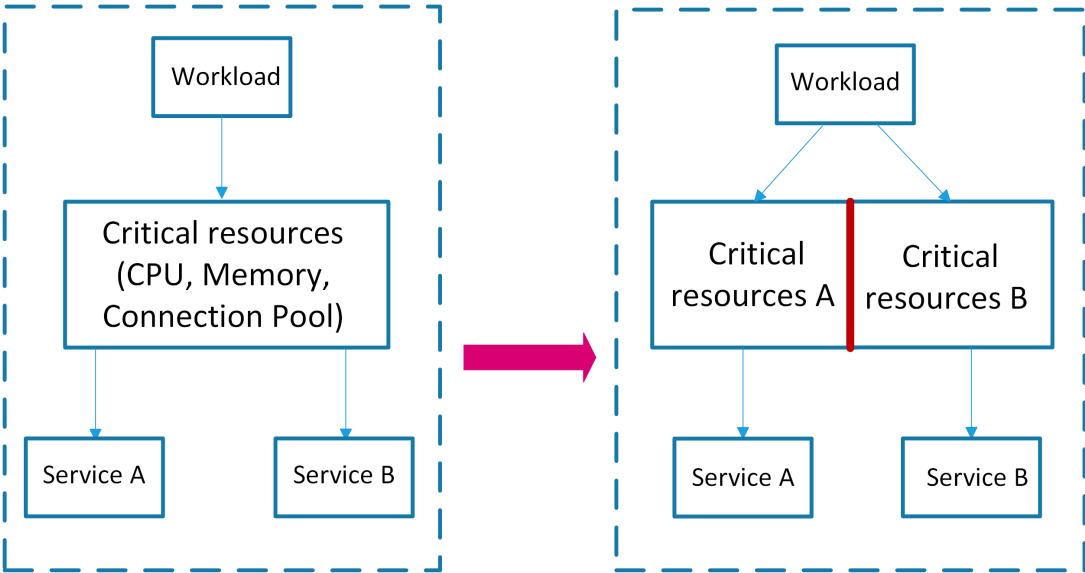
В случае, если основной сервер базы данных упал, система автоматически переключается на резервный сервер, минимизируя простой в работе приложения.

- **Паттерн Bulkhead**

Это паттерн проектирования, который предполагает разделение ресурсов и операций для предотвращения распространения сбоев.

**Как работает:**

Ресурсы (например, потоки, память) изолированы таким образом, чтобы сбой в одной части системы не влиял на другие части.



**Варианты реализации:**

- 1. **Thread-level Bulkhead:** Изоляция на уровне потоков.
- 2. **Process-level Bulkhead:** Изоляция на уровне процессов.

**Пример:**

Если в микросервисной архитектуре один из микросервисов перегружен, Bulkhead гарантирует, что этот сбой не затронет другие микросервисы.

- **Timeouts and Retries**

Механизмы установления времени ожидания и автоматических повторных попыток для сетевых (API) или других типов операций.

**Как работает:**

Если операция не завершается за установленное время, она либо отменяется, либо инициируется повторная попытка.

**Варианты реализации:**

- 1. **Exponential Backoff**: Увеличение времени между попытками экспоненциально.
- 2. **Jitter**: Случайное изменение времени между попытками.

**Пример:**

При временной недоступности API, система автоматически повторяет запрос, применяя стратегию Exponential Backoff, чтобы не перегрузить API.

- **Graceful Degradation**

Ограничение функциональности системы для обеспечения продолжения работы основных функций в случае сбоя.

**Как работает:**

В случае сбоя некритичные функции отключаются, сохраняя при этом работоспособность основных функций.

**Варианты реализации:**

- 1. **Feature Flags** (говорили о них ранее): Возможность динамического отключения определенных функций.
- 2. **Rate Limiting**: Ограничение числа запросов.

**Пример:**

Если сервис рекомендаций товаров не работает, интернет-магазин может отключить эту функцию, чтобы основной процесс покупки оставался неповрежденным.

- **Chaos Engineering**

Практика искусственного введения сбоев в систему для проверки её отказоустойчивости.

**Как работает:**

Создаются эксперименты, в которых система подвергается различным видам сбоев, чтобы убедиться, что она может выдерживать их.

**Варианты реализации:**

- 1. **Fault Injection:** Искусственное введение ошибок в систему.
- 2. **Traffic Shaping:** Изменение характеристик трафика для имитации различных условий.

**Пример:**

С помощью Chaos Monkey (инструмент от Netflix) регулярно отключают случайные сервисы в продакшн-среде, чтобы удостовериться, что система способна к самовосстановлению.

**Организационные практики:**

- **Планы восстановления**

Это заранее подготовленные сценарии действий для реагирования на различные типы сбоев и инцидентов. Эти планы часто включают в себя шаги по диагностике проблем, восстановлению ключевых систем и коммуникации с заинтересованными сторонами. Они регулярно обновляются и тестируются, чтобы удостовериться в их актуальности и эффективности.

- **Post-Mortem анализ**

Анализ проводится после каждого крупного или учебного инцидента. Составляется отчёт со всеми данными - когда произошла проблема, кто обнаружил, кто выполнил какие действия и так далее. Это систематический процесс, в ходе которого анализируются данные, логи и другая информация для выявления причин сбоев и определения путей улучшения системы. Это также время для анализа эффективности реакции команды: как быстро была найдена проблема, как эффективно была проведена коммуникация и так далее.