**Использование Redis как in-memory-хранилища и кэша в системе с RabbitMQ**

Показать, как можно использовать Redis для временного хранения данных и кэширования в распределенной системе с RabbitMQ, повышая производительность и устойчивость системы. (Вместо Redis, можно взять: Apache Ignite или Memcached)

**Основная идея:**

* **Publisher** по-прежнему публикует сообщения в RabbitMQ.
* **Consumers** при получении сообщения:
  1. Сначала проверяют, нет ли уже обработанного результата или промежуточных данных в Redis.
  2. Если в Redis нет — берет сообщение, обрабатывают его, записывает в БД и кэшируют результат в Redis.
  3. Если в Redis есть (например, такое сообщение уже обрабатывалось другим Consumer’ом), то записывают в БД и не делают повторную запись.

### ****Что нужно продумать:****

1. **Стратегию хранения данных в Redis:**

* Какие ключи и структуры использовать?
* Что будет храниться: результат обработки сообщения, флаг о том, что сообщение уже обработано, или временный статус? (можно свой вариант)
* Продумать и реализовать Eviction policy или TTL

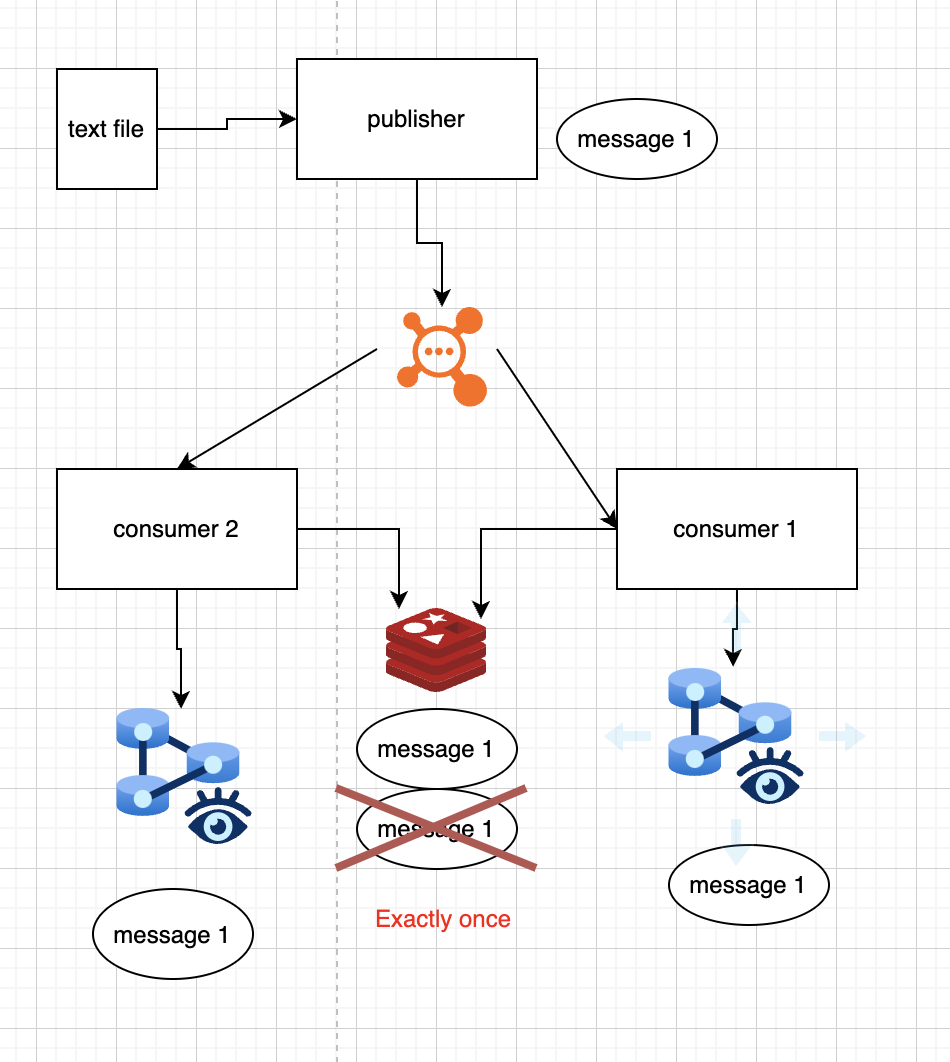
1. **Проблему "гонки":**

* Если два Consumers одновременно обрабатывают одно сообщение, как избежать **дублирования**? (проблема exactly onсe)

### ****В README Вы должны описать:****

* Как используется Redis (что хранится, каков формат ключей и тд)
* Как Вы решаете проблему с повторной обработкой сообщений в Redis
* Какая Eviction policy выбрана или как реализован TTL и почему
* Сценарии сбоя: что происходит, если Redis перезапущен или очищен (можно продумать синхронизацию с базой)
* Как вы решили проблему гонки

Абстрактная схема итогового сервиса:

****