# НА ШАГ БЛИЖЕ К ПЕРЕДОВОМУ ОПЫТУ ИТ-ХОЛДИНГА Т1

Видеокурс

«Практика построения высокопроизводительных и отказоустойчивых решений»

Вебинар 1

# APACHE KAFKA

# ИНФОРМАЦИЯ ОБ ЭКСПЕРТАХ

**Спикер курса**

## Великанов Даниил

Главный разработчик

Более 10 лет работы в Фин.Тех интеграции

# ПРАВИЛА

- Включённые камеры и микрофоны
- Вопросы голосом через поднятие руки в DION
- Вопросы после основной части
- Общение в чате: доброжелательность и уважение
- Запись вебинара будет

# ПЛАН УРОКА

## Сегодня вы изучите следующие темы:

- история Apache Kafka;
- цели и задачи, решаемые Apache Kafka;
- инструменты управления и ключевые элементы конфигурирования;
- способы работы с Apache Kafka.

# ЦЕЛИ И ЗАДАЧИ, РЕШАЕМЫЕ APACHE KAFKA

# ЦЕЛИ И ЗАДАЧИ, РЕШАЕМЫЕ APACHE KAFKA

## Цели Apache Kafka

- Обеспечение высокой производительности и масштабируемости.
- Гарантия надёжности и устойчивости к сбоям.
- Поддержка реального времени для обработки данных.
- Упрощение интеграции различных систем и сервисов.

# ЦЕЛИ И ЗАДАЧИ, РЕШАЕМЫЕ APACHE KAFKA

## Цели Apache Kafka

- Обеспечение высокой производительности и масштабируемости.
- Гарантия надёжности и устойчивости к сбоям.
- Поддержка реального времени для обработки данных.
- Упрощение интеграции различных систем и сервисов.

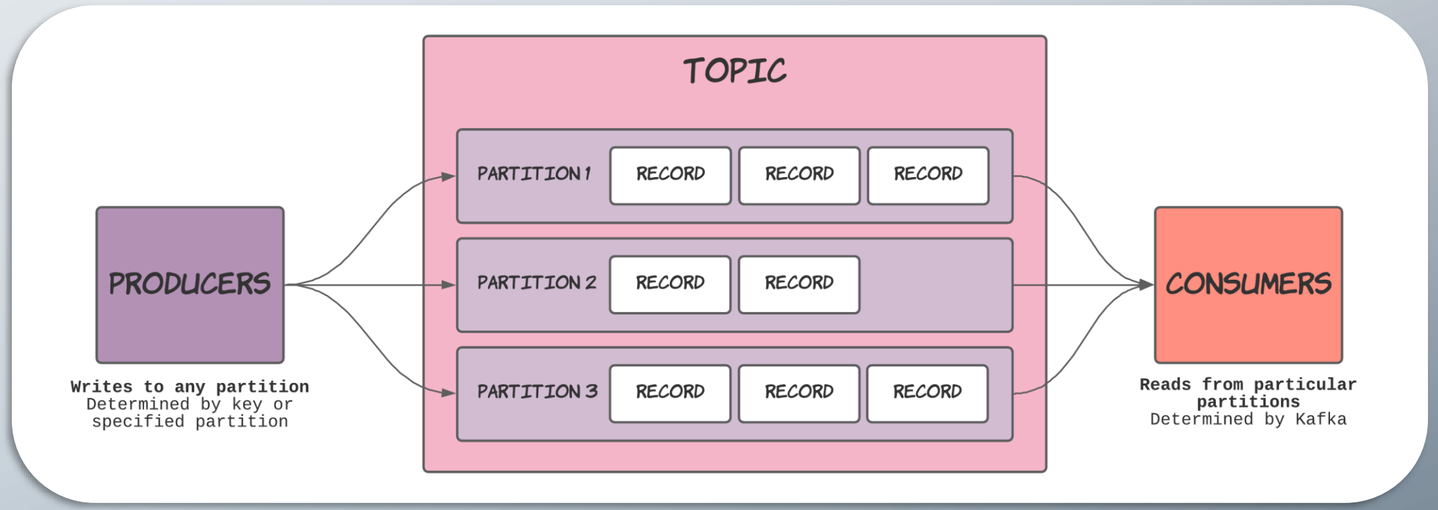
## Задачи, которые решает Apache Kafka

- Логирование: Kafka часто используется для централизованного сбора и хранения логов из различных систем. Это упрощает мониторинг и анализ логов.
- Мониторинг: Kafka позволяет собирать метрики и события из различных систем в реальном времени, что упрощает мониторинг их состояния.
- Аналитика в реальном времени: Kafka позволяет обрабатывать и анализировать данные в реальном времени, что особенно важно для приложений, требующих мгновенного отклика.
- Микросервисная архитектура: Kafka упрощает взаимодействие между микросервисами, обеспечивая надёжную и масштабируемую передачу сообщений.

# УСТРОЙСТВО APACHE KAFKA

## Основные объекты

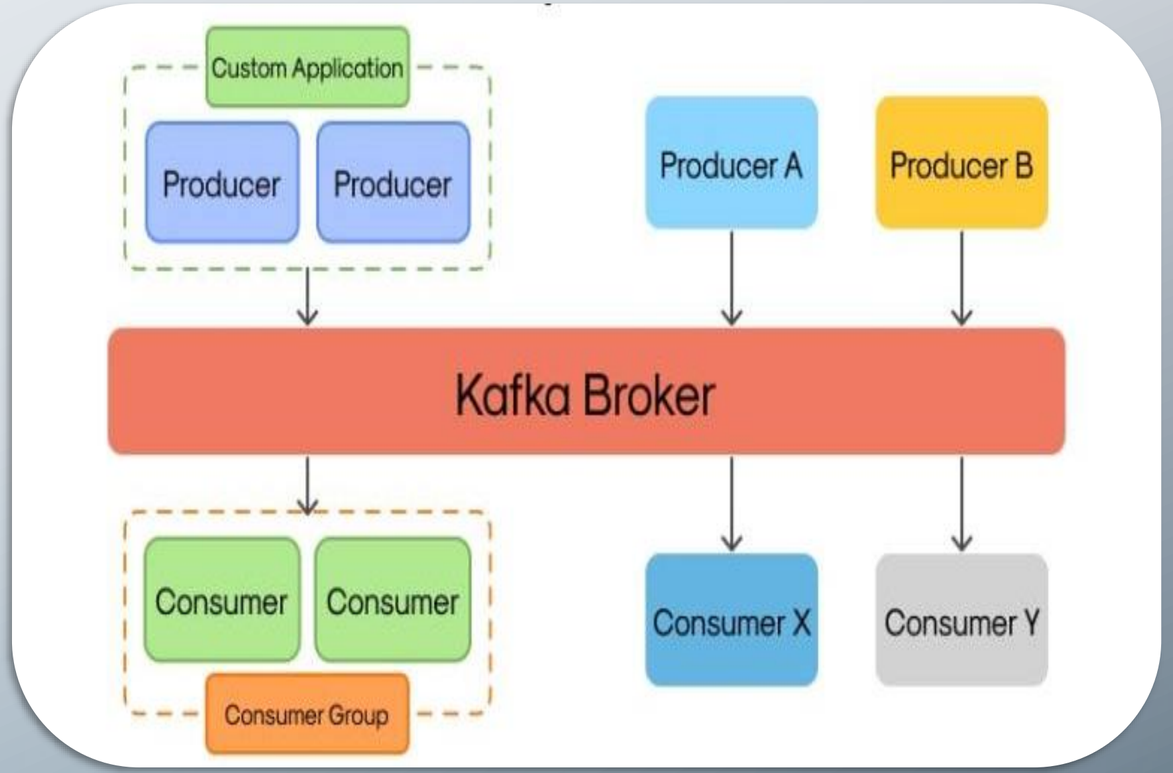
- **Топики (Topics)**
  Логические каналы, по которым передаются сообщения в Kafka.
  Каждый топик представляет собой поток данных, который может быть прочитан консюмерами.
  Топики могут содержать одну или несколько партиций.
- **Партиции (Partitions)**
  Части топиков, которые позволяют распределять нагрузку и обеспечивать параллельную обработку данных. Партиции позволяют масштабировать топики, распределяя данные между несколькими брокерами.
- **Брокеры (Brokers)**
  Серверы, которые хранят данные партиций и обрабатывают запросы от продюсеров и консюмеров. Брокеры могут быть организованы в кластеры для обеспечения надёжности и масштабируемости. Каждый брокер хранит одну или несколько партиций топиков.



# УСТРОЙСТВО APACHE KAFKA

## Основные компоненты

1. Продюсеры (Producers) → отправка сообщений.
2. Настройка acks (подтверждений) → определение количества брокеров, подтверждающих получение сообщений:
   - acks=0: продюсер не ждёт подтверждения от брокера. Это самый быстрый, но и самый ненадёжный способ;
   - acks=1: продюсер ждёт подтверждения от лидера партиции. Это баланс между производительностью и надёжностью;
   - acks=all: продюсер ждёт подтверждения от всех реплик партиции. Это самый надёжный, но и самый медленный способ.
3. Консюмеры (Consumers) → чтение сообщений.
4. Группы консюмеров → получение уникальных сообщений.
5. Оффсеты (Offsets) → определение позиции сообщения в партиции.
6. Брокеры (Brokers) → хранение данных и управление ими.
7. ZooKeeper → координация и синхронизация брокеров.

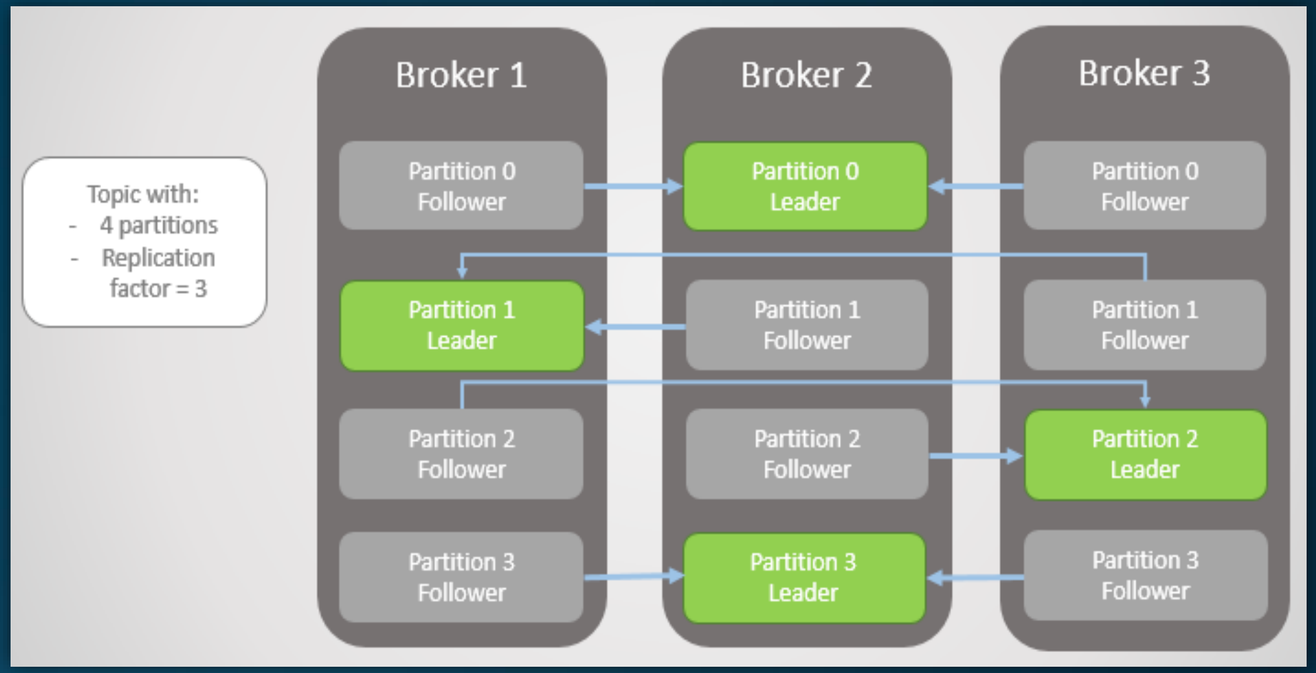


# МАСШТАБИРОВАНИЕ И НАДЁЖНОСТЬ

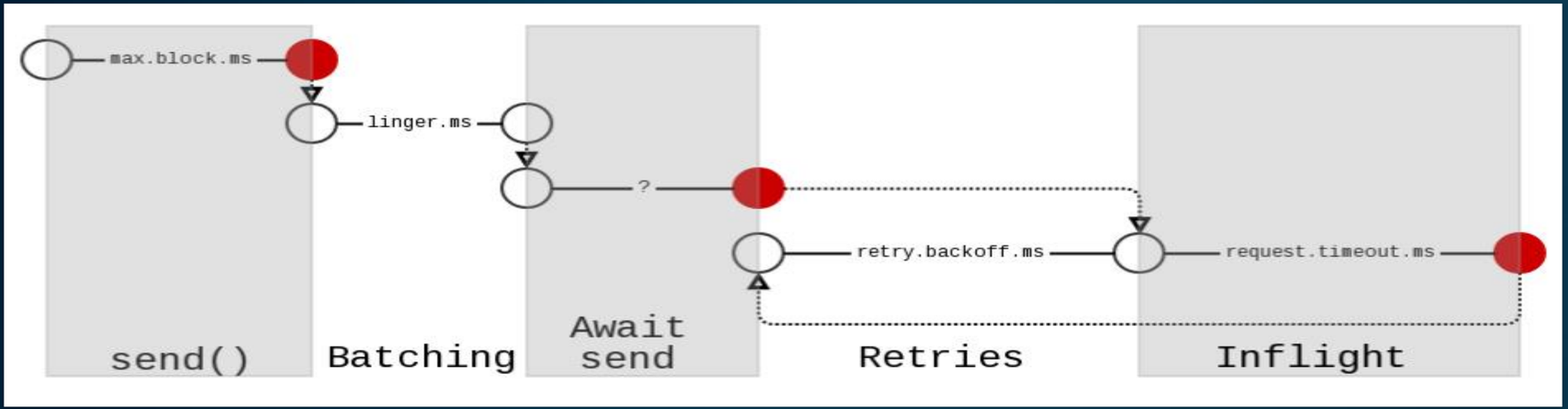
**Горизонтальное масштабирование → добавление новых брокеров в кластер**

**Балансировка нагрузки → оптимальное распределение данных и запросов между брокерами**

- Фактор репликации → определение количества реплик для каждой партиции.
- Настройка минимального количества синхронных реплик (Min ISR) → определение минимального количества реплик, которые должны быть синхронизированы с лидером партиции.



# МАСШТАБИРОВАНИЕ И НАДЁЖНОСТЬ




**Оптимизация производительности**

- Настройка размера батча (Batch Size) → определение количества сообщений, отправляемых продюсером в одном батче.
- Настройка времени ожидания батча (Linger Time) → определение времени ожидания продюсера перед отправкой батча сообщений.

# СИЛЬНЫЕ И СЛАБЫЕ СТОРОНЫ KAFKA

## Преимущества

- Высокая производительность и масштабируемость благодаря параллельной обработке сообщений;
- Гибкость в обработке потоков данных с возможностью интеграции с другими системами (Hadoop, Spark и т.д.). поддержка различных клиентов и языков программирования.
- Устойчивость к сбоям и возможность воспроизведения данных, что обеспечивает отказоустойчивость.
- Реактивная архитектура — позволяет строить приложения, реагирующие на события.

## Недостатки

- Сложность настройки — требует значительных усилий для правильной конфигурации.
- Управление потоками данных — может быть сложно при сложной архитектуре.
- Латентность — в некоторых сценариях может быть выше, чем у других систем.
- Ориентирован на задачи — не всегда подходит для простых очередей сообщений.
- Недостаток поддерживаемых форматов — ограниченные возможности по обработке сложных типов данных.

# ПАТТЕРНЫ И АНТИ ПАТТЕРНЫ KAFKA

## Паттерны использования Kafka

- 1. Стремление к событиям — используйте Kafka для построения событийно-ориентированной архитектуры, что позволяет улучшить взаимодействие между сервисами.
- 2. Микросервисы — применяйте Kafka как центральный узел для асинхронной передачи данных между микросервисами.
- 3. Логи и мониторинг — эффективно используйте Kafka для сбора логов и метрик, позволяя анализировать данные в реальном времени.
- 4. Интеграция с большими данными — комбинируйте Kafka с системами обработки данных, такими как Hadoop или Spark, для анализа потоков данных.
- 5. Хранение событий — реализуйте хранилище событий для длительного хранения и анализа исторических данных.

## Анти паттерны использования Kafka

- 1. Использование как простой брокер сообщений — Kafka предназначен для потоковой обработки, а не для простых очередей, что может привести к неправильному использованию его возможностей.
- 2. Неправильный выбор ключей — отсутствие хорошо продуманных ключей для партиционирования может создать проблемы с производительностью и балансировкой нагрузки.
- 3. Чрезмерная сложность — использование Kafka для слишком простых задач может привести к ненужной сложности в архитектуре.
- 4. Игнорирование ретеншн-политик — не управляя политиками хранения данных, можно столкнуться с проблемами нехватки дискового пространства.
- 5. Неаккуратное управление схемами — отсутствие схемы для сообщений может привести к несовместимости данных и ошибкам во время обработки.

# ДЕМОНСТРАЦИЯ ИНСТРУМЕНТОВ УПРАВЛЕНИЯ И КЛЮЧЕВЫХ ЭЛЕМЕНТОВ КОНФИГУРИРОВАНИЯ

# KAFKA TOOL

**Рассмотрим, как использовать Offset Explorer, а именно:**

- как создать новый топик с определённым количеством партиций и фактором репликации;
- как настроить права доступа для продюсеров и консюмеров, чтобы обеспечить безопасность данных.


**Kafka System Tools**

- Kafka Migration Tool
- Mirror Maker
- Consumer Offset Cheker

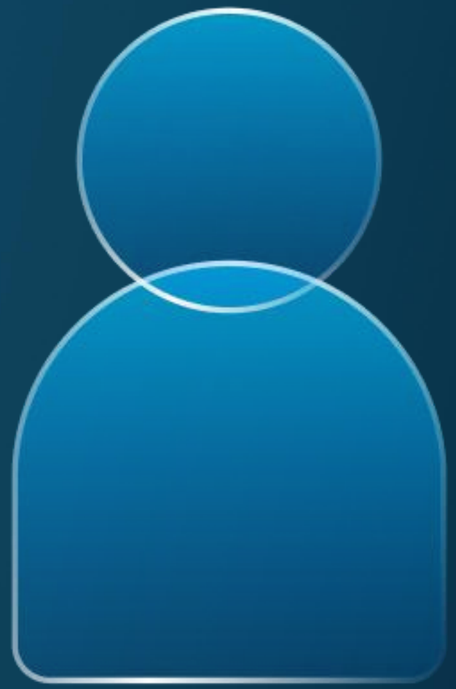
**Kafka Replication Tools**

- Create Topic Tool
- List Topic Tool
- Add Partition Tool

# ДЕМОНСТРАЦИЯ СПОСОБОВ РАБОТЫ С APACHE KAFKA

# КЛИЕНТ

**Рассмотрим, как настроить и использовать клиент для отправки и получения сообщений в Kafka на примерах, написанных на языке программирования Python.**



# СЕРВИС НА JAVA

**Рассмотрим, как создать сервис на Java, который будет взаимодействовать с Kafka.**

**Изучим примеры продюсеров и консюмеров, а также обработку ошибок и настройку конфигурации.**

# KAFKA TOOL

**Apache Kafka** — это мощный инструмент для потоковой передачи данных, который обеспечивает высокую производительность, надёжность и масштабируемость.

# РЕКОМЕНДАЦИИ

**Для закрепления полученных знаний и отработки навыков рекомендуем:**

- попробовать настроить собственный кластер Kafka;
- поэкспериментировать с различными настройками и сценариями использования.

Это поможет вам лучше понять возможности и особенности работы с Kafka.

# ВАШИ ВОПРОСЫ?


TI
МАИ