**Разработать архитектуру хранилища больших данных для заданного сценария использования в развитии малого и среднего бизнеса на основе применения технологии Big Data.**

**Задачи:**

**Шаг 1:** Провести анализ требований: определить источники данных; выявить типы данных (структурированные, полуструктурированные, неструктурированные); оценить объемы данных и скорость их поступления; определить требования к аналитике и отчетности.

**Шаг 2:** Выбрать компоненты архитектуры: выбрать систему распределенного хранения (например, Hadoop HDFS, Amazon S3); определить систему обработки данных (например, Apache Spark, Flink); выбрать систему управления метаданными (например, Apache Atlas); определить инструменты для ETL процессов (например, Apache NiFi, Talend); выбрать инструменты для визуализации и аналитики (например, Tableau, Power BI).

**Шаг 3:** Спроектировать архитектуру: разработать схему потоков данных; определить компоненты для обеспечения безопасности и управления доступом; спроектировать систему мониторинга и логирования; разработать стратегию масштабирования.

- Создать диаграммы архитектуры: использовать draw.io для создания визуального представления архитектуры; включить все основные компоненты и связи между ними.

- Описать компоненты: для каждого компонента архитектуры предоставить краткое описание его роли и функций.

- Обосновать выбор: объяснить причины выбора конкретных технологий и компонентов.

- Рассмотреть вопросы производительности и масштабируемости бизнеса: описать, как архитектура обеспечивает высокую производительность и масштабируемость.

Примерный алгоритм выполнения задания:

http://95.131.149.21/moodle/mod/assign/view.php?id=896

**Вариант 1. Крупный онлайн-ритейлер**

- Объем данных: 500 ТБ в год, рост 50% ежегодно.

- Скорость получения данных: до 5000 транзакций в секунду.

- Типы данных: 60% структурированные, 30% полуструктурированные, 10% неструктурированные.

- Требования к обработке: анализ поведения пользователей в реальном времени, прогнозирование спроса.

- Доступность: 99.99%, время отклика <5 секунд.

- Безопасность: шифрование, соответствие 152-ФЗ и PCI DSS

**Шаг 1: Определение требований**

* 1. Объем данных
     1. Ожидаемый объем 500 ТБ в год
     2. Ожидаемый рост: 50% ежегодно
  2. Скорость получения данных:
     1. Мобильный приложения и веб-сайты: до 5000 транзакция в секунду
     2. Социальные сети: обновление каждые 15 минут.
     3. CRM системы: в режиме реального времени.
  3. Типы данных:
     1. Структурированные: транзакционные данные (информация о продажах, покупках, возвратах, включая идентификаторы товаров, количество, цена, дата, время транзакции), данные о клиентах, складские данные, данные CRM – 60%
     2. Полуструктурированные: логи событий (данные о взаимодействиях пользователей с сайтом и мобильным приложением, включая время на странице, клики, навигацию), данные о продуктах (описание категорий атрибутов товаров в формате JSON/XML), обратная связь от клиентов – 30%
     3. Неструктурированные: о товарах (изображения и видео товаров), массовые рассылки (текстовые и мультимедийные данные с предложениями и акциями), посты в социальных сетях – 10%
  4. Требования к обработке:
     1. Прогнозирование спроса: в режиме реального времени.
     2. Анализ поведения пользователей: в режиме реального времени.
     3. Персонализация предложений: в режиме реального времени.
     4. Обработка транзакций: в режиме реального времени.
     5. Сегментация клиентов: ежемесячно
  5. Доступность данных:
     1. Время отклика для аналитических запросов: <5 секунд.
     2. Доступность системы: 99.99% (допустимое время простоя ~ 8.5 часов в год)
  6. Безопасность данных:
     1. Шифрование
     2. Соответствие 152-ФЗ и PCI DSS

**📘 152-ФЗ — Федеральный закон РФ №152 "О персональных данных"**

Это основной российский закон, регулирующий **сбор, хранение, обработку и защиту персональных данных граждан РФ**.

**Ключевые положения:**

* **Согласие пользователя** — необходимо получать добровольное согласие на обработку персональных данных.
* **Цель обработки** — данные можно собирать и использовать только с заранее определённой и законной целью.
* **Безопасность** — операторы должны принимать меры по защите данных от несанкционированного доступа, утечек, изменений и т. д.
* **Хранение в РФ** — в некоторых случаях персональные данные граждан РФ должны физически храниться на серверах внутри страны.

Пример: если у вас интернет-магазин и вы принимаете заказы с ФИО и телефонами, вы обязаны соблюдать 152-ФЗ.

**💳 PCI DSS — Payment Card Industry Data Security Standard**

Это международный **стандарт безопасности данных платёжных карт**, разработанный такими компаниями, как Visa, MasterCard, American Express и др.

**Основные цели:**

* Защита данных держателей карт (PAN, CVV и др.)
* Снижение рисков компрометации платёжной информации
* Обеспечение безопасного хранения, обработки и передачи платёжных данных

**Требования включают:**

* Использование **брандмауэров** и **шифрования**
* Контроль доступа к данным
* Регулярное **тестирование систем безопасности**
* Разделение обязанностей между сотрудниками

Пример: если вы принимаете оплату по картам на сайте или храните данные клиентов для повторной оплаты, вы должны соответствовать PCI DSS.

**Шаг 2: Выбор модели хранилища данных: компоненты архитектуры**

Выбор модели Hybrid Data Storage, т.к. несмотря на то, что 60% данных компании структурированные, все-таки остальные 40% занимают полуструктурированные и неструктурированные данные, для которых подойдет озеро данных.

Выбор схемы «Снежинка», т.к. она экономит место, что важно для данной компании, у которой по условию объем обрабатываемых и хранимых данных 500 ТБ в год, да еще и ожидается ежегодный рост этого значения на 50%. Тем более, «Снежинка» подразумевает нормализованные таблица измерений.

2.1. Источники данных

⎯ Мобильное приложение и веб-сайт.

⎯ Социальные сети (ВКонтакте, Telegram, Одноклассники, Tik Tok, Дзен).

⎯ CRM системы.

⎯ Платежные системы банков (обработка платежей на стороне банка).

⎯ Внешние API (данные о погоде для специализации предложений и рассылок, курс валют, данные о поведении клиента на сторонних приложениях или веб-сайтах, данные о браузерных запросах пользователей для персонализации подборок товаров).

2.2. Слой сбора данных

⎯ Apache Pulsar – для сбора потоковых данных.

⎯ Fluentd – для сбора логов.

2.3. Слой хранения данных

⎯ Apache Cassandra – для быстрого доступа к большим объемам данных, для хранения структурированных данных.

⎯ Amazon S3 – для хранения сырых данных, особенно при росте 50% в год.

⎯ PostgeSQL для структурированных данных.

2.4. Слой обработки данных

⎯ Apache Spark – для пакетной и потоковой обработки.

⎯ Apache Flink– для обработки в реальном времени.

⎯ Apache Hive – для SQL-подобных запросов к большим данным.

2.5. Слой аналитики и машинного обучения

⎯ Jupiter Notebook – для аналитик и ML

⎯ PowerBI – для визуализации и дашбордов.

⎯ Apache Druid – для аналитики в режиме реального времени.

2.6. Слой управления данными

⎯ Apache Ranger – для контроля доступа и аудита

2.7. Слой оркестрации и мониторинга

⎯ Apache Airflow – для оркестрации.

⎯ Prometheus – для мониторинга и алертинга.

**Объяснение выбора технологий**

## **1. Гибридное хранилище дан ных**

**Причина выбора:**

* В компании 60% — структурированные данные, 40% — полу- и неструктурированные. Требуется одновременно реляционное и масштабируемое объектное хранилище.

**Выбранные технологии:**

* **Apache Cassandra** — масштабируемая NoSQL-база с высокой скоростью чтения/записи. Отлично подходит для распределенного хранения структурированных данных, где требуется быстрый доступ.
* **Amazon S3** — облачное объектное хранилище для сырого, полу- и неструктурированного контента (JSON, изображения, логи). Масштабируется практически без ограничений.
* **PostgreSQL** — надежная реляционная СУБД, подходит для аналитических запросов к нормализованным таблицам (схема «Снежинка»), и обладает расширенной поддержкой JSON.

## **2. Слой сбора данных**

**Причина выбора:**

* Необходима поддержка масштабируемого потока событий и эффективный сбор логов.

**Выбранные технологии:**

* **Apache Pulsar** (или в схеме — JupyterHive как аналог) — высокопроизводительный брокер сообщений с поддержкой pub/sub и очередей. Масштабируется горизонтально и лучше справляется с многопоточностью по сравнению с Kafka.
* **Fluentd** — легковесный, расширяемый инструмент для сбора логов и маршрутизации данных. Поддерживает множество плагинов для интеграции.

## **3. Слой обработки данных**

**Причина выбора:**

* Обеспечить как пакетную, так и потоковую обработку с минимальной задержкой.

**Выбранные технологии:**

* **Apache Spark** — для высокопроизводительной пакетной обработки (ETL, агрегирование, трансформации).
* **Apache Flink** — для потоковой обработки в реальном времени (например, поведение пользователей, события).
* **Apache Hive** — SQL-интерфейс к большим данным, облегчает работу аналитиков и интеграцию с BI-инструментами.

## **4. Слой аналитики и машинного обучения**

**Причина выбора:**

* Необходима поддержка как интерактивного анализа, так и построения ML-моделей.

**Выбранные технологии:**

* **Jupyter Notebook** — де-факто стандарт для аналитиков и дата-сайентистов, поддерживает Python, ML-библиотеки (scikit-learn, TensorFlow, PyTorch).
* **Power BI** — мощный инструмент визуализации, хорошо интегрируется с PostgreSQL, Hive, и позволяет строить дашборды.
* **Apache Druid** — предназначен для OLAP-аналитики в реальном времени, обеспечивает мгновенный отклик на запросы.

## **5. Слой управления данными**

**Причина выбора:**

* Контроль доступа, аудит и безопасность критичны при работе с чувствительными данными.

**Выбранные технологии:**

* **Apache Ranger** — централизованный контроль доступа ко всем Hadoop-совместимым источникам. Гибкие политики безопасности.

## **6. Слой оркестрации и мониторинга**

**Причина выбора:**

* Нужна автоматизация пайплайнов и контроль состояния систем.

**Выбранные технологии:**

* **Apache Airflow** — лидер среди инструментов оркестрации. Поддерживает сложные зависимости, повторный запуск задач, визуализацию DAG’ов.
* **Prometheus** — система мониторинга и алертинга с высокой производительностью и поддержкой сбора метрик из практически любой системы.

## **7. Общая архитектура (Снежинка + масштабируемость)**

* **Снежинка** — экономит пространство хранения за счёт нормализации таблиц (актуально при росте объёмов).
* Поддерживает **горизонтальное масштабирование** по всем уровням: от источников до аналитики.

**Масштабирование**

**1. Гибридное хранилище данных (Hybrid Data Storage)**

* Использование **Apache Cassandra** и **PostgreSQL** для работы со структурированными данными обеспечивает **низкую задержку запросов** и **горизонтальное масштабирование** (особенно в Cassandra).
* **Amazon S3** применяется как масштабируемое хранилище для полу- и неструктурированных данных, с высокой устойчивостью и практически неограниченным объемом хранения.

**2. Потоковая и пакетная обработка данных**

* **Apache Spark** и **Apache Flink**:
  + Spark обеспечивает **параллельную пакетную обработку больших объемов данных**, работая в распределенной среде.
  + Flink — для **низколатентной обработки в реальном времени**, что критично для персонализации, динамических цен и мониторинга.
* Это разделение обработки позволяет достигать как **высокой производительности**, так и **гибкой масштабируемости** в зависимости от нагрузки.

**3. Масштабируемый сбор данных**

* **Apache Pulsar** и **Fluentd** масштабируются горизонтально, обрабатывая **миллионы событий в секунду**, поддерживая устойчивый поток логов и данных даже при пиковой нагрузке.

**4. Расширяемая аналитика и ML**

* Использование **Jupyter Notebook** и **Apache Druid**:
  + Обеспечивает **интерактивный анализ** и **аналитику в реальном времени**.
  + **Druid** масштабируется горизонтально, поддерживает быстрое агрегирование и фильтрацию по большим объемам данных.

**5. Оркестрация и управление**

* **Apache Airflow** обеспечивает **устойчивое планирование и оркестрацию процессов**, легко масштабируется при увеличении DAG-пайплайнов.
* **Prometheus** отслеживает нагрузку и метрики в реальном времени, позволяя **оперативно реагировать на узкие места** в производительности.

**6. Безопасность и управление доступом**

* **Apache Ranger** масштабируем для множества политик контроля доступа и аудита, без снижения производительности.

**7. Архитектурные особенности масштабируемости**

* **Микросервисный подход** в источниках данных и обработке позволяет масштабировать отдельные компоненты независимо.
* Возможность **автоматического масштабирования** в облаке (например, на Amazon S3 и с Pulsar/Flink) делает архитектуру гибкой к изменяющимся требованиям бизнеса.

