etl

Построения потока данных в организации.

Хэнделинг задач разных этапов потока данных: extract-transform-load, обработка, анализ.

Хэнделинг разных типов потоков данных.

| **Характеристика** | **ETL (традиционный подход)** | **Data Engineering (современный подход)** |
| --- | --- | --- |
| Фокус | Извлечение, трансформация, загрузка данных | Проектирование, построение и поддержка дата-систем |
| Инструменты | GUI-инструменты, визуальные пайплайны | Код (Python, Scala), Spark, Airflow, dbt и т.п. |
| Подход | Чаще монолитный и «ручной» | Автоматизированный, CI/CD, инфраструктура как код |
| Архитектура | Data Warehouse, ETL | Data Lake, ELT, streaming, lakehouse |
| Навыки | SQL, настройка ETL-инструментов | Программирование, DevOps, cloud, distributed systems |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Key words

Современные data инфраструктуры:

ETL-инструменты:

Apache Spark

Hadoop-кластеры (вероятно, HDFS + YARN)

Разделение ETL монолита на микросервисы

Монолит → микросервисы, каждый выполняет отдельную бизнес-функцию.

Это даёт:

Масштабируемость (горизонтально)

Устойчивость

Разделение ответственности по командам

Переезд с "железа" в Kubernetes

Физическая инфраструктура → кластер Kubernetes (k8s)

Позволяет:

Автоматизировать деплой

Легко масштабировать сервисы

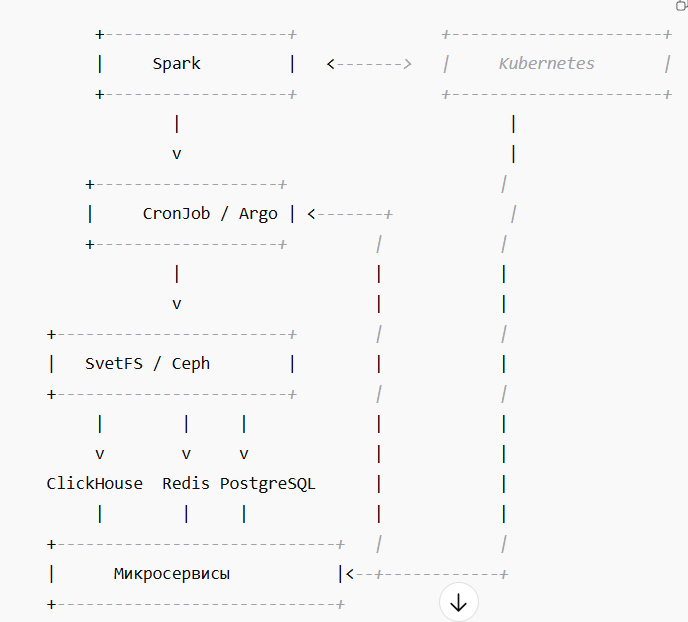
Использовать DevOps практики (CI/CD, ArgoCD и т.д.)

4. 🚛 ETL-инструменты остались, но стали облачными

* Spark остался, но теперь запускается в Kubernetes, не в YARN.  
  + Возможно, используется Spark-on-Kubernetes runtime.
* Это удобно:  
  + Автоматическое масштабирование
  + Использование стандартных образов контейнеров
  + Интеграция с остальной инфраструктурой (логгирование, мониторинг)

Регистрация и оркестрация — пока через CronJob, дальше — Argo

* Сейчас ETL и другие задачи регистрируются как Kubernetes CronJobs.
* Планируете перейти на Argo Workflows — мощный инструмент для оркестрации DAG-джоб в k8s.  
  + У соседей (команд или департаментов) он уже используется.
  + Позволяет красиво описывать пайплайны как YAML + UI.



Оркестраторы

# Сравнительная таблица для поверхностного понимания

| **Характеристика** | **Airflow** | **NiFi** | **K8s CronJob** | **Argo Workflows** |
| --- | --- | --- | --- | --- |
| 🔧 Тип системы | Оркестратор задач (DAG, Python) | GUI для потоков данных (Flow-based) | Планировщик задач в Kubernetes | Kubernetes-native оркестратор DAG |
| 🧠 Логика выполнения | DAG (Python-код) | Flow-based (GUI-конфигурации) | Bash / Pod / job | DAG (YAML/JSON, GitOps, event-driven) |
| 🎯 Основное назначение | ETL/ELT, ML pipeline, скедулинг | Data routing, enrichment, протоколы | Простые периодические задачи | Сложные DAG-пайплайны в Kubernetes |
| 📦 Контейнеризация | Частично, через Helm/Executor | Тяжело контейнеризуется | Полностью K8s-native | K8s-native |
| ⚙️ Поддержка DAG | Да (в коде на Python) | Частично, визуальные потоки | Нет | Да (YAML, GitOps) |
| 💻 UI | Да, веб-интерфейс | Да, drag-n-drop UI | Нет | Да, красивый UI |
| 🧩 Интеграции с источниками | Много коннекторов (PostgreSQL, S3 и т.д.) | Очень много протоколов (SFTP, MQTT) | Ручная настройка в образе | Через кастомные steps, templates |
| 📂 Работа с файлами / потоками | Плохо, нужно писать код | Отлично (stream, routing) | Только скриптами | Средне (можно, но не "из коробки") |
| 🔄 Запуск по событию | Ограниченно (через sensors) | Да | Нет (только по cron) | Да (через event-driven triggers, Webhooks) |
| 📜 Хранение истории | В БД (метаданные, логи) | Да (в своём хранилище) | Логи только в Pod/Log system | Да (через CRD и хранилище Kubernetes) |
| 🚀 Масштабируемость | Средняя (scheduler может быть узким местом) | Плохо масштабируется | Отличная (Kubernetes-native) | Отличная (спроектирован для Kubernetes) |
| 🔒 Безопасность и доступ | Хорошая (RBAC, auth) | Много уязвимостей без донастроек | Полностью под контролем k8s RBAC | Kubernetes RBAC, токены, OIDC |
| 🧠 Порог вхождения | Средний (Python + DAG API) | Низкий для простых задач (GUI), высокий для прод | Очень низкий (shell-скрипты) | Средний (YAML + понимание Kubernetes) |

# Сценарии использования

| **Сценарий** | **Лучший выбор** | **Почему** |
| --- | --- | --- |
| Простой ETL раз в день | **Airflow / Cron** | Airflow — с логами, Cron — проще всего |
| Сложный DAG с ML-тренировкой | **Airflow / Argo** | DAG + контроль зависимостей |
| Потоковые данные (IoT, REST, Kafka) | **NiFi** | Встроенные коннекторы, визуальные потоки |
| Простые cron-задачи внутри k8s (backup, Spark run) | **K8s CronJob** | K8s-native, без лишнего веса |
| GitOps-архитектура пайплайнов | **Argo Workflows** | YAML-пайплайны, версионирование, CI/CD-friendly |
| Интеграция с BI/аналитикой | **Airflow** | Поддержка SQL/S3/BigQuery/PostgreSQL из коробки |
| Выгрузка/обогащение данных из FTP/SFTP/REST | **NiFi** | Поддержка всех источников без программирования |
| Комплексная ETL-платформа с историей, retry и alerting | **Airflow** | Отработанный стек с метаданными и логированием |

# Итого

| **Хочешь...** | **Выбор** |
| --- | --- |
| Простоту, минимум зависимостей | **CronJob** |
| Управляемый оркестратор для DAG | **Airflow** |
| Визуальные потоки данных | **NiFi** |
| GitOps, Kubernetes, CI/CD и DAG | **Argo** |