OTUS

Data Engineer

Лучшие практики по приготовлению данных. Загрузка, обработка, организация хранения и доступа к данным с использованием современных инструментов

Длительность курса: 115 академических часов

1 Введение

1 Инженер данных. Задачи, навыки, инструменты, потребность на рынке

Цели занятия:

познакомиться и обсудить правила работы и общения;
выяснить, кто такой инженер данных;
сформулировать задачи инженера данных и ожидания бизнеса.

Краткое содержание:

Data-driven компании;
роль аналитики в компаниях;
профессии работающие с данными;
Data Engineer: потребность и ценность, задачи и навыки;
создание ценности и основные вызовы (challenges).

Домашние задания

1 Список навыков Data Engineer

Цель: В этом ДЗ, вам нужно сделать обзор вакансий Data Engineer по нескольким ресурсам (hh.ru, работа.ру и д.р.).
Составить список требуемых навыков, продуктов и технологий.

Найти и изучить несколько ресурсов с вакансиями Data Engineer (hh.ru, работа.ру и д.р.).
Выписать требования к навыкам и опыту работы с продуктами и технологиями.
Обобщить.

2 **Архитектура систем обработки данных**

Цели занятия:

проследить эволюцию подходов работы с данными;
получить представление о технологиях и инструментах;

Краткое содержание:

развитие ИТ;
появление Big Data;
Lambda и Kappa архитектуры;
современные архитектуры обработки данных;

---

3 **On-premise и облака**

Цели занятия:

выяснить что такое on-premise;
научиться использовать облачную инфраструктуру и понимать отличия от on-premise;

Краткое содержание:

варианты развёртывания: on-premise, private cloud, public cloud;
облачные решения: IaaS, PaaS, SaaS;
отличия от on-premise. Yandex.Cloud;

---

4 **Docker**

Цели занятия:

обсудить контейнеризацию;
рассмотреть компоненты Docker;

Краткое содержание:

контейнеризация: краткий обзор;
компоненты docker: engine, cli, registry;

Домашние задания

1 Регистрация аккаунта в облаке

Цель: Зарегистрироваться в Yandex.Cloud. Создать Linux VM. Установить в ней Docker. Запустить простой образ.

1. Создайте аккаунт в Яндекс-почте или используйте уже существующий (https://yandex.ru);
2. Подключите двухфакторную авторизацию для вашего аккаунта (https://yandex.ru/support/passport/authorization/twofa-on.html);
3. На главной странице Яндекс-облака (https://cloud.yandex.ru) нажмите кнопку "Подключиться";
4. На стартовой странице вашей консоли (https://console.cloud.yandex.ru/) выберете пункт "Биллинг" (https://console.cloud.yandex.ru/billing);
5. На странице "Биллинг" выберете пункт "Список аккаунтов" и создайте новы платёжный аккаунт;
6. Перейдите в созданный платёжный аккаунт и в разделе "Обзор", нажмите кнопку "Активировать промокод" и введите полученный промокод;
7. Зайдите на дашборд каталога, выберете "Compute cloud" и нажмите "Создать ВМ";
8. Создадите VM с Ubuntu;
9. Добавьте SSH-ключ в форме конфигуратора виртуальной машины;
10. Запустите VM, подключитесь к ней и установите Docker Engine по инструкциям:
- https://docs.docker.com/engine/install/ubuntu/
- https://docs.docker.com/engine/install/linux-postinstall/
11. Запустить простой образ: docker run hello-world
12. Сделать снимок экрана и приложить в качестве результата выполнения задания.

2 Data Lake

1 Что такое Data Lake

Цели занятия:

узнать что такое Data Lake, как надо строить Data Lake;
познакомиться с Hadoop и его компонентами;

Краткое содержание:

понятие Data Lake;
построение Data Lake;
знакомство с Hadoop;

Домашние задания

1 Hadoop в Docker

Цель: Запустить Hadoop в Docker. Настроить службы.

1. Убедитесь, что у Linux VM, созданной в предыдущем ДЗ не менее 16 GB RAM
2. Запустите VM
3. Запустите Hortonworks HDP Sandbox (https://hub.docker.com/r/hortonworks/sandbox-hdp) в Docker:
- https://www.cloudera.com/tutorials/sandbox-deployment-and-install-guide/3.html
- https://www.cloudera.com/tutorials/sandbox-architecture.html
- https://www.cloudera.com/tutorials/learning-the-ropes-of-the-hdp-sandbox.html
4. Проверить и запустить службы HDFS, YARN, Hive, Spark2, Zeppelin Notebook и Data Analytics Studio

2 **Форматы данных и их особенности**

Цели занятия:

изучить факторы выбора формата хранения данных;
понять разницу между Row-based и Column-based форматами;
познакомиться с наиболее распространенными форматами;

Краткое содержание:

особенности форматов файлов;
текстовые форматы;
бинарные форматы;

---

3 **Распределенные файловые системы.**

Цели занятия:

узнать зачем нужны распределённые файловые системы и познакомиться с их особенностями;
понять как устроен HDFS и научиться с ней работать;
познакомится с объектными хранилищами на примере S3;

Краткое содержание:

распределённые файловые системы;
архитектура HDFS;
работа с HDFS с помощью CLI;
Object Storarge, сходство и отличия от HDFS на примере S3;

---

4 **SQL-доступ к Hadoop**

Цели занятия:

понять зачем SQL для работы с Hadoop;
узнать что такое Hive и как с ним работать;

Краткое содержание:

потребность в SQL для работы с Hadoop;
архитектура Hive и модели данных. HiveQL;

---

5 **Apache Spark. Архитектура приложения**

Цели занятия:

познакомиться со Spark;
узнать как развернуть Spark, запускать задания и работать интерактивно;

Краткое содержание:

что такое Spark;
архитектура приложений;
запуск заданий;
интерактивная работа;

---

6 **Apache Spark. API, Оптимизация**

Цели занятия:

познакомиться со Spark API;
узнать как оптимизировать приложения и решать проблемы;

Краткое содержание:

Spark API: RDD, DataFrame;
Dataset, Pandas API;
оптимизация и решение проблем;

Домашние задания

1 Сборка витрины на Spark

Цель: В этом задании предлагается собрать статистику по криминогенной обстановке в разных районах Бостона. В качестве исходных данных используется датасет https://www.kaggle.com/AnalyzeBoston/crimes-in-boston

Цель задания - разработать программу построения витрины.
Результат - ссылка на репозиторий с кодом.

Программа должна запускаться через spark-submit.
Пути к данным и к результату должны передаваться в качестве аргументов вызова.

Инструкция:
1) Загрузить данные.

2) Проверить данные на корректность, наличие дубликатов. Очистить.

3) Собрать витрину (агрегат по районам (поле district)) со следующими метриками:
- crimes_total - общее количество преступлений в этом районе;
- crimes_monthly - медиана числа преступлений в месяц в этом районе;
- frequent_crime_types - три самых частых crime_type за всю историю наблюдений в этом районе, объединенных через запятую с одним пробелом “, ” , расположенных в порядке убывания частоты;
- crime_type - первая часть NAME из таблицы offense_codes, разбитого по разделителю “ - ” (например, если NAME “BURGLARY - COMMERICAL - ATTEMPT”, то crime_type “BURGLARY”);
- lat - широта координаты района, рассчитанная как среднее по всем широтам инцидентов;
- lng - долгота координаты района, рассчитанная как среднее по всем долготам инцидентов.

4) Сохранить витрину в один файл в формате .parquet в папке path/to/output_folder.

Подсказки:
- Функция percentile_approx может посчитать медиану.
- Конкретный месяц идентифицируется не только номером месяца, но и номером года.
- В справочнике кодов есть дубликаты. Нужно выбрать уникальные коды, взяв любое из названий.

7 **Потоковая обработка данных**

Цели занятия:

познакомиться с потоковой обработкой данных;
познакомиться с Kafka;

Краткое содержание:

потоковая обработка. Архитектура Kafka;
работа с Kafka;

3 DWH

1 Что такое DWH

Цели занятия:

узнать что такое DWH;
познакомится с подходами к проектированию;

Краткое содержание:

понятие хранилища данных;
почему появились DWH и какие задачи решают;
отличия OLAP от OLTP;
подходы к проектированию: "по Инмону", "по Кимбалу";

2 **Проектирование DWH**

Цели занятия:

познакомиться с этапами проектирования;
получить базовые знания реляционной модели;
получить базовое понимание Dimensional Modeling;
получить базовое понимание Data Vault;
получить базовое понимание Anchor Modeling;

Краткое содержание:

ER-диаграммы;
нормальные формы;
Dimensional Modeling;
Data Vault;
Anchor Modeling;

Домашние задания

1 Проетирование хранилища данных по модели Data Vault.

Цель: Из 4 источников данных (таблицы во операционных базах) спроектировать три слоя хранилища данных (STG, ODS и DDS) по модели Data Vault.

Из 4 источников данных (таблицы в операционных базах) спроектировать три слоя хранилища данных (STG, ODS и DDS) по модели Data Vault.

---

3 **Аналитические MPP-базы данных**

Цели занятия:

изучить архитектуру кластерных аналитических СУБД;
изучить примеры реализации MPP-баз данных;

Краткое содержание:

MPP-базы данных: что это такое, для чего они нужны и их особенности;
обзор Greenplum;

---

4 **ClickHouse**

Цели занятия:

узнать, что такое ClickHouse и понять области его применения;
понять как начать работу с ClickHouse и где искать ответы на свои вопросы;

Краткое содержание:

введение в ClickHouse;
развёртывание ClickHouse;
работа с ClickHouse;

---

5 **Развертывание BI-решения**

Цели занятия:

сформулировать назначение систем класса BI;
изучить принципы работы BI-инструментов и решаемые им задачи;
сравнить и выбрать BI-решение для конкретного кейса;

Краткое содержание:

обзор популярных BI-решений;
варианты развертывания: self-hosted vs managed;
задание метрик, фильтров, сегментов;
сборка аналитических дашбордов: лучшие практики;

Домашние задания

1 Витрина + BI

Цель: 1. Установить Metabase:

Использовать виртуальную машину и Docker в Yandex.Cloud.

2. Подключиться к источнику данных.

Использовать любой доступный источник данных:

- Jaffle shop (postgres) из занятия по dbt.
- Любой датасет из блока Data Lake.
- ClickHouse Playground: https://clickhouse.tech/docs/en/getting-

started/playground/
- Любой другой.

3. Создать дашборд:

- Несколько видов визуализации (таблицы, графики, числа).
- Фильтры.
- Блок с комментариями.

Выслать скриншоты своего дашборда с комментариями.

https://gist.github.com/kzzzr/0d9ab5898866de3db2b66c79c111d967

---

6 **Разбор ДЗ по 1 и 2 модулю**

Цели занятия:

сравнить своё решение ДЗ из модуля 1 и 2 с эталонным;

Краткое содержание:

пример решения ДЗ из модулей 1 и 2;
основные нюансы и часто встречающиеся ошибки;

4 NoSQL

1 **NoSQL. Key-Value**

Цели занятия:

познакомиться с NoSQL: что это, предпосылки к появлению, особенности;
изучить отличия NoSQL от SQL систем;
изучить классификацию NoSQL систем: Key-Value, Wide-column, Document-oriented, Графовые (Graph);

Краткое содержание:

концепция NoSQL;
классификация NoSQL СУБД;

---

2 **NoSQL. Wide-column**

Цели занятия:

познакомиться с Wide-column;
познакомиться с HBase;
познакомиться с Cassandra;

Краткое содержание:

обзор HBase;
обзор Cassandra;

---

3 **NoSQL. Document-oriented**

Цели занятия:

познакомиться с Document-oriented хранилищами;

Краткое содержание:

особенности Document-oriented баз;
обзор MongoDB;

---

4 **Разбор ДЗ по 3 модулю**

Цели занятия:

сравнить своё решение ДЗ из модуля 3 с эталонным;

Краткое содержание:

пример решения ДЗ из модуля 3;
основные нюансы;
часто встречающиеся ошибки;

---

5 **ELK**

Цели занятия:

познакомиться с основными концепциями Elasticsearch;
рассмотреть возможности обработки данных с использованием Logstash;

Краткое содержание:

основные концепции;
основы анализа и поиска;

Домашние задания

1 Анализ веб-логов с помощью ELK

Цель: В рамках ДЗ нужно будет загрузить данные в ElasticSearch и построить дашборд в Kibana.

https://github.com/Gorini4/elk_demo - в README репозитория содержится полная инструкция с подсказками.

1. Склонируйте репозиторий в директорию elk_demo
2. Зайдите в эту директорию и разверните инфраструктуру, выполнив в терминале docker-compose up
3. Отредактируйте файл clickstream.conf
4. Загрузите данные веб-логов, выполнив команду ./load_data.sh
5. Перейдите по адресу http://localhost:5601 и создайте отчет (dashboard), показывающий распределение запросов с разными кодами ответов (status_code) по времени

1 **Выгрузка данных из внешних систем**

Цели занятия:

изучить классификацию источников для выбора правильного способа загрузки;
научиться загружать источники с помощью NiFi;

Краткое содержание:

типы, форматы источников и способы их загрузки;
использование Apache Ni-Fi в качестве инструмента интеграции данных;

Домашние задания

1 Загрузка сырых данных

Цель: В данном ДЗ мы настроим data pipeline в Apache NiFi, который будет:

- получать данные, отправляемые по HTTP (по сути, работать как API);
- анализировать содержимое того, что ему прислали;
- выполнять некоторые преобразования (склейку);
- сохранять результат в отдельную директорию в соответствии с контентом.

Конечный продукт:

1) работающий pipeline NiFi в контейнерном окружении;
2) примеры curl-команд, приводящих к разным результатам;
3) текстовые данные в соответствующих директориях контейнерной файловой системы.

1) Скачиваем себе образ Apache NiFi (если еще не сделали этого на занятии):

docker pull apache/nifi:latest

2) Собираем контейнер с mapping-ом 2-х портов: через который будем локально смотреть в интерфейс и через который будем общаться через

curl:

docker run --name nifi -p 9090:9090 -p 8081:8081 -d -e NIFI_WEB_HTTP_PORT='9090' apache/nifi:latest

3) Создаем стартовую точку pipeline-а:

ListeHTTP-процессор.
Порт: 8081 (его мы заранее мэппили для docker-контейнера)
Base Path: loglistener

4) Второй узел. Он будет перенаправлять данные в соответствии с тем, что пришло на наш веб-сервер извне:

RouteOnContent-процессор.
Подключаем его связью "success" к ListeHTTP.
Match Requirement: "content must contain match"
Добавляем ему новое свойство errortext со значением "ERROR".
Теперь из узла способно выходить 2 потока: для flowfiles с ключевым значением (под названием errortext) и второй: unmatched

5) Проверяем работоспособность 1-го узла:

Нажимаем "Play" для ListeHTTP-процессора
Из командной строки делаем: " curl --data "someRandomText" 127.0.0.1:8081/loglistener"
Жмем Refresh (после клика ПКМ)
Видим, что в него пришли данные

6) Добавляем узел, который будет объединять несколько файлов в один:

Добавляем MergeContent-процессор.
Создаем 2 связи из RouteOnContent в MergeContent: errortext и unmatched
Настраиваем узел (ПКМ => configure):
* Scheduling -> Run Schedule: 10 sec
* Merge Strategy: Bin-Packing Algorithm
* Correlation Attribute Name: RouteOnContent.Route - этот параметр позволит нам писать файлы в разные директории в дальнейшем
* Merge Format: Binary Concatenation
* Maximum Number of Entries: 500
* Maximum Bin Age: 90s (максимальное время жизни flow-файла, после которого его принудительно выведет из pipeline)

Мы будем склеивать текстовые файлы, чтобы не плодить их в большом количестве.
Поэтому укажем формат склейки:
* Delimiter Strategy: Text
* Demarcator: открываем и в поле нажимаем Shift+Enter (это аналог \n- перевод каретки на новую строку)

7) Сохраняем полученные артефакты в облачный s3 Yandex.Cloud:

PutS3Object-процессор
* Directory: nifiHW/${RouteOnContent.Route}/${now()}.txt

Создаем бакет nifiHW через интерфейс.
У нас будет 2 папки в s3: merged и unmatched
Делаем связи от узла MergeContent вида "merged" и "unmatched"

Дополнительно требуется разобраться с тем, как начать писать в бакет через API.

8) Запускаем весь pipeline

9) Шлем много разных запросов через cli: содержащих текст ERROR внутри и нет.

Примеры:

curl --data "adbrsgbndt ERROR fevrtb" 127.0.0.1:8081/loglistener
curl --data "uiebveovne" 127.0.0.1:8081/loglistener

10) Наблюдаем за тем, что происходит внутри pipeline-а.

11) Заходим через интерфейс в s3 Yandex.Cloud и смотрим, какие новые папки и файлы появились внутри.

2 **Автоматизация пайплайнов и оркестрация – 1**

Цели занятия:

научиться использовать оркестраторы для автоматизации процессов обработки данных;
написать DAG'и для создания сложных процессов обработки, состоящих из множества этапов;

Краткое содержание:

назначение оркестраторов;
сравнение самых популярных оркестраторов - Oozie и Airflow;
устройство Airflow и способы его использования для построения сложных процессов обработки данных;

---

3 **Автоматизация пайплайнов и оркестрация – 2**

Цели занятия:

научиться строить более сложные пайплайны Airflow;
понять архитектуру Airflow и способы деплоя;
научиться выполнять базовые операции с AF: тестирование, бэкфилл;

Краткое содержание:

сенсоры, сабдаги и другие специальные операторы;
архитектура Airflow - webserver, scheduler, worker;
использование Airflow CLI;

Домашние задания

1 Подготовка и установка на расписание DAG выгрузки данных из источников

Цель: В данном ДЗ мы настроим автоматический data pipeline, который будет получать данные из публичного API и складывать их в БД для дальнейшего анализа.
Конечный продукт:
1) Работающий облачный инстанс Apache Airflow.
2) Data pipeline, содержащий в себе несколько task-ов и "крутящийся" по расписанию Airflow.
3) Работающий облачный инстанс СУБД, куда Airflow заливает данные, получаемые из внешнего

API.
4) Данные в СУБД.

Часть из операций мы разбирали на занятии. При необходимости - можно пересмотреть запись.

1) Создаем Виртуальную машину с Apache Airflow 2.0 в Yandex-Cloud.

2) Создаем "managed instance" PostgreSQL/ClickHouse/MySQL - по выбору в Yandex-Cloud.
Создаем БД "analytics".

3) Добавляем наш psql в Connections через UI Airflow.

4) Выбираем один из 2-х API, с которым будем работать:
- Положение Международной Космической Станции на текущий момент времени (timestamp-latitude-longitude). Source: http://api.open-notify.org/iss-now.json
- Курс BTC: https://docs.coincap.io/#2a87f3d4-f61f-42d3-97e0-3a9afa41c73b
Тут нас интересует следующий endpoint: "api.coincap.io/v2/rates/bitcoin"

5) Создаем схему данных (таблицу в бд analytics) с названиями и типами полей, релевантными тому, что будем забирать из API.

6) Описываем DAG (программируем на Python) для получения данных с периодичностью 30 min.
Сам оператор для обращения к API можно выбрать любой.
Для простоты рекомендуется слать GET-запросы через python-библиотеку requests (`PythonOperator`), либо через bash (`BashOperator`) с помощью curl.

Концептуальная схема Dag-а: отправить запрос в API->распарсить пришедший результат->положить данные в БД (сделать insert в таблицу)

7) Кладем .py файл с DAG-ом в нужную директорию виртуалки с airflow.

8) Запускаем DAG тумблером в UI Airflow.

		9) Отлаживаем DAG до работоспособного состояния. В интерфейсе Airflow (облачный инстанс) есть информация о наборе успешно завершившихся Dag runs (темно-зеленые кружки) + в БД есть данные за >5 периодов времени. В качестве проверки мы зайдем в вам airflow webserver и отправим sql-запрос в таблицу с данными. 10) Проверяем, что данные появились в БД. 11) Поздравляю, вы завершили ДЗ!
4	**Data Quality. Управление качеством данных**	**Цели занятия:** получить представление о том, что такое качество данных; выяснить как Data Quality влияет на выводы и принимаемые решения; рассмотреть стратегии управления качеством данных; **Краткое содержание:** основные метрики качества данных; причины нарушения качества и стратегии реагирования; измерение, мониторинг, исправление; демонстрация: тесты актуальности, кросс-проверки источник <-> DWH;

5	**DevOps практики. CI + CD**	**Цели занятия:** изучить основы DevOps-практик; рассмотреть подходы к построению CI-CD pipelines применительно к аналитическим задачам; продемонстрировать несколько инструментов: Jenkins, Github Actions; **Краткое содержание:** культура DevOps; работа в команде; Continuous Integration / Continuous Delivery;
6	**Управление метаданными**	**Цели занятия:** научиться собирать метаданные в каталог данных; **Краткое содержание:** что такое метаданные; источники метаданных; Data Catalog и Data Discovery; Atlas.
7	**Мониторинг**	**Цели занятия:** рассмотреть инструменты мониторинга: Prometheus, Zabbix, Graphite, Grafana; понять специфику мониторинга процессов обработки данных; **Краткое содержание:** для чего нужен мониторинг и какие проблемы он решает; паттерны и антипаттерны построения систем мониторинга; сравнение решений представленных на рынке;

8 **Разбор ДЗ по 4 и 5 модулю**

Цели занятия:

сравнить своё решение ДЗ из модулей 3 и 4 с эталонными решениями;

Краткое содержание:

примеры решения ДЗ из модуля 4 и 5;
основные нюансы;
часто встречающиеся ошибки;

6 Выпускной проект

1 Выбор темы и организация проектной работы

Цели занятия:

выбрать и обсудить тему проектной работы;
спланировать работу над проектом;
ознакомиться с регламентом работы над проектом.

Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;
требования к результату проекта и итоговой документации.

Домашние задания

1 Проектная работа

Цель: В качестве курсового проекта необходимо придумать дизайн и архитектуру, а затем - имплементировать data-driven приложение в выбранной доменной области.
Работающее приложение должно быть представлено в качестве репозитория в GitHub.

Подробнее см. в фале: https://docs.google.com/document/d/1Onz9oWn_TfdWZmBhRCOIUAyKKLX3gXqcliMLdhzBoow/edit?usp=sharing

---

2 Консультация по проектам и домашним заданиям

Цели занятия:

получить ответы на вопросы по проекту, ДЗ и по курсу.

Краткое содержание:

вопросы по улучшению и оптимизации работы над проектом;
затруднения при выполнении ДЗ;
вопросы по программе.

3	**Защита проектных работ**	**Цели занятия:** защитить проект и получить рекомендации экспертов. **Краткое содержание:** презентация проектов перед комиссией; вопросы и комментарии по проектам.