> РАБОЧЕЕ ОКРУЖЕНИЕ

> Об инструментах

Прежде всего давайте познакомимся с рабочим окружением и инструментами, которые мы будем использовать для работы с данными:

- ClickHouse — колоночная база данных для хранения пользовательских событий;
- Redash — инструмент для написания SQL-запросов и базовой визуализации;
- Superset — полноценная BI-система;
- Jupyter Lab — среда для написания кода на Python;
- GitLab — репозиторий для хранения кода.

Теперь давайте настроим все инструменты и посмотрим, как они связаны друг с другом.

> ClickHouse и Redash

Начнём с ClickHouse и Redash. Перейдите по ссылке и войдите в Redash. Давайте посмотрим, к каким таблицам в ClickHouse у нас есть доступ. Выберите нужную схему данных и напишите первый простой запрос, чтобы посмотреть, как сохраняются события просмотров постов.

```
SELECT
      *
FROM simulator.feed_actions
WHERE toDate(time) = today() AND action = 'view'
ORDER BY time DESC
LIMIT 10
```

Давайте посмотрим на последние 10 событий, которые наши пользователи совершили, просматривая посты.

Каждый раз, когда пользователь просматривает или лайкает пост, соответствующее событие сохраняется в таблицу. Поэтому она постоянно пополняется новыми данными об активности пользователей, которые и днём и ночью пользуются нашим приложением.

Вторая таблица, к которой у нас есть доступ, — это данные об отправке сообщений. Изучите эту таблицу самостоятельно с помощью запросов.

При помощи Redash вы можете не только удобно смотреть результаты выполнения запросов, но и визуализировать полученные данные. Давайте посмотрим, как распределилась активность пользователей, просматривавших посты в течение вчерашнего дня.

Чтобы получить количество просмотров по 15-минутным интервалам за вчерашний день, выполним следующий запрос:

```
SELECT
    toStartOfFifteenMinutes(time) as t,
    count(user_id)
FROM simulator.feed_actions
WHERE toDate(time) = yesterday() AND action = 'view'
GROUP BY t
```

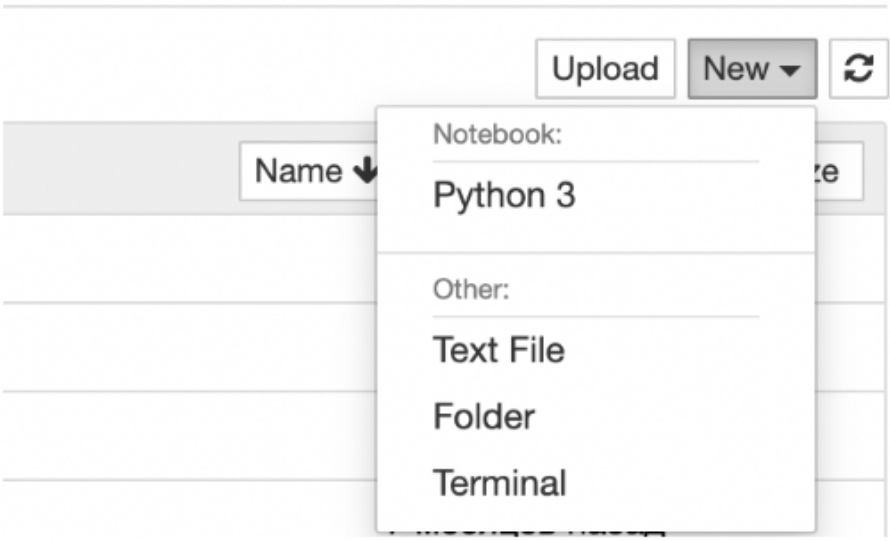
Перейдём в настройки графиков и визуализируем результат. Отложим время по оси `X` и число событий по оси `Y`. Легко заметить суточный паттерн использования приложения. Пик активности приходится на вечер, при этом ночью и утром приложением пользуются не так часто.

Попробуйте самостоятельно поискать какой-нибудь паттерн использования приложения в течение недели.

> Jupyter Lab и GitLab

Теперь давайте разберёмся, как и где мы будем писать код на Python. Для этого необходимо авторизоваться в Jupyter. Для начала давайте склонируем репозиторий, в котором уже есть несколько полезных функций для работы с данными.

Создайте новый терминал, нажав на меню `New` в верхнем правом углу, и сгенерируйте пару ssh ключей: приватный и публичный. Они понадобятся для работы с удалённым репозиторием в GitLab:



Выполните в терминале команду `ssh-keygen -t ed25519`. В результате появится путь, по которому вам предложат сохранить ключи. Согласитесь и нажмите Enter:

```
jupyter-an.karpov@lab:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/jupyter-an.karpov/.ssh/id_ed25519):
```

Дважды введите пароль, который потребуется для использования пары ключей. Обратите внимание, что вводимые символы не видны в терминале из соображений безопасности.

Если вы получили следующее изображение, генерация ключей прошла успешно:

```
The key's randomart image is:
+--[ED25519 256]--+
|                 |
|          . o    |
|         . o = . |
|        . o + = +|
|         S o *oo.=|
|          = .Eo%+.|
|         + .o.# *.|
|     .  . o o= o *|
|      oo.o  oo  o*|
+----[SHA256]-----+
jupyter-an.karpov@lab:~$
```

Напечатайте и скопируйте публичную часть ключа:

`cat ~/.ssh/id_ed25519.pub`

Публичный ключ выглядит так:

ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIMnhwMxD8ju8VY3uF5kzEbDQSwkNqy0trW79luay6UHF

Важно: никогда не выкладывайте в открытый доступ приватную часть ключа, это может привести к утечке ваших данных!

Теперь необходимо добавить публичную часть ключа в GitLab, чтобы клонировать репозиторий. Авторизуйтесь в GitLab — здесь вам будет доступен репозиторий `analyst_simulator`, в котором мы будем хранить код для работы с данными. Вы можете изучить, какие файлы и скрипты уже есть в репозитории.

Пропишите публичный ssh ключ в настройках GitLab. Скопируйте команду для клонирования репозитория по ssh, вернитесь в терминал и склонируйте репозиторий.

Теперь в файловом менеджере Jupyter появилась папка analyst_simulator, в которой хранится код из нашего репозитория. Создайте новый Jupyter Notebook в этой папке, импортируйте pandas, уже имеющийся модуль для подключения к ClickHouse и попробуйте выполнить какой-нибудь запрос к ClickHouse таблице прямо из Jupyter Notebook.

Например, давайте загрузим топ 10 постов и их статистику за вчерашний день и сохраним результат в Pandas Dataframe:

```
import pandas as pd
import pandahouse as ph

connection = {'host': 'http://clickhouse.beslan.pro:8080',
                      'database':'simulator',
                      'user':'student',
                      'password':'dpo_python_2020'
                     }

query = '''
select post_id,
       countIf(action = 'view') as views,
       countIf(action = 'like') as likes,
       uniq(user_id) as uniq_users
from {db}.feed_actions
where toDate(time) = yesterday()
group by post_id
order by views desc
limit 10
'''
```

```
df = ph.read_clickhouse(query, connection=connection)
```

Таким образом, мы можем читать данные из ClickHouse как из Jupyter, так и из Redash.

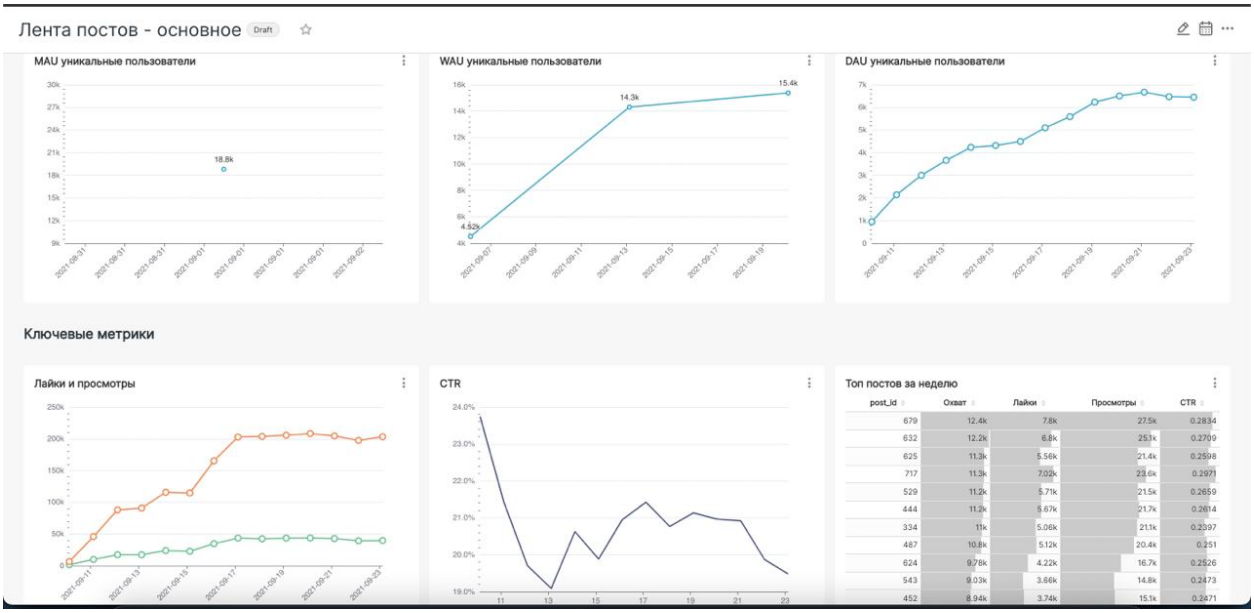
Redash больше подходит для быстрых несложных запросов. Если данные требуют сложной обработки, то бывает удобнее выгрузить данные в Pandas Dataframe и продолжить работать с ними на Python.

> Superset

В лекции также упоминался Superset.

Superset — это облачное приложение, с помощью которого можно обрабатывать, визуализировать и анализировать очень большие объёмы данных.

Мы вернёмся к нему позже в третьем уроке, но уже сейчас вы можете авторизоваться и познакомиться с его интерфейсом.



В дальнейшем мы будем использовать Superset для построения дашбордов и выполнения наших SQL-запросов.