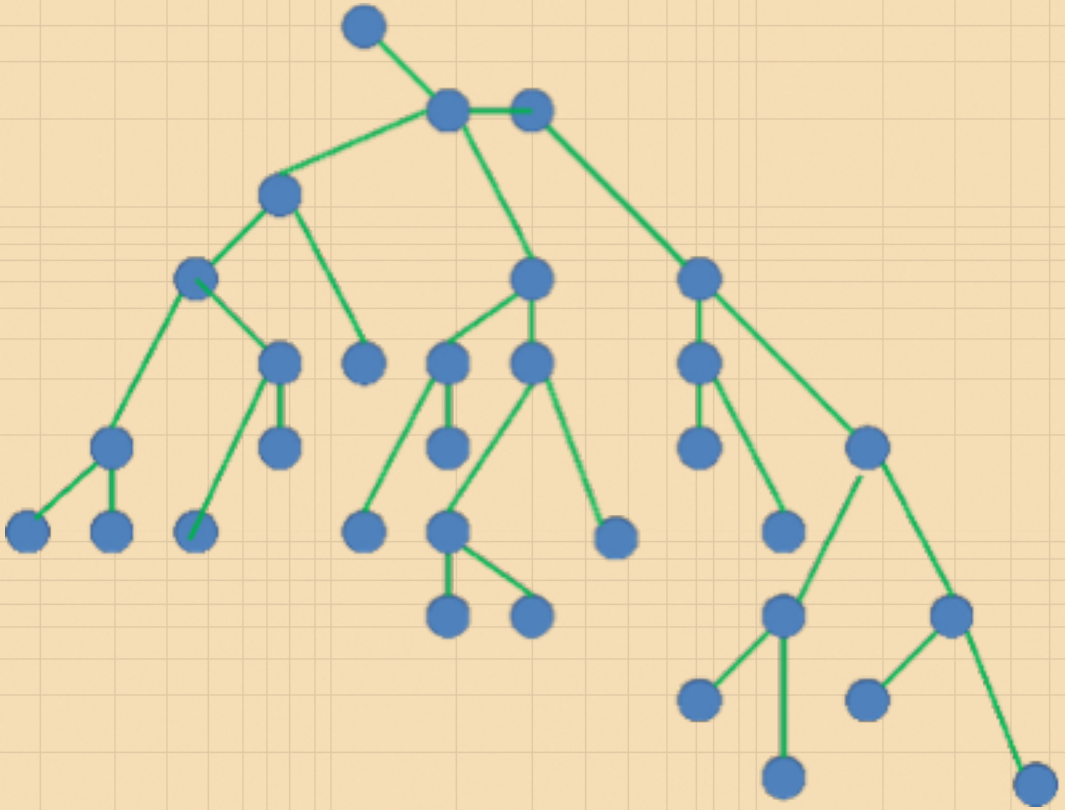
Анализ данных в командной строке

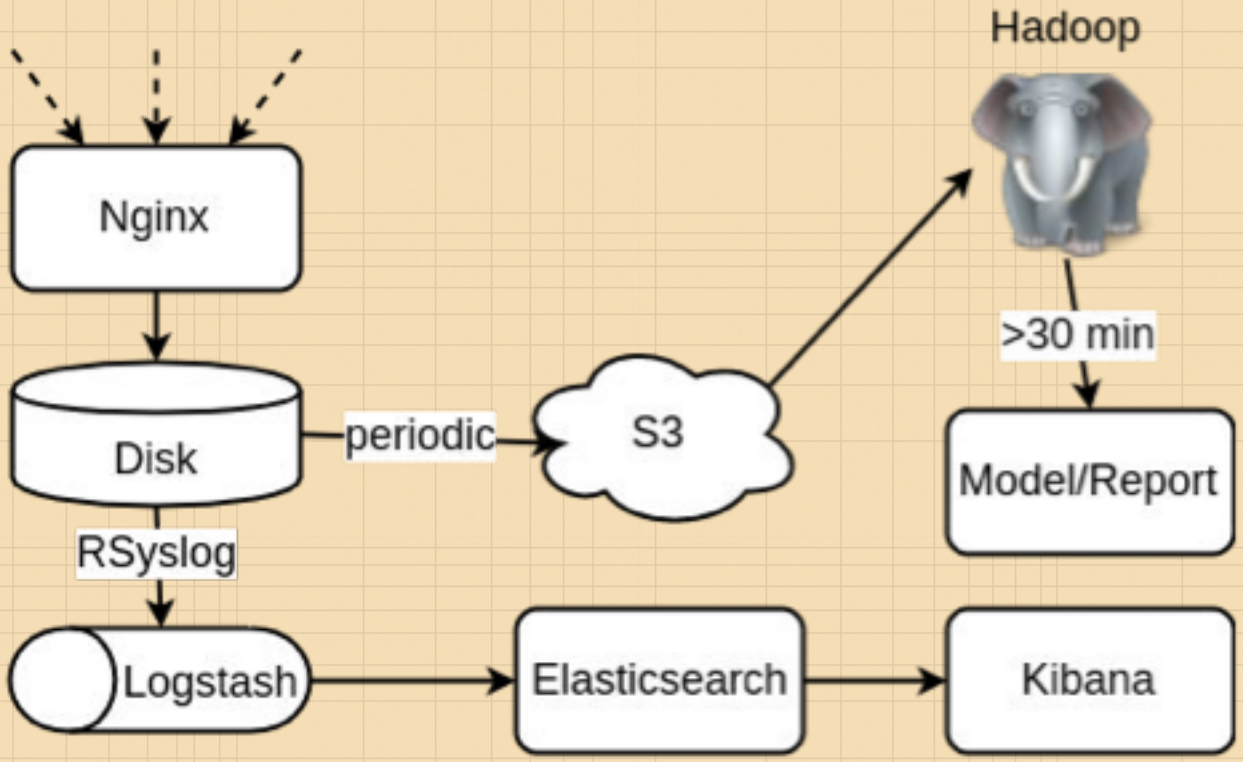
Николай Марков (@enchantner)

Пайплайн

- Airflow/Luigi/Jenkins
- Bash
- RabbitMQ/Apache Kafka
- SQL
- MongoDB/HBase
- ELK
- …
- PROFIT



Пайплайн


Hadoop
Nginx
Disk
periodic
S3
>30 min
Model/Report
RSyslog
Logstash
Elasticsearch
Kibana

Данные

- В базах данных наподобие PostgreSQL
- Чуть более специализированные форматы - HDF5
- Различные API
- Файлы на HDFS или в локальных файлах
- Запакованные каким-то архиватором (tar.gz, lzo, xz, zip)
- С датами или таймстемпами в названиях
- Иногда в сложной структуре каталогов
- В формате CSV/TSV (колонки данных) или JSON

Bash

- Есть практически на любой *nix/BSD системе
- Любые операции элементарно автоматизируются написанием скриптов
- Большинство кода УЖЕ написано за нас
- Огромное количество программ имеют CLI либо версию, работающую в окне терминала (rtorrent, midnight commander, мессенджеры, почтовые клиенты, архиваторы, браузеры)

Часто используемые команды

wc -l

cat

tar -xvf

sort

chmod u+x

echo

ps aux

ls -la

find

grep

less

Еще команды

~$ seq [first [incr]] last # последовательность чисел
~$ tr 'first' 'second' # замена символов во входных строках
~$ zcat / gunzip -c # распаковка файла с выводом на терминал
~$ head -n 10 # вывод первых нескольких строк
~$ tail -n 10 # вывод последних нескольких строк
~$ uniq -c # подсчет количества уникальных строк

~$ tail -n+100 # вывод последних строк, начиная со строки 100
~$ zgrep # grep по архивированным файлам
~$ seq -f "Line %g" 10 # последовательность с шаблоном

Кодировка

enca *и* **iconv**

Трубоукладка

~$ cat file.csv | grep -v Title | tr ‘n’ ‘0’ | awk ‘{ print $5 }’

Система реализует буферизацию, разделение данных на куски и управление памятью за нас

~$ set -o pipefail # перехват кодов выхода

~$ { echo "1"; echo "2"; } | *other_command* # объединение вывода

~$ sleep 3 | *echo 1* # немного магии
~$ cat file | *(grep "foo"* || *true)* | less # еще немного магии

https://habrahabr.ru/post/195152/

http://ryanstutorials.net/linuxtutorial/piping.php

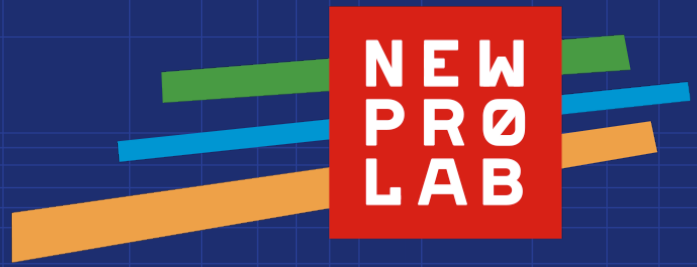
Web API

~$ curl -s https://www.gutenberg.org/files/76/76-0.txt

~$ curl [-XGET|-XPOST|-XPUT|-XHEAD]

~$ curl -s http://url/some_file | tr '[:upper:]' '[:lower:]' | grep -oE '\w+' | sort | uniq -c | sort -nr -k1,1 | head -n 100

Часто лучше просто сохранять файл



Sed и Awk

Потоковый редактор Sed

```
~$ cat file.txt | sed 's/First/Second/'  # потоковая замена
~$ cat file.txt | sed 's/First/Second/g'  # жадная потоковая замена
~$ cat file.txt | sed 's:/one/path:/another/path:g'  # другой разделитель
~$ cat file.txt | sed 's/[a-z*]/(&)/g'  # ссылка на ту же строку
~$ cat file.txt | sed 's/(Foo)[a-z]*/\1fel/g'  # ссылка на ту же строку
~$ sed -r/-E ... # расширенные регулярные выражения
```

http://www.grymoire.com/Unix/Sed.html

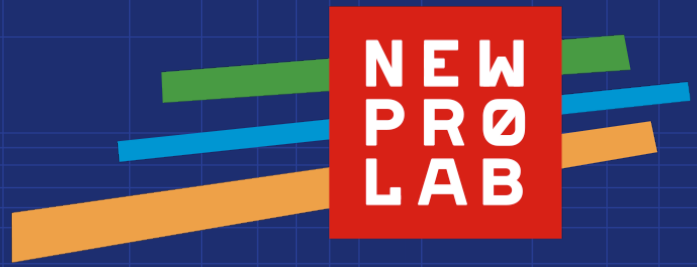
Язык обработки Awk

awk > cut

```
~$ cat file.txt | awk -F':' '{ print $2 }'  # вывод поля по номеру
~$ cat file.txt | awk '{ print $2 "," $1 }' # join двух полей запятой
~$ cat file.txt | awk 'BEGIN { OFS="\t" }{$1=$1; print $1,$2 }' # меняем разделитель
~$ cat file.txt | awk '{ x+=$2 } END { print x }'  # суммируем поле 2
~$ cat file.txt | awk -v home=$HOME/project '{ print home "/data.txt" }'
```

Переменные:
NF - число полей
NR - число строк

http://www.grymoire.com/Unix/Awk.html



Ближе к данным

Работа с CSV

csvkit - модуль Python для продвинутой работы с CSV
~$ pip install csvkit

in2csv, csvcut, csvlook, csvjson, csvsql, csvsort

~$ in2csv imdb-250-1996-2011-lists-only.xlsx 2>/dev/null | csvsql --query "select Title,Year from stdin where Year<2009" | csvsort -r -c Year | head -n 10 | csvlook

https://csvkit.readthedocs.org/

Работа с JSON

~$ sudo apt-get install jq

~$ cat file.txt | jq '.Title, .Year' # имеем дело с одним объектом

~$ cat file.txt | jq -c '.[].Title' # имеем дело со списком

~$ cat file.txt | jq --raw-input --slurp --arg home $HOME/project 'split("\n") | map(select(length > 0)) | . as $items | reduce range(0; $items|length) as $i ({}; . + { ($items[$i]): $home })'

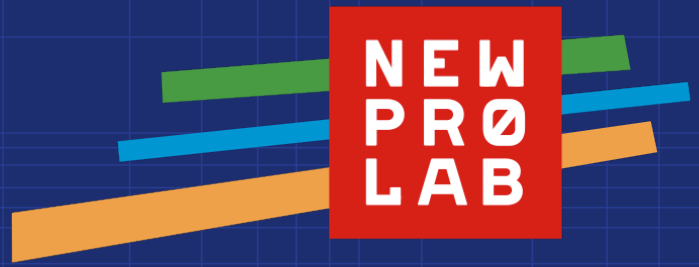
http://hyperpolyglot.org/json

https://stedolan.github.io/jq/manual/

Еще магия с jq

```
~$ cat file.txt | csvjson --stream | jq -c 'if .createdDate != "" then .createdDate =
(.standardRegCreatedDate | split(" ") | .[0:2] | join("T") + "Z" ) else .createdDate
= "9999-01-01T00:00:00Z" | to_entries | map(select(.key | contains("rawText") |
not ) ) | from_entries'
```

https://stedolan.github.io/jq/manual/



Параллелизация

xargs

Подстановка результата выполнения одной команды в качестве аргумента другой команде

```
~$ find . -name "*.mp3" -print0 | xargs -0 ls

~$ find . -name "*.sh" -print0 | xargs -0 -I {} mv {} ~/back.scripts

~$ cat file.csv | csvcut -c "Field" | xargs  # убираем переносы строк
```

http://bit.ly/1sS01aP

еще xargs

```
~$ cat file.txt | xargs -l bash -c 'echo hdfs dfs -get $0 $1' | xargs -I {} -d '\n' -n1 -P8 -t bash -c "eval {}"
```

http://bit.ly/1sS01aP

GNU Parallel

```
~$ ls *.wav | parallel lame {} -o {}.mp3
```

```
~$ python makelist.py | parallel -j+2 'wget "{}" -O - | python parse.py'
```

```
~$ cat file.txt | awk '{ print "{\"index\": {} }","\n" $0 }' | parallel --pipe -N500 curl -s
-XPOST localhost:9200/items/entry/_bulk --data-binary @- > /dev/null
```

man parallel_tutorial

http://bit.ly/22zcqhE

More tips and tricks

```
~$ cat file.csv | parallel --colsep "\t" echo {2} {1} {3}

~$ cat file.csv | awk '!a[$1]++'  # sort -u не нужно

~$ cat file.csv | pv --line-mode -b > /dev/null  # ОЧЕНЬ КРУТО

~$ cat file.csv | peco | some_other_process  # выбор строк вручную

~$cat 76-0.txt | tr '[:upper:]' '[:lower:]' | grep -oE '\w+' | sort | uniq -c | sort -nr -k1,1 |
head -n 50 | awk '{ print $2 "\t" $1 }' | gnuplot -p -e "set term png; set xtic rotate;
plot '-' using (column(0)):2:xtic(1) smooth freq with boxes" > test.png
```

Shameless links

https://xakep.ru/2016/05/17/console-magic/

https://xakep.ru/2016/06/07/console-magic-2/