



**Московский городской педагогический университет
Департамент информатики, управления и технологий**

Аналитика больших данных : подходы и инструменты

Босенко Т.М.



Введение в **Apache Spark**

Apache Spark - это быстрый механизм обработки данных **в памяти**, который позволяет работникам, работающим с данными, эффективно выполнять потоковую передачу, машинное обучение или рабочие нагрузки **SQL**, требующие быстрого итеративного доступа к наборам данных.

Выполняет вычисления в памяти!

- **Apache Spark** имеет усовершенствованный механизм выполнения **DAG**, который поддерживает ациклический поток данных и вычисления **в памяти**.
- **В 100 раз** быстрее в памяти и в 10 раз быстрее, даже когда работает на диске, чем **MapReduce**.

Общая модель программирования, которая позволяет разработчикам написать приложение, **составив произвольные операторы**.

Spark позволяет легко комбинировать различные обработки моделирование без проблем в одном приложении.

Пример:

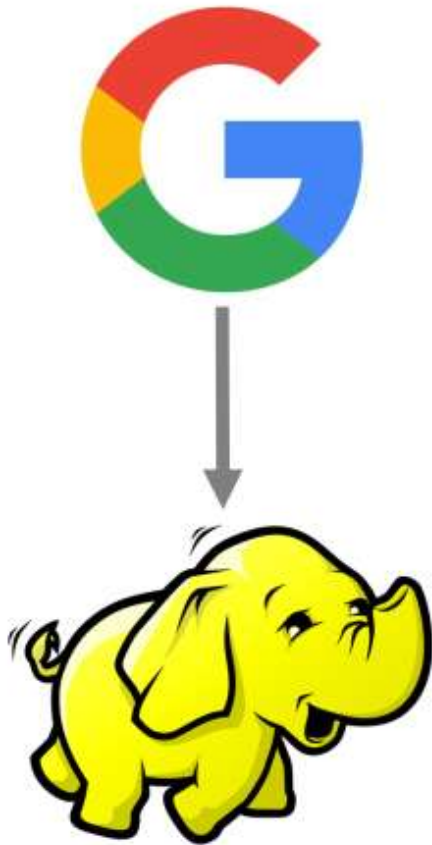
- Классификация данных с помощью библиотеки машинного обучения **Spark**.
- Поточковая передача данных через источник через **Spark Streaming**.
- Запрос полученных данных в реальном времени с помощью **Spark SQL**.



Apache Spark

Формирование озера данных Data Lake

Концепция озера данных

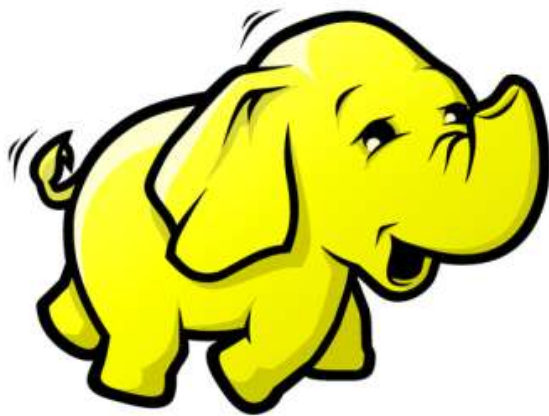


Google File System

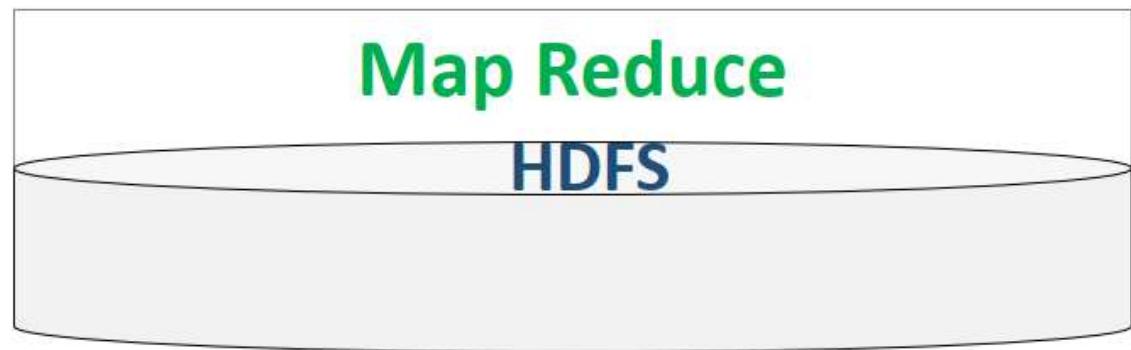


Hadoop Distributed File System

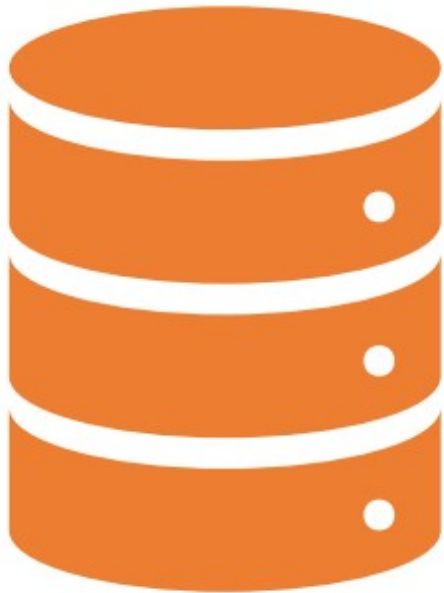
Концепция озера данных



HADOOP



Концепция озера данных

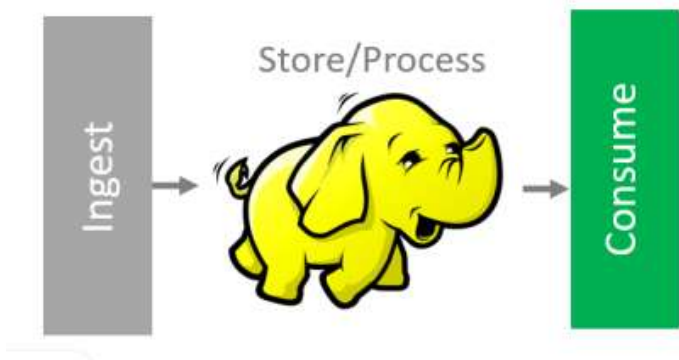


1. Teradata
2. Exadata

Концепция озера данных

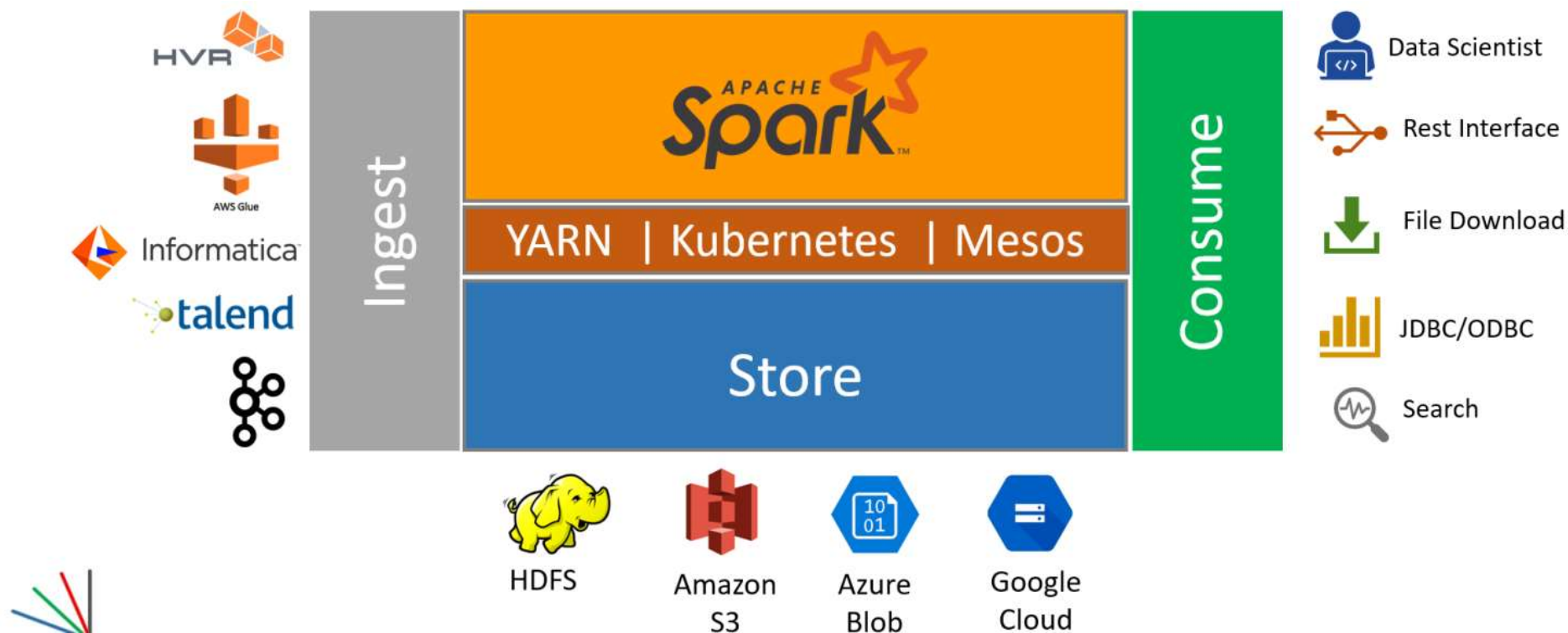


1. Вертикальная масштабируемость
2. Высокая себестоимость



1. Горизонтальная масштабируемость
2. Низкие капитальные затраты.

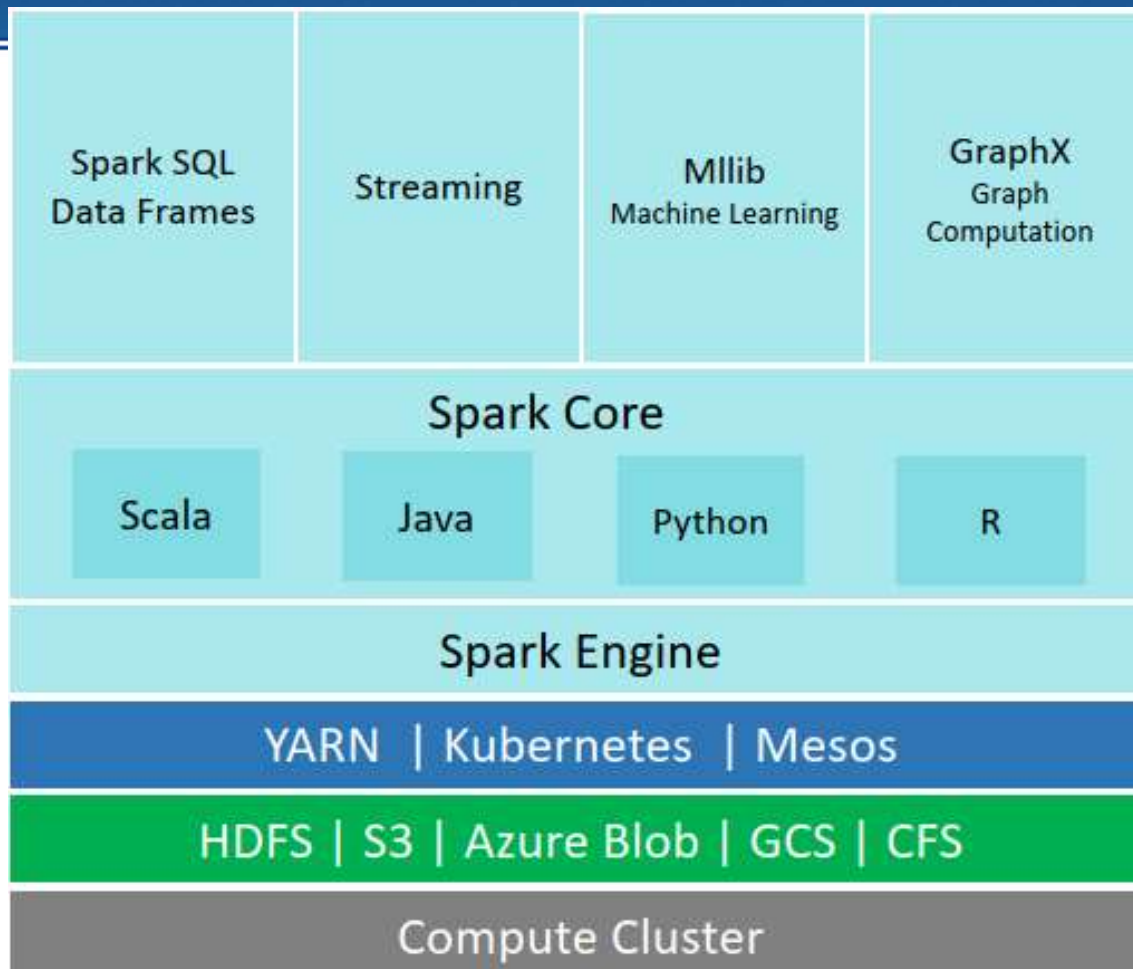
Концепция озера данных



Apache Spark

Введение в Spark

Spark Ecosystem



Apache Spark

Среды Spark



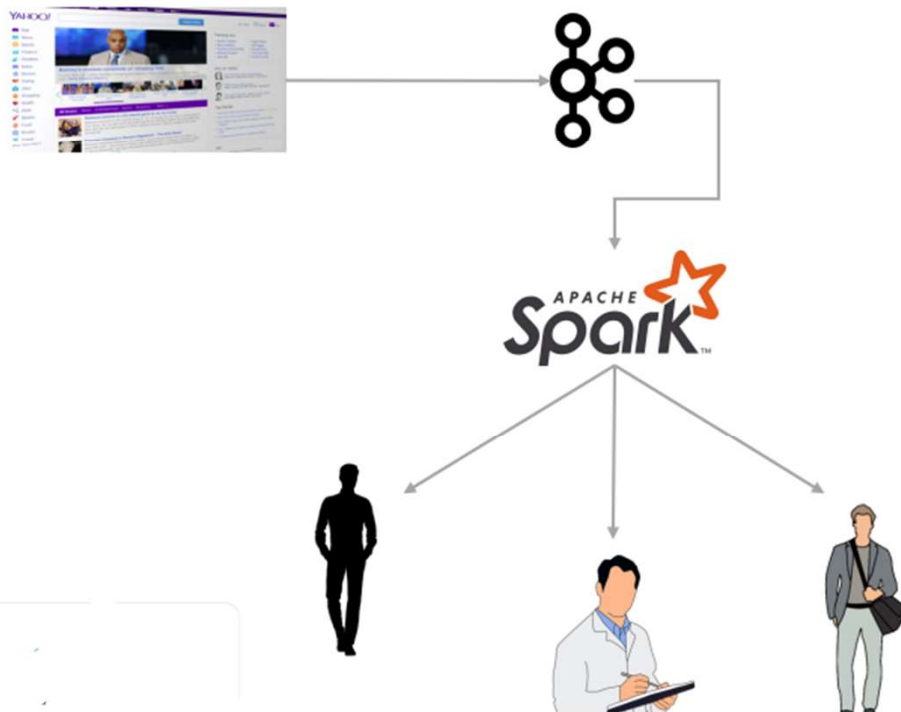
1. Local Mode – Command line REPL
2. Development Scala IDE – IntelliJ IDEA
3. Databricks Cloud – Notebooks
4. Cloudera Cluster – Zeppelin Notebooks
5. Other Options – Cloud offerings

Apache Spark

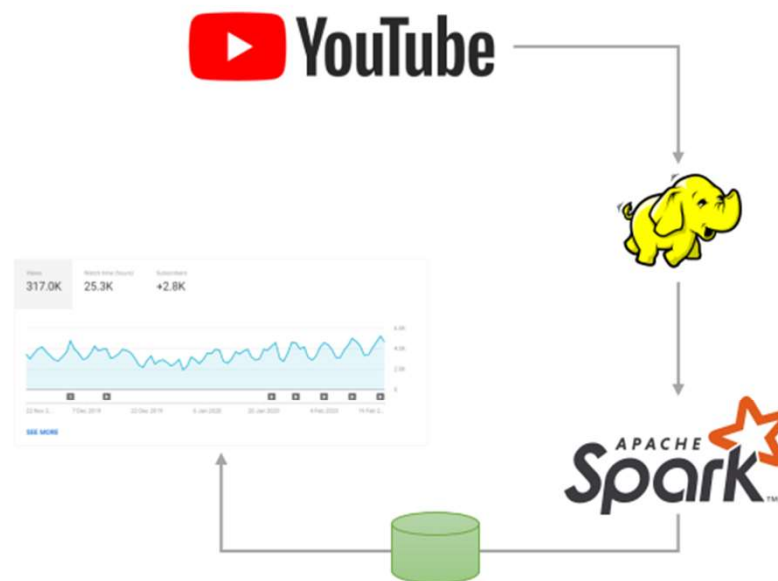
Методы исполнения

Методы исполнения

Stream Processing



Batch Processing



Как выполнять программы Spark?



1. **Интерактивные клиенты**

Spark-Shell, Ноутбук

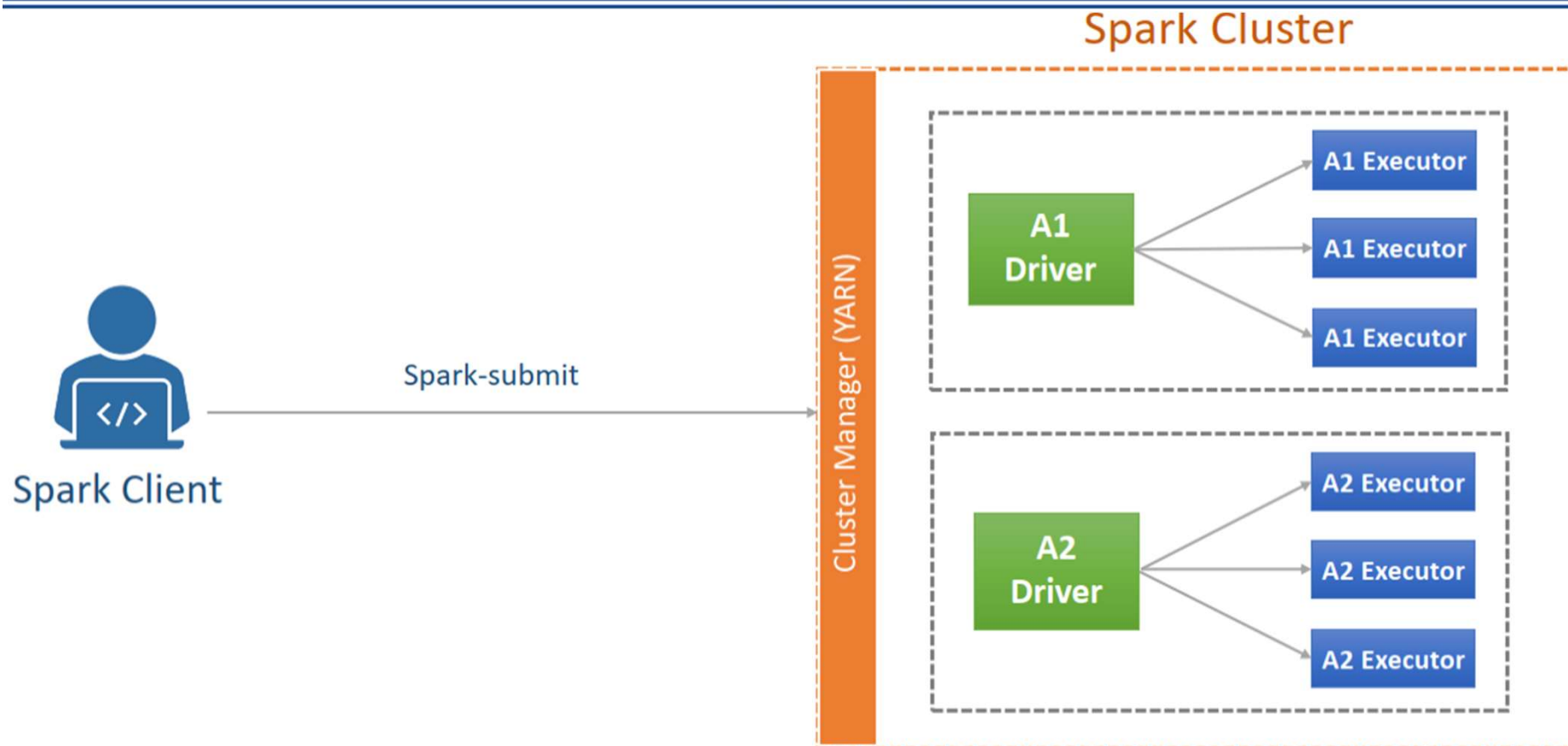
2. **Выполнение заданий**

Spark-submit, Блокнот Databricks, Rest API

Apache Spark

Модель обработки

Модель обработки

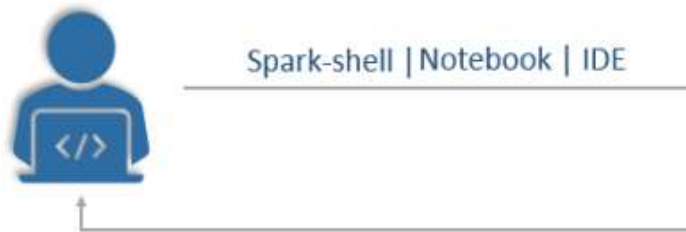


Apache Spark

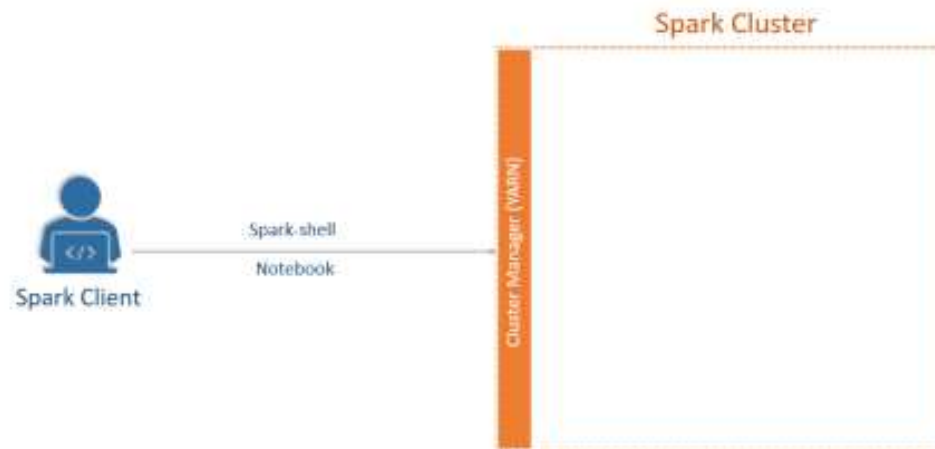
Spark Local и режимы выполнения

Spark Local и режимы выполнения

Как **Spark** работает на локальном компьютере?



Как **Spark** работает с интерактивными клиентами?



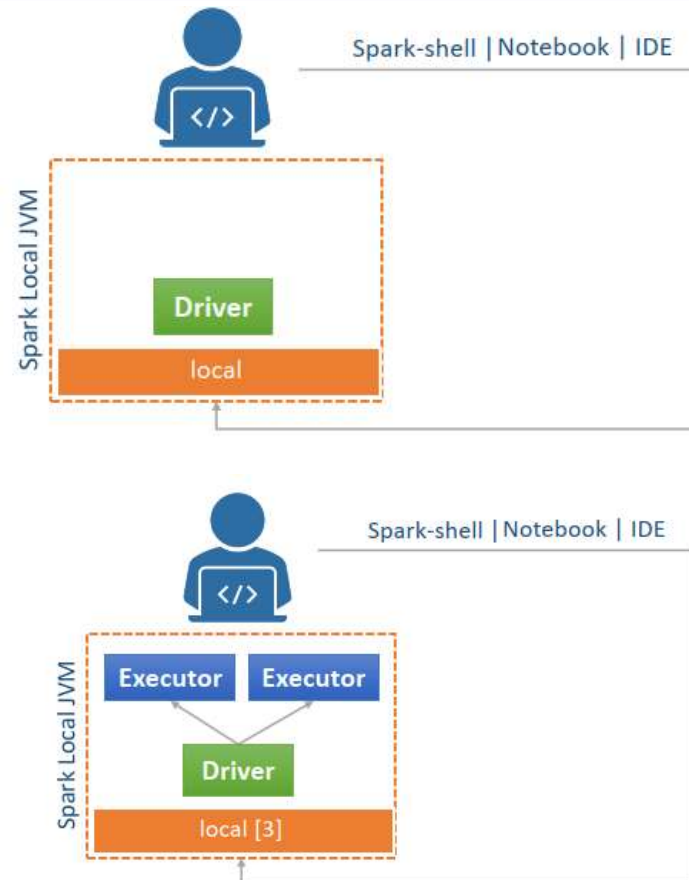
Менеджеры кластеров Spark



1. local[n]
2. YARN
3. Kubernetes
4. Mesos
5. Standalone

Spark Local

1. local[n]
2. YARN
3. Kubernetes
4. Mesos
5. Standalone

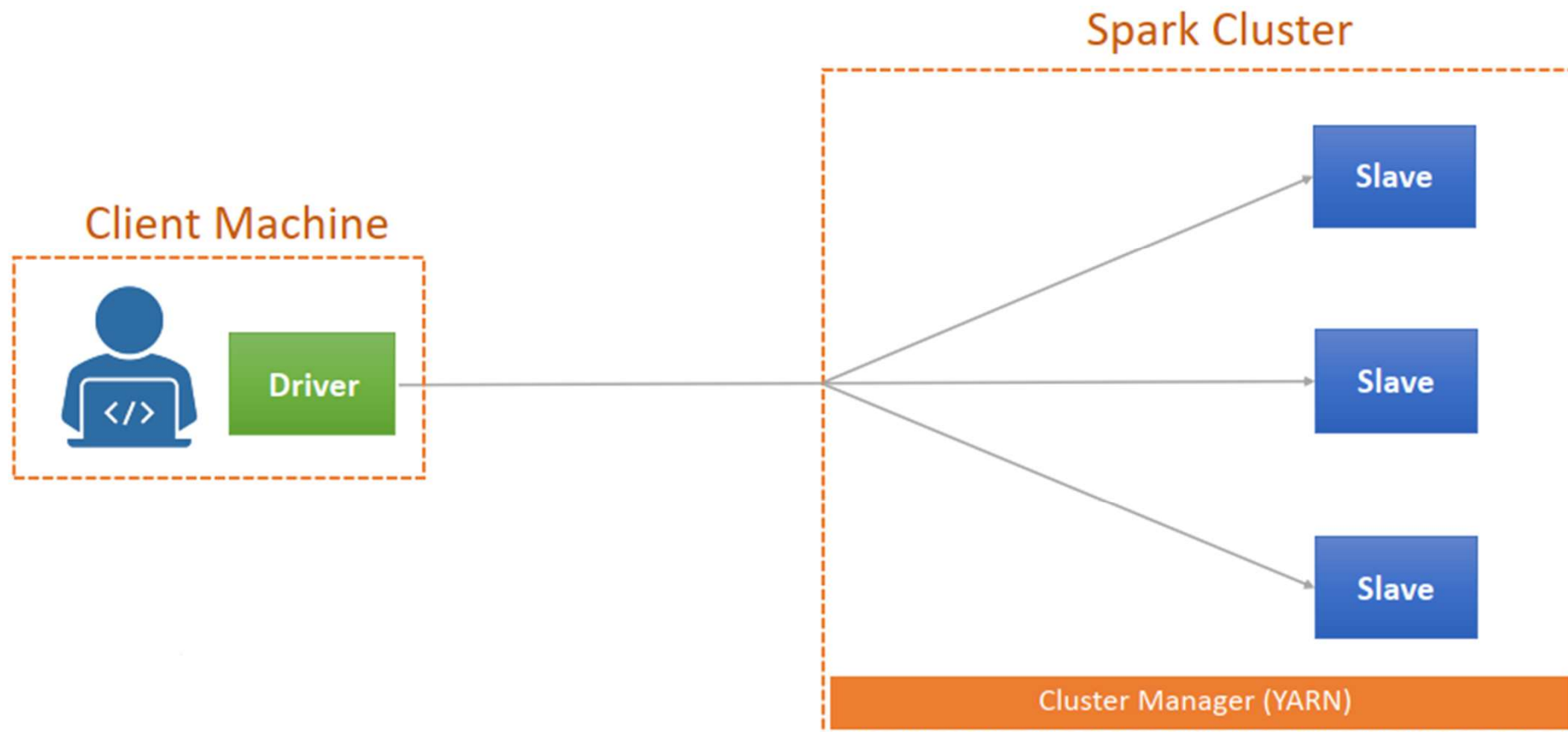


Режимы выполнения Spark



1. Client
2. Cluster

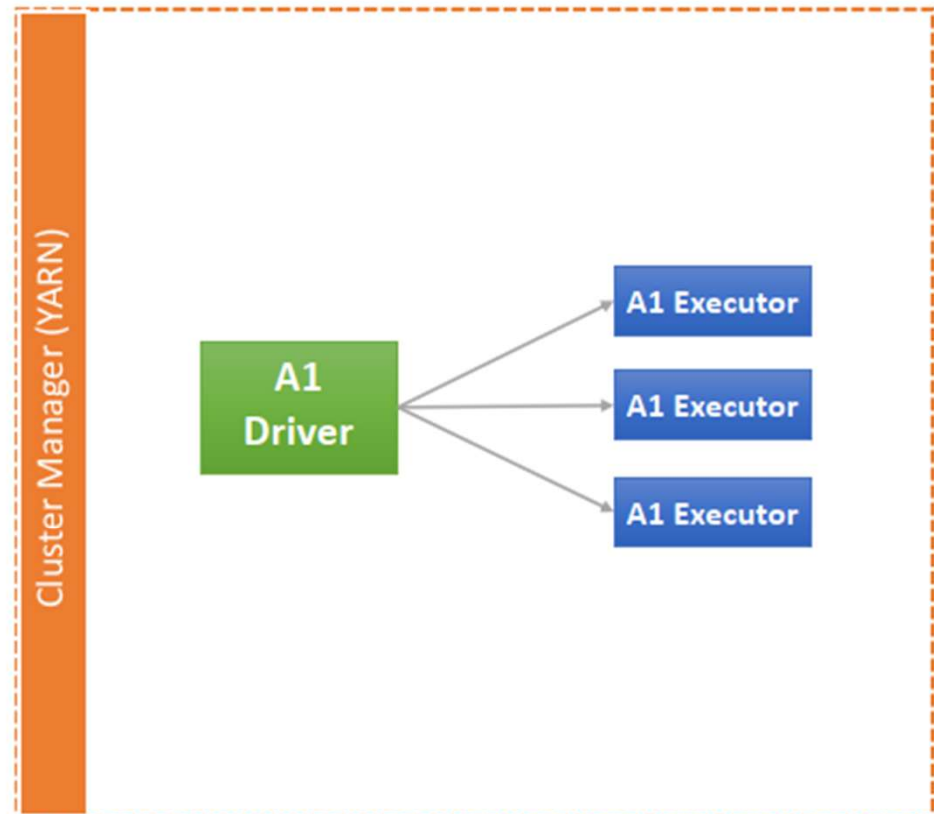
Модель обработки – Client Mode



Модель обработки – Cluster mode



Spark Cluster



Модели обработки Spark

Cluster Managers

1. local[n]
2. YARN
3. Kubernetes
4. Mesos
5. Standalone

Execution Modes

- 1. Client**
- 2. Cluster**

Execution Tools

1. IDE, Notebook
2. Submit

Apache Spark

Spark Local и режимы выполнения

run spark-shell in local-client-mode.

Cluster	Mode	Tool
Local	Client Mode	spark-shell, Notebook, IDE

Cluster	Mode	Tool
YARN	Client Mode	spark-shell, Notebook
YARN	Cluster Mode	spark-submit



Apache Spark

Install

<https://www.youtube.com/watch?v=YanzUI-30pI&list=PL589M8KPPT1YP2lN8nXbqfheRwAnzdDLx&index=9>
<https://neoserver.ru/help/osnovnie-komandi-redaktora-vi-vim>

Install Apache Spark on Ubuntu

Step 1: Install Java on Ubuntu

Step 2: Download the spark package

Step 3: Setting environment variable

Step 4: Verification of Spark Installation

1 step Install Java



```
hdmaster@linux: ~  
hdmaster@linux:~$ java -version  
openjdk version "1.8.0_282"  
OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~20.04-b08)  
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)  
hdmaster@linux:~$
```

ЕСЛИ НЕ УСТАНОВЛЕНА JAVA

`sudo apt-get install default-jdk`

2. Download the Apache Spark package

<https://spark.apache.org/downloads.html>



Download

Libraries ▾

Documentation ▾

Examples

Community ▾

Developers ▾

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: **spark-3.1.1-bin-hadoop2.7.tgz**
4. Verify this release using the 3.1.1 [signatures](#), [checksums](#) and [project release KEYS](#).

Note that, Spark 2.x is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12. Spark 3.0+ is pre-built with Scala 2.12.

2. Download the Apache Spark package



COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"

Projects ▾

People ▾

Community ▾

License ▾

Sponsors ▾



We suggest the following mirror site for your download:

<https://apache-mirror.rbc.ru/pub/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz>

Other mirror sites are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

HTTP

<https://apache-mirror.rbc.ru/pub/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz>

Сохраните его в каталоге

/opt/spark

(`mkdir -p /opt/spark`)

Поскольку это файл **tar**. Поэтому нам нужно распаковать его, используя команду ниже

2. Download the Apache Spark package

```
sudo su –  
mkdir –p /opt/spark  
cd /opt/spark  
root@linux:/opt/spark# wget https://apache-mirror.rbc.ru/pub/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz
```



COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"



Projects ▾

People ▾

Community ▾

License ▾

Sponsors ▾

We suggest the following mirror site for your download:

<https://apache-mirror.rbc.ru/pub/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz>

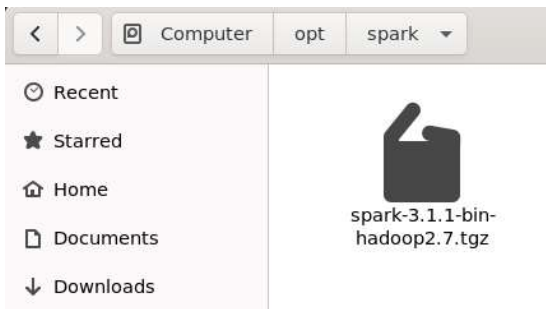
Other mirror sites are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

Please only use the backup mirrors to download KEYS, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.

HTTP

<https://apache-mirror.rbc.ru/pub/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz>



2. Download the Apache Spark package

```
root@linux:/opt/spark# ll
total 219128
drwxr-xr-x 2 root root      4096 мар 12 00:12 ./
drwxr-xr-x 4 root root      4096 мар 12 00:10 ../
-rw-r--r-- 1 root root 224374704 фев 22 05:45 spark-3.1.1-bin-hadoop2.7.tgz
root@linux:/opt/spark#
```

tar -xvf spark-3.1.1-bin-hadoop2.7.tgz

```
root@linux:/opt/spark# ls
spark-3.1.1-bin-hadoop2.7  spark-3.1.1-bin-hadoop2.7.tgz
```

3 step Setting up the environment variable

- Переходим **/opt/spark/ spark-3.1.1-bin-hadoop2.7/bin** и устанавливаем переменные окружения.
- Устанавливаем **SPARK_HOME** в **.bashrc** файл.
- **vi ~/.bashrc**
- Переходим в конец файла и в режиме **i** вводим текст

```
SPARK_HOME=/opt/spark/spark-3.1.1-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```
- **:wq** - записать файл и выйти;

3 step Setting up the environment variable

- сохраните его(:wq) и обновите файл **Source**, чтобы отразить изменения, внесенные в файл **bashrc**.

```
root@linux:/opt/spark# vi ~/.bashrc
```

```
root@linux:/opt/spark# source ~/.bashrc
```

Запускаем scala

```
root@linux:/opt/spark# spark-shell
```

```
root@linux:/opt/spark# spark-shell
21/03/12 00:57:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform...
s where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://linux:4040
Spark context available as 'sc' (master = local[*], app id = local-1615499828533).
Spark session available as 'spark'.
Welcome to

  ____ _
 / ___ \| | | |
/ /   \| |_| |
\ \   /| | | |
 \___/\_| |_| |
      |_|_|_|

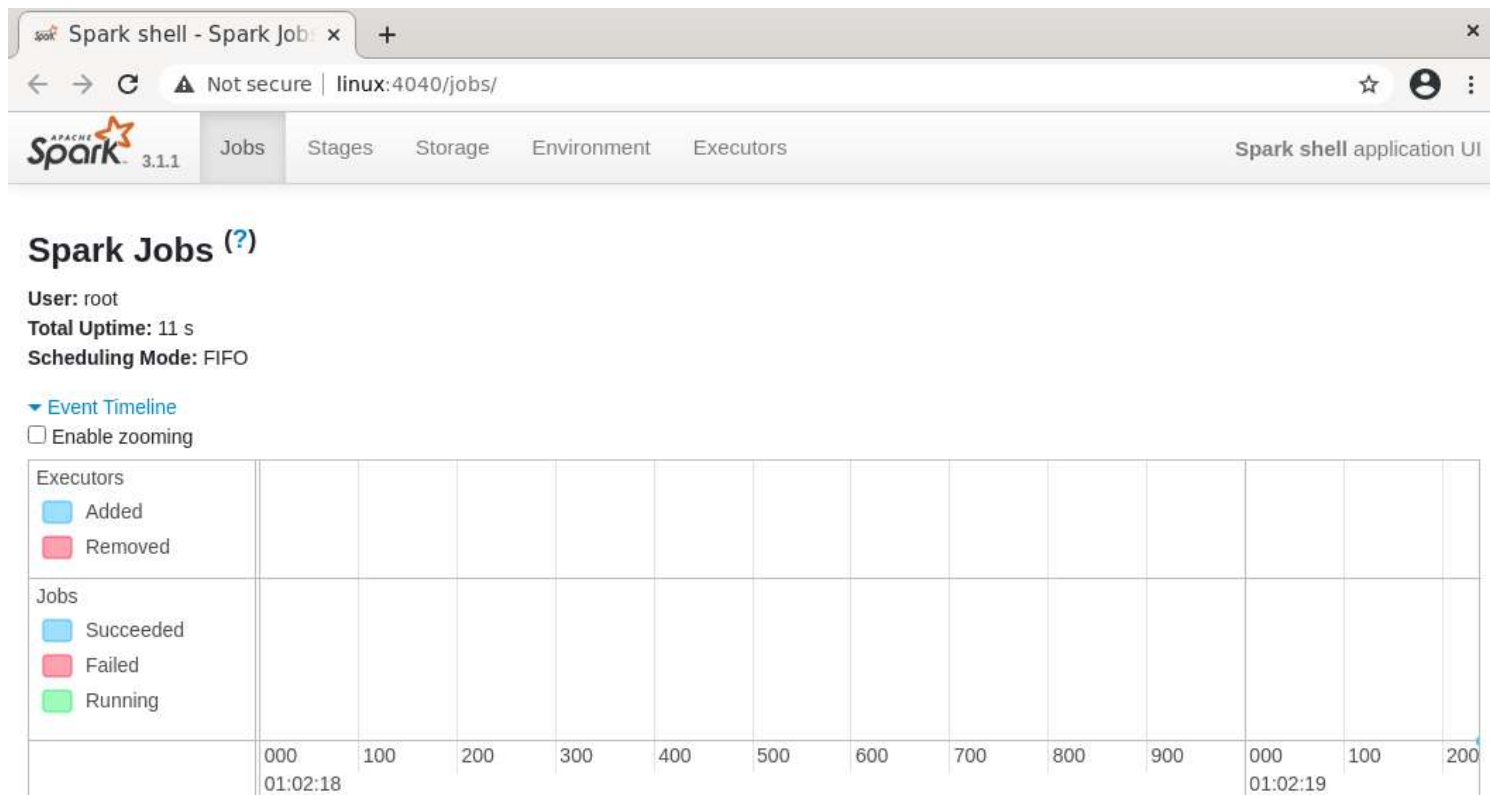
version 3.1.1

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_282)
Type in expressions to have them evaluated.
Type :help for more information.

scala> root@linux:/opt/spark#
```

Проверка веб-интерфейса SCALA

http://linux:4040





RDD

(Устойчивые распределенные наборы данных)

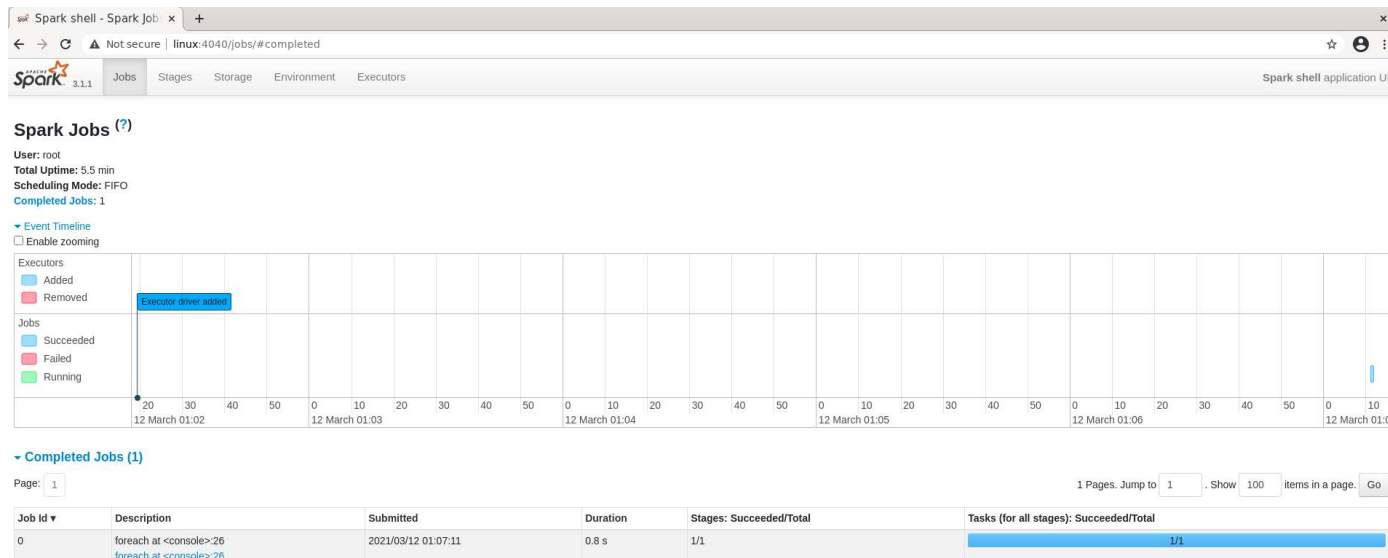
- **RDD** могут содержать любые типы объектов, включая определяемые пользователем классы.
- **RDD** - это просто объединение очень большого набора данных. В **Spark** вся работа выражается либо в создании новых RDD, либо в преобразовании существующих **RDD**, либо в вызове операций над RDD для вычисления результата.
- Под капотом **Spark** автоматически распределяет данные, содержащиеся в RDD, по вашему кластеру и распараллеливает операции, которые вы выполняете над ними.

Проверка веб-интерфейса SCALA

```
scala> val num=Array(1,2,3,454,32)
num: Array[Int] = Array(1, 2, 3, 454, 32)

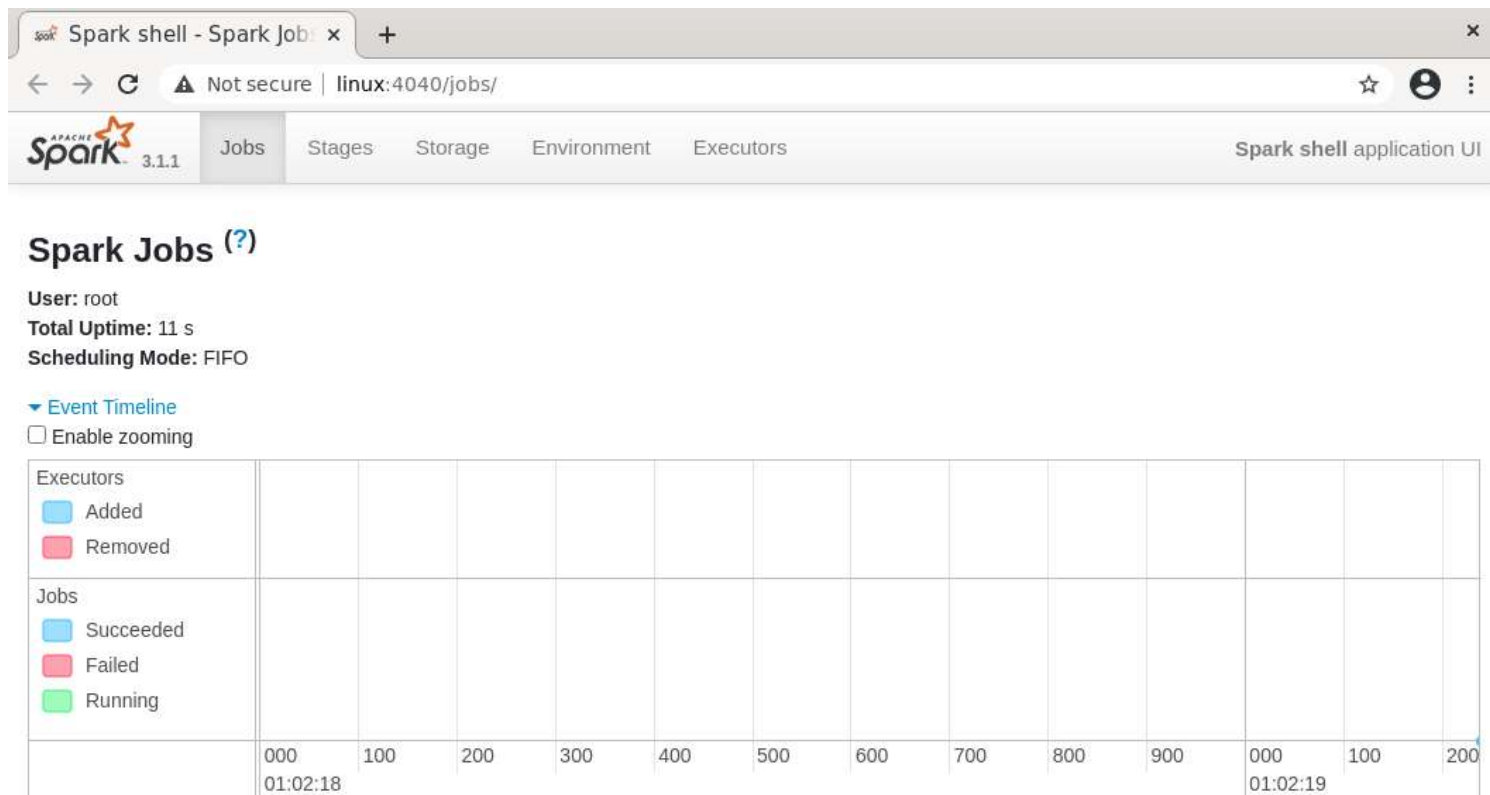
scala> val data=sc.parallelize(num)
data: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:26

scala> data.foreach(println)
1Stage 0:> (0 + 1) / 1]
2
3
454
32
```



Проверка веб-интерфейса SCALA

http://linux:4040



Apache Spark

Форматы файлов

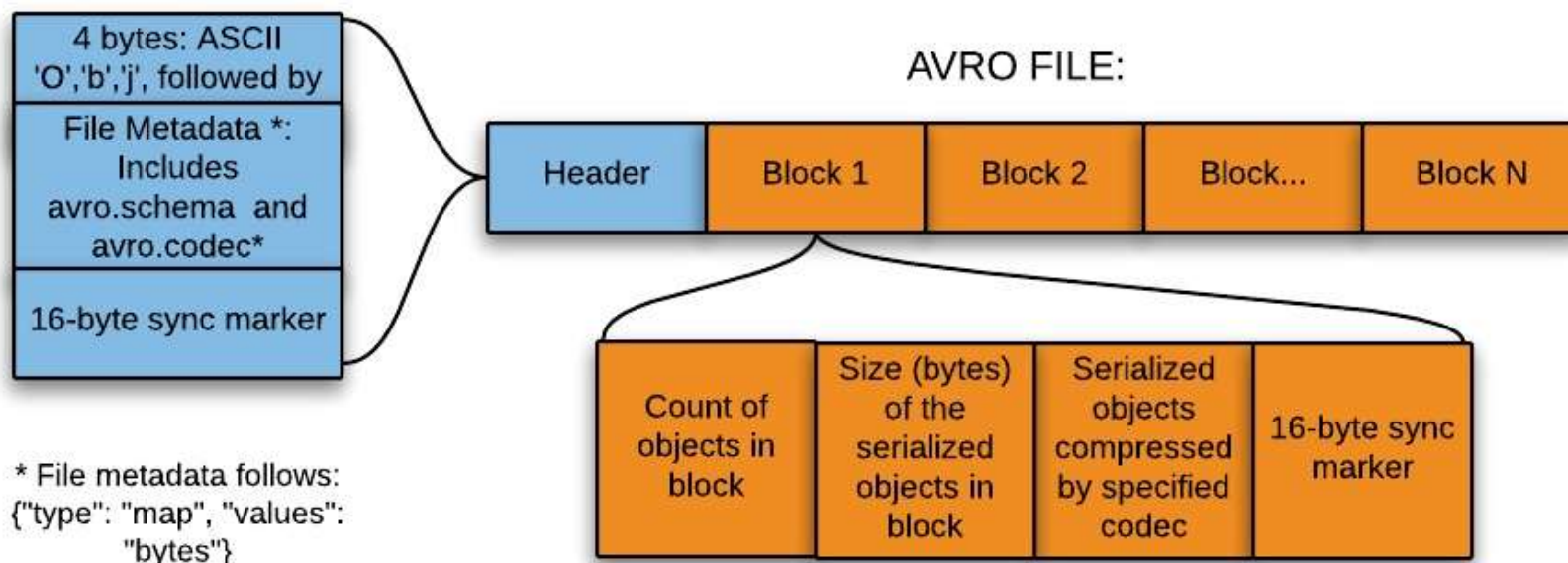
Форматы файлов

- Серьезное узкое место в производительности приложений с поддержкой **HDFS**, таких как **MapReduce** и **Spark** — время поиска, чтения, а также записи данных.
- Эти проблемы усугубляются трудностями в управлении большими наборами данных, если у нас не фиксированная, а эволюционирующая схема, или присутствуют некие ограничения на хранение.
- Обработка больших данных увеличивает нагрузку на подсистему хранения — **Hadoop** хранит данные избыточно для достижения отказоустойчивости. Кроме дисков, нагружаются процессор, сеть, система ввода-вывода и так далее. По мере роста объема данных увеличивается и стоимость их обработки и хранения.

- Различные форматы файлов в **Hadoop** придуманы для решения именно этих проблем. Выбор подходящего формата файла может дать некоторые существенные преимущества:
1. **Более быстрое время чтения.**
 2. **Более быстрое время записи.**
 3. **Разделяемые файлы.**
 4. **Поддержка эволюции схем.**
 5. **Расширенная поддержка сжатия.**

Формат файлов Avro

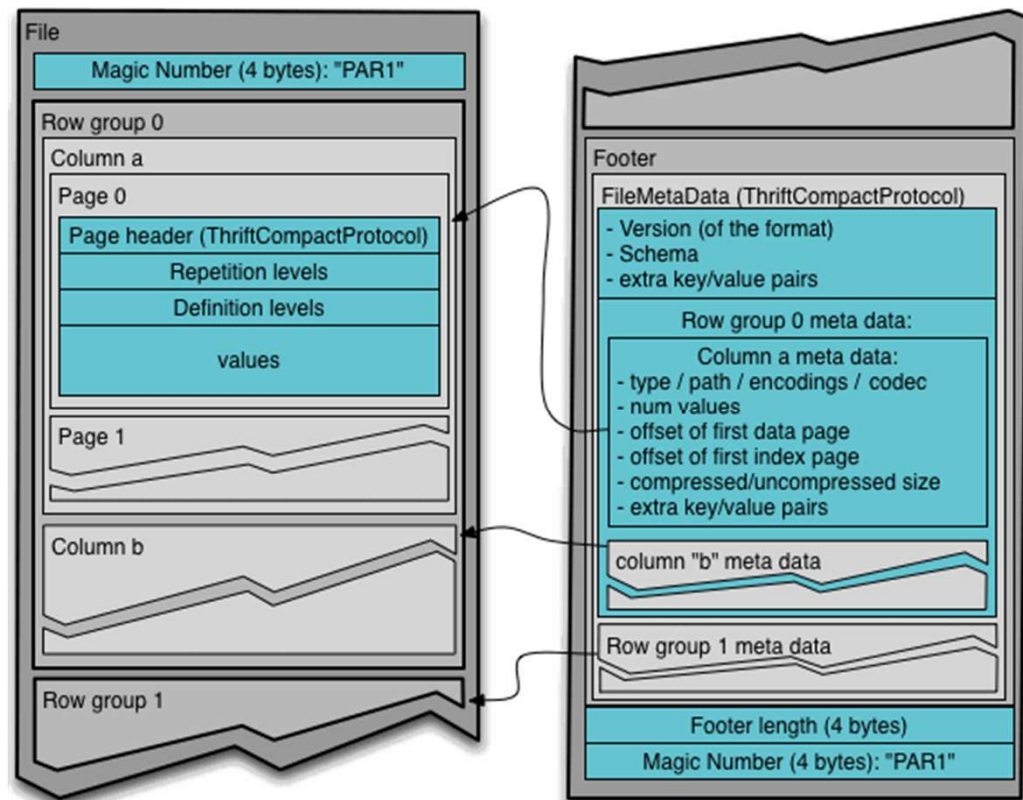
- Для **сериализации данных** широко используют **Avro** — это основанный на строках, то есть **строковый, формат хранения данных в Hadoop**.
- Он хранит схему в формате JSON, облегчая ее чтение и интерпретацию любой программой. Сами данные лежат в двоичном формате, компактно и эффективно.



Формат файлов Parquet

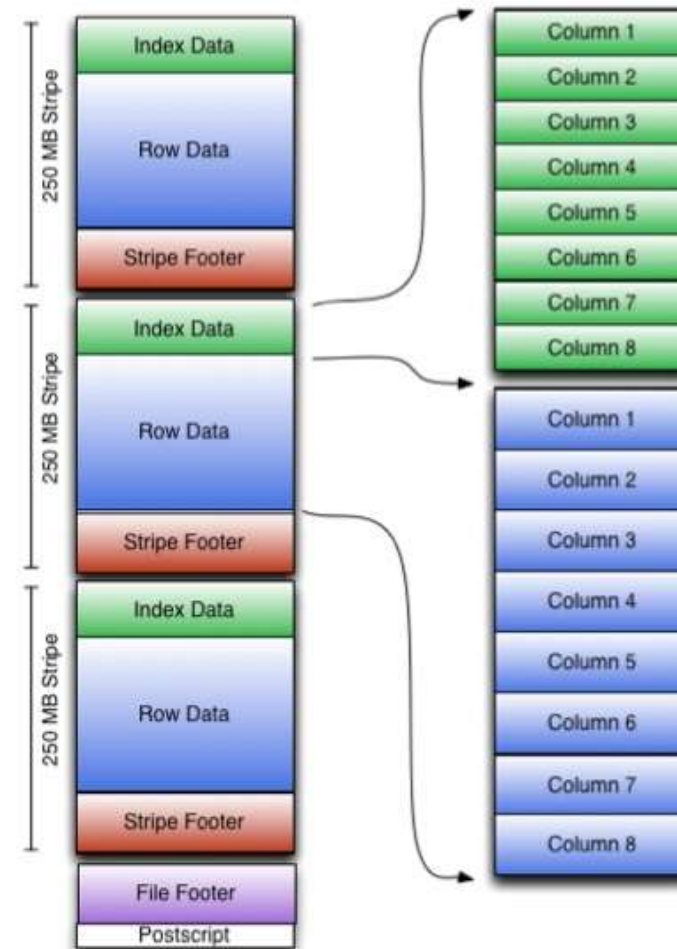
- **Parquet** — **открытый** формат файлов для Hadoop, который хранит **вложенные структуры данных в плоском столбчатом формате**.

По сравнению с традиционным строчным подходом, Parquet более эффективен с точки зрения хранения и производительности.



Формат файлов ORC

- **Оптимизированный строково-столбчатый формат файлов (Optimized Row Columnar, ORC)**
- предлагает очень эффективный способ хранения данных и был разработан, чтобы преодолеть ограничения других форматов. Хранит данные в идеально компактном виде, позволяя пропускать ненужные детали — при этом не требует построения больших, сложных или обслуживаемых вручную индексов.



- **XML - расширяемый язык разметки**
- **XML** является выбором по умолчанию для обмена данными, так-как практически у каждого языка есть синтаксический анализатор, будь то Java, .NET или любой другой язык, поэтому легко извлечь определенные данные из XML, и есть схемы, которые могут проверять XML.
- XML также широко используется в сфере услуг: банковские услуги, онлайн-магазины розничной торговли, интеграция промышленных систем и т.д.

Структура XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <сотрудники>
    <id>01</id>
    <имя>Влад</имя>
    <команда>Разработчик</команда>
    <технология>Веб</технология>
    <должность>Инженер</должность>
  </сотрудники>
</root>
```

Формат файлов JSON



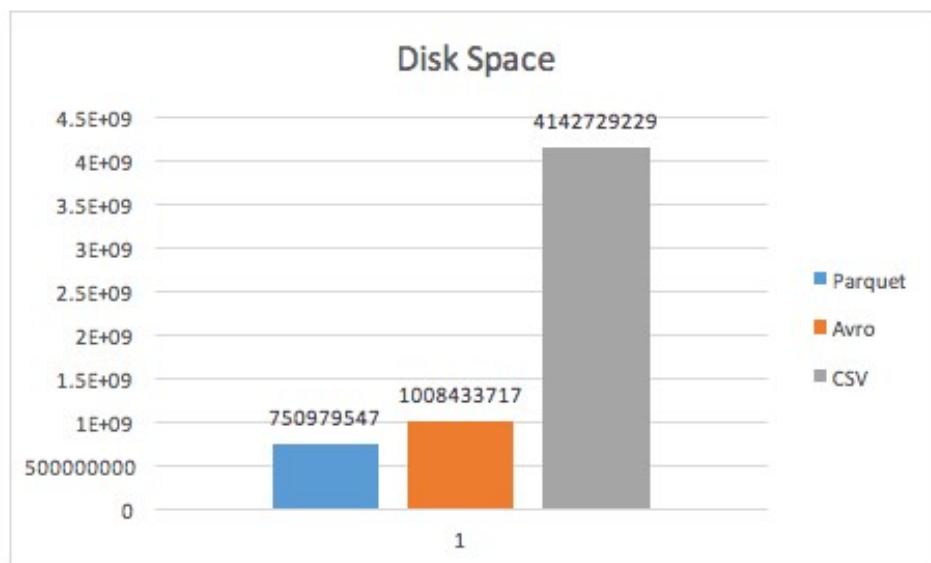
- **JSON** - текстовый формат обмена данными, основанный на JavaScript.
- **JSON** требует меньше кода и имеет меньший размер, что ускоряет обработку и передачу данных. Не смотря на то, что **JSON** написан на JavaScript, он не зависит от языка.
- Он не имеет каких-либо мощных функций, связанных с проверкой и схемой, которые есть у **XML**.

Структура JSON

```
{  
  "сотрудники": [{  
    "id": "01",  
    "имя": "Влад",  
    "команда": "Разработчик",  
    "технология": "Веб",  
    "должность": "Инженер",  
  }]  
}
```

Data Source

Сжатие файлов Big Data, Apache Parquet, Avro, CSV



Сравнение форматов данных Big Data, Apache Parquet, JSON, CSV

Spark Format Showdown		File Format		
		CSV	JSON	Parquet
A t t r i b u t e	Columnar	No	No	Yes
	Compressable	Yes	Yes	Yes
	Splittable	Yes*	Yes**	Yes
	Human Readable	Yes	Yes	No
	Nestable	No	Yes	Yes
	Complex Data Structures	No	Yes	Yes
	Default Schema: Named columns	Manual	Automatic (full read)	Automatic (instant)
	Default Schema: Data Types	Manual (full read)	Automatic (full read)	Automatic (instant)

- CSV Read the CSV and Write it to Parquet
- Parquet Read from Parquet and Write it to JSON
- JSON Read from JSON and Write it to ORC
- ORC Read from ORC and Write it to Avro
- AVRO Read from Avro and Write it to XML
- XML Read from XML and Write it to CSV

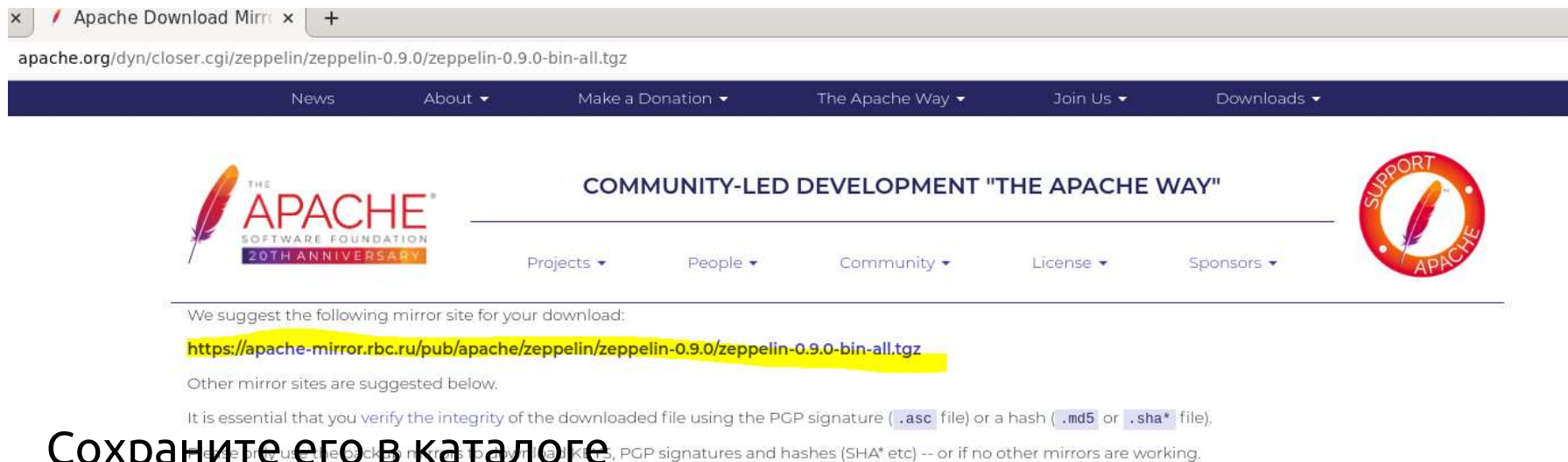
Apache zeppelin

Install

<http://zeppelin.apache.org/download.html>

1. Download the Zeppelin

<http://zeppelin.apache.org/download.html>



The screenshot shows the Apache Zeppelin download page. At the top, there's a browser tab titled 'Apache Download Mirror' and the URL 'apache.org/dyn/closer.cgi/zeppelin/zeppelin-0.9.0/zeppelin-0.9.0-bin-all.tgz'. Below the browser window, the Apache Software Foundation logo is on the left, and the text 'COMMUNITY-LED DEVELOPMENT "THE APACHE WAY"' is in the center. To the right is a circular 'SUPPORT APACHE' logo. Below the logo, there are navigation links: 'News', 'About', 'Make a Donation', 'The Apache Way', 'Join Us', and 'Downloads'. Further down, there are links for 'Projects', 'People', 'Community', 'License', and 'Sponsors'. The main content area says 'We suggest the following mirror site for your download:' followed by a highlighted URL: 'https://apache-mirror.rbc.ru/pub/apache/zeppelin/zeppelin-0.9.0/zeppelin-0.9.0-bin-all.tgz'. Below this, it says 'Other mirror sites are suggested below.' and 'It is essential that you verify the integrity of the downloaded file using the PGP signature (.asc file) or a hash (.md5 or .sha* file).' At the bottom, there's a note: 'Please print out the quickstart instructions, PGP signatures and hashes (SHA* etc) -- or if no other mirrors are working.'

Сохраните его в каталоге

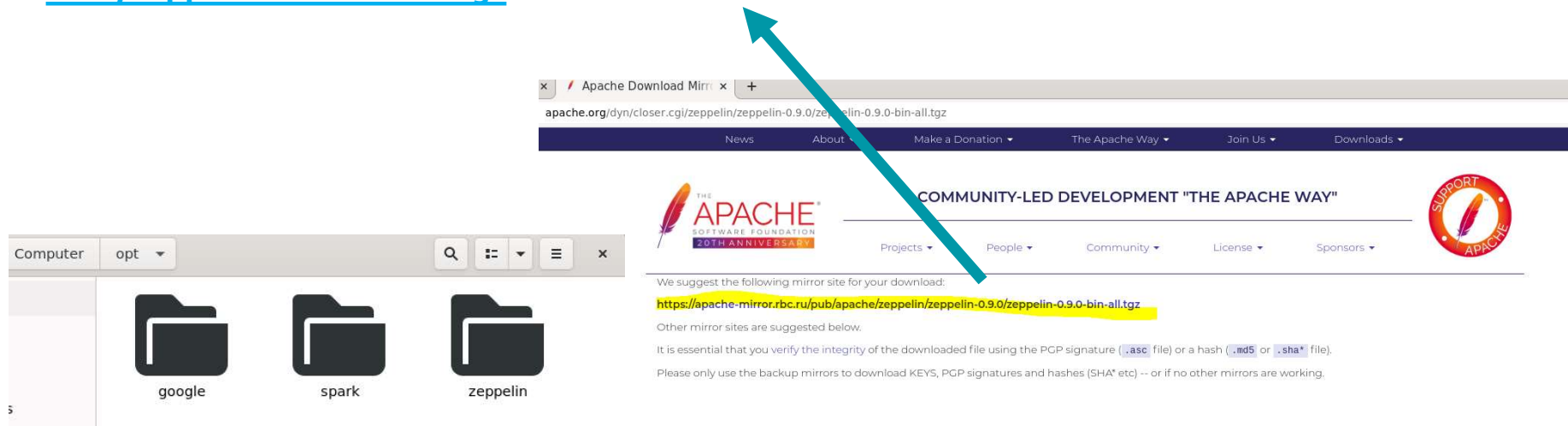
/opt/zeppelin

(**mkdir -p /opt/zeppelin**)

Поскольку это файл **tar**. Поэтому нам нужно распаковать его, используя команду ниже

2. Download the Apache Spark package

```
hdmaster@linux:~$ sudo su -  
[sudo] password for hdmaster:  
root@linux:~# mkdir -p /opt/zeppelin  
root@linux:~# cd /opt/zeppelin  
root@linux:/opt/zeppelin# wget https://apache-mirror.rbc.ru/pub/apache/zeppelin/zeppelin-0.9.0/zeppelin-0.9.0-bin-all.tgz
```



2. Download ZEPPELIN

```
root@linux:/opt/zeppelin# ll
total 1503972
drwxr-xr-x 2 root root      4096 мар 12 03:37 ./
drwxr-xr-x 5 root root      4096 мар 12 03:31 ../
-rw-r--r-- 1 root root 1540051033 дек 20 09:37 zeppelin-0.9.0-bin-all.tgz
root@linux:/opt/zeppelin#
```

root@linux:/opt/zeppelin# tar -xvf zeppelin-0.9.0-bin-all.tgz

```
root@linux:/opt/zeppelin# ls
zeppelin-0.9.0-bin-all zeppelin-0.9.0-bin-all.tgz
```

3 step Setting up the environment variable

- Переходим `/opt/zeppelin/zeppelin-0.9.0-bin-all/bin` и устанавливаем переменные окружения.

root@linux:/opt/zeppelin# cd /opt/zeppelin/zeppelin-0.9.0-bin-all

```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# ls
bin    interpreter  lib          licenses    NOTICE    README.md    zeppelin-web-angular-0.9.0.war
conf   k8s          LICENSE      notebook    plugins    zeppelin-web-0.9.0.war
```

root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# cd conf

```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# ls
configuration.xsl  log4j2.properties  log4j.properties2  shiro.ini.template  zeppelin-env.sh.template
interpreter-list   log4j.properties  log4j_yarn_cluster.properties  zeppelin-env.cmd.template  zeppelin-site.xml.template
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf#
```

- **root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# mv zeppelin-env.sh.template zeppelin-env.sh**
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# mv zeppelin-site.xml.template zeppelin-site.xml
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# cd /usr/lib/jvm

```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# mv zeppelin-env.sh.template zeppelin-env.sh
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# mv zeppelin-site.xml.template zeppelin-site.xml
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# cd /usr/lib/jvm
root@linux:/usr/lib/jvm# ls
```

3 step Setting up the environment variable

- **root@linux:/opt/zeppelin# nano zeppelin-env.sh**

```
root@linux:/usr/lib/jvm# cd -  
/opt/zeppelin/zeppelin-0.9.0-bin-all/conf  
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# nano zeppelin-env.sh
```

Переходим в конец файла и в режиме вводим текст

- **export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64**
- **export SPARK_HOME=/opt/spark**
- **^o** - записать файл и **^x** выйти;

- **Переходим в каталог zeppelin и запускаем демон:**
- **root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# cd ..**
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# ls

3 step Setting up the environment variable

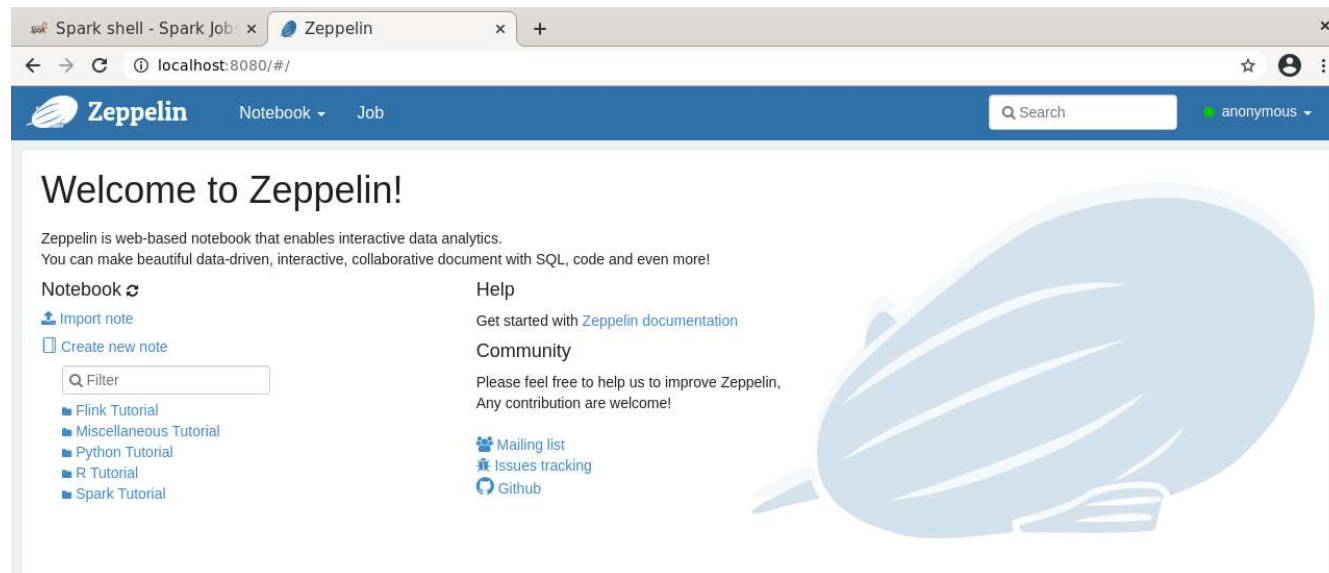
```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/conf# cd ..
```

```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# ls
```

```
bin      interpreter  lib      licenses  NOTICE  README.md  zeppelin-web-angular-0.9.0.war
conf     k8s          LICENSE  notebook  plugins  zeppelin-web-0.9.0.war
```

Запускаем ZEPPELIN

- **root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# ./bin/zeppelin-daemon.sh start**





Working with Big Data

TLC Trip Record Data - TLC - NYC.gov

<https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

Working with Big Data

```
hdmaster@linux:~$ cd /opt/zeppelin/
hdmaster@linux:/opt/zeppelin$ ls
zeppelin-0.9.0-bin-all  zeppelin-0.9.0-bin-all.tgz
hdmaster@linux:/opt/zeppelin$ cd zeppelin-0.9.0-bin-all
hdmaster@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all$ ls
bin          lib          notebook    run
conf         LICENSE     NOTICE     webapps
interpreter  licenses    plugins     zeppelin-web-0.9.0.war
k8s          logs        README.md   zeppelin-web-angular-0.9.0.war
hdmaster@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all$ mkdir tripdatatest
mkdir: cannot create directory 'tripdatatest': Permission denied
hdmaster@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all$ sudo su -
[sudo] password for hdmaster:
root@linux:~# cd /opr/zeppelin/^C
root@linux:~# cd /opt/zeppelin/zeppelin-0.9.0-bin-all
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# mkdir tripdatatest
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# ls
bin          LICENSE     plugins     zeppelin-web-0.9.0.war
conf         licenses    README.md   zeppelin-web-angular-0.9.0.war
interpreter  logs        run
k8s          notebook    tripdatatest
lib          NOTICE     webapps
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# cd tripdatatest
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/tripdatatest#
```

Working with Big Data

- `root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# wget https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2020-01.csv`

www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Get the latest on the COVID-19 Vaccine
Information on COVID-19 Zones
Agency service suspensions/reductions

NYC Taxi & Limousine Commission

Русский Translate Text-Size

About Passengers Drivers Vehicles Businesses TLC Online Search

About TLC Data and Research TLC Initiatives Contact TLC

TLC Trip Record Data

The yellow and green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP). The trip data was not created by the TLC, and TLC makes no representations as to the accuracy of these data.

The For-Hire Vehicle ("FHV") trip records include fields capturing the dispatching base license number and the pick-up date, time, and taxi zone location ID (shape file below). These records are generated from the FHV Trip Record submissions made by bases. Note: The TLC publishes base trip record data as submitted by the bases, and we cannot guarantee or confirm their accuracy or completeness. Therefore, this may not represent the total amount of trips dispatched by all TLC-licensed bases. The TLC performs routine reviews of the records and takes enforcement actions when necessary to ensure, to the extent possible, complete and accurate information.

Expand All Collapse All

2020

Due to COVID-19 and its impact on the daily operations of small businesses, TLC granted smaller bases an extension on trip record submissions. Trip and trip-related data for these bases will be updated as it becomes available.

January	July
<ul style="list-style-type: none"> Yellow Taxi Trip Records (CSV) Green Taxi Trip Records (CSV) For-Hire Vehicle Trip Records (CSV) High Volume For-Hire Vehicle Trip Records (CSV) 	<ul style="list-style-type: none"> Yellow Taxi Trip Records (CSV) Green Taxi Trip Records (CSV) FHV Trip Records (CSV) High Volume For-Hire Vehicle Trip Records (CSV)

Working with Big Data

```
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all# wget
https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2020-01.csv
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/tripdatatest# ls
yellow_tripdata_2020-01.csv
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/tripdatatest# du
579708.
root@linux:/opt/zeppelin/zeppelin-0.9.0-bin-all/tripdatatest# ls -l
total 579704-rw-r--r-- 1 root root 593610736 июл 29 2020
yellow_tripdata_2020-01.csv
```



Спасибо за внимание.