

Streaming de Dados em Tempo Real: Aula 2

Prof. Felipe Timbó



Ementa (dia 2)

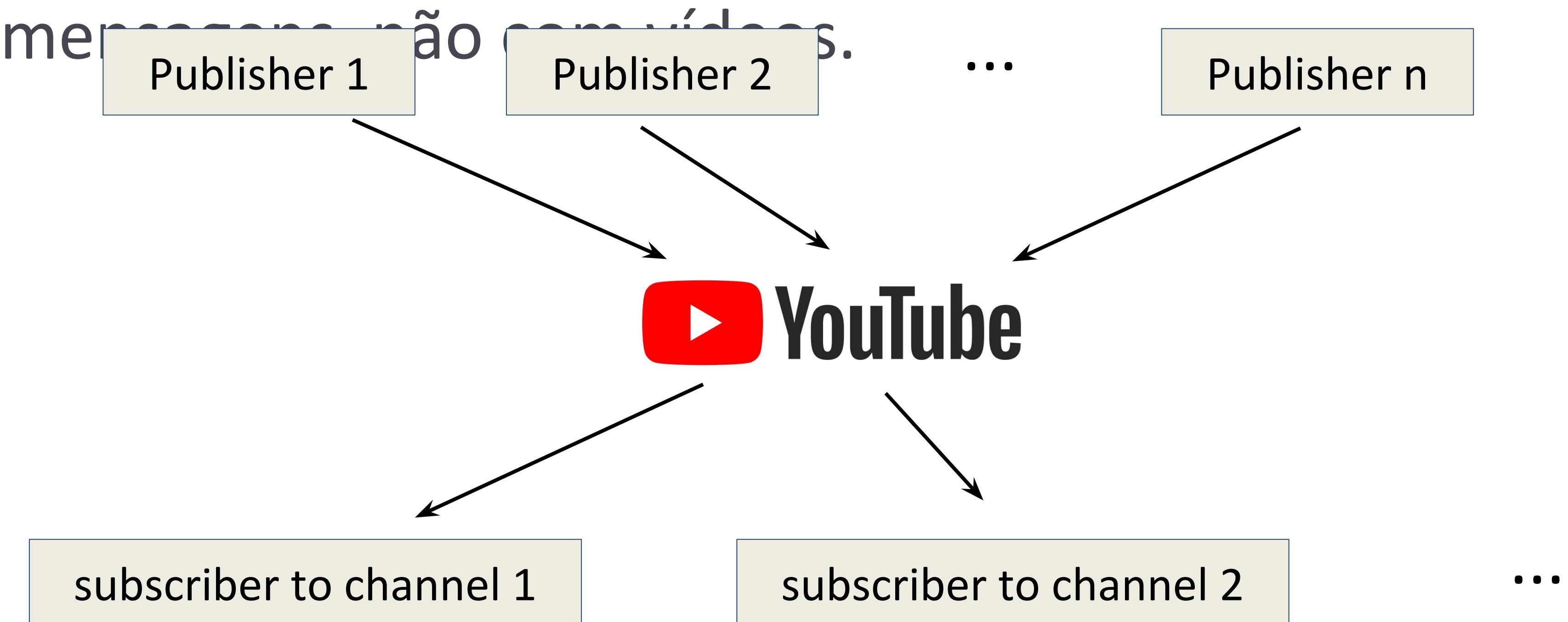
- Apache Kafka
- Como processar os dados de Streaming
- Apache Spark
- Resolução de problemas com Spark

Apache Kafka

(cont.)

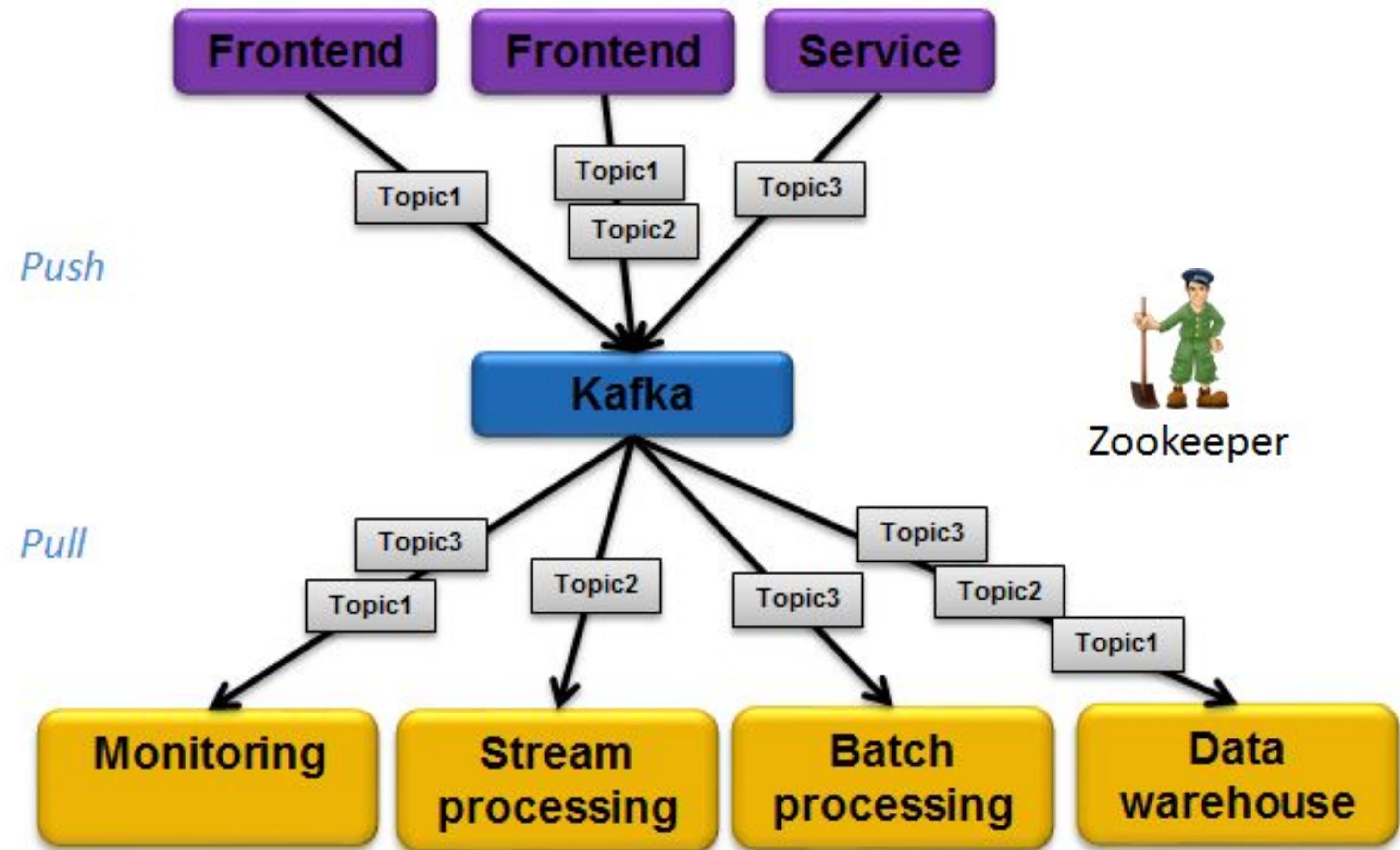
Publisher-subscriber system

- Kafka pode ser visto como um sistema publisher/subscriber, como o Youtube, mas com mensagens não como vídeos.



Kafka: relembrando

Producers

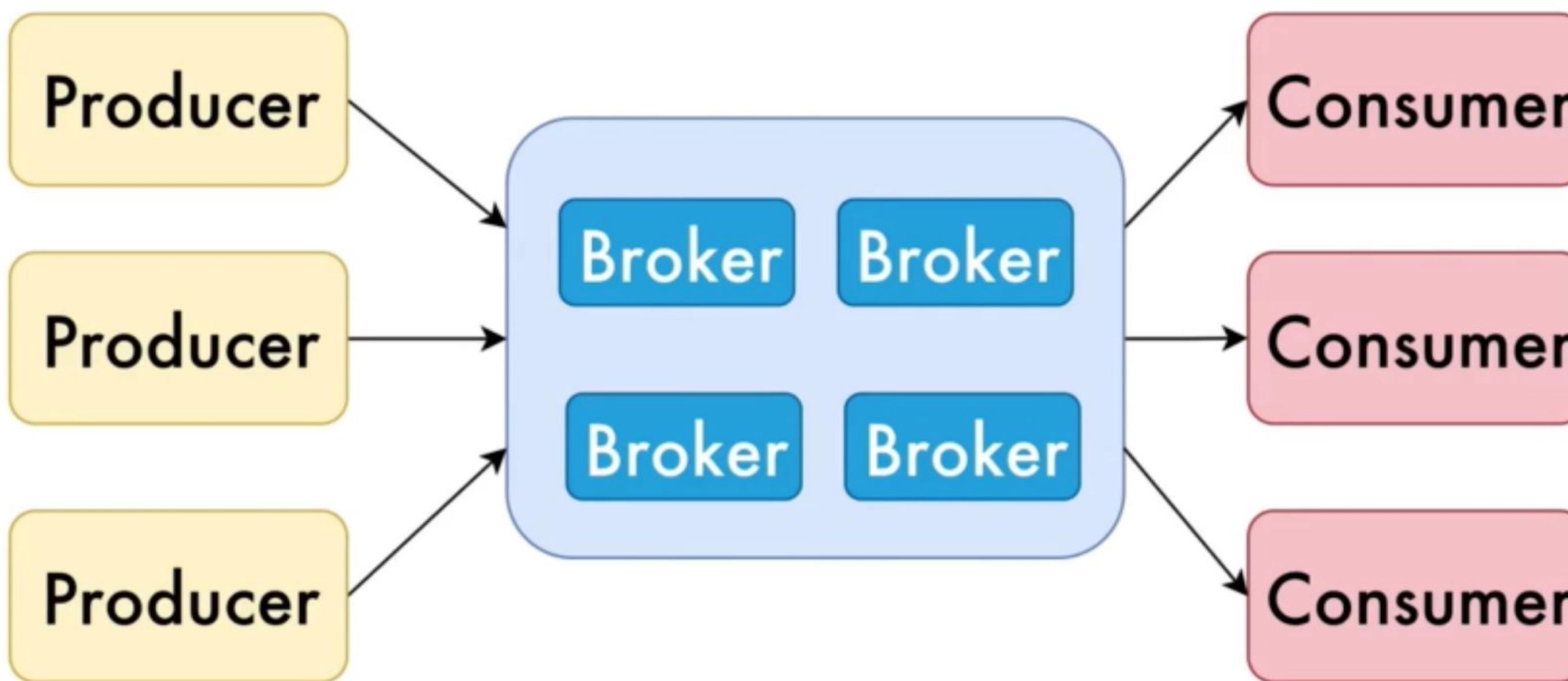


Broker

Consumers

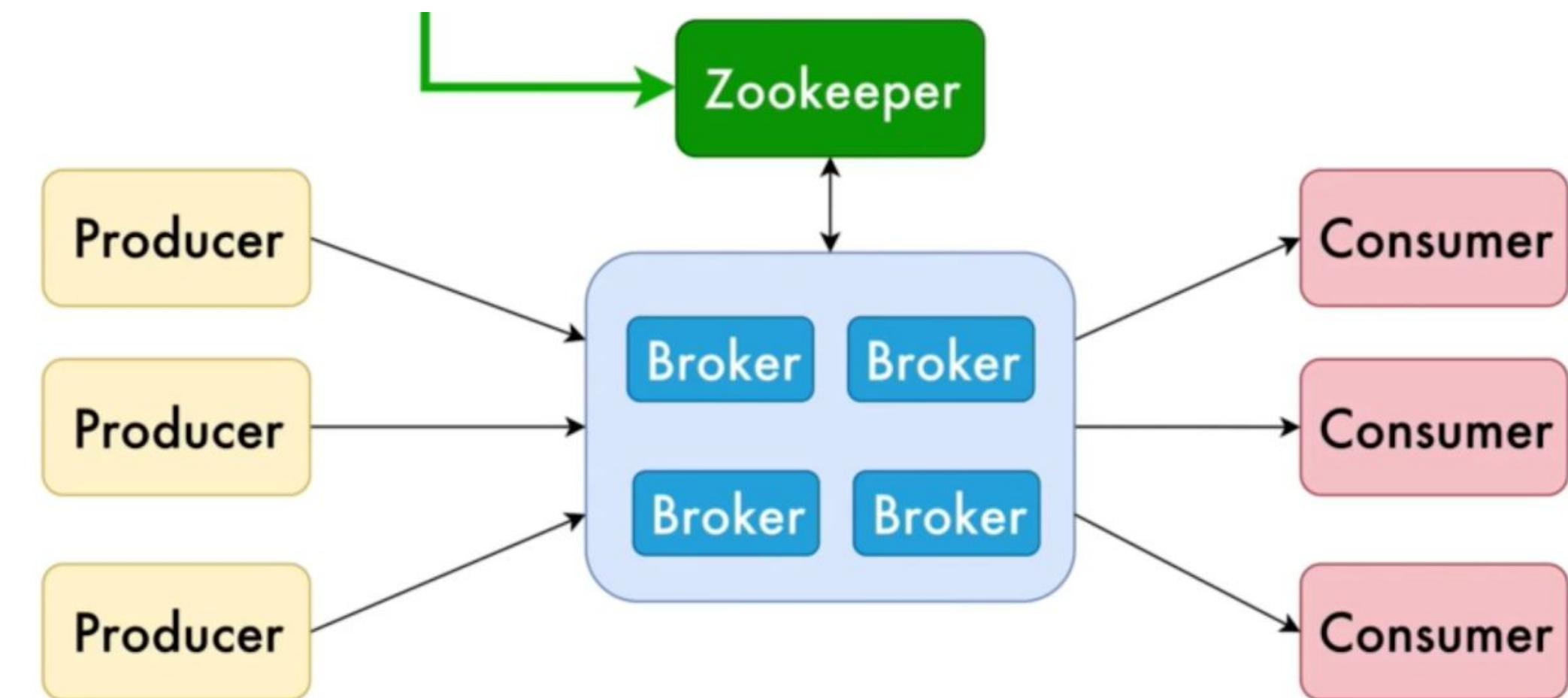
Kafka na vida real

- Em aplicações do mundo real: LinkedIn, Netflix:
 - Aplicações distribuídas, isto é, mais de um Broker



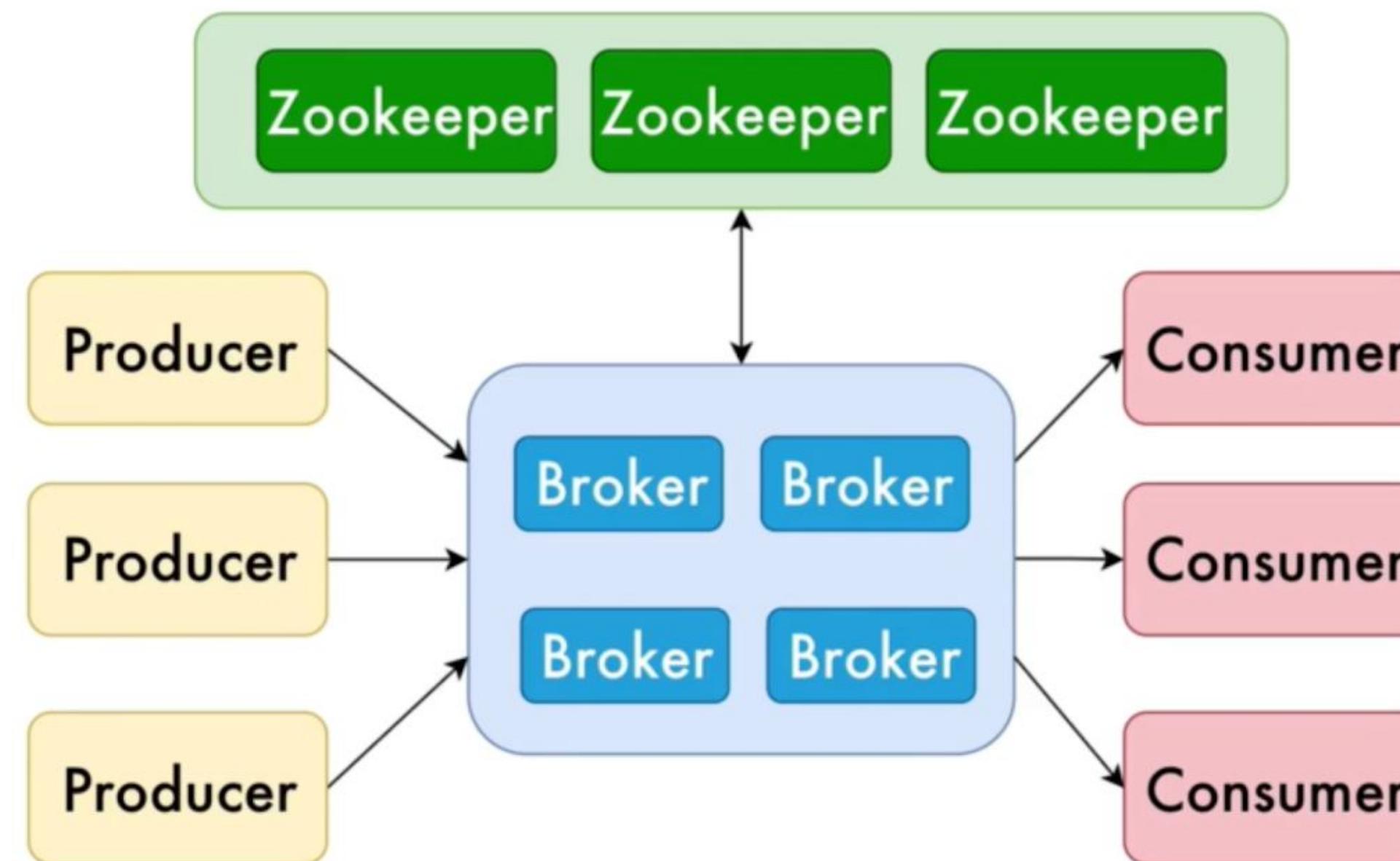
Zookeeper

- Utilizado não só pelo Apache Kafka, mas pelo Apache Hadoop também, por exemplo.
- Mantém uma lista de brokers ativos
- Elege controller
- Gerencia as configurações de tópicos e partições.



Zookeeper na vida real

- Múltiplos zookeepers rodando, para garantir a tolerância a falhas e os benefícios dos sistemas distribuídos.



Instalando o KAFKA

Instalando o Apache Kafka

Realizar o download do Apache Kafka:

```
➤ curl  
http://ftp.unicamp.br/pub/apache/kafka/2.6.0/ka  
fka_2.12-2.6.0.tgz -o Downloads/kafka.tgz
```

```
posgrad@posgrad-vm:~$ curl http://ftp.unicamp.br/pub/apache/kafka/2.6.0/kafka_2.  
12-2.6.0.tgz -o Downloads/kafka.tgz  
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current  
          Dload  Upload Total   Spent    Left Speed  
100 62.6M  100 62.6M    0      0  528k      0  0:02:01  0:02:01 --:--:-- 345k  
posgrad@posgrad-vm:~$
```

Instalando o Apache Kafka

Criar um diretório chamado kafka e entrar nele:

- mkdir kafka
- cd kafka

Extrair os arquivos que estão na pasta download para o diretório criado:

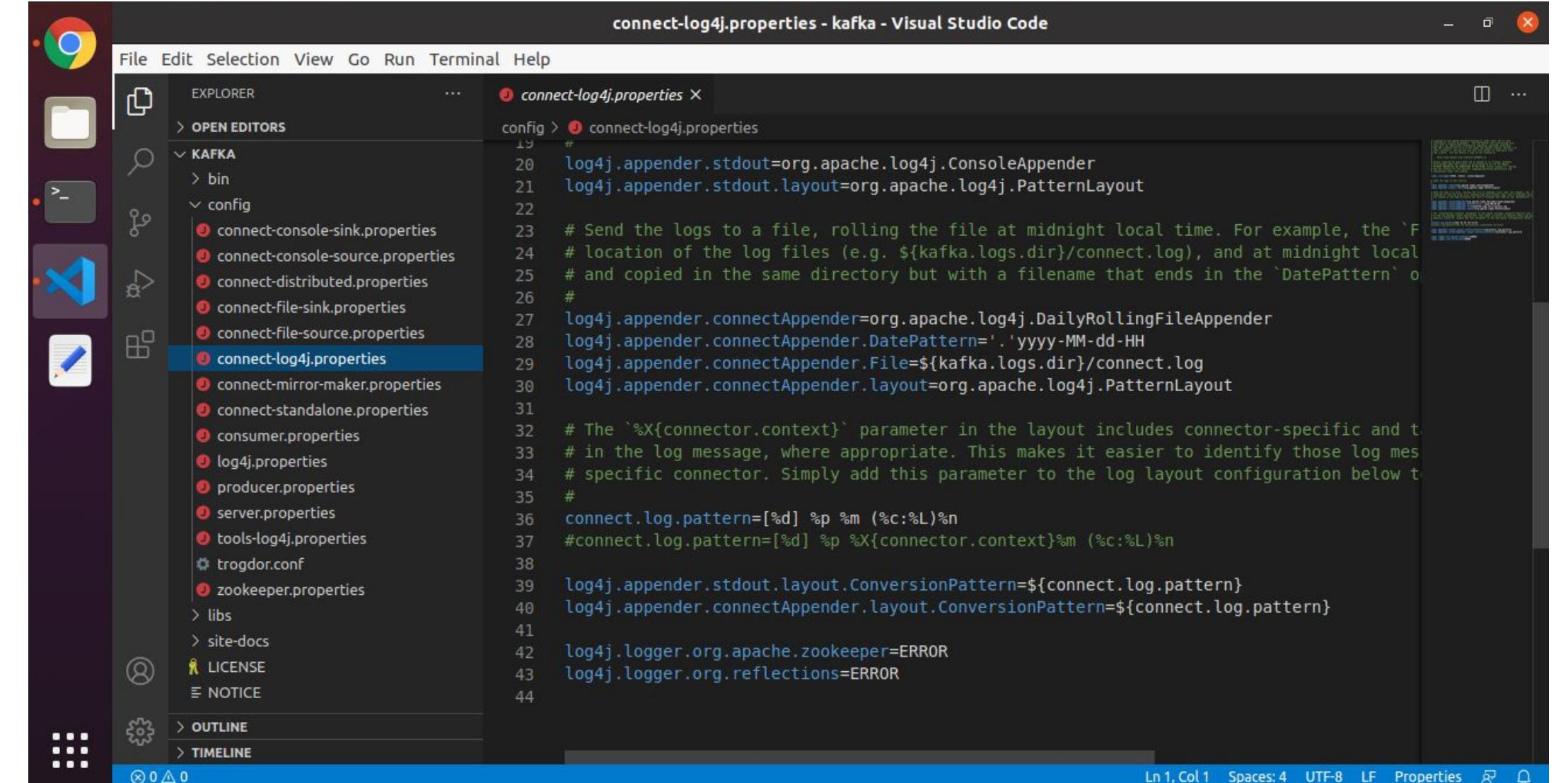
- tar -xvzf ~/Downloads/kafka.tgz --strip 1

Visualizando os arquivos do Kafka

Abrir o VS Code

Clicar em Open Folder

Selecionar a pasta kafka



The screenshot shows the Visual Studio Code interface with the title bar "connect-log4j.properties - kafka - Visual Studio Code". The left sidebar displays a file tree under the "KAFKA" folder, with "connect-log4j.properties" selected. The main editor area shows the content of the "connect-log4j.properties" file:

```
connect-log4j.properties - kafka - Visual Studio Code

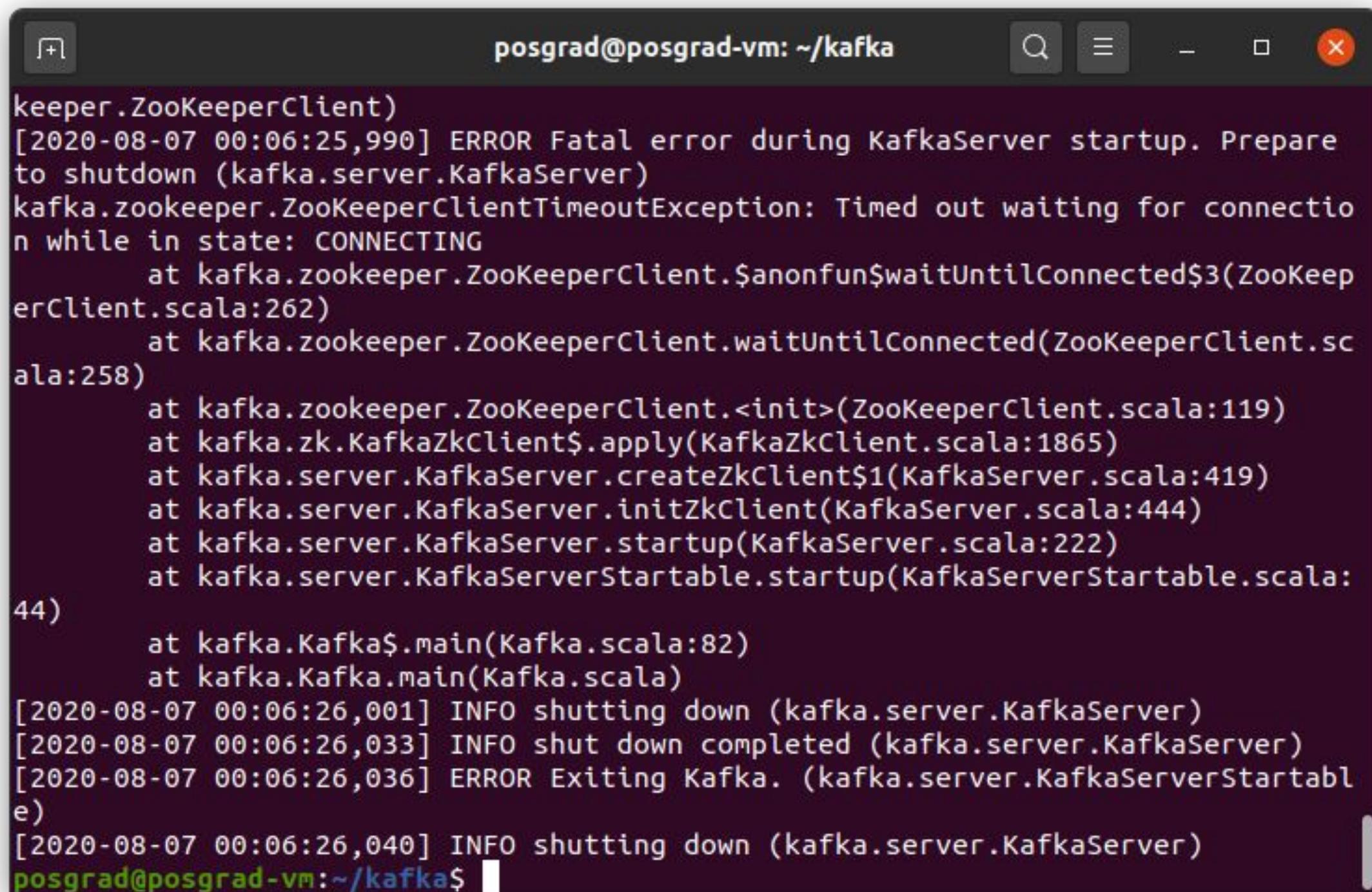
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
KAFKA
> bin
config
  connect-console-sink.properties
  connect-console-source.properties
  connect-distributed.properties
  connect-file-sink.properties
  connect-file-source.properties
  connect-log4j.properties
  connect-mirror-maker.properties
  connect-standalone.properties
  consumer.properties
  log4j.properties
  producer.properties
  server.properties
  tools-log4j.properties
  trogdr.conf
  zookeeper.properties
> libs
> site-docs
LICENSE
NOTICE
OUTLINE
TIMELINE

connect-log4j.properties ×
config > connect-log4j.properties
19 #
20 log4j.appenders.stdout=org.apache.log4j.ConsoleAppender
21 log4j.appenders.stdout.layout=org.apache.log4j.PatternLayout
22
23 # Send the logs to a file, rolling the file at midnight local time. For example, the `F
24 # location of the log files (e.g. ${kafka.logs.dir}/connect.log), and at midnight local
25 # and copied in the same directory but with a filename that ends in the `DatePattern` o
26 #
27 log4j.appenders.connectAppender=org.apache.log4j.DailyRollingFileAppender
28 log4j.appenders.connectAppender.DatePattern='yyyy-MM-dd-HH
29 log4j.appenders.connectAppender.File=${kafka.logs.dir}/connect.log
30 log4j.appenders.connectAppender.layout=org.apache.log4j.PatternLayout
31
32 # The `'%X{connector.context}'` parameter in the layout includes connector-specific and t
33 # in the log message, where appropriate. This makes it easier to identify those log mes
34 # specific connector. Simply add this parameter to the log layout configuration below t
35 #
36 connect.log.pattern=[%d] %p %m (%c:%L)%n
37 #connect.log.pattern=[%d] %p %X{connector.context}%m (%c:%L)%n
38
39 log4j.appenders.stdout.layout.ConversionPattern=${connect.log.pattern}
40 log4j.appenders.connectAppender.layout.ConversionPattern=${connect.log.pattern}
41
42 log4j.logger.org.apache.zookeeper=ERROR
43 log4j.logger.org.reflections=ERROR
44
```

The status bar at the bottom indicates "Ln 1, Col 1 Spaces: 4 UTF-8 LF Properties ⌂ ⌂".

Inicializando o Servidor Kafka

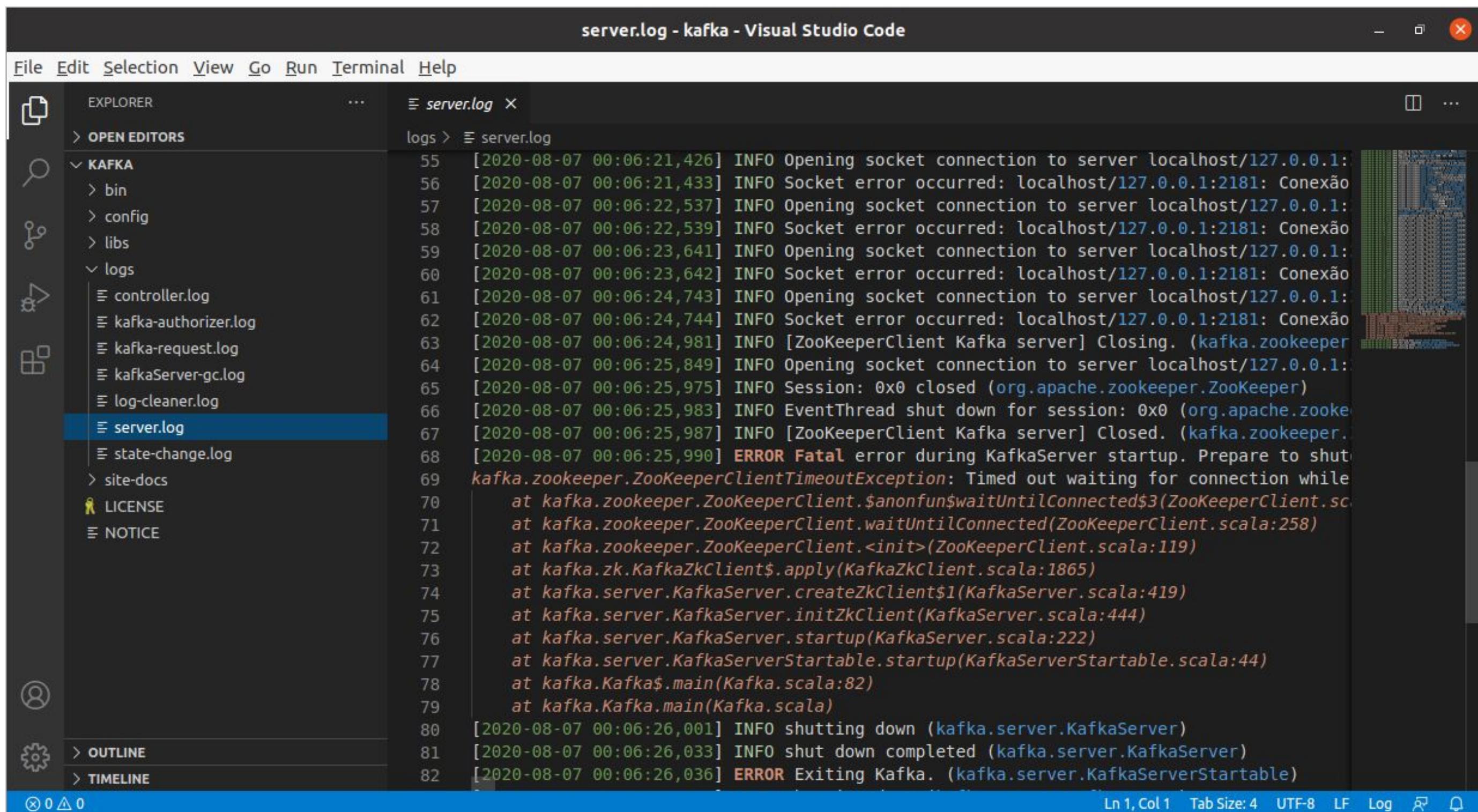
➤ bin/kafka-server-start.sh config/server.properties



```
keeper.ZooKeeperClient)
[2020-08-07 00:06:25,990] ERROR Fatal error during KafkaServer startup. Prepare
to shutdown (kafka.server.KafkaServer)
kafka.zookeeper.ZooKeeperClientTimeoutException: Timed out waiting for connectio
n while in state: CONNECTING
    at kafka.zookeeper.ZooKeeperClient.$anonfun$waitUntilConnected$3(ZooKeep
erClient.scala:262)
    at kafka.zookeeper.ZooKeeperClient.waitUntilConnected(ZooKeeperClient.sc
ala:258)
    at kafka.zookeeper.ZooKeeperClient.<init>(ZooKeeperClient.scala:119)
    at kafka.zk.KafkaZkClient$.apply(KafkaZkClient.scala:1865)
    at kafka.server.KafkaServer.createZkClient$(KafkaServer.scala:419)
    at kafka.server.KafkaServer.initZkClient(KafkaServer.scala:444)
    at kafka.server.KafkaServer.startup(KafkaServer.scala:222)
    at kafka.server.KafkaServerStartable.startup(KafkaServerStartable.scala:
44)
    at kafka.Kafka$.main(Kafka.scala:82)
    at kafka.Kafka.main(Kafka.scala)
[2020-08-07 00:06:26,001] INFO shutting down (kafka.server.KafkaServer)
[2020-08-07 00:06:26,033] INFO shut down completed (kafka.server.KafkaServer)
[2020-08-07 00:06:26,036] ERROR Exiting Kafka. (kafka.server.KafkaServerStartabl
e)
[2020-08-07 00:06:26,040] INFO shutting down (kafka.server.KafkaServer)
posgrad@posgrad-vm:~/kafka$
```

Observando os logs de erro do kafka

No VS Code, abrir a pasta logs, e o arquivo server.log



The screenshot shows a Visual Studio Code window titled "server.log - kafka - Visual Studio Code". The left sidebar has "OPEN EDITORS" expanded, showing a tree view of a "KAFKA" directory containing files like "bin", "config", "libs", "logs", and several log files: "controller.log", "kafka-authorizer.log", "kafka-request.log", "kafkaServer-gc.log", "log-cleaner.log", "server.log" (which is selected and highlighted in blue), and "state-change.log". The main editor area displays the contents of the "server.log" file, which is a log of Kafka server startup. Lines 55 through 82 are shown, detailing socket connections, errors, and finally a fatal error at line 68:

```
55 [2020-08-07 00:06:21,426] INFO Opening socket connection to server localhost/127.0.0.1:  
56 [2020-08-07 00:06:21,433] INFO Socket error occurred: localhost/127.0.0.1:2181: Conexão  
57 [2020-08-07 00:06:22,537] INFO Opening socket connection to server localhost/127.0.0.1:  
58 [2020-08-07 00:06:22,539] INFO Socket error occurred: localhost/127.0.0.1:2181: Conexão  
59 [2020-08-07 00:06:23,641] INFO Opening socket connection to server localhost/127.0.0.1:  
60 [2020-08-07 00:06:23,642] INFO Socket error occurred: localhost/127.0.0.1:2181: Conexão  
61 [2020-08-07 00:06:24,743] INFO Opening socket connection to server localhost/127.0.0.1:  
62 [2020-08-07 00:06:24,744] INFO Socket error occurred: localhost/127.0.0.1:2181: Conexão  
63 [2020-08-07 00:06:24,981] INFO [ZooKeeperClient Kafka server] Closing. (kafka.zookeeper.  
64 [2020-08-07 00:06:25,849] INFO Opening socket connection to server localhost/127.0.0.1:  
65 [2020-08-07 00:06:25,975] INFO Session: 0x0 closed (org.apache.zookeeper.ZooKeeper)  
66 [2020-08-07 00:06:25,983] INFO EventThread shut down for session: 0x0 (org.apache.zooke  
67 [2020-08-07 00:06:25,987] INFO [ZooKeeperClient Kafka server] Closed. (kafka.zookeeper.  
68 [2020-08-07 00:06:25,990] ERROR Fatal error during KafkaServer startup. Prepare to shut  
kafka.zookeeper.ZooKeeperClientTimeoutException: Timed out waiting for connection while  
70 at kafka.zookeeper.ZooKeeperClient.$anonfun$waitForConnected$3(ZooKeeperClient.sc  
71 at kafka.zookeeper.ZooKeeperClient.waitForConnected(ZooKeeperClient.scala:258)  
72 at kafka.zookeeper.ZooKeeperClient.<init>(ZooKeeperClient.scala:119)  
73 at kafka.zk.KafkaZkClient$.apply(KafkaZkClient.scala:1865)  
74 at kafka.server.KafkaServer.createZkClient$1(KafkaServer.scala:419)  
75 at kafka.server.KafkaServer.initZkClient(KafkaServer.scala:444)  
76 at kafka.server.KafkaServer.startup(KafkaServer.scala:222)  
77 at kafka.server.KafkaServerStartable.startup(KafkaServerStartable.scala:44)  
78 at kafka.Kafka$.main(Kafka.scala:82)  
79 at kafka.Kafka.main(Kafka.scala)  
80 [2020-08-07 00:06:26,001] INFO shutting down (kafka.server.KafkaServer)  
81 [2020-08-07 00:06:26,033] INFO shutdown completed (kafka.server.KafkaServer)  
82 [2020-08-07 00:06:26,036] ERROR Exiting Kafka. (kafka.server.KafkaServerStartable)
```

The status bar at the bottom shows "Ln 1, Col 1 Tab Size: 4 UTF-8 LF Log ⌂".

Inicializando o Zookeeper

➤ bin/zookeeper-server-start.sh config/zookeeper.properties

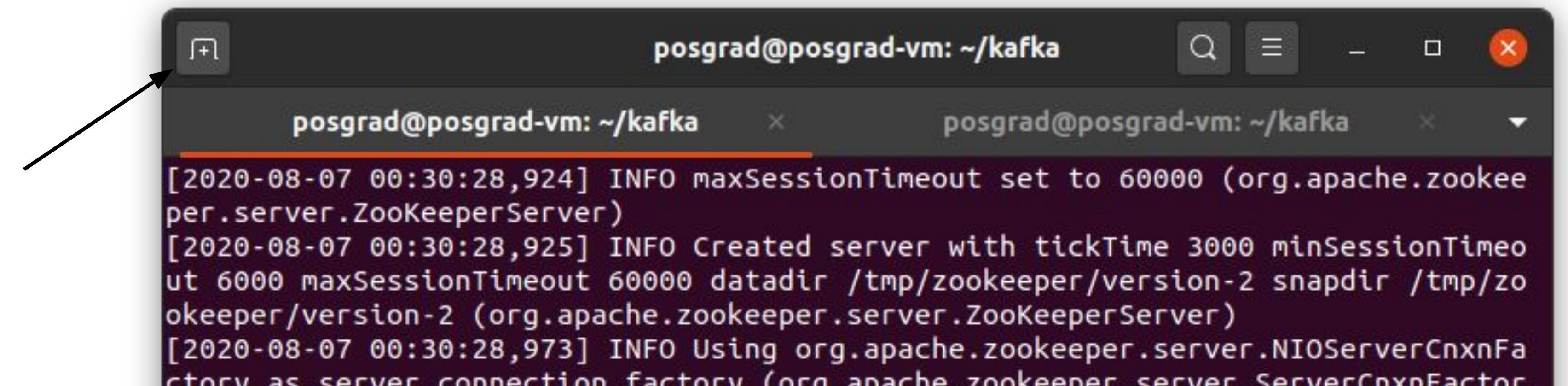
Zookeeper por padrão escuta a porta 2181

Observar arquivo ***zookeeper.properties*** na pasta ***config***

```
connect-standalone.properties      15 # the directory where the snapshot is stored.
                                16 dataDir=/tmp/zookeeper
                                17 # the port at which the clients will connect
                                18 clientPort=2181
                                19 # disable the per-ip limit on the number of connections since this is a non-production
                                20 maxClientCnxns=0
```

Inicializando o Servidor Kafka (novamente)

Em outro terminal



```
[2020-08-07 00:30:28,924] INFO maxSessionTimeout set to 60000 (org.apache.zookeeper.server.ZooKeeperServer)
[2020-08-07 00:30:28,925] INFO Created server with tickTime 3000 minSessionTimeout 6000 maxSessionTimeout 60000 datadir /tmp/zookeeper/version-2 snapdir /tmp/zookeeper/version-2 (org.apache.zookeeper.server.ZooKeeperServer)
[2020-08-07 00:30:28,973] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.NIOServerCnxnFactory)
```

Inicie o Kafka novamente:

➤ bin/kafka-server-start.sh config/server.properties

Voilà

Zookeeper **localhost:2181**

Kafka server (broker) **localhost:9092**

Utilizando o KAFKA via CLI

Criando um tópico

Em outro terminal: **Terminal 1**

```
➤ bin/kafka-topics.sh --create --bootstrap-server  
localhost:9092 --topic cidades
```

Observar o que acontece na criação de um tópico:

```
➤ cd logs  
➤ ls
```

Listar os tópicos existentes:

```
➤ cd ..  
➤ bin/kafka-topics.sh --list --zookeeper  
localhost:2181
```

Produzindo mensagens

Acessando o kafka broker para enviar mensagens

```
➤ bin/kafka-console-producer.sh --broker-list  
localhost:9092 --topic cidades  
    > Fortaleza  
    > Sobral  
    > Canindé  
    > Russas  
    > Quixadá
```

Consumindo mensagens

Em outro terminal: **Terminal 2**

```
> bin/kafka-console-consumer.sh --bootstrap-server  
localhost:9092 --topic cidades
```

De volta ao **Terminal 1** digitar:

- > New York
- > Dubai
- > Rio de Janeiro
- > Buenos Aires

Consumindo mensagens do início

Terminal 2

Parar o terminal 2:

➤ `ctrl+c`

Executar novamente para obter mensagens desde o início:

➤ `bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic cidades --from-beginning`

Fatos importantes

Kafka armazena mensagens mesmo se elas já tenham sido consumidas por um de seus “consumers”

Algumas mensagens podem ser lidas múltiplas vezes por diferentes “consumers” em diferentes momentos

Criando um novo consumidor

Em outro terminal: Terminal 3

```
➤ bin/kafka-console-consumer.sh --bootstrap-server  
localhost:9092 --topic cidades
```

Múltiplos consumidores e múltiplos produtores podem
trocar mensagens via clusters kafka.

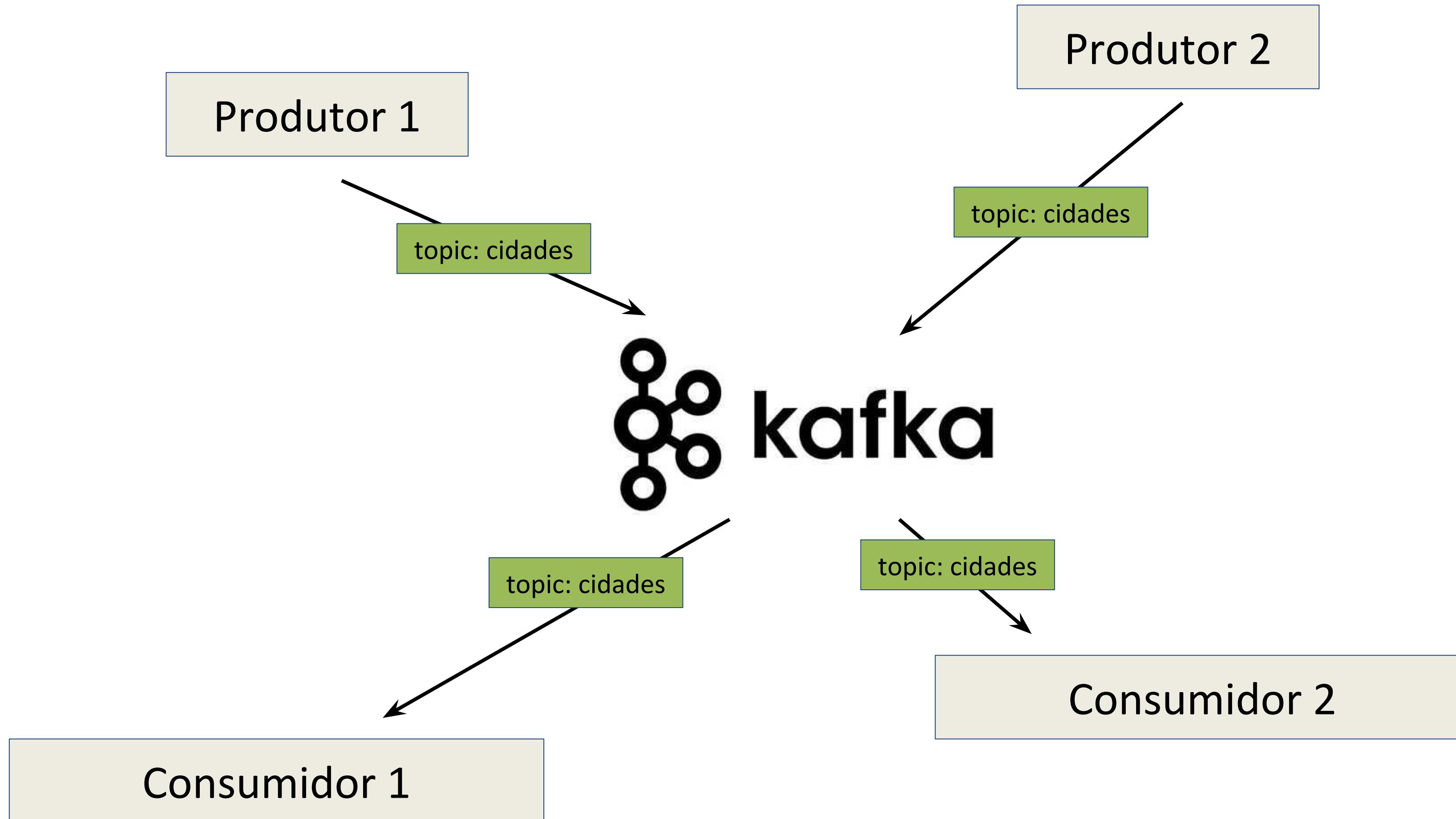
Criando um novo produtor

Em outro terminal: Terminal 4

```
➤ bin/kafka-console-producer.sh --broker-list  
localhost:9092 --topic cidades  
    > Barcelona
```

Produtores não sabem nada sobre outros produtores.

Esquema até então...



Ao encerrar o produtor 2
(Terminal 4) o que acontece
aos consumidores?

Ao deletar o produtor 2

Resposta: Apenas deixam de receber mensagens do produtor excluído, mas continuam operando normalmente.

Produtores recebem a mensagem do cluster kafka, independente dos consumidores.

Agora, você deve deletar todos os produtores/consumidores

Agrupar consumidores

Criar um consumidor em um grupo específico

Terminal 1

```
➤ bin/kafka-console-consumer.sh --bootstrap-server  
localhost:9092 --topic test --group dw  
--from-beginning
```

Listar os grupos de consumidores

Terminal 2

```
➤ bin/kafka-consumer-groups.sh --bootstrap-server  
localhost:9092 --list
```

Detalhar grupo de consumidores

Terminal 2

```
➤ bin/kafka-consumer-groups.sh --bootstrap-server  
localhost:9092 --group dw --describe
```

Comandos básicos do KAFKA - Resumo

Start Broker

```
bin/kafka-server-start.sh config/server.properties
```

Start Zookeeper

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
bin/kafka-topics.sh \
--bootstrap-server localhost:9092 \
--create \
--replication-factor 1 \
--partitions 3 \
--topic test
```

Create new topic

```
bin/kafka-topics.sh \
--bootstrap-server localhost:9092 \
--list
```

List all topics

```
bin/kafka-topics.sh \
--bootstrap-server localhost:9092 \
--describe \
--topic test
```

Details about the topic

```
bin/kafka-console-producer.sh \
--broker-list localhost:9092 \
--topic test
```

Start console producer

```
bin/kafka-console-consumer.sh \
--bootstrap-server localhost:9092 \
--topic test \
--from-beginning
```

Start console consumer

```
bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--list
```

List all consumer groups

```
bin/kafka-consumer-groups.sh \
--bootstrap-server localhost:9092 \
--group test \
--describe
```

Consumer group details

Atividade 1



1. Crie um tópico com o nome de vocês;
2. Liste os tópicos e verifique se o seu foi criado;
3. Gere dados para o tópico criado.

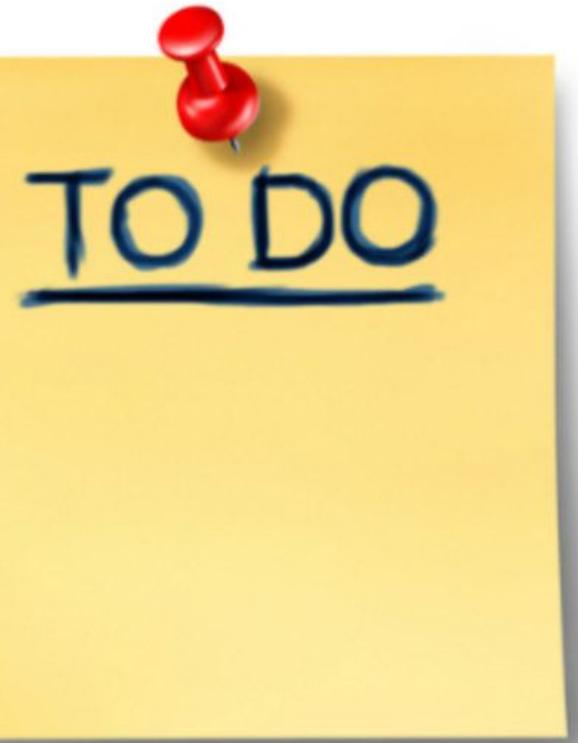
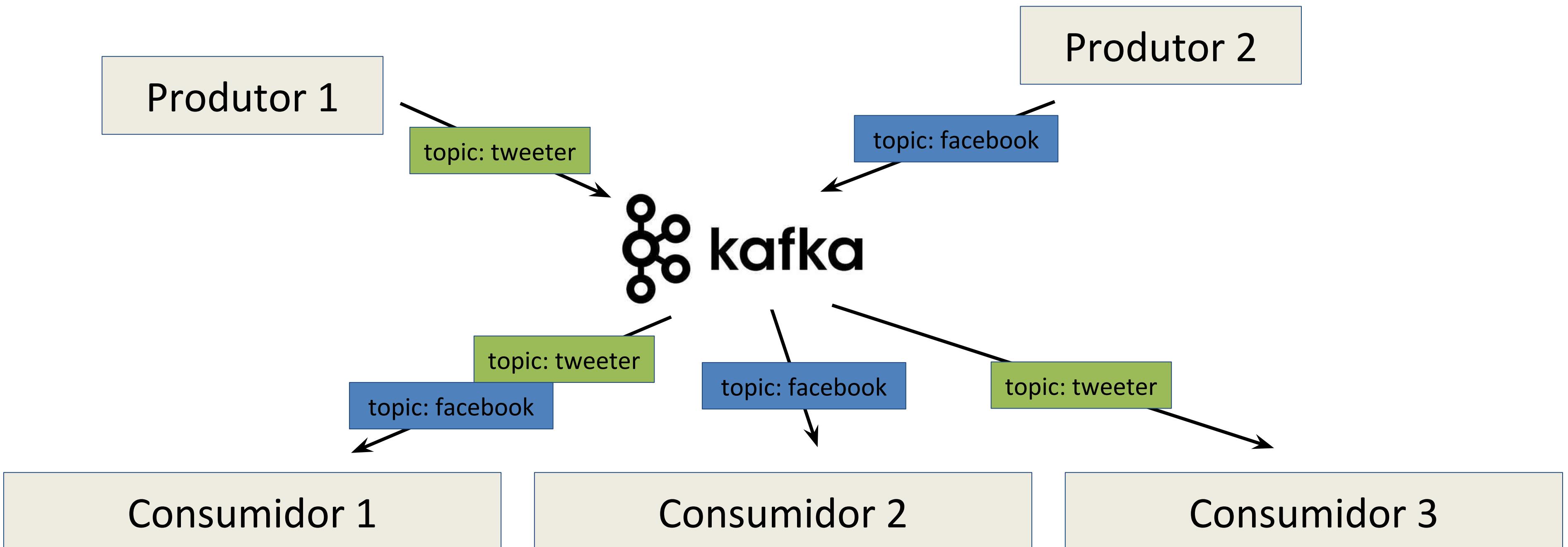
Obs.:

Endereço do zookeeper: localhost:2181

broker-list: localhost:9092

Atividade 2

Crie o seguinte esquema de troca de mensagens via streaming de dados no Apache KAKFA:



KAFKA - Prática 2

Produtor/consumidor com Python

Instalar dependência do kafka no python

- sudo apt install python3-pip
- pip3 install kafka-python

Abrir scripts no VS Code e criar os seguintes scripts:

- consumer.py
- producer.py

Criar e executar os scripts juntamente com o Professor.

KAFKA - Prática 3

Produtor/consumidor

Instalar dependência do fake names no python

```
➤ pip3 install Faker
```

Baixar códigos python produtor/consumidor

```
➤ wget www.lia.ufc.br/~timbo/streaming/producer.py  
➤ wget www.lia.ufc.br/~timbo/streaming/consumer.py
```

Abrir scripts no VS Code

Atividade 3



1. Altere o script para serem enviados dados de 2 em 2 segundos;
2. Crie um segundo consumidor, e altere o produtor para enviar os dados para ambos os consumidores;
3. Imprima nos consumidores apenas o nome das pessoas Fake que estão sendo produzidas, ao invés do conteúdo completo da mensagem;
4. Imprima no consumidor 1 o nome, e no consumidor 2 o sobrenome da pessoa Fake que é produzido.

KAFKA - Prática 4

Produtor/consumidor

Baixar código único python produtor/consumidor

➤ wget www.lia.ufc.br/~timbo/streaming/producer_consumer.py

Rodar script no VS Code

➤ python3 producer_consumer.py

Atividade 4



1. Gere dados de quatro producers simultâneos;
2. Aumente a frequência de geração das tuplas (geração mais rápida);
3. Filtre e imprima apenas por tuplas que possuem valores de peso maiores que 80;
4. Filtre e imprima apenas por tuplas que possuem valores de IMC acima de 35 ($IMC = peso/altura^2$).

Spark

Spark

- “*A fast and general engine for large-scale data processing*”
- Manipulações, transformações e análises complexas
 - Aprendizagem de máquina
 - Mineração de dados
 - Análise de grafos
 - Streaming
- Linguagens: Java, Python e Scala

 **Spark - Python Developer**
Prudens Inc

3 d

New York, New York Mais Nova Iorque vagas >
Our client is looking for Experienced Spark Developer. Candidate must have Pyspark and Python experience. All the data is stored in a parquet data format so potential candidates...

[in](#) Candidatura simplificada

 **APACHE SPARK Developer**
Wipro Limited

5 d

Sunnyvale, California Mais Sunnyvale vagas >
You should have good experience in application of standard software development principles. As a Senior APACHE SPARK Developer, you are responsible for development, support, ...

 **Apache Spark Developer - Top Pay For Expertise!!**
CyberCoders

Novo

New York City, NY, US Mais Nova Iorque vagas >
We are on a mission to create a new language for companies and their partners to communicate, understand, and improve each other's security posture. Apache Spark experience a MUST. [www.cybercoders.com](#)

[in](#) Candidatura simplificada

 **Spark Developer**
Apex Systems

Novo

Charlotte, NC, US Mais Charlotte vagas >
Apex Systems is looking for a Java Developer with Attivio experience. Experience with unit test frameworks including Junit, Jasmine, Karma, Mockito and PowerMock. Experience ... [itcareers.apexsystems.com](#)

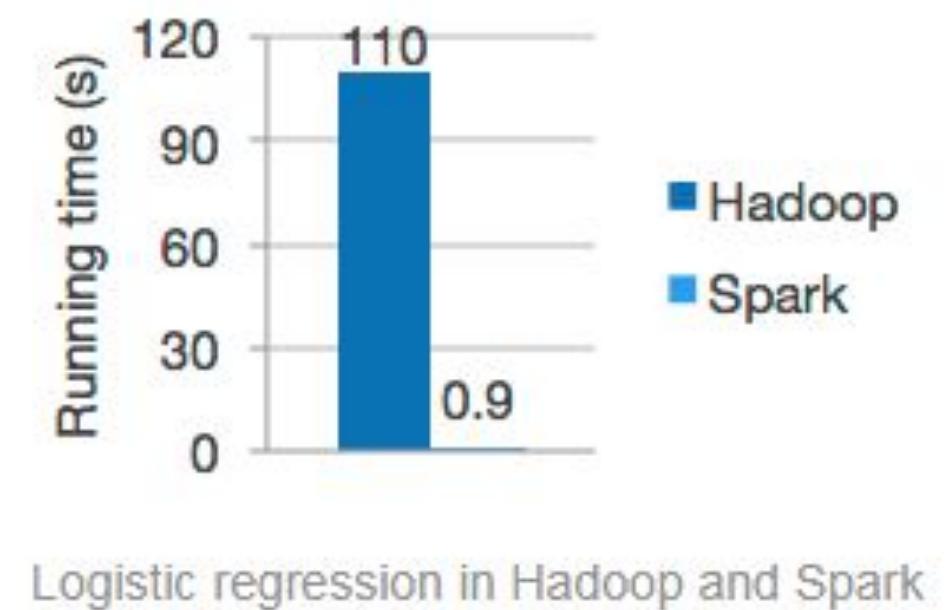
Spark - Quem usa?

- Amazon
- Ebay: análise de log e agregação
- NASA JPL: Deep Space Network
- Groupon
- TripAdvisor
- Etc. <https://spark.apache.org/powerd-by.html>



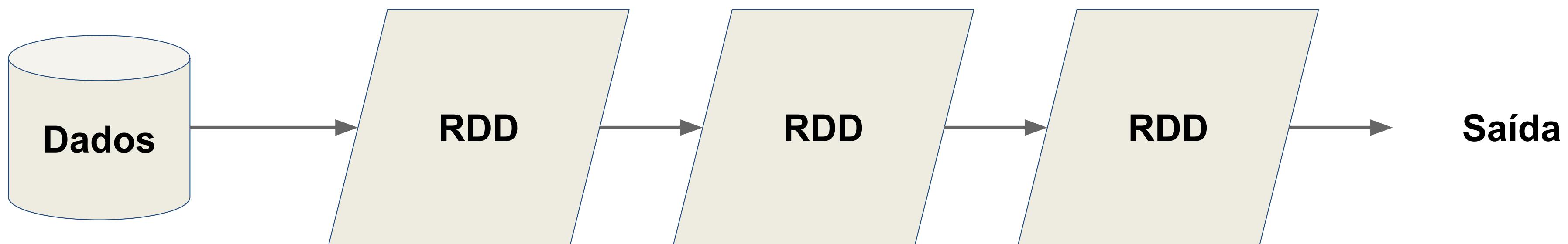
Spark - Desempenho

- 100x mais rápido que o Hadoop MapReduce em memória
- 10x mais rápido que o Hadoop MapReduce em disco
- 2x a 5x menos código



RDD - Resilient Distributed Dataset

- Abstração fornecida pelo Spark para a manipulação de dados.
- Representação de um dado distribuído pelos nós do cluster que pode ser operado em paralelo.
- Transformações e Ações



RDD - Resilient Distributed Dataset

Particionado

Dividido em nós
de um cluster

Imutável

RDDs, uma vez
criados, não podem
ser alterados

Resiliente

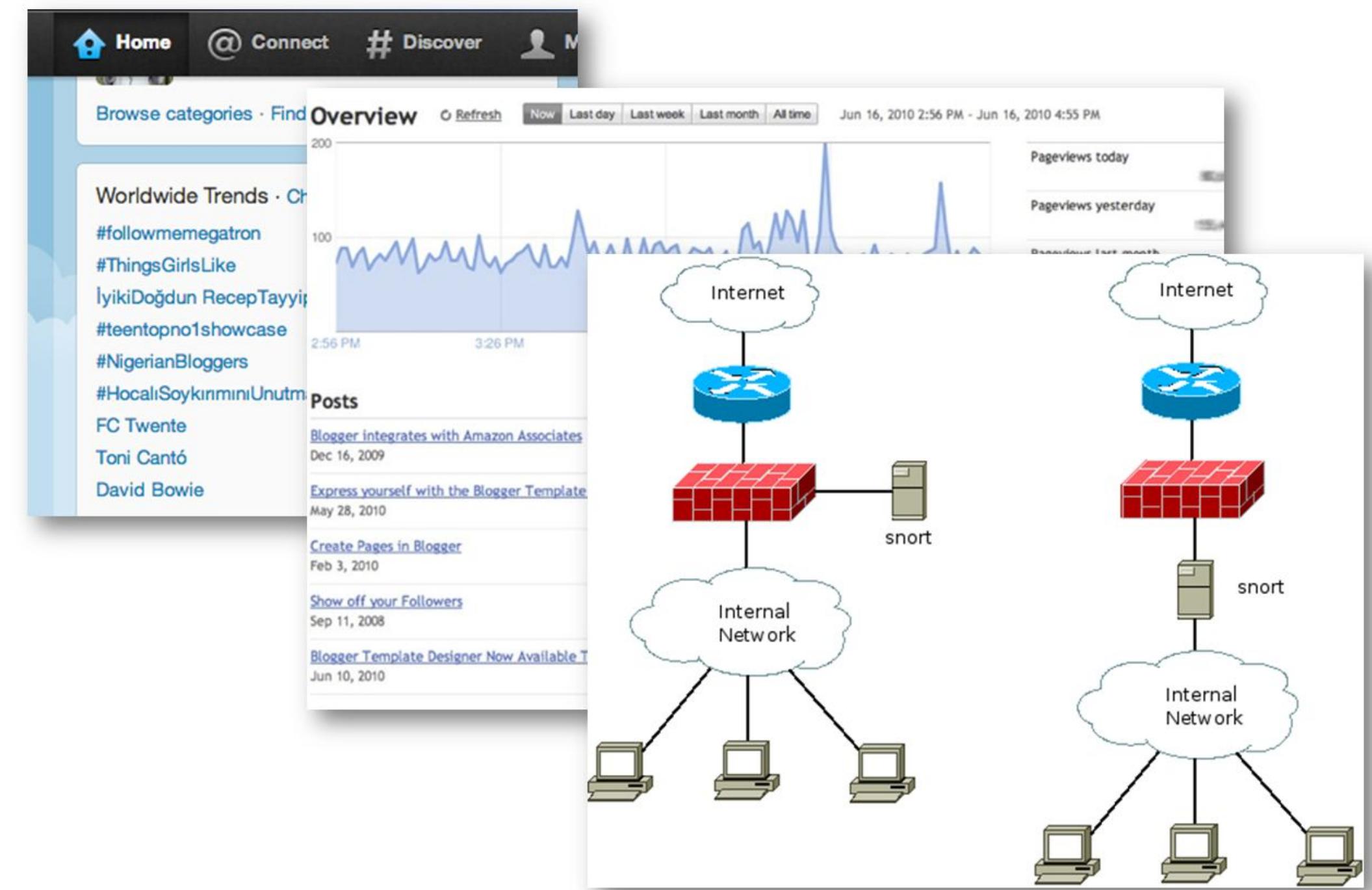
Podem ser
reconstruídos mesmo
se um nó caia

O que é Spark Streaming?

- Framework para processamento de streaming em larga escala
 - Suporta até 100 nós
 - Latência de segundos, em alguns casos
 - Integra-se com o modelo de processamento Spark em batch
 - Integra-se também com KAFKA, Flume, entre outros.

Motivação

- Análise de streaming de dados em tempo real.
 - Tendências em redes sociais
 - Estatísticas web
 - Detecção de intrusão de sistemas
 - etc.



Streaming de Dados

2016-12-30 09:09:57,862 INFO

org.apache.hadoop.http.HttpServer2: Jetty bound to port
56745

2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26

2016-12-30 09:09:58,037 INFO org.mortbay.log: Started
HttpServer2\$SelectChannelConnectorWithSafeStartup@localhost:
56745

2016-12-30 09:09:58,124 INFO

org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServe
r: Listening HTTP traffic on /0.0.0.0:50075

2016-12-30 09:09:58,239 INFO

Instalando o Spark

Instalando Hadoop no Ubuntu

Baixar Spark + Hadoop

```
➤ wget  
http://ftp.unicamp.br/pub/apache/spark/spark-2  
.4.6/spark-2.4.6-bin-hadoop2.7.tgz
```

Extrair os arquivos

```
➤ tar xvf spark-*
```

Mover os arquivos para o diretório *opt/spark*

```
➤ sudo mv spark-2.4.6-bin-hadoop2.7 /opt/spark
```

Configurando o ambiente Spark

Instalar Python 2.7

```
➤ sudo apt install python
```

Configurar variáveis de ambiente

```
➤ echo "export SPARK_HOME=/opt/spark" >> ~/.profile  
➤ echo "export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin" >> ~/.profile  
➤ echo "export PYSPARK_PYTHON=/usr/bin/python2.7" >> ~/.profile  
➤ echo "export JAVA_HOME=/usr/lib/jvm/default-java" >> ~/.profile
```

Carregar o arquivo .profile

```
➤ source ~/.profile
```

Configurando o ambiente Spark

Editar o arquivo .bashrc

➤ gedit ~/.bashrc

Adicionar as linhas a seguir ao final do arquivo:

```
export SPARK_HOME=/opt/spark  
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin  
export PYSPARK_PYTHON=/usr/bin/python2.7  
export JAVA_HOME=/usr/lib/jvm/default-java
```

Carregar o arquivo .profile e .bashrc

➤ source ~/.bashrc

Rodando Spark

Iniciar spark no ambiente standalone

- start-master.sh

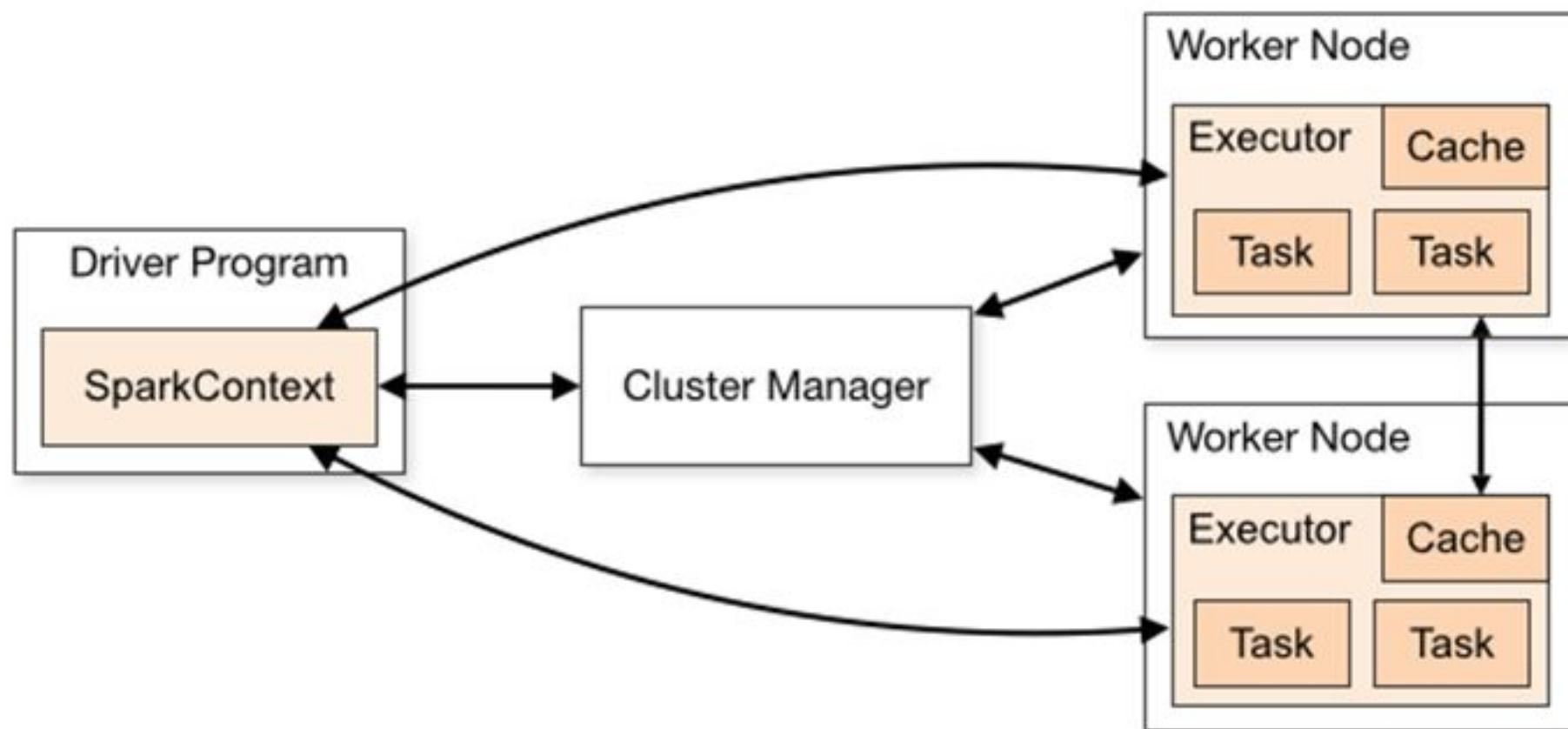
Observar no navegador o seguinte endereço:

- <http://127.0.0.1:8080/>

SparkContext

- Responsável por criar os RDDs

```
conf = SparkConf().setAppName(appName).setMaster(master)  
sc = SparkContext(conf=conf)
```



Criando um SparkContext

Example 2-7. Initializing Spark in Python

```
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf = conf)
```

Example 2-8. Initializing Spark in Scala

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._

val conf = new SparkConf().setMaster("local").setAppName("My App")
val sc = new SparkContext(conf)
```

Example 2-9. Initializing Spark in Java

```
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;

SparkConf conf = new SparkConf().setMaster("local").setAppName("My App");
JavaSparkContext sc = new JavaSparkContext(conf);
```

Criando um SparkContext em Python

Para criar um SparkContext:

```
➤ sc.stop()  
➤ from pyspark import SparkConf, SparkContext  
➤ conf = SparkConf().setAppName("MyApp")  
➤ sc = SparkContext(conf=conf)
```

Criando dados para “brincar”

Criar um arquivo de entrada:

```
➤ sudo su  
➤ cd ~  
➤ gedit students.csv
```

Iniciar o spark shell

➤ pyspark

Welcome to
██████████ version 2.1.

Using Python version 2.7.15rc1 (default, Apr 15 2018 21:51:34)
SparkSession available as 'spark'.

Nome	Idade
John	23
Jim	22
Emily	41
Nina	50
Susan	31

Brincando com os dados

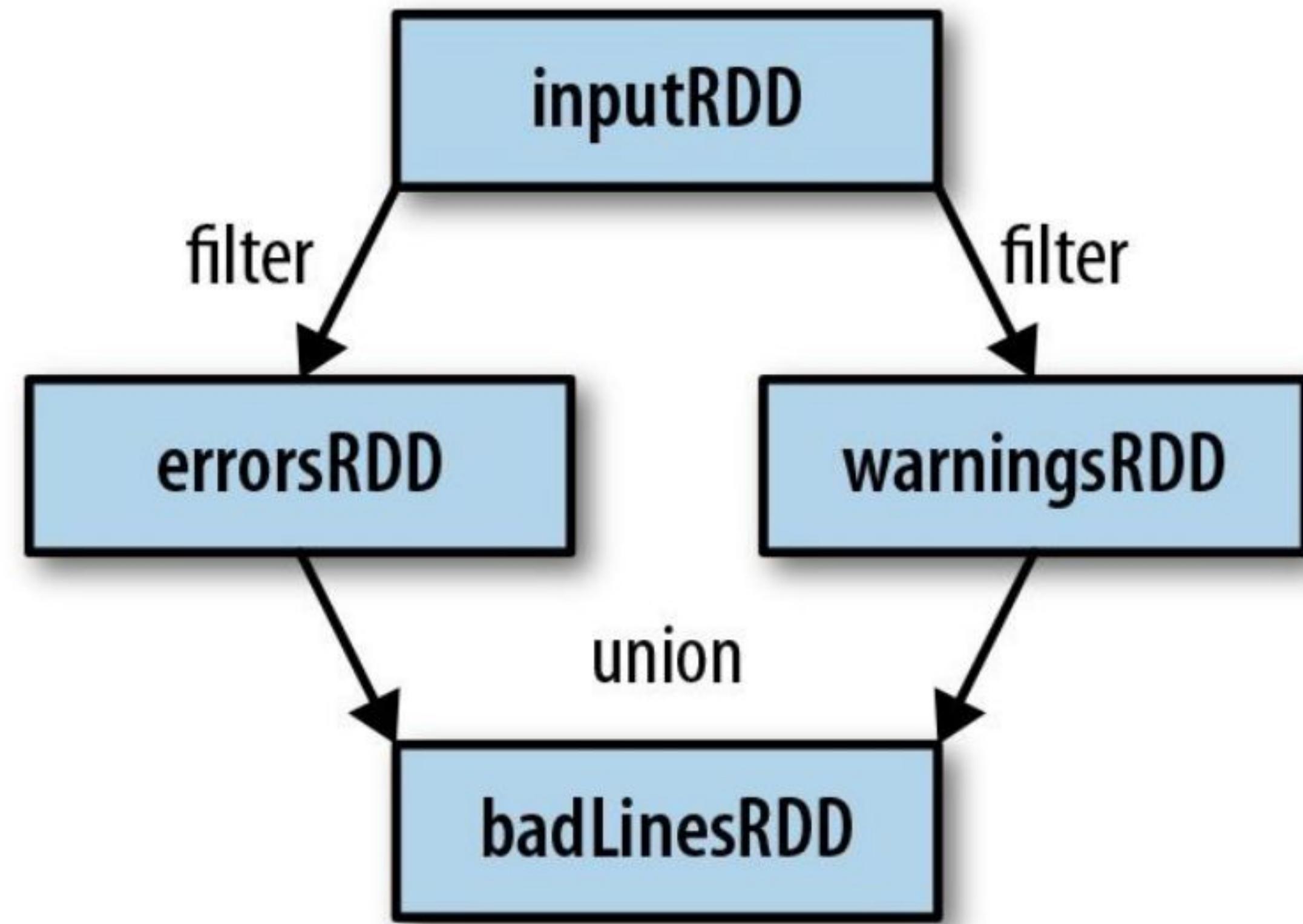
- students =
sc.textFile("file:///home/posgrad/students.csv")
- students.take(2)
- someStudents = students.filter(lambda row: 'i' in row)

- nums = sc.parallelize([1, 2, 3, 4])
- nums.map(lambda x: x+1)
- nums2 = sc.parallelize([4, 5, 6])
- nums.union(nums2).take(10)
- nums.intersection(nums2).take(10)

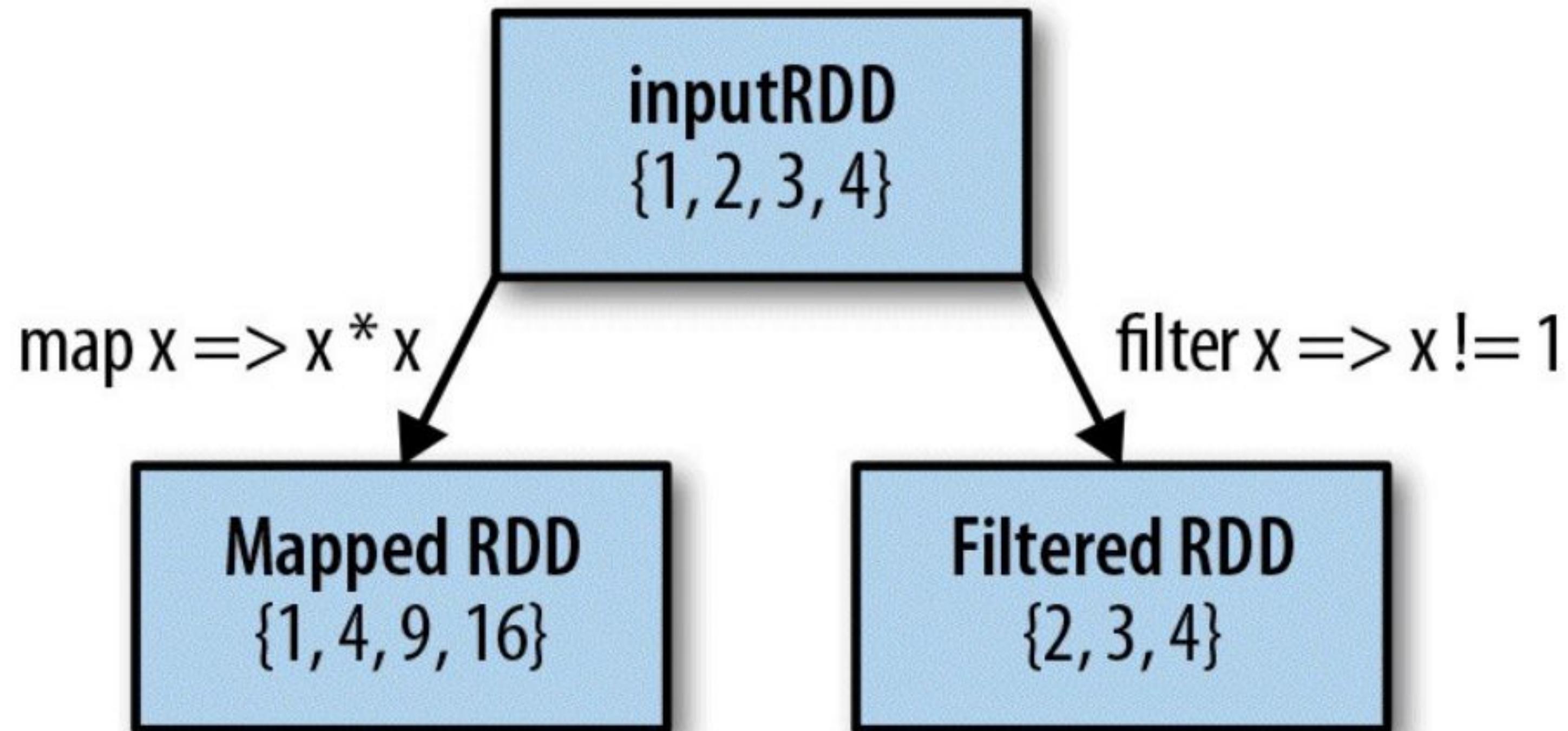
Transformações

- map
- flatmap
- filter
- distinct
- sample
- union, intersection, subtract, cartesian
- etc.

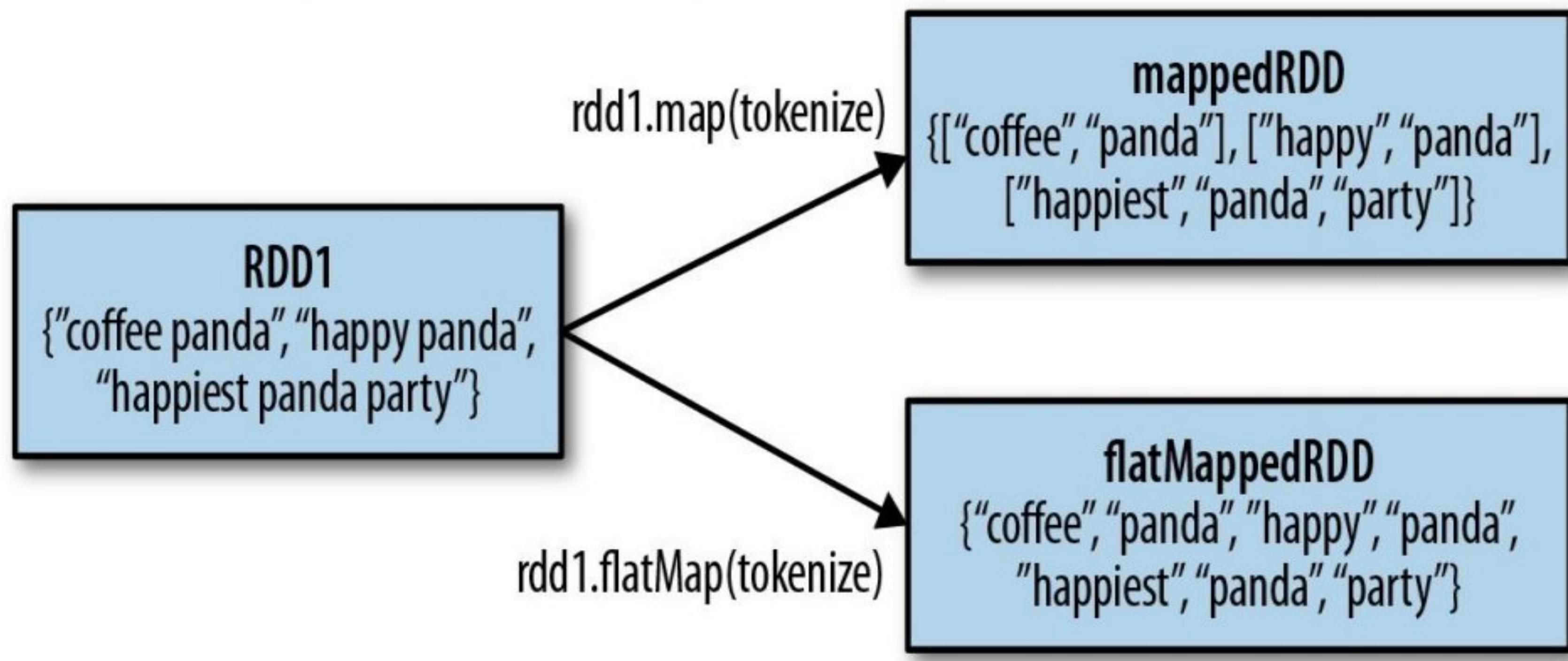
Transformações



Transformações



Transformações



Transformações

RDD1

{coffee, coffee, panda,
monkey, tea}

RDD2

{coffee, money, kitty}

RDD1.distinct()
{coffee, panda,
monkey, tea}

RDD1.union(RDD2)
{coffee, coffee, coffee,
panda, monkey,
monkey, tea, kitty}

RDD1.intersection(RDD2)
{coffee, monkey}

RDD1.subtract(RDD2)
{panda, tea}

Transformações (em Scala)

Transformação	Descrição	Exemplo	Resultado
map(func)	retorna um RDD depois de processar cada elemento com a função <i>func</i>	sc.parallelize(Array(1, 2, 3, 4, 5)).map (x => x + 10)	{ 11, 12, 13, 14, 15 }
filter(func)	Retorna um RDD depois de selecionar os elementos onde <i>func</i> é verdadeira	sc.parallelize(Array(1, 2, 3, 4, 5)).filter(x => x == 3)	3
flatMap(func)	Similar ao map, mas cada item pode ser mapeado para 0 ou mais items (<i>func</i> deve retornar uma sequência, ao invés de um item).	sc.parallelize(List("this is line 1", "this is the second line"), 2).flatMap(_.split(" ")).foreach {println _}	{this, is, the, second, line, this, is, line, 1}
mapPartitions(func)	Similar ao map, mas executa <i>func</i> separadamente em cada partição (bloco). A função <i>func</i> deve ser do tipo <i>Iterator<T> => Iterator<U></i> quando invocada sobre um RDD do tipo T.	sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8)), 3).mapPartitions (x => x.map((t: (Int, Int)) => t._1 + t._2)).foreach {println }	{3, 3, 3, 7, 11, 15, 15}

Transformações (em Scala)

Transformação	Descrição	Exemplo	Resultado
<code>union(otherDataset)</code>	Retorna uma coleção que contém a união de todos os elementos da fonte e do argumento.	<code>sc.parallelize(Array(1, 2, 3, 4, 5)).union(sc.parallelize(Array(6,7,8,9,10))).foreach(println _)</code>	{1,2,3,4,5,6,7,8,9,10}
<code>intersection(otherDataset)</code>	Retorna um RDD que contém uma interseção dos elementos da fonte e do argumento.	<code>sc.parallelize(Array(1, 2, 3, 4, 5)).intersection(sc.parallelize(Array(4,5,6,7,8))).foreach(println _)</code>	{5,4}
<code>distinct([numTasks])</code>	Retorna uma coleção que contém os elementos distintos da fonte.	<code>sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8))).distinct().foreach { println _ }</code>	{(1,2), (5,6), (7,8), (3,4)}
<code>groupByKey([numTasks])</code>	Quando invocado sobre uma coleção de pares (k, v), retorna um par (k, Iterable<v>).	<code>sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8))).groupByKey().foreach { println _ }</code>	(1,CompactBuffer(2, 2, 2)) (3,CompactBuffer(4)) (7,CompactBuffer(8, 8)) (5,CompactBuffer(6))
<code>reduceByKey(func, [numTasks])</code>	Quando invocado sobre uma coleção de pares (k, v), retorna uma coleção de pares (k, v) onde os valores de cada chave são agregados usando pela função <code>func</code> , que deve ser do tipo (v, v).	<code>sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8))).reduceByKey(_ + _).foreach { println _ }</code>	(1,6) (3,4) (7,16) (5,6)

Transformações (em Scala)

Transformação	Descrição	Exemplo	Resultado
sortByKey ([ascending], [numTasks])	Quando invocado sobre uma coleção de pares (k, v), onde k implementa Ordered, retorna uma coleção de pares (k, v) ordenados pela chaves.	sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8)), 1).sortByKey().foreach {println _}	(1,2) (1,2) (1,2) (3,4) (5,6) (7,8) (7,8)
join (otherDataset, [numTasks])	Quando invocado sobre uma coleção de pares do tipo (k, v) e (k, w), retorna uma coleção de pares (k, (v, w)).	sc.parallelize(Array ((1, "um"), (2, "dois"), (3, "tres"))).join(sc.parallelize(Array ((1, "one"), (2, "two"), (3, "three")))).foreach { println _ }	(1,(um,one)) (3,(tres,three)) (2,(dois,two))
cartesian (otherDatasets)	Quando invocado sobre uma coleção de dados do tipo T e U, retorna o conjunto de pares (T, U).	sc.parallelize (1 to 2).cartesian(sc.parallelize(10 to 11)).foreach {println _ }	(1,10) (1,11) (2,10) (2,11)

Ações

- collect
- count
- countByValue
- take
- top
- reduce
- etc.

Obs. Lazy evaluation: nada acontece até uma dessas funções serem chamadas!

Ações (em Scala)

Ações	Descrição	Exemplo	Resultado
reduce(func)	Agrega os elementos de uma coleção usando a função <i>func</i> (que precisa de dois argumentos, retornando um). A função deve ser acumulativa ou associativa, para ser computada corretamente em paralelo.	sc.parallelize(Array(1, 2, 3, 4, 5)).reduce((a, b) => a+b)	15
collect()	Retorna todos os elementos de uma coleção como um vetor dentro do programa <i>driver</i> .	sc.parallelize(Array(1, 2, 3, 4, 5)).collect	Array(1, 2, 3, 4, 5)
count()	Retorna o número de elementos dentro da coleção de dados.	sc.parallelize(Array(1, 2, 3, 4, 5)).count	5
first()	Retorna o primeiro elemento da coleção de dados.	sc.parallelize(Array(1, 2, 3, 4, 5)).first	1
take(n)	Return an array with the first <i>n</i> elements of the dataset.	sc.parallelize(Array(1, 2, 3, 4, 5)).take(2)	Array(1, 2)

Ações (em Scala)

Ações	Descrição	Exemplo	Resultado
takeSample(withReplacement, num, [seed])	Retorna um vetor com uma amostra aleatória de <i>num</i> selecionando opcionalmente uma semente para os números aleatóreos. O parâmetro <i>withReplacement</i> diz respeito a repetir os elementos na amostra ou não.	sc.parallelize(Array(1, 2)).takeSample(true, 4)	Array(1, 1, 2, 1)
takeOrdered(<i>n</i>, [<i>ordering</i>])	Retorna os primeiros <i>n</i> elementos de um RDD usando ou a ordem natural ou um comparador customizado.	sc.parallelize(Array(1, 2, 3, 4, 5)).takeOrdered(3)	Array(1, 2, 3)
saveAsTextFile(path)	Escreve todos os elementos da coleção de dados em um arquivo texto no sistema de arquivos local ou em qualquer sistema de arquivos suportado pelo Hadoop.	sc.textFile("README.md").saveAsTextFile("README.md2")	README.md2/_SUCCESS README.md2/part-00000 README.md2/part-00001
saveAsSequenceFile(path) (Java and Scala)	Escreve todos os elementos da coleção de dados como um <i>SequenceFile</i> do Hadoop no sistema de arquivos local, ou em qualquer sistema de arquivos suportado pelo Hadoop.	sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8))).saveAsSequenceFile("SEQ")	SEQ/_SUCCESS SEQ/part-00000 SEQ/part-00001

Ações (em Scala)

Ações	Descrição	Exemplo	Resultado
saveAsObjectFile(path) (Java and Scala)	Escreve todos os elementos da coleção de dados em um formato simples usando serialização Java, que pode ser deserializada posteriormente usando <i>SparkContext.objectFile()</i> .	sc.textFile("README.md").saveAsObjectFile("README.obj")	README.obj/_SUCCESS README.obj/part-00000 README.obj/part-00001
countByKey()	Somente disponível em RDDs do tipo (k, v) hashmap de pares (K, Int) com o número de ocorrência de cada chave.	sc.parallelize(Array((1,2), (1,2), (1,2), (3,4), (5,6), (7,8), (7,8))).countByKey	Map(1 -> 3, 3 -> 1, 7 -> 2, 5 -> 1)
foreach(func)	Executa uma função <i>func</i> em cada elemento da coleção de dados. Geralmente utilizada para atualizar contadores ou interagir com sistemas de arquivos externos. Observação: modificar <i>accumulators</i> fora de um <i>foreach</i> pode um comportamento indefinido.	sc.parallelize(Array(1, 2, 3, 4, 5)).foreach { println _ }	{1,2,3,4,5}

Como seria WordCount em Spark?

```
text_file = sc.textFile("hdfs://...")  
counts = text_file.flatMap(lambda line: line.split(" ")) \  
    .map(lambda word: (word, 1)) \  
    .reduceByKey(lambda a, b: a + b)  
counts.saveAsTextFile("hdfs://...")
```

Spark

Aplicações no modo standalone

WordCount em Spark

1. Download dos dados e do script

- cd ~
- wget <http://lia.ufc.br/~timbo/streaming/shakespeare.txt>
- wget <http://lia.ufc.br/~timbo/streaming/wordcount.py>

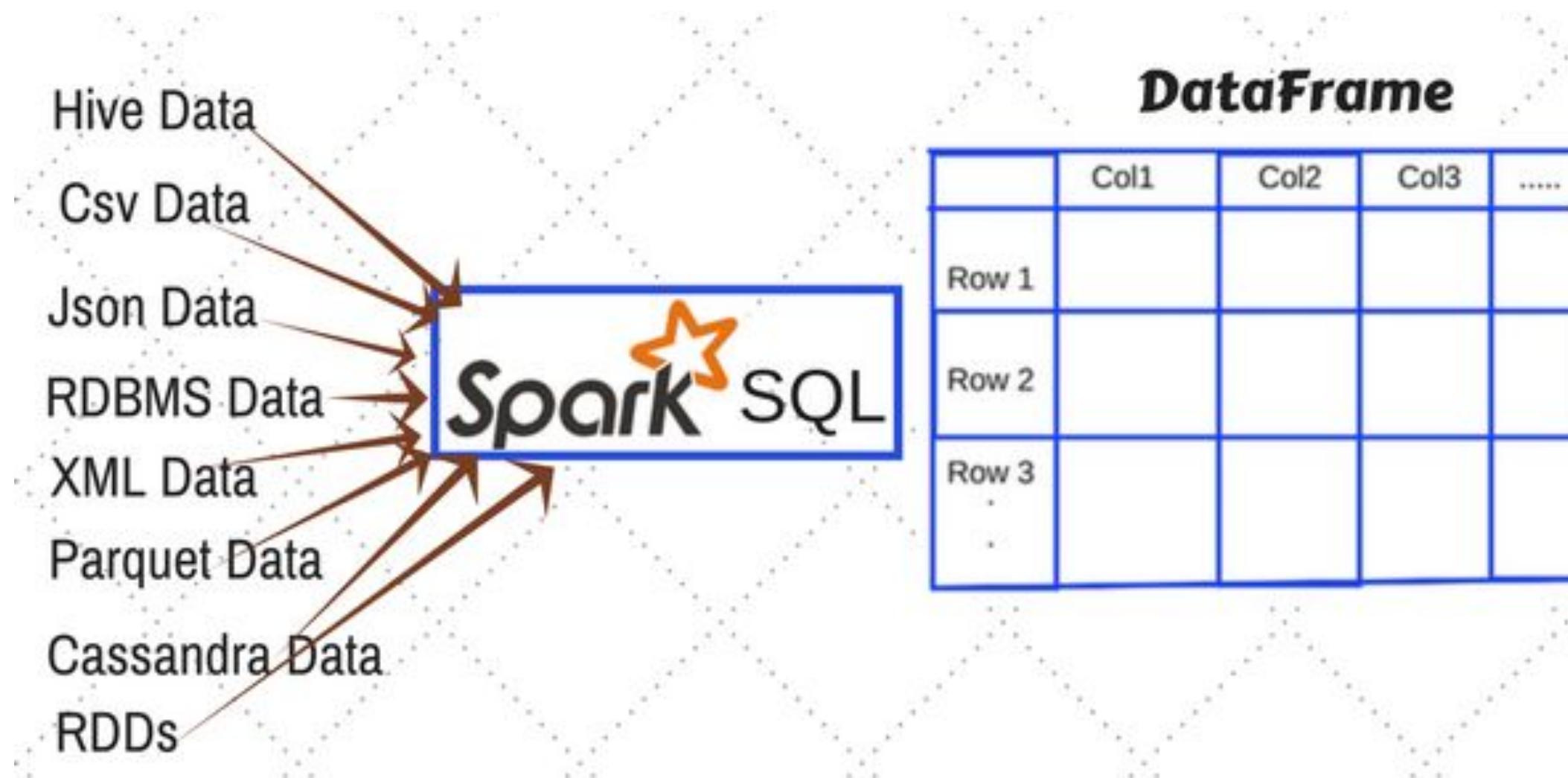
2. Rodando o script utilizando Spark:

- spark-submit wordcount.py

Spark 2

Spark 2 - Dataframes

- Extensão do RDD
- Pode rodar consultas SQL
- Possui um esquema (maneira melhor de lidar)



Spark 2 - Exemplo

Baixar conjunto de dados de pessoas

➤ wget <http://lia.ufc.br/~timbo/streaming/people.json>

Criando um SparkSession

Para criar um SparkSession:

- `from pyspark.sql import SparkSession`
- `spark = SparkSession.builder.appName("MyApp").getOrCreate()`

Criando um dataframe de pessoas

- `df = spark.read.json("file:///home/posgrad/people.json")`

Dataframes - funcionalidades

- `dataFrame.show()`
- `dataFrame.select("algum_campo")`
- `dataFrame.filter(dataFrame("algum_campo") > x)`
- `dataFrame.groupBy(dataFrame("algum_campo")).mean()`
- `dataFrame.rdd().map(funcao_map)`

Brincando com Dataframes

- df.show()
- df.printSchema()
- df.select("name").show()
- df.select(df['name'], df['age'] + 1).show()
- df.filter(df['age'] > 21).show()
- df.groupBy("age").count().show()
- df.orderBy("age", ascending = False).take(2)