

# Spark – Big Data Processing

Aula 8



# Semantix<sup>®</sup>

All about data

# Quem sou eu?



## Rodrigo Augusto Rebouças

Engenheiro de dados da Semantix  
Instrutor do Semantix Academy

### Contatos

[rodrigo.augusto@semantix.com.br](mailto:rodrigo.augusto@semantix.com.br)  
[linkedin.com/in/rodrigo-reboucas](https://www.linkedin.com/in/rodrigo-reboucas)



# Spark e Kafka



# Spark & Kafka

- Formas de integrar Kafka no Spark
  - Spark Streaming (DStreams)
    - Versão  $\geq$  Spark 0.7.0
    - <https://spark.apache.org/docs/2.4.1/streaming-kafka-integration.html>
  - Structured Streaming
    - Versão  $\geq$  Spark 2.0.0 (2.3)
    - <https://spark.apache.org/docs/2.4.4/structured-streaming-kafka-integration.html>

# Spark Streaming com Kafka

	<a href="#">spark-streaming-kafka-0-8</a>	<a href="#">spark-streaming-kafka-0-10</a>
Broker Version	0.8.2.1 or higher	0.10.0 or higher
API Maturity	Deprecated Spark 2.3	Stable
Language Support	Scala, Java, Python	Scala, Java
Receiver DStream	Yes	No
Direct DStream	Yes	Yes
SSL / TLS Support	No	Yes
Offset Commit API	No	Yes
Dynamic Topic Subscription	No	Yes

- <https://spark.apache.org/docs/2.4.1/streaming-kafka-integration.html>

# Spark Streaming com Kafka

- Spark Streaming (DStreams)
  - Scala e Java
    - Versão  $\geq$  Spark 0.7.0
    - <https://spark.apache.org/docs/latest/streaming-kafka-0-10-integration.html>
  - Python
    - Versão  $\geq$  Spark 1.2
    - Versão  $\leq$  Spark 2.3
    - <https://spark.apache.org/docs/2.4.1/api/python/pyspark.streaming.html#pyspark.streaming.kafka.KafkaUtils>
- Receber dados de um ou mais tópicos do Kafka
  - Configurar os parâmetros do StreamContext
  - Configurar os parâmetros do Kafka
  - Configurar o Dstreams para leitura dos tópicos

# Revisão de Kafka



# Exemplos de comandos no Kafka

- Acessar o container do Kafka
  - `docker exec --it kafka bash`
- Criação de tópico
  - `kafka-topics --bootstrap-server kafka:9092 --topic topicTeste --create --partitions 1 --replication-factor 1`
- Criação do produtor
  - `kafka-console-producer --broker-list kafka:9092 --topic topicTeste`
- Criação do consumidor
  - `kafka-console-consumer --bootstrap-server kafka:9092 --topic topicTeste`



# Exemplos de comandos no Kafka

- Criação do produtor com Chave/Valor
  - `kafka-console-producer --broker-list kafka:9092 --topic topicTeste --property parse.key=true --property key.separator=,`
- Criação do produtor enviando um arquivo
  - `kafka-console-producer --broker-list kafka:9092 --topic topicTeste < file.log`
- Criação do produtor enviando um arquivo com Chave/Valor
  - `kafka-console-producer --broker-list kafka:9092 --topic topicTeste --property parse.key=true --property key.separator=: < file.log`

# Spark Streaming com Kafka

- Dependências

# Spark Streaming com Kafka 0.10 - Dependências

- Adicionar pacote do Kafka
    - packages org.apache.spark:spark-streaming-kafka-0-10\_<versãoScala>:<versãoSpark>
  - Exemplo
    - \$ kafka-topics --version
- 2.3.0
- Link: <https://docs.confluent.io/current/installation/versions-interoperability.html>
    - Confluent Platform: 5.3.x (Ex. 5.3.1-css)
    - Apache Kafka: 2.3.x
  - \$ spark-shell
    - Scala: 2.11.12
    - Spark: 2.4.1
  - \$ spark-shell --packages org.apache.spark:spark-streaming-kafka-0-10\_2.11:2.4.1
  - SBT:  
libraryDependencies += "org.apache.spark" % "spark-streaming-kafka-0-10" % "2.4.1"

# Spark Streaming com Kafka

- Estrutura do código e importações

# Spark Streaming com Kafka

- Processos básicos para Leitura de dados do Kafka

```
scala> import ...
```

```
scala> val ssc = new StreamingContext( ... )
```

```
scala> val kafkaParams = Map[String, Object]( ... )
```

```
scala> val dstream = KafkaUtils.createDirectStream[String, String]( ... )
```

```
scala> dstream.map( ... )
```

```
scala> ssc.start()
```

# Importar packages

- Packages do Kafka 010 e Streaming

```
scala> import org.apache.kafka.clients.consumer.ConsumerRecord
```

```
scala> import org.apache.kafka.common.serialization.StringDeserializer
```

```
scala> import org.apache.spark.streaming.{Seconds, StreamingContext}
```

```
scala> import org.apache.spark.streaming.kafka010._
```

```
scala> import org.apache.spark.streaming.kafka010.LocationStrategies.PreferConsistent
```

```
scala> import org.apache.spark.streaming.kafka010.ConsumerStrategies.Subscribe
```

# Spark Streaming com Kafka

- **Parâmetros do Kafka**

# Parâmetros do Kafka

- Usar API Kafka para configuração

```
scala> val kafkaParams = Map[String, Object](  
  "bootstrap.servers" -> "localhost:9092",  
  "key.deserializer" -> classOf[StringDeserializer],  
  "value.deserializer" -> classOf[StringDeserializer],  
  "group.id" -> "app-teste",  
  "auto.offset.reset" -> "earliest",  
  "enable.auto.commit" -> (false: java.lang.Boolean)  
)
```



# Spark Streaming com Kafka

- Criação e Visualização do DStream

# Spark Streaming com Kafka 0.10

- Location Strategies
  - PreferConsistent
    - Distribuir partições uniformemente entre os executores disponíveis
  - PreferBrokers
    - Se seus executores estiverem nos mesmos hosts que seus corretores kafka
  - PreferFixed
    - Especifique um mapeamento explícito de partições para hosts
- Consumer Strategies
  - Subscribe
    - Inscrever-se em uma coleção fixa de tópicos
  - SubscribePattern
    - Use um regex para especificar tópicos de interesse
  - Assign
    - Especificar uma coleção fixa de partições

# Criação do Dstream para o tópico do Kafka

- Usar estratégia do Kafka

```
scala> val ssc = new StreamingContext(sc, Seconds(10))
scala> val topics = Array("topicA")
scala> val dstream = KafkaUtils.createDirectStream[String, String](
    ssc,
    LocationStrategies.PreferConsistent,
    ConsumerStrategies.Subscribe[String, String](topics, kafkaParams)
)
```

# Visualizar DStream

- Mapear a estrutura do Tópico

```
scala> val info_dstream = dstream.map(record => (  
    record.topic,  
    record.partition,  
    record.key,  
    record.value  
))
```

```
scala> info_dstream.print()
```

# Exercício 1 Kafka – Spark Streaming

1. Preparação do ambiente no Kafka
  - a) Criar o tópico “topic-spark” com 1 partição e o fator de replicação = 1
  - b) Inserir as seguintes mensagens no tópico:
    - Msg1
    - Msg2
    - Msg3
  - c) Criar um consumidor no Kafka para ler o “topic-spark”

# Exercício 1 Spark – Spark Streaming

1. Criar um consumidor em Scala usando Spark Streaming para ler o “topic-spark” no cluster Kafka “kafka:9092”
2. Visualizar o tópico com as seguintes informações
  - Nome do tópico
  - Partição
  - Valor
3. Salvar o tópico no diretório `hdfs://namenode:8020/user/<nome>/kafka/dstream`

## Exercício 2 Kafka – Spark Streaming

1. Preparação do ambiente no Kafka
  - a) Criar o tópico “topic-kvspark” com 2 partições e o fator de replicação = 1
  - b) Inserir as seguintes mensagens no tópico (Chave, Valor):
    - 1, Msg1
    - 2, Msg2
    - 3, Msg3
  - c) Criar um consumidor no Kafka para ler o “topic-kvspark”

## Exercício 2 Spark – Spark Streaming

1. Criar um consumidor em Scala usando Spark Streaming para ler o “topic-kvspark” no cluster Kafka “kafka:9092”
2. Visualizar o tópico com as seguintes informações
  - Nome do tópico
  - Partição
  - Chave
  - Valor
3. Salvar o tópico no diretório `hdfs://namenode:8020/user/<nome>/kafka/dstreamkv`





# Semantix<sup>®</sup>

All about data

[contato@semantix.com.br](mailto:contato@semantix.com.br)

[www.semantix.com.br](http://www.semantix.com.br)