**A VM da Cloudera utilizada nestes exercícios de Spark está disponível em:**

<http://training.cloudera.com/cloudera/VMs/Cloudera-Training-Spark-Developer-VM-5.0a-vmware.zip>

**EXERCÍCIO 1 - SPARK**

**Objetivo:** Neste exercício iremos praticar a utilização de RDDs no Spark.

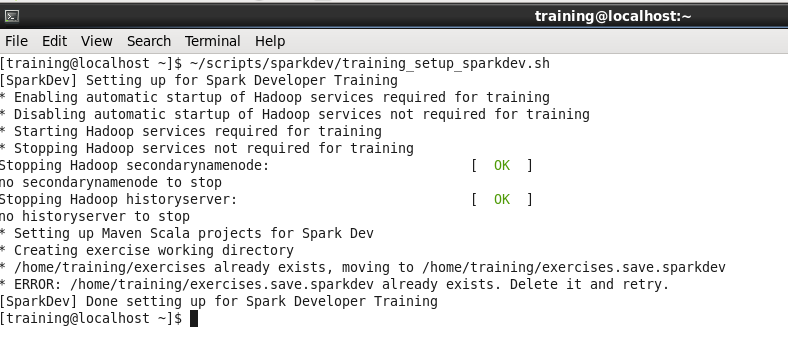
**Tempo estimado:** 20 minutos.

**Descrição da atividade:** Serão inicializados os serviços do Spark por meio de um script de setup, após isso será criado um arquivo de texto de teste que será utilizado, na sequência, no processamento com Spark para contagem de palavras desse texto. O RDD também será armazenado em cache para comparar a velocidade de processamento.

**Login da máquina virtual:** training (usuário) e training (senha).

1. Abra um terminal no Linux ( - na barra superior) e execute o script para inicializar os serviços do Spark:

~/scripts/sparkdev/training\_setup\_sparkdev.sh



1. Crie um arquivo texto de teste no Linux com o seguinte conteúdo, executando obs: copie o conteúdo depois pressione ‘i’ no terminal para poder colar o conteúdo.

gedit /home/training/teste\_spark.log

Conteúdo:

# Inicio

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

Teste Spark !!!

# Fim

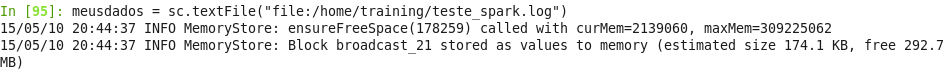
1. Abra o pyspark (Spark Shell para interação utilizando a linguagem Python):

pyspark



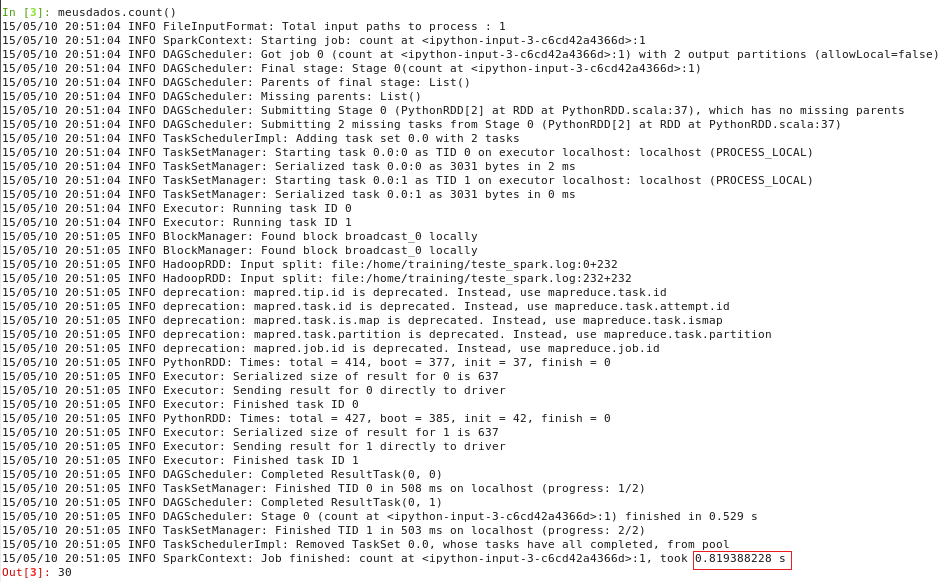
1. Crie um RDD abrindo o arquivo texto. Tome cuidado para colar todo o comando em uma única linha:

meusdados = sc.textFile("file:/home/training/teste\_spark.log")



1. Conte as linhas do RDD meusdados. E verifique o tempo de execução:

meusdados.count()



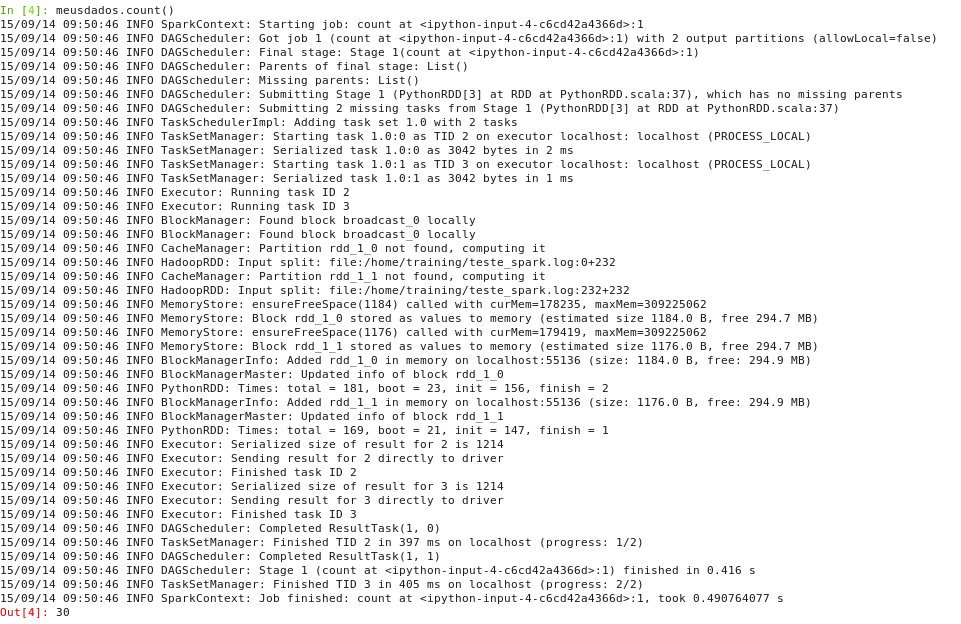
1. Armazene no cache (em memória) o RDD meusdados:

meusdados.cache()



1. Conte novamente o meusdados mais 2 vezes. Repare na mudança na velocidade do tempo de execução.

meusdados.count()

****

****

**EXERCÍCIO 2 - SPARK**

**Objetivo:** Neste exercício iremos praticar algumas operações de transformação e ação em RDDs.

**Tempo estimado:** 30 minutos.

**Descrição da atividade:** Vamos iniciar o exercício lendo logs fictícios de um web server Apache, simulando um site provedor de serviços de telefonia. Vamos realizar filtros de linhas e colunas específicas e imprimi-las na tela. Na sequência, salvaremos o RDD final em um diretório local no Linux. Também iremos acessar a interface web dos serviços Spark Master e Worker.

1. Vamos abrir um arquivo de logs do web server. Abra um segundo terminal no Linux ( - na barra superior) e leia as primeiras dez linhas do arquivo executando o comando:

head /home/training/training\_materials/sparkdev\

/data/weblogs/2014-03-15.log



1. No primeiro terminal, crie um Spark Context gerando um RDD do arquivolog:

arquivolog="file:/home/training/\

training\_materials/sparkdev/data/weblogs/\

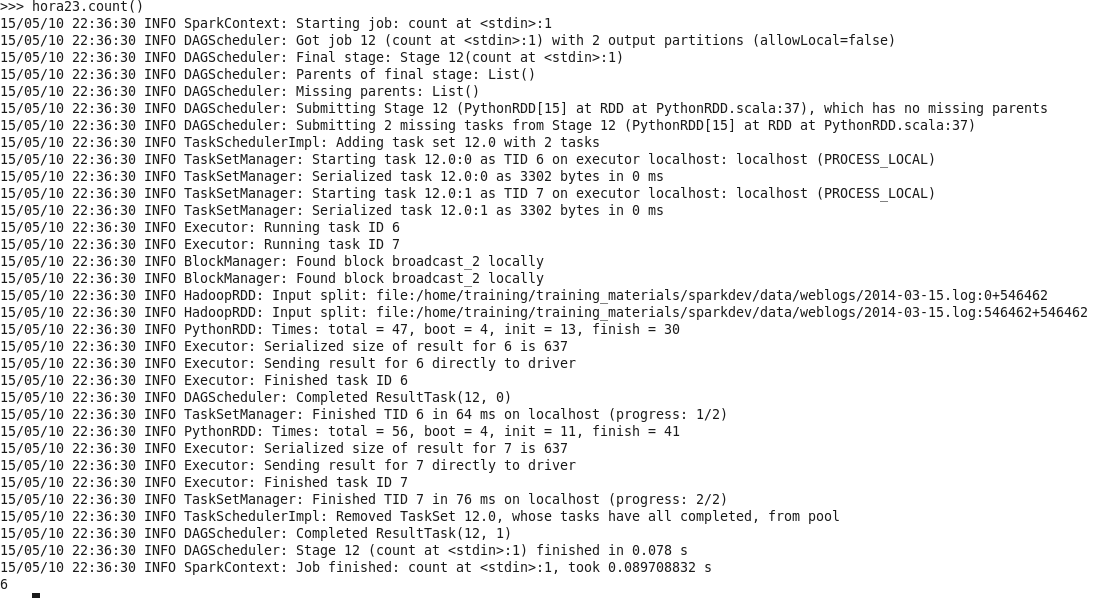
2014-03-15.log"

logs = sc.textFile(arquivolog)



1. Realize a seguinte **transformação (filter)** para criar um RDD que contenha apenas as linhas com os acessos (requests) realizados às 23:00 horas. Obs: lambda é uma função anônima no python. Execute o comando:

hora23=logs.filter(lambda x: "2014:23:00" in x)



1. Realize a seguinte **ação (take)** para obter os três primeiros registros do RDD:

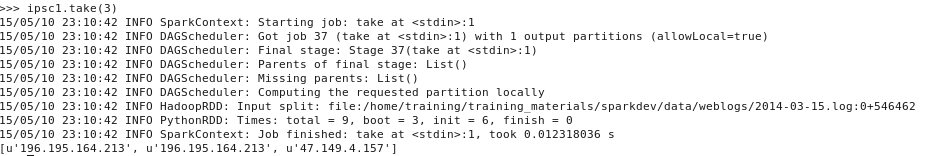
hora23.take(3)



1. Execute uma **transformação (map)** para filtrar apenas a coluna IP e na sequencia uma **ação (take)** para obter os três primeiros IPs:

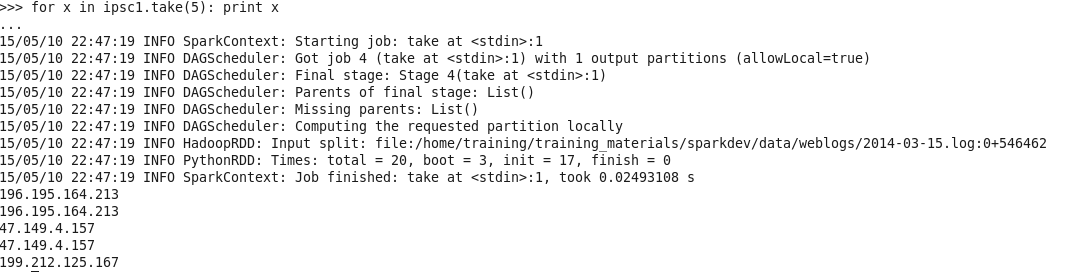
ipsc1 = logs.map(lambda s: s.split(" ")[0], )

ipsc1.take(3)



1. Realize a **ação (take)** para imprimir os cinco primeiros IPs:

for x in ipsc1.take(5): print x

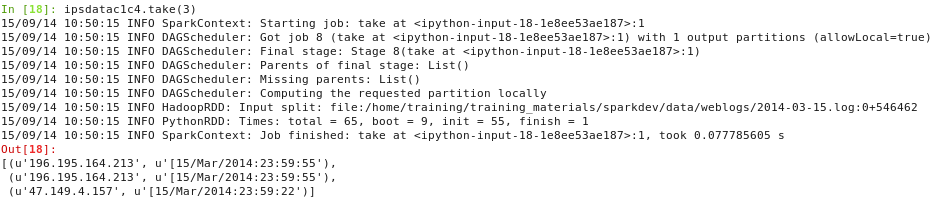


1. Execute uma **transformação (map)** para filtrar as colunas IP e data/horário e na sequencia uma **ação (take)** para obter os três primeiros registros:

ipsdatac1c4 = logs.map(lambda s: (s.split(" ")[0],s.split(" ")[3]))

Rddc1c4

ipsdatac1c4.take(3)

****

1. Salve o RDD (ipcs1) final em um arquivo texto no Linux:

ipsdatac1c4.saveAsTextFile("file:/home/training/listaip")



1. Em um novo terminal no Linux, inicie o spark master por meio do comando:

sudo service spark-master start



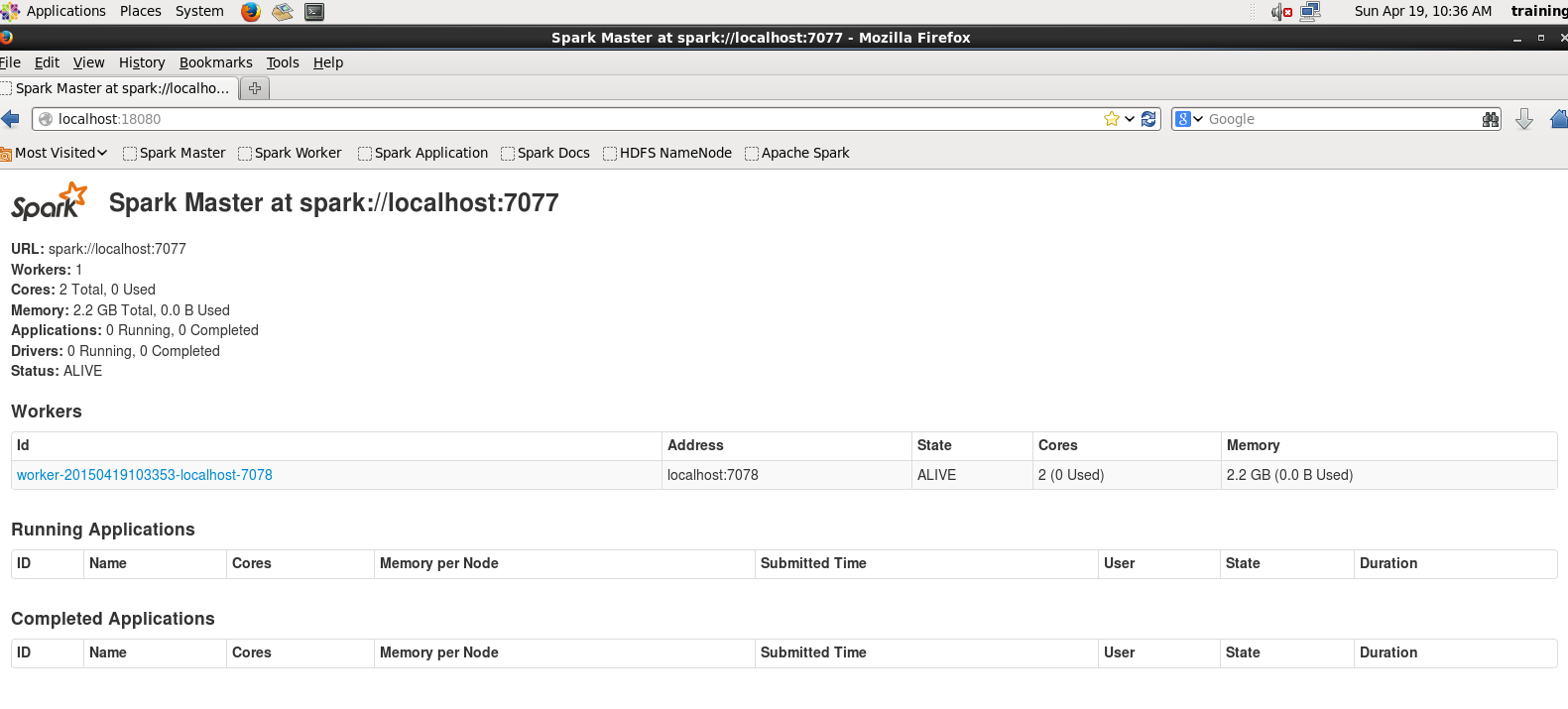
1. Reinicie o serviço do spark worker por meio do comando:

sudo service spark-worker restart



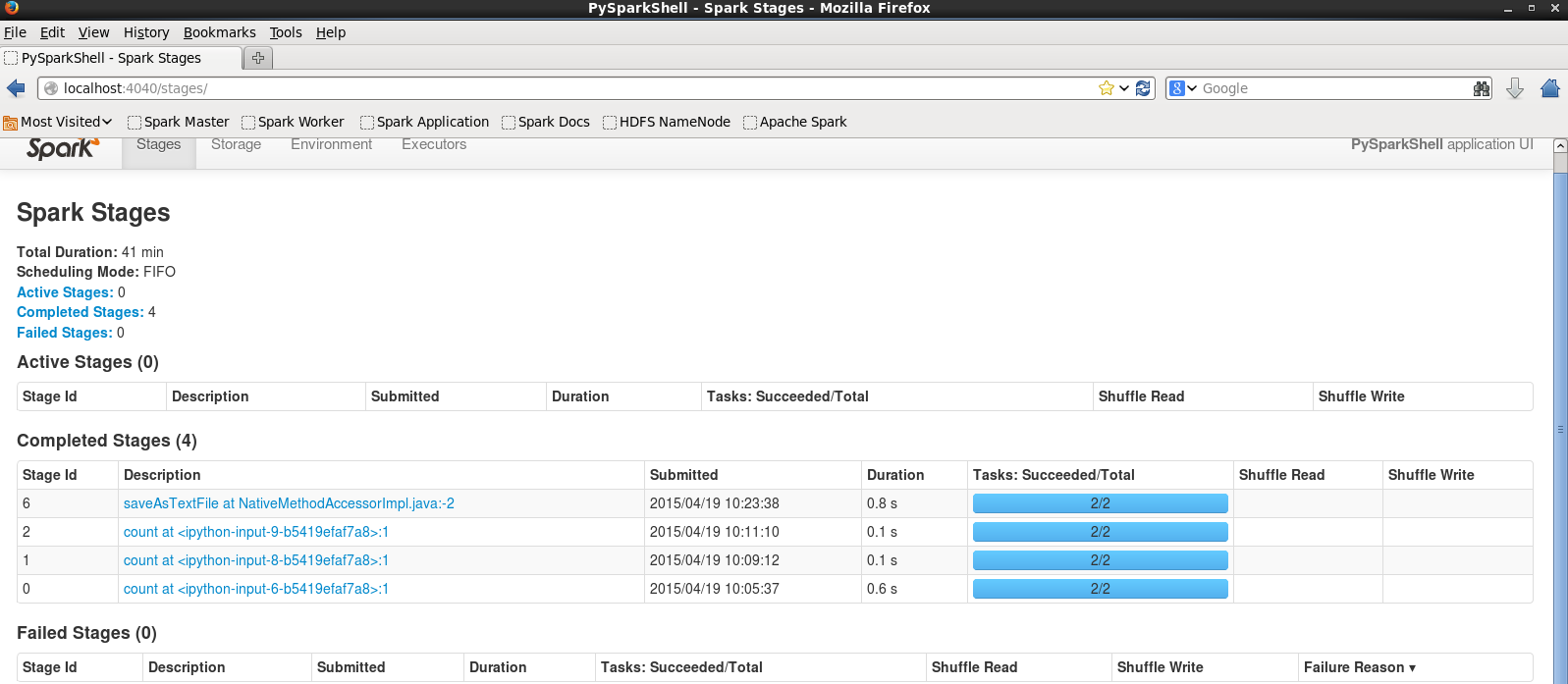
1. Verifique a interface web do Spark Master abrindo o navegador Firefox na VM:

[http://localhost:18080](http://localhost:18080/)



1. Verifique a interface web do Spark Worker abrindo o navegador Firefox na VM:

[http://localhost:4040](http://localhost:4040/)



**EXERCÍCIO 3 – SPARK**

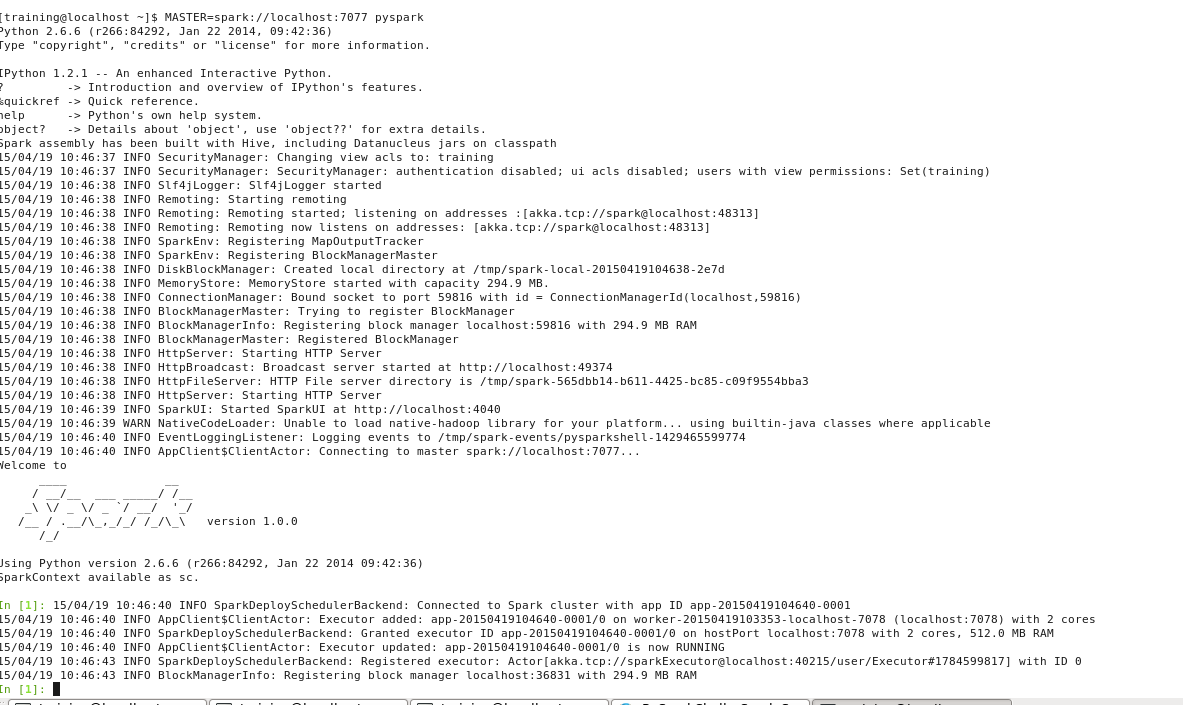
**Objetivo**: Rodando o Cluster com Spark Standalone - continuação do exercício 1.

**Tempo estimado:** 15 minutos.

**Descrição da atividade:** Neste exercício serão carregados 182 arquivos de log no HDFS e depois será realizado um processo de contagem de linhas (count) em todos esses arquivos por meio do Spark.

1. Feche o terminal antigo no pyspark e abra um novo terminal utilizando o serviço do pyspark com cluster master:

MASTER=spark://localhost:7077 pyspark



1. Confirme a sua conexão no master digitando no shell:

sc.master

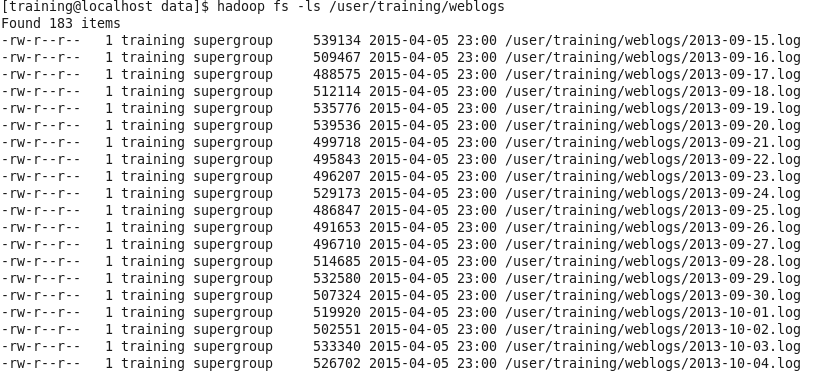


1. Abra um **novo terminal no Linux** e carregue alguns arquivos de logs do servidor web no HDFS:

cd ~/training\_materials/sparkdev/data

hdfs dfs -put weblogs /user/training/

hdfs dfs -ls /user/training/weblogs

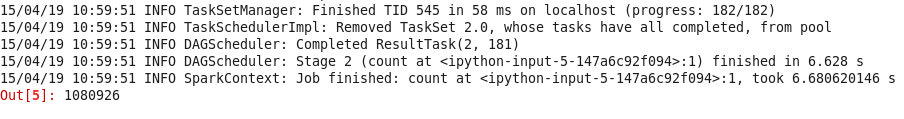


1. **No terminal com o Spark**, execute uma operação simples no cluster para contar a quantidade de registros contidos nos arquivos dos weblogs armazenados no HDFS:

sc.textFile("weblogs/\*").count()



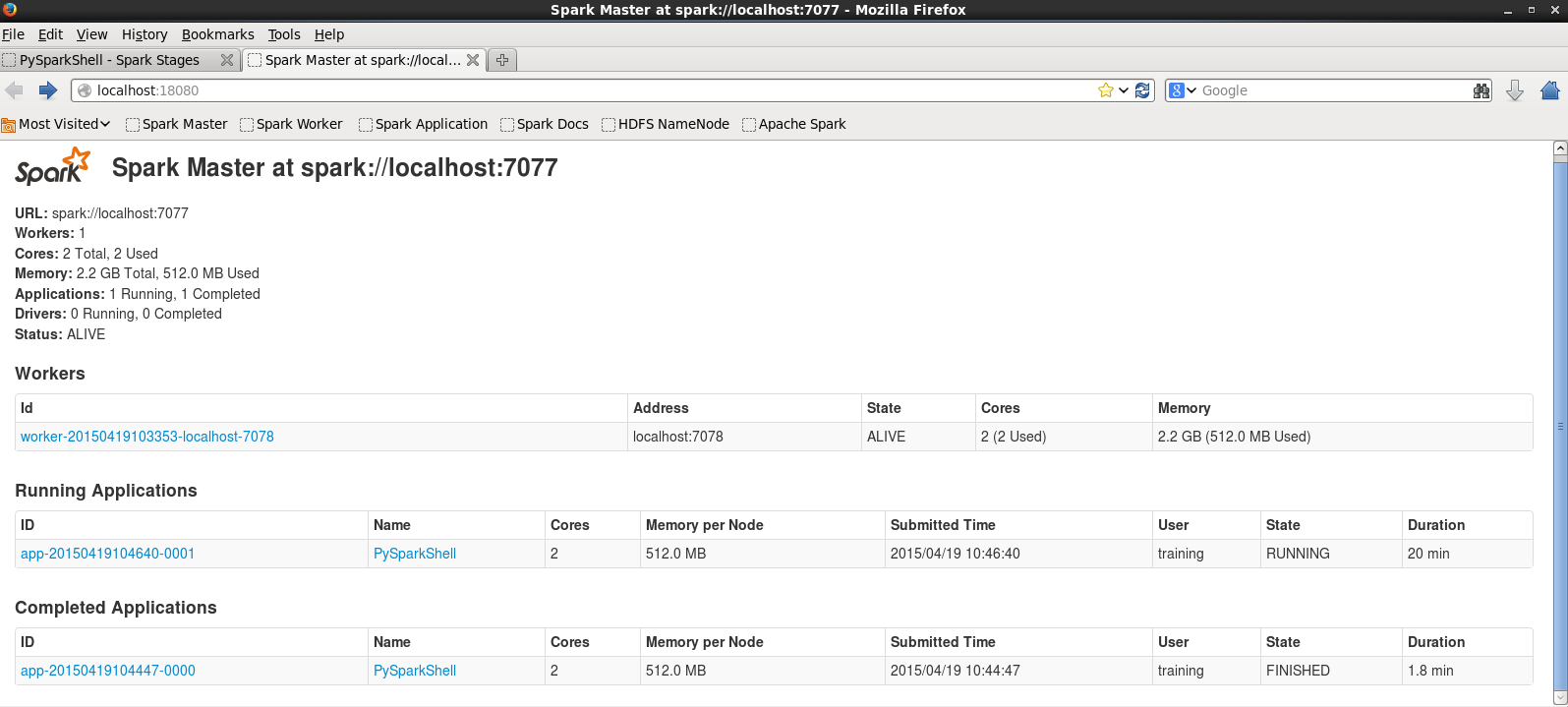
.....



1. Abra o browser Firefox e verifique a interface do Spark Standalone Master e veja como ocorreu o processamento:

<http://localhost:18080>

Clique no ID da aplicação em execução para ver mais detalhes.



**EXERCÍCIO 4 – SPARK**

**Objetivo**: Utilizar o Spark Streaming para realizar contagem de palavras das mensagens em real time.

**Tempo estimado:** 20 minutos.

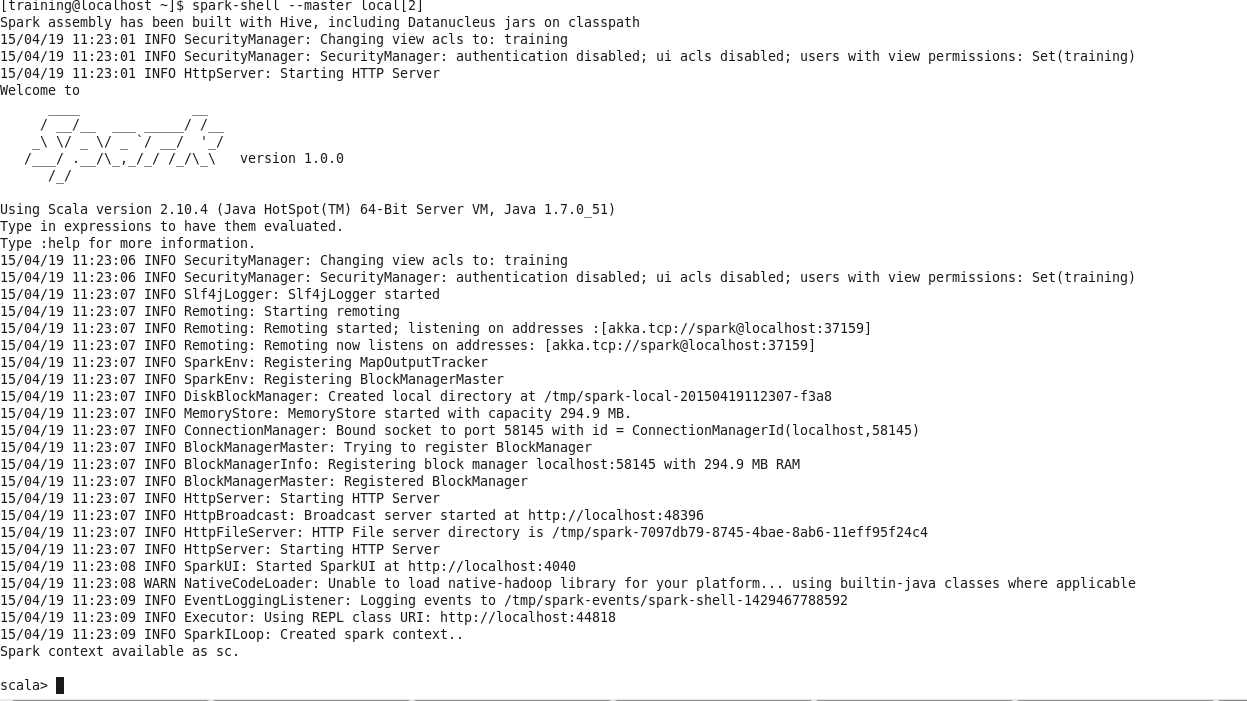
**Descrição da atividade:** Será utilizado processamento do Spark Streaming para realizar a contagem (counts) de palavras chegando na porta 4444 no servidor. O código utilizado foi desenvolvido na linguagem scala.

1. Feche todos os terminais com Spark Shell.
2. Em um novo terminal no Linux, digite o seguinte comando que abrirá a porta 4444 para enviar mensagens. Deixe-o aberto:

nc -lkv 4444

1. Em um novo terminal, inicie o Scala Shell com uma opção especial para abrir 2 threads:

spark-shell --master local[2]

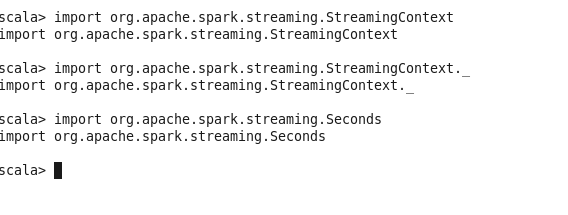


1. Importe as classes do Spark Streaming:

import org.apache.spark.streaming.StreamingContext

import org.apache.spark.streaming.StreamingContext.\_

import org.apache.spark.streaming.Seconds



1. Crie um contexto com uma janela de 10 segundos:

val streaming = new StreamingContext(sc,Seconds(10))

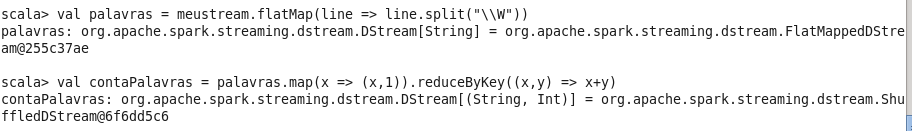
1. Crie um DStream que leia dados da porta 4444:

val meustream = streaming.socketTextStream("localhost",4444)

1. Use operações de transformação **map** escritas na linguagem scala para contar a ocorrência de palavras:

val palavras = meustream.flatMap(line => line.split("\\W"))

val contaPalavras = palavras.map(x => (x,1)).reduceByKey((x,y) => x+y)



1. Execute a **ação (print)** que imprima as 10 primeiras palavras em cada batch:

contaPalavras.print()

1. Inicie o contexto do Spark Streaming.

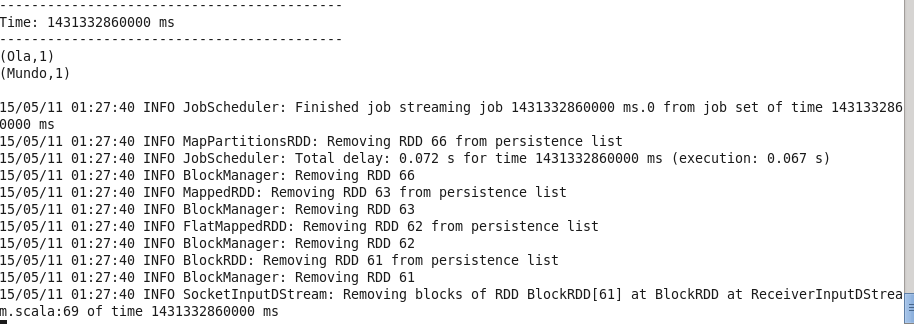
streaming.start()

1. Volte ao terminal onde você rodou o comando “nc -lkv 4444”e digite a frase:

“Olá Mundo.”



1. Volte à janela do Spark Shell (processamento Streaming) para ver o resultado.



1. Siga digitando outros textos com palavras repetidas na tela do terminal e aperte enter para ver o Spark Streaming em ação.