Enviar tarefas do Dataproc para dados não estruturados

**visão global**

Os dados não estruturados incluem dados sem um esquema e dados que possuem uma estrutura, mas que não são úteis para o objetivo pretendido.

Neste laboratório, você aprenderá sobre o Spark e a estrutura de Conjuntos de Dados Distribuídos Resilientes (RDDs) e operações para trabalhar com dados grandes e dados não estruturados.

**Objetivos**

Neste laboratório, você executará as seguintes tarefas:

* Explore o HDFS e o Cloud Storage
* Use o PySpark interativo para aprender sobre as funções RDDs, Operations e Lambda

**O que você precisará**

Para concluir este laboratório, você precisará de:

* Acesso a um navegador de internet padrão (navegador Chrome recomendado).
* Tempo. Observe o tempo de **conclusão** do laboratório no Qwiklabs. Essa é uma estimativa do tempo necessário para concluir todas as etapas. Planeje sua programação para que você tenha tempo de concluir o laboratório. Depois de iniciar o laboratório, você não poderá pausar e retornar mais tarde (começará na etapa 1 toda vez que iniciar um laboratório).
* O tempo de **acesso** do laboratório é o tempo que seus recursos de laboratório estarão disponíveis. Se você concluir seu laboratório com o tempo de acesso ainda disponível, poderá explorar o Google Cloud Platform ou trabalhar em qualquer seção do laboratório marcada "se tiver tempo". Quando o tempo de acesso acabar, seu laboratório terminará e todos os recursos serão encerrados.
* Você **NÃO** precisa de uma conta ou projeto do Google Cloud Platform. Uma conta, projeto e recursos associados são fornecidos a você como parte deste laboratório.
* Se você já tiver sua própria conta do GCP, não a use para este laboratório.
* Se o seu laboratório solicitar que você faça login no console, **use apenas a conta do aluno fornecida pelo laboratório** . Isso impede que você incorra em cobranças por atividades de laboratório em sua conta pessoal do GCP.

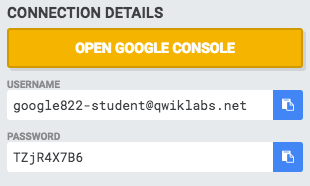
**Comece seu laboratório**

Quando estiver pronto, clique em **Iniciar Laboratório** . Você pode acompanhar o progresso do seu laboratório com a barra de status na parte superior da tela.

**Importante** O que está acontecendo durante esse tempo? Seu laboratório está gerando recursos do GCP para você nos bastidores, incluindo uma conta, um projeto, recursos dentro do projeto e permissão para você controlar os recursos necessários para executar o laboratório. Isso significa que, em vez de gastar tempo manualmente configurando um projeto e construindo recursos do zero como parte de seu laboratório, você pode começar a aprender mais rapidamente.

**Encontre o nome de usuário e a senha do GCP do seu laboratório**

Para acessar os recursos e o console deste laboratório, localize o painel Detalhes da Conexão no Qwiklabs. Aqui você encontrará o ID da conta e a senha da conta que você usará para fazer login no Google Cloud Platform:



Se o seu laboratório fornecer outros identificadores de recursos ou informações relacionadas à conexão, ele também aparecerá nesse painel.

**Tarefa 1: preparação**

Um cluster Dataproc foi preparado para você. Se você fizer login no GCP antes de a barra de progresso informar que o "Laboratório está em execução", talvez seja necessário esperar alguns minutos para que o cluster faça a transição de "Fornecimento" para "Em execução" antes que o cluster conclua a configuração.

Você estará executando a maioria das etapas de laboratório do nó mestre do cluster em uma janela do terminal SSH.

1. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Dataproc** > **Clusters** .
2. Localize o cluster denominado cluster de **dataproc** . Em qual região e zona ele está localizado? A região e a zona foram selecionadas automaticamente para você pela Qwiklabs.
3. Observe o intervalo de armazenamento temporário do Cloud Storage definido para este cluster. Esse intervalo tem o mesmo nome que o ID do projeto, que é uma maneira conveniente de tornar o nome globalmente exclusivo.
4. Clique no nome **dataproc-cluster** para ir para a página Detalhes do cluster.
5. A página de detalhes do cluster é aberta na guia "Visão geral". Clique na guia "Instâncias de VM".
6. Na linha da VM denominada **dataproc-cluster-m,** você verá que ela tem a função de mestre e que há um link SSH ao lado dela. Clique em **SSH** para abrir uma janela de terminal para o nó mestre.

**Tarefa 2. Ativar o Acesso Seguro da Web ao Cluster Dataproc**

Criar uma regra de firewall restritiva usando tags de destino, endereço IP e protocolo

Crie uma regra de firewall que permita acesso somente ao nó mestre do endereço IP do seu computador. Somente as portas 8088 (Interface de trabalho do Hadoop) e 9870 (interface do Administrador do Hadoop) serão permitidas.

Verifique se a tag de rede está definida no nó mestre

Verifique se a tag de rede "hadoopaccess" está definida no nó mestre. Isso aplicará a regra de firewall ao nó mestre, dando acesso ao seu laptop.

1. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Compute Engine** > **Instâncias de VM** .
2. Clique no nó mestre, **dataproc-cluster-m** .
3. Verifique se, em Network Tags, ele lista **hadoopaccess** .
4. Se a tag não estiver lá, clique em **EDITAR** .
5. Em Network Tags, adicione a tag: **hadoopaccess**
6. Clique em **Salvar** .

Identifique o endereço IP do navegador

Você usará o endereço IP do navegador para permitir que seu navegador local se conecte ao cluster Dataproc.

1. Encontre o endereço IP do navegador do seu computador abrindo uma janela do navegador e visualizando <http://ip4.me/> Copie o endereço IP.

Crie a regra de firewall

1. No console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Rede VPC** > **Regras de firewall** .
2. Clique em **Criar regra de firewall** .
3. Especifique o seguinte e deixe as configurações restantes como seus padrões:

|  |  |
| --- | --- |
| **Propriedade** | **Valor**  (digite o valor ou selecione a opção conforme especificado) |
| **Nome** | allow-hadoop |
| **Rede** | padrão |
| **Prioridade** | 1000 |
| **Direção de tráfego** | Entrada |
| **Ação no jogo** | Permitir |
| **Alvos** | Tags de destino especificadas |
| **Tags alvo** | hadoopaccess |
| **Intervalos de IP de origem** | <your-IP>/32 |
| **Protocolos e portas especificados** | Verifique tcp e digite o número da porta 9870,8088 |

1. Clique em **Criar** .

Levará alguns minutos para que a regra de firewall se torne ativa.

**Tarefa 3. Preparar os dados**

Copie os arquivos de amostra para o diretório inicial do nó mestre

Os arquivos de amostra necessários já foram arquivados no nó mestre. Você precisará copiá-los em seu diretório de usuários com o seguinte comando.

1. Na janela do terminal SSH do Nó Mestre.

cd

cp -r /training .

ls

1. Na janela do terminal SSH do Nó Mestre. Agora você deve ter o diretório **/ training**em seu diretório inicial. E deve ter arquivos dentro dele.

**Tarefa 4. Explorar o HDFS e o Cloud Storage**

Use o cliente do Master Node HDFS

O Dataproc exige um intervalo do Cloud Storage para organizar seus próprios arquivos durante a inicialização. Além disso, você pode usar esse intervalo para se comunicar com o cluster, por exemplo, testando programas de aplicativos PySpark ou dados no intervalo.

1. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Compute Engine** > **Instâncias de VM.**
2. Na linha do nó mestre, **dataproc-cluster-m** , clique em **SSH** .
3. Na janela do terminal **dataproc-cluster-m** , digite o seguinte:

hadoop fs -ls /

Exemplo de saída:

18/09/10 13:45:53 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.8-hadoop2

Found 3 items

drwxrwxrwt - mapred hadoop 0 2018-09-10 11:51 /hadoop

drwxrwxrwt - mapred hadoop 0 2018-09-10 11:52 /tmp

drwxrwxrwt - hdfs hadoop 0 2018-09-10 11:51 /user

1. Existem dois arquivos de texto de amostra na pasta **/ training** . O arquivo **road-not-taken.txt** é um poema de Robert Frost de apenas 24 linhas. O arquivo **sherlock-holmes.txt** contém as obras coletadas de Sherlock Holmes por Arthur Conan Doyle. ( *As primeiras 277 linhas de texto foram movidas para este arquivo para facilitar o processamento de dados:* gs: //cloud-training/gcpdei/sherlock-copyleft.txt)

cd ~/training

ls

1. Importe os arquivos de dados para o HDFS no cluster.

hadoop fs -mkdir /sampledata

hadoop fs -copyFromLocal road-not-taken.txt /sampledata/.

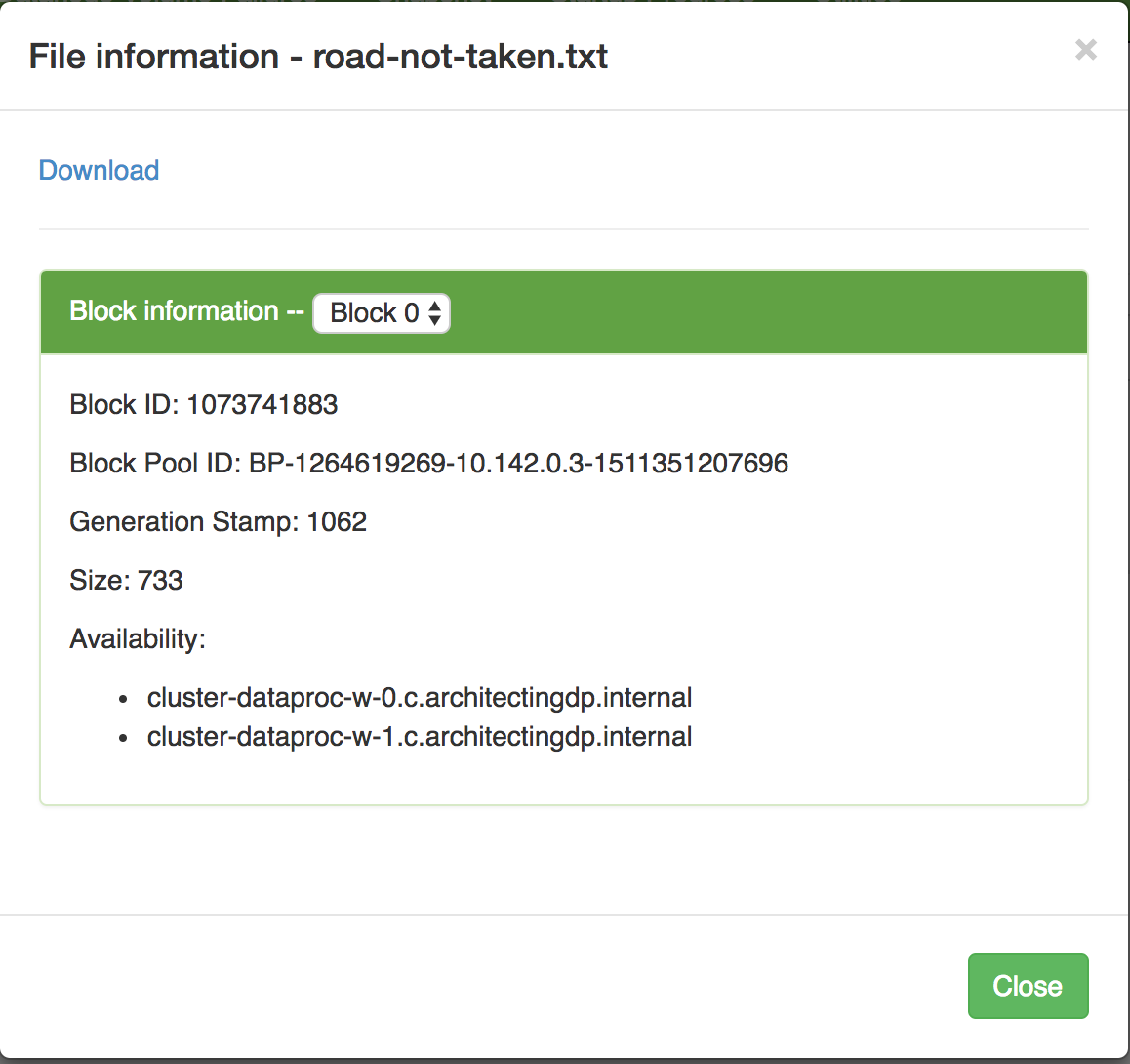
hadoop fs -copyFromLocal sherlock-holmes.txt /sampledata/.

1. Verifique se os arquivos existem no HDFS.

hadoop fs -ls /sampledata

1. Retorne ao navegador para a Interface de administração do Hadoop ou abra uma nova janela e navegue até **<External\_IP>: 9870** .
2. Em **Utilitários** , selecione **Navegar no sistema de arquivos** .
3. Na coluna **Nome** , clique em **sampledata** (se você não vir sampledata, atualize a página).
4. Na coluna **Nome** , clique em **road-not-taken.txt** .

Uma janela pop-up semelhante à seguinte deve ser exibida.



Isso mostra que o arquivo se encaixa em um único bloco HDFS. Observe no menu suspenso **Informações** do **Bloco** , que o arquivo está localizado apenas no **Bloco 0** . E que o bloco é duplicado no nó do trabalhador 0 e no nó do trabalhador 1.

1. Clique em **Fechar** .
2. Feche a janela ou a guia Interface de Administração do Hadoop.

**Tarefa 5. Use o PySpark interativo para aprender sobre as funções RDDs e Lambda**

PySpark interativo

A versão interativa do PySpark fornece um excelente ambiente para aprender a semântica do Spark e como trabalhar com RDDs, transformações e ações. Nesta seção, você estará trabalhando com a cópia dos arquivos de amostra armazenados no cluster no HDFS.

O PySpark é um interpretador Read-Evaluate-Print-Loop (REPL). Também conhecido como shell de linguagem. O REPL lê uma única expressão do usuário, avalia e imprime o resultado. Em seguida, ele faz um loop e executa outra expressão única. O feedback imediato de etapa única de um REPL o torna útil para explorar e aprender um idioma ou sistema. A limitação é que o REPL não possui um contexto de estado próprio, ao contrário de shells mais sofisticados.

<https://en.wikipedia.org/wiki/Read%E2%80%93eval%E2%80%93print_loop>

1. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Compute Engine** > **Instâncias de VM.**
2. Na linha do nó mestre, **dataproc-cluster-m** , clique em **SSH** .
3. Na janela do terminal **dataproc-cluster-m** .
4. Inicie a sessão interativa digitando o seguinte:

pyspark

Levará alguns momentos para inicializar. Quando estiver pronto, você verá algo assim:

Python 2.7.9 (default, Jun 29 2016, 13:08:31)

[GCC 4.9.2] on linux2

Type "help", "copyright", "credits" or "license" for more information.

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

ivysettings.xml file not found in HIVE\_HOME or HIVE\_CONF\_DIR,/etc/hive/conf.dist/ivysettings.xml will be used

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0

/\_/

Using Python version 2.7.9 (default, Jun 29 2016 13:08:31)

SparkSession available as 'spark'.

1. Crie um RDD do Spark (Conjunto de Dados Distribuído Resiliente) lendo o arquivo de texto do HDFS. Use a função type () do python para identificar o tipo de objeto. E use um método interno para o objeto contar o número de linhas.

Um conjunto de dados distribuído resiliente (RDD) é uma abstração sobre dados no armazenamento. O RDD é opaco para o local e a replicação de dados que ele contém. Para confiabilidade, os RDDs são resilientes (tolerantes a falhas) à perda de dados devido a falhas de nó. Um gráfico de linhagem RDD é usado para recalcular partições danificadas ou ausentes. E, para eficiência, o Spark pode optar por processar uma parte em um local ou outro, com base na disponibilidade da CPU nesse local ou com base na latência da rede ou na proximidade de outros recursos.

O benefício dessa abstração é que ela permite que as operações em um RDD tratem o RDD como um único objeto e ignorem a complexidade de como os dados são localizados, replicados e migrados dentro do RDD. Todos esses detalhes são deixados para o Spark.

Entrar:

lines = sc.textFile("/sampledata/sherlock-holmes.txt")

type(lines)

lines.count()

lines.take(15)

Exemplo:

>>> lines = sc.textFile("/sampledata/sherlock-holmes.txt")

>>> type(lines)

<class 'pyspark.rdd.RDD'>

>>> lines.count()

12652

>>> lines.take(15)

[u'', u'THE ADVENTURES OF SHERLOCK HOLMES by ARTHUR CONAN DOYLE', u'', u'', u'', u'', u'A Scandal in Bohemia', u'Th

e Red-headed League', u'A Case of Identity', u'The Boscombe Valley Mystery', u'The Five Orange Pips', u'The Man wit

h the Twisted Lip', u'The Adventure of the Blue Carbuncle', u'The Adventure of the Speckled Band', u"The Adventure

of the Engineer's Thumb"]

>>>

1. Use uma transformação de centelha flatMap () para dividir as sentenças em palavras.

Entrar:

words = lines.flatMap(lambda x: x.split(' '))

type(words)

words.count()

words.take(15)

Exemplo:

>>> words = lines.flatMap(lambda x: x.split(' '))

>>> type(words)

<class 'pyspark.rdd.PipelinedRDD'>

>>> words.count()

107265

>>> words.take(15)

[u'', u'THE', u'ADVENTURES', u'OF', u'SHERLOCK', u'HOLMES', u'by', u'ARTHUR', u'CONAN', u'DOYLE', u'', u'', u'', u'

', u'A']

>>>

1. Use uma transformação de mapa de centelha () para criar pares. O primeiro elemento do par é a palavra. O segundo elemento do par é o número de caracteres (comprimento) da palavra.

Entrar:

pairs = words.map(lambda x: (x,len(x)))

type(pairs)

pairs.count()

pairs.take(5)

Exemplo:

>>> pairs = words.map(lambda x: (x,len(x)))

>>> type(pairs)

<class 'pyspark.rdd.PipelinedRDD'>

>>> pairs.count()

107265

>>> pairs.take(5)

[(u'', 0), (u'THE', 3), (u'ADVENTURES', 10), (u'OF', 2), (u'SHERLOCK', 8)]

>>>

1. O objetivo é contar o número de palavras de vários comprimentos - quantas palavras com 10 caracteres são usadas nos livros. Modifique o mapa () para que ele crie o comprimento da palavra e '1' para cada instância. Este é um método comum "orientado por colunas" para contar instâncias de maneira paralelizável.

Entrar:

pairs = words.map(lambda x: (len(x),1))

pairs.take(5)

Exemplo:

>>> pairs = words.map(lambda x: (len(x),1))

>>> pairs.take(5)

[(0, 1), (3, 1), (10, 1), (2, 1), (8, 1)]

>>>

1. Use um método parallelizable (add) para acumular as instâncias. A função add será chamada dentro da transformação Spark reduceByKey ().

Entrar:

from operator import add

wordsize = pairs.reduceByKey(add)

type(wordsize)

wordsize.count()

wordsize.take(5)

Exemplo:

>>> from operator import add

>>> wordsize = pairs.reduceByKey(add)

>>> type(wordsize)

<class 'pyspark.rdd.PipelinedRDD'>

>>> wordsize.count()

22

>>> wordsize.take(5)

[(0, 2756), (2, 18052), (4, 19456), (6, 8622), (8, 4664)]

>>>

1. Converta o RDD em um objeto Python para facilitar a saída.

Entrar:

output = wordsize.collect()

type(output)

for (size,count) in output: print(size, count)

Exemplo:

>>> output = wordsize.collect()

>>> type(output)

<type 'list'>

>>> for (size,count) in output: print(size, count)

...

(0, 2756)

(2, 18052)

(4, 19456)

(6, 8622)

(8, 4664)

(10, 1730)

(12, 585)

(14, 159)

(16, 31)

(18, 8)

(20, 4)

(1, 5141)

(3, 22939)

(5, 12044)

(7, 6615)

(9, 2980)

(11, 1035)

(13, 352)

(15, 75)

(17, 12)

(19, 4)

(21, 1)

1. O que aconteceu? Por que a lista está fora de ordem? É porque a ação collect () foi paralelizada e os resultados foram reunidos. Use o método sortByKey () do Spark para classificar os pares antes de coletá-los em uma lista.

Entrar:

output = wordsize.sortByKey().collect()

for (size,count) in output: print(size, count)

Exemplo:

>>> output = wordsize.sortByKey().collect()

>>> for (size, count) in output: print(size, count)

...

(0, 2756)

(1, 5141)

(2, 18052)

(3, 22939)

(4, 19456)

(5, 12044)

(6, 8622)

(7, 6615)

(8, 4664)

(9, 2980)

(10, 1730)

(11, 1035)

(12, 585)

(13, 352)

(14, 159)

(15, 75)

(16, 31)

(17, 12)

(18, 8)

(19, 4)

(20, 4)

(21, 1)

>>>

1. O programa a seguir recria exatamente o que você fez nas etapas anteriores. Em cada etapa, você executou uma transformação Spark em um RDD, que retorna os resultados em um RDD separado.

O Spark não executa operações imediatamente. Ele usa uma abordagem chamada "avaliação lenta".

Existem duas classes de operações: transformações e ações. Uma transformação recebe um RDD como entrada e gera outro RDD como saída. Você pode pensar em uma transformação como uma solicitação. Explica ao Spark o que você quer que seja feito, mas não informa ao Spark como fazer isso.

Ações como "coletar", "contar" ou "tirar" produzem um resultado, como um número ou uma lista.

Quando o Spark recebe transformações, ele as armazena em um DAG (Directed Acyclic Graph), mas não as executa nesse momento. Quando o Spark recebe uma ação, ele examina as transformações no DAG e os recursos disponíveis (número de trabalhadores no cluster) e cria pipelines para executar o trabalho com eficiência.

1. Nas etapas anteriores, você emitiu uma ação após cada transformação para ver os resultados intermediários. No programa abaixo, a única ação é "coletar" no final.

from operator import add

lines = sc.textFile("/sampledata/sherlock-holmes.txt")

words = lines.flatMap(lambda x: x.split(' '))

pairs = words.map(lambda x: (len(x),1))

wordsize = pairs.reduceByKey(add)

output = wordsize.sortByKey().collect()

1. Para permitir que o Spark realize sua mágica, o programa precisa construir uma cadeia de transformações usando o operador dot. Quando é passado para o Spark dessa maneira, o Spark compreende as várias etapas e os resultados de uma transformação devem ser passados ​​para a próxima transformação. Isso permite que o Spark organize o processamento da maneira que decidir com base nos recursos disponíveis no cluster. Assim, enquanto o programa acima é mais legível, com "words", "pairs" e "wordsize" chamados, o programa a seguir faz a mesma coisa sem nomear os resultados intermediários. Eles são funcionalmente idênticos.

output2 = lines.flatMap(lambda x: x.split(' ')).map(lambda x: (len(x),1)).reduceByKey(add).sortByKey().collect()

for (size, count) in output2: print(size, count)

1. Digite **exit ()** para sair do PySpark. Deixe o terminal SSH do Nó Principal aberto. Você vai usá-lo na próxima tarefa.

**Tarefa 6: Envie um trabalho do PySpark a partir do nó mestre**

1. No terminal SSH do nó mestre, use **nano** ou **vi** para criar o arquivo **wordcount.py** .
2. Copie e cole o seguinte código no arquivo.

from pyspark.sql import SparkSession

from operator import add

import re

print("Okay Google.")

spark = SparkSession\

.builder\

.appName("CountUniqueWords")\

.getOrCreate()

lines = spark.read.text("/sampledata/road-not-taken.txt").rdd.map(lambda x: x[0])

counts = lines.flatMap(lambda x: x.split(' ')) \

.filter(lambda x: re.sub('[^a-zA-Z]+', '', x)) \

.filter(lambda x: len(x)>1 ) \

.map(lambda x: x.upper()) \

.map(lambda x: (x, 1)) \

.reduceByKey(add) \

.sortByKey()

output = counts.collect()

for (word, count) in output:

print("%s = %i" % (word, count))

spark.stop()

1. Execute o trabalho.

spark-submit wordcount.py

Exemplo de saída

Okay Google.

"'ABOUT = 1

"'ABSOLUTE = 1

"'AH!' = 2

"'AH, = 2

"'AMPLE.' = 1

"'AND = 10

"'ARTHUR!' = 1

"'AS = 1

"'AT = 1

"'BECAUSE = 1

"'BRECKINRIDGE, = 1

Visualizar o trabalho na janela do JobTracker do Hadoop

1. Enquanto o aplicativo estiver em execução, retorne à janela do Hadoop JobTracker. Examine as seguintes páginas na interface do Hadoop Applications em <External IP>: 8088. Você pode encontrar o nó mestre IP externo no console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Compute Engine** > **VM Instances** .
2. Explore esses links na interface. Eles mapeiam para o ciclo de vida do trabalho. ENVIADO, ACEITO, EXECUTANDO, TERMINADO, FALHOU, MATOU

Procure o trabalho na página Tarefas do Dataproc

1. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Dataproc** > **Trabalhos** . Por que não há empregos listados?

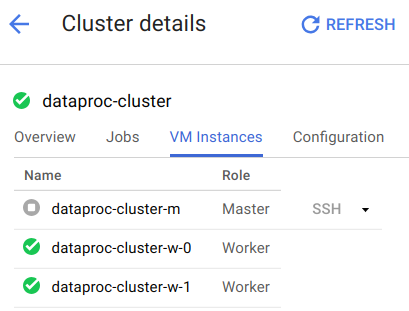
Os aplicativos do Datalab que você executou anteriormente não são exibidos na página Dataproc> Tarefas no Console. Somente aplicativos enviados do Console são rastreados no Console.

**Tarefa 7: tempo limite do nó principal**

O nó mestre é configurado com um recurso de tempo limite automático. Se nenhum trabalho for submetido por um longo período de tempo, o Nó Mestre irá PARAR para reduzir os custos. Os outros nós continuam operando conforme necessário para manter o HDFS. Você pode verificar esse estado na página Detalhes do cluster. Para restaurar a capacidade de enviar trabalhos, você precisará reiniciar o nó mestre.

1. Verifique se o nó mestre expirou.
2. No Console, no **menu Navegação** ( 7a91d354499ac9f1.png), clique em **Dataproc** > **Clusters** .
3. Clique no seu cluster, **cluster-dataproc** . Em seguida, clique na guia **Instâncias** da **VM** .

Exemplo:



1. Se o Nó Principal tiver expirado, clique em seu nome, **dataproc-cluster-m** .
2. Na parte superior da página de **detalhes da instância** da **VM** , clique em **START** .

**Termine seu laboratório**

Quando você tiver concluído seu laboratório, clique em **Finalizar Laboratório** . O Qwiklabs remove os recursos que você usou e limpa a conta para você.

Você terá a oportunidade de avaliar a experiência do laboratório. Selecione o número de estrelas aplicável, digite um comentário e clique em **Enviar** .

O número de estrelas indica o seguinte:

* 1 estrela = muito insatisfeito
* 2 estrelas = insatisfeito
* 3 estrelas = neutra
* 4 estrelas = Satisfeito
* 5 estrelas = muito satisfeito

Você pode fechar a caixa de diálogo se não quiser fornecer feedback.

Para comentários, sugestões ou correções, use a guia **Suporte** .

Última atualização: 2018-09-19

© 2018 Google LLC Todos os direitos reservados. Google e o logotipo do Google são marcas registradas do Google LLC. Todos os outros nomes de empresas e produtos podem ser marcas registradas das respectivas empresas com as quais estão associados.