

# Documentação

## Criação do Data Lake na Amazon AWS:

- Arquitetura:
  - 1 namenode de 8 GB de RAM.  
Máquina: m4.large, 2 CPU's, 8 GB de RAM
  - 2 datanodes de 4 GB de RAM cada.  
Máquinas: t2.medium, 2 CPU's, 4 GB de RAM

Sistema Operacional em todas as máquinas: Amazon Linux 2.

Atentar para a criação da Key e seu download posterior.

Após as criações das máquinas a conexão deve ser feita via Putty.

- 1) Usar o Putty Gen para converter a chave.
- 2) Clicar em load (marcar todos os arquivos) e selecionar a key que foi feita o download.
- 3) Salvar private key (.ppk)
- 4) Se conectar a instância via SSH
  - 4.1 SSH -> Auth -> Colocar a chave gerada no gen.
  - 4.2 Voltar em Session e colocar o DnS público da instância.
- 5) O usuário inicialmente será o ec2-user.

## Criação do usuário hadoop:

OBS: a criação deverá ser feita nas 3 máquinas.

- Criando o usuário:  
`sudo useradd -m hadoop`
- Gerando uma senha o usuário:  
`sudo passwd hadoop`
- Dar permissão de sudo para o usuário hadoop:

```
sudo vi /etc/sudoers
```

Editar o arquivo:

```
hadoop    ALL=(ALL)  ALL
```

Fazer o reboot:

```
Sudo reboot
```

## Instalação o Java JDK:

OBS1: a criação deverá ser feita nas 3 máquinas.

OBS2: Essa instalação deve ser feita como usuário hadoop.

1) cd /opt/

2) instalar o wget:

```
sudo yum install wget
```

3) Fazer o download:

```
sudo wget http://datascienceacademy.com.br/blog/aluno/JDK/jdk-8u181-linux-x64.tar.gz
```

4) Descompactar o arquivo:

```
sudo tar -xvf jdk-8u181-linux-x64.tar.gz
```

5) Renomear o arquivo JDK:

```
sudo mv jdk1.8.0_171/ jdk
```

6) Remover o arquivo:

```
sudo rm -rf jdk-8u181-linux-x64.tar.gz
```

7) Ajustar os privilégios:

```
sudo chown -R root:root jdk
```

8) Configurar as variáveis de ambiente:

8.1 cd ~

```
vi .bash_profile
```

8.2 # Java JDK 1.8

```
export JAVA_HOME=/opt/jdk
```

```
export JRE_HOME=/opt/jdk/jre
```

```
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

8.3 source .bash\_profile

#### 8.4 Verificar a versão do Java:

Java -version

### **Estabelecer a conexão SSH sem senha**

OBS1: a criação deverá ser feita nas 3 máquinas.

OBS2: Essa instalação deve ser feita como usuário ec2-user.

- 1) Criar o diretório .ssh na pasta hadoop:

```
sudo mkdir /home/hadoop/.ssh
```

- 2) Mudar o proprietário do diretório criado:

```
sudo chown -R hadoop:hadoop /home/hadoop/
```

- 3) Copiar as authorized-keys para a pasta hadoop:

```
sudo cp ~/.ssh/authorized_keys /home/hadoop/.ssh/authorized_keys
```

- 4) Alterar o proprietário:

```
sudo chown hadoop:hadoop /home/hadoop/.ssh/authorized_keys
```

- 5) Alterar permissão:

```
sudo chmod 600 /home/hadoop/.ssh/authorized_keys
```

### **Configurando o SSH**

OBS: a criação deverá ser feita nas 3 máquinas.

- 1) Configurar o arquivo sshd-config:

```
sudo vi /etc/ssh/sshd_config
```

As seguintes configurações devem estar ativas:

```
Port 22
#AddressFamily any
ListenAddress 0.0.0.0
ListenAddress ::
PubkeyAuthentication yes
```

2) Reiniciar o SSH:

```
sudo systemctl restart sshd.service
```

3) **Apenas no namenode e como usuário hadoop:**

Gerar as chaves públicas e privadas em .ssh

```
cd ~
ssh-keygen
```

4) Copiar a chave:

**Em cada node slave, copiar a chave id\_rsa.pub do servidor node master**

Usar sudo vi id\_rsa.pub e colar a chave

Basta usar cat na chave pública e copiar para os slaves (diretório .ssh)

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Portanto em cada node teremos a configuração da figura abaixo:

```
hadoop@ip-172-31-8-58:~/.ssh
```

```
[hadoop@ip-172-31-8-58 ~/.ssh]$ ls -la
total 8
drwxr-xr-x 2 hadoop hadoop  47 Mar  7 10:37 .
drwx----- 4 hadoop hadoop 136 Mar  7 10:37 ..
-rw----- 1 hadoop hadoop 818 Feb 20 18:45 authorized_keys
-rw-rw-r-- 1 hadoop hadoop 430 Feb 20 18:42 id_rsa.pub
[hadoop@ip-172-31-8-58 ~/.ssh]$
```

5) Testando a conexão:

Fazer nas 3 máquinas.

ssh hadoop@<dns privado>

Conexões autorizadas sem senha.

## **Instalação e configuração do Hadoop no nodemaster**

OBS1: o Hadoop deverá ser instalado no nodemaster e depois copiado para os datanodes.

OBS2: Como usuário hadoop

1) Cd /opt/

2) Fazer o download do Hadoop:

```
sudo wget https://archive.apache.org/dist/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
```

3) Descompactar:

```
sudo tar -xvf hadoop-3.3.1.tar.gz
```

4) Renomear e mudar o proprietário:

```
sudo mv /opt/hadoop-3.1.0 /opt/hadoop
```

```
sudo chown -R hadoop:hadoop /opt/hadoop
```

5) Remover o arquivo:

```
sudo rm -rf hadoop-3.3.1.tar.gz
```

6) Testar a instalação:

```
cd /opt/hadoop/bin
```

```
./hadoop version
```

7) Configurando as variáveis de ambiente para o usuário hadoop:

```
# Hadoop
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

8) Fazendo o source:

```
source .bash_profile
```

Portanto teremos a seguinte configuração no arquivo .ssh\_profile:

```
hadoop@ip-172-31-0-29:~
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export PATH

# Java JDK 1.8
export JAVA_HOME=/opt/jdk
export JRE_HOME=/opt/jdk/jre
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin


# Hadoop
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

# Instalando e Configurando o Hadoop no NameNode

- 1) Editar o arquivo `$HADOOP_HOME/etc/hadoop/hadoop-env.sh` e adicionar as linhas:

```
export JAVA_HOME=/opt/jdk
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR="${HADOOP_HOME}/etc/hadoop"
export PATH="${PATH}:${HADOOP_HOME}/bin"
```

O arquivo `hadoop-env.sh` ficará assim:

 `hadoop@ip-172-31-0-29:/etc`

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Set Hadoop-specific environment variables here.
export JAVA_HOME=/opt/jdk
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR="${HADOOP_HOME}/etc/hadoop"
export PATH="${PATH}:${HADOOP_HOME}/bin"
#
##
## THIS FILE ACTS AS THE MASTER FILE FOR ALL HADOOP PROJECTS.
## SETTINGS HERE WILL BE READ BY ALL HADOOP COMMANDS. THEREFORE,
## ONE CAN USE THIS FILE TO SET YARN, HDFS, AND MAPREDUCE
## CONFIGURATION OPTIONS INSTEAD OF xxx-env.sh.
##
## Precedence rules:
##
## {yarn-env.sh|hdfs-env.sh} > hadoop-env.sh > hard-coded defaults
```

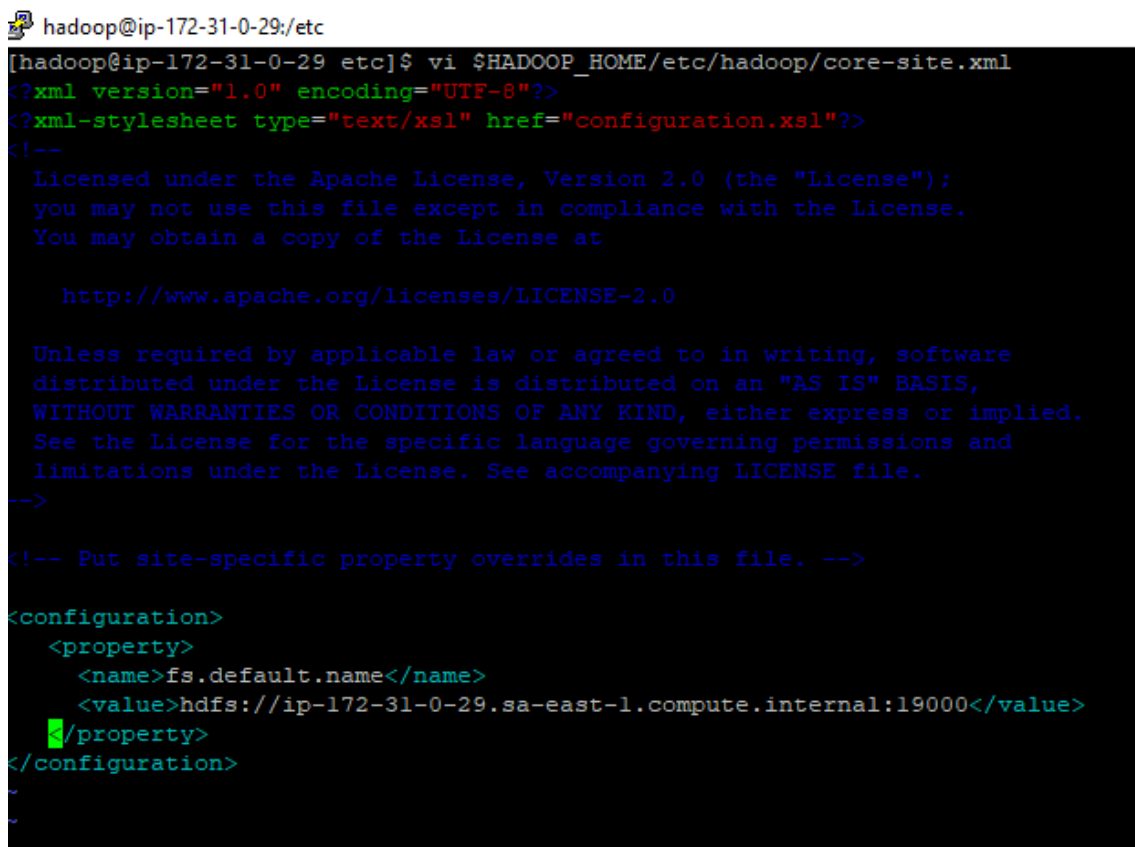
- 2) Editar o arquivo `$HADOOP_HOME/etc/hadoop/core-site.xml` e adicionar as linhas:

Vale a pena salientar que o DnS privado do masternode deverá ser inserido aqui.

vi \$HADOOP\_HOME/etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://ip-172-31-0-29.sa-east-1.compute.internal:19000</value>
  </property>
</configuration>
```

Logo o arquivo core-site.xml ficará assim:



```
hadoop@ip-172-31-0-29:/etc
[hadoop@ip-172-31-0-29 etc]$ vi $HADOOP_HOME/etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://ip-172-31-0-29.sa-east-1.compute.internal:19000</value>
  </property>
</configuration>
```

3) Agora é preciso criar os diretórios abaixo:

```
mkdir /opt/hadoop/dfs
mkdir /opt/hadoop/dfs/data
mkdir /opt/hadoop/dfs/namespace_logs
```

4) Precisamos agora editar o arquivo hdfs-site.xml



```
vi $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/opt/hadoop/dfs/namespace_logs</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/opt/hadoop/dfs/data</value>
  </property>
</configuration>
```

Logo teremos:

```
hadoop@ip-172-31-0-29:/etc
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/opt/hadoop/dfs/namespace_logs</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/opt/hadoop/dfs/data</value>
  </property>
</configuration>
```

- 5) Neste ponto precisamos editar o arquivo workers e inserir os DnS privados dos dois datanodes.

vi \$HADOOP\_HOME/etc/hadoop/workers

Portanto ficamos com

```
hadoop@ip-172-31-0-29:/etc
ip-172-31-8-58.sa-east-1.compute.internal
ip-172-31-13-45.sa-east-1.compute.interna
```

6) Continuando precisamos editar o arquivo mapred-site.xml e neste arquivo será inserido o DnS privado do namenode.

vi \$HADOOP\_HOME/etc/hadoop/mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.job.user.name</name>
    <value>hadoop</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value>ip-172-31-0-29.sa-east-1.compute.internal:8032</value>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
```

```

<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
</property>

<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
</property>

</configuration>

```

Logo o arquivo da seguinte maneira:

```

<configuration>
  <property>
    <name>mapreduce.job.user.name</name>
    <value>hadoop</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value>ip-172-31-0-29.sa-east-1.compute.internal:8032</value>
  </property>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>

  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>

  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>

</configuration>

```

## Instalação do Hadoop nos slaves

- 1) Em cada slave executar os comandos abaixo.

```
sudo mkdir /opt/hadoop
cd /opt
sudo chown -R hadoop:hadoop /opt/hadoop/
```

OBS: Apesar de parecer o começo do mesmo procedimento adotado no namenode não iremos repetir todo o processo. Na verdade, vamos apenas copiar os arquivos de instalação para cada slave.

- 2) No namenode e como usuário hadoop.

```
cd ~
scp -rv /opt/hadoop hadoop@ip-172-31-8-58.sa-east-1.compute.internal:/opt
scp -rv /opt/hadoop hadoop@ip-172-31-13-45.sa-east-1.compute.internal:/opt
```

- 3) Formatar o namenode

```
hdfs namenode -format
```

- 4) Inicializar o HDFS

```
$HADOOP_HOME/sbin/start-dfs.sh
```

- 5) Inicializar o YARN

```
$HADOOP_HOME/sbin/start-yarn.sh
```

Podemos agora acessar o cluster via browser:

<http://ec2-18-230-145-251.sa-east-1.compute.amazonaws.com:9870/>

Se a conexão não ocorrer pode ser o firewall. Neste vamos verificar o status do firewall:

```
service firewalld status
```

Então paramos o serviço:

```
service firewalld stop
```

## Overview 'ip-172-31-0-29.sa-east-1.compute.internal:19000' (✓active)

Started:	Tue Mar 08 07:13:31 -0300 2022
Version:	3.3.1, ra3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled:	Tue Jun 15 02:13:00 -0300 2021 by ubuntu from (HEAD detached at release-3.3.1-RC3)
Cluster ID:	CID-a746073d-70fd-4f62-9691-02b1a297cc34
Block Pool ID:	BP-846668435-172.31.0.29-1645523541088

## Summary

Security is off.  
Safemode is off

6) Por fim vamos fazer uma análise do cluster:

```
hdfs dfsadmin -report
```

E temos como saída:

Configured Capacity: 17154662400 (15.98 GB)

Present Capacity: 10387283968 (9.67 GB)

DFS Remaining: 10283700224 (9.58 GB)

DFS Used: 103583744 (98.79 MB)

DFS Used%: 1.00%

Replicated Blocks:

Under replicated blocks: 11

Blocks with corrupt replicas: 0

Missing blocks: 0

Missing blocks (with replication factor 1): 0

Low redundancy blocks with highest priority to recover: 0

Pending deletion blocks: 0

Erasure Coded Block Groups:

Low redundancy block groups: 0

Block groups with corrupt internal blocks: 0

Missing block groups: 0

Low redundancy blocks with highest priority to recover: 0

Pending deletion blocks: 0

-----

Live datanodes (2):

Name: 172.31.13.45:9866 (ip-172-31-13-45.sa-east-1.compute.internal)

Hostname: ip-172-31-13-45.sa-east-1.compute.internal

Decommission Status : Normal

Configured Capacity: 8577331200 (7.99 GB)

DFS Used: 51593216 (49.20 MB)

Non DFS Used: 3383468032 (3.15 GB)

DFS Remaining: 5142269952 (4.79 GB)

DFS Used%: 0.60%

DFS Remaining%: 59.95%

Configured Cache Capacity: 0 (0 B)

Cache Used: 0 (0 B)

Cache Remaining: 0 (0 B)

Cache Used%: 100.00%

Cache Remaining%: 0.00%

Xceivers: 0

Last contact: Tue Mar 08 10:14:28 UTC 2022

Last Block Report: Tue Mar 08 10:13:37 UTC 2022

Num of Blocks: 11

Name: 172.31.8.58:9866 (ip-172-31-8-58.sa-east-1.compute.internal)

Hostname: ip-172-31-8-58.sa-east-1.compute.internal

Decommission Status : Normal

Configured Capacity: 8577331200 (7.99 GB)

DFS Used: 51990528 (49.58 MB)  
Non DFS Used: 3383910400 (3.15 GB)  
DFS Remaining: 5141430272 (4.79 GB)  
DFS Used%: 0.61%  
DFS Remaining%: 59.94%  
Configured Cache Capacity: 0 (0 B)  
Cache Used: 0 (0 B)  
Cache Remaining: 0 (0 B)  
Cache Used%: 100.00%  
Cache Remaining%: 0.00%  
Xceivers: 0  
Last contact: Tue Mar 08 10:14:28 UTC 2022  
Last Block Report: Tue Mar 08 10:13:37 UTC 2022  
Num of Blocks: 12

## Testando o cluster

Para finalizar o projeto se faz necessário testar o nosso cluster, para tanto vamos fazer um job de contagem de palavras de um determinado arquivo (questões.csv).

- 1) Criar o diretório Download onde vamos copiar nosso arquivo .csv.

```
mkdir Downloads
```

- 2) Estando no diretório criado fazemos o download do arquivo .csv:

```
wget
```

```
http://datascienceacademy.com.br/blog/aluno/RFundamentos/Datasets/Parte2/questoes.csv
```

- 3) O que precisamos fazer agora é copiar o arquivo para o HDFS, mas primeiro vamos criar o diretório datasets onde vamos colocar o arquivo .csv.

```
hdfs dfs -mkdir /datasets
```

- 4) Copiando o arquivo e listando:

```
hdfs dfs -put questoes.csv /datasets  
hdfs dfs -ls /datasets
```

- 5) Chegou a hora de executar o job de processamento de MapReduce via YARN (contagem de palavras no arquivo)

```
yarn jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-  
mapreduce-examples-3.3.1.jar wordcount "/datasets/questoes.csv"  
output
```

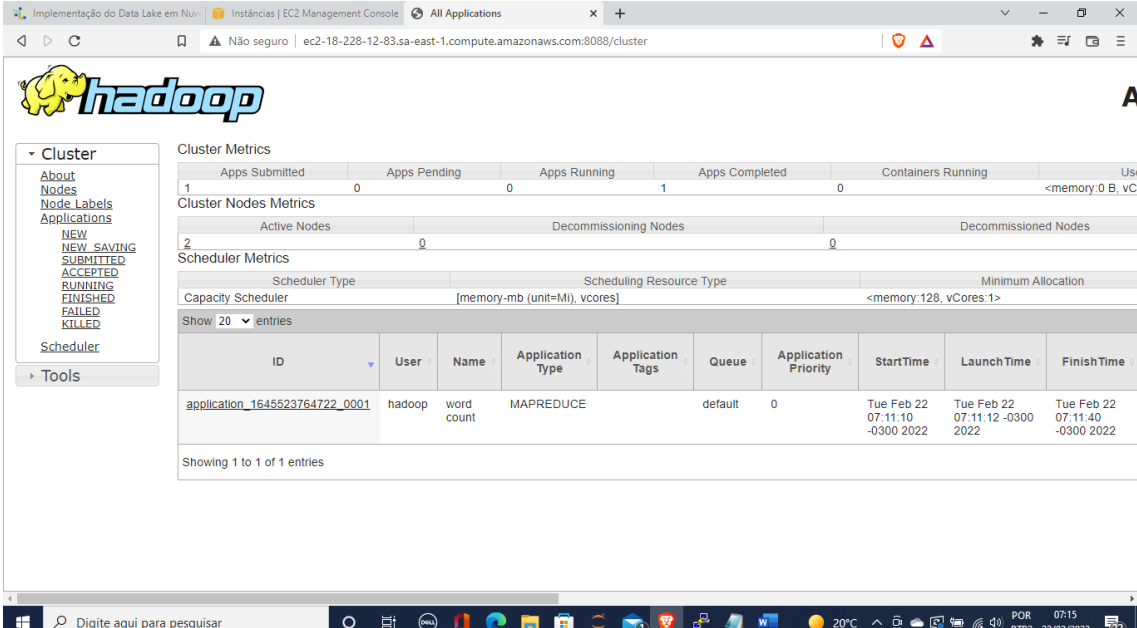
- 6) Checando a execução do Job:

```
hdfs dfs -cat output/part-r-00000  
yarn node -list  
yarn application -list
```

- 7) Novamente podemos acessar via browser:

```
http://ec2-18-230-145-251.sa-east-  
1.compute.amazonaws.com:8088/cluster
```





The screenshot shows the Amazon EMR console interface for a Hadoop cluster. The top navigation bar includes the AWS logo, the cluster name, and the EC2 Management Console link. The left sidebar contains a navigation menu with options like Cluster, Nodes, Node Labels, Applications, and Tools. The main content area displays the cluster's status and metrics.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Use
1	0	0	1	0	<memory:0 B, vCores:1>

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
2	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:128, vCores:1>

**Applications**

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1645523764722_0001	hadoop	word count	MAPREDUCE		default	0	Tue Feb 22 07:11:10 -0300 2022	Tue Feb 22 07:11:12 -0300 2022	Tue Feb 22 07:11:40 -0300 2022

Showing 1 to 1 of 1 entries

8) Por fim paramos o cluster:

```
$HADOOP_HOME/sbin/stop-yarn.sh
$HADOOP_HOME/sbin/stop-dfs.sh
```

## Conclusão

O objetivo do presente trabalho era implementar um cluster Hadoop na nuvem usando a Amazon AWS. O cluster foi implementado e testado com sucesso. Vale a pena salientar que tal tarefa não se mostrou trivial, mas ainda sim é menos complexa do que a implementação on-premise.

Tendo o cluster criado e configurado fica para trabalhos futuros as adições das camadas como o Apache Flume ou o Apache NiFi, Kafka, Hive entre outras.