محمد حسین شهبازی گزارش تمرین سوم فاز ۰ و ۱ ۴۰۰۳۱۰۸۸

فاز صفر:

نمایش موفقیت آمیز ساخت کانتینر ها:

```
[+] Running 13/13
 ✔ Volume "hind_hadoop_namenode"              Created                   0.0s
 ✔ Volume "hind_hadoop_historyserver"         Created                   0.0s
 ✔ Volume "hadoop-distributed-file-system"    Created                   0.0s
 ✔ Volume "hind_hadoop_datanode"              Created                   0.0s
 ✔ Container hadoop-datanode                  Started                  14.3s
 ✔ Container hadoop-historyserver             Started                  13.6s
 ✔ Container jupyter-notebook                 Started                  13.3s
 ✔ Container hadoop-resourcemanager           Started                  12.4s
 ✔ Container hadoop-nodemanager-1             Started                  13.0s
 ✔ Container spark-master                     St...                    12.8s
 ✔ Container hadoop-namenode                  Started                  14.5s
 ✔ Container spark-worker-2                   Started                  20.2s
 ✔ Container spark-worker-1                   Started                  20.0s
```

نمایش کانتینر های ایجاد شده:

```
                                                                                                                NAMES
8ce01b8510b2   hind-hadoop-historyserver     "/entrypoint.sh /run…"   5 days ago    Exited (255) 3 minutes ago
                                                                                                                hadoop-historyserver
f02f89627ca6   hind-spark-worker-1           "/bin/sh -c 'bin/spa…"   8 days ago    Exited (255) 3 minutes ago
                                                                                                                spark-worker-1
4a07689d26fd   hind-spark-worker-2           "/bin/sh -c 'bin/spa…"   8 days ago    Exited (255) 3 minutes ago
                                                                                                                spark-worker-2
c74af73a8273   hind-spark-master             "/bin/sh -c 'bin/spa…"   8 days ago    Exited (255) 3 minutes ago
0.0.0:8080->8080/tcp, :::8080->8080/tcp                                                                         spark-master
c916a423be56   hind-jupyter-notebook         "/bin/sh -c 'jupyter…"   8 days ago    Exited (255) 3 minutes ago
0.0.0:8888->8888/tcp, :::8888->8888/tcp                                                                         jupyter-notebook
85529e06dd59   hind-hadoop-datanode          "/entrypoint.sh /run…"   8 days ago    Up 2 minutes (healthy)
                                                                                                                hadoop-datanode
81846a6b9729   hind-hadoop-namenode          "/entrypoint.sh /run…"   8 days ago    Up 2 minutes (healthy)
0.0.0:9870->9870/tcp, :::9870->9870/tcp                                                                         hadoop-namenode
d33b89cd17aa   hind-hadoop-resourcemanager   "/entrypoint.sh /run…"   8 days ago    Up 2 minutes (healthy)
                                                                                                                hadoop-resourcemanager
6f2329ac8474   hind-hadoop-nodemanager-1     "/entrypoint.sh /run…"   8 days ago    Up 2 minutes (healthy)
```

توضیح کانتینر های ایجاد شده:

NameNode: گره اصلی در HDFS Hadoop که ابرداده سیستم(metadata) فایل را نگهداری و مدیریت می کند.

DataNode: گره‌های ذخیره کننده داده در HDFS که بلوک‌هایی از داده‌ها را زمانی که NameNode به آنها گفته می‌شود ذخیره و بازیابی می‌کنند.

ResourceManager: مرجع مرکزی در YARN برای مدیریت منابع و زمان‌بندی کار ها.

NodeManager: مدیر فریمورک هر ماشین در YARN که مسئول کانتینرها، نظارت بر استفاده از منابع آنها و گزارش همان به ResourceManager است.

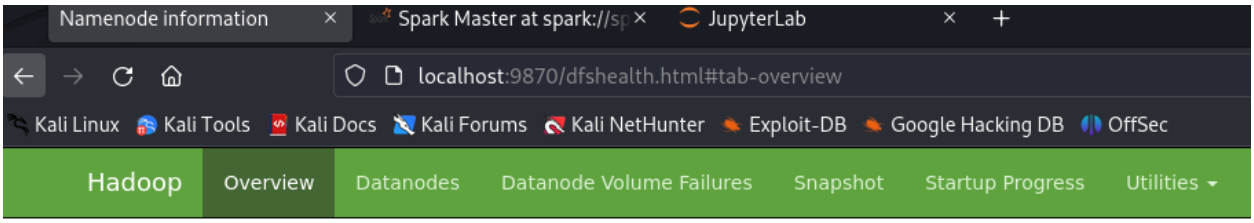
Spark Master: نقطه مرکزی و نقطه ورود خوشه Spark. مسئولیت توزیع وظایف به Sparkworker را بر عهده دارد.

Spark Worker: گره های پردازنده در کلاستر Spark که وظایف توزیع شده توسط Spark Master را اجرا می کنند.

Jpyter Notebook: یک برنامه وب منبع باز که امکان ایجاد و به اشتراک گذاری اسناد حاوی live code، Visualization را فراهم می کند. اغلب با Spark برای تجزیه و تحلیل داده ها استفاده می شود.

نمایش UI برای Hadoop و Spark و Jupyter :

Hadoop:



Overview 'hadoop-namenode:9000' (active)

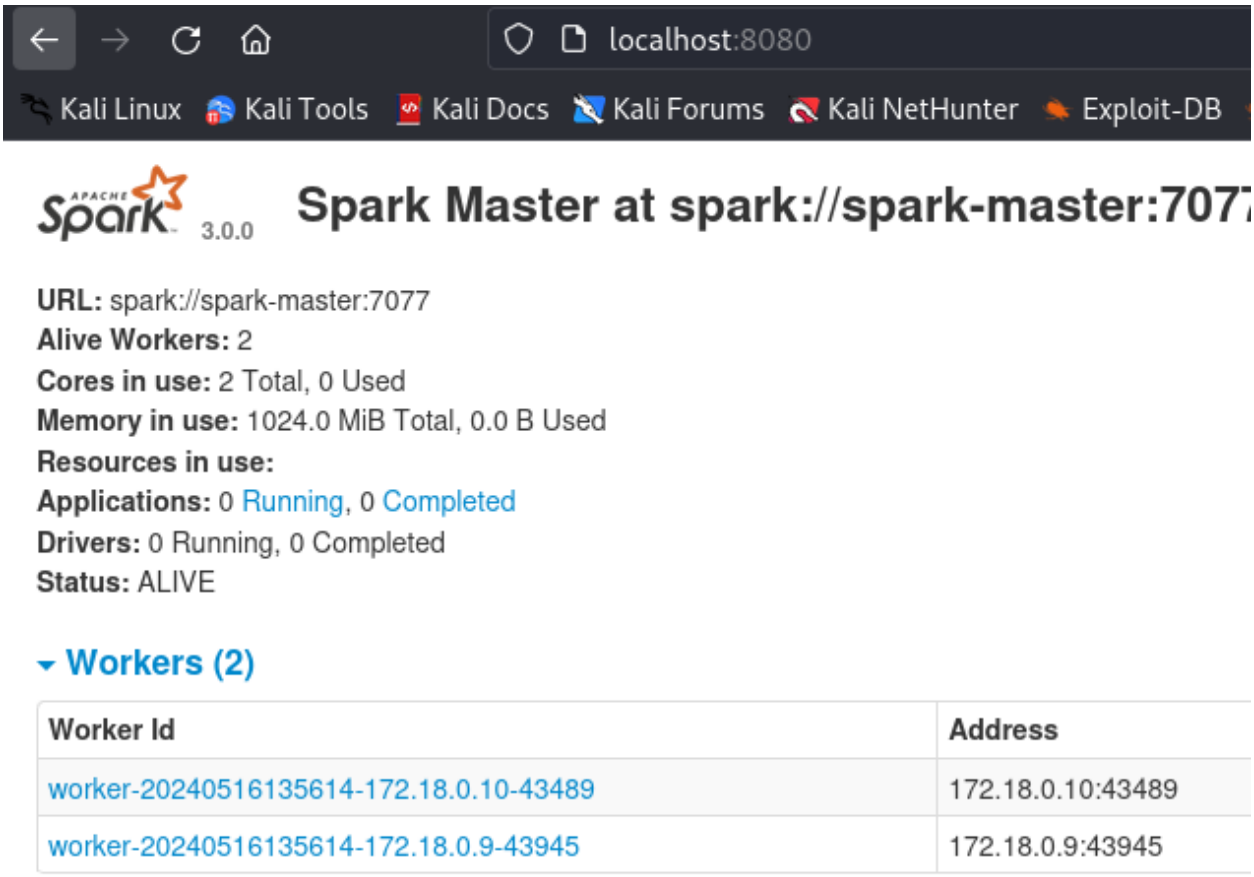
Started:	Thu May 16 09:56:36 -0400 2024
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 04:22:00 -0400 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-45fe742d-4292-41b5-bb37-50e9ac728ac2
Block Pool ID:	BP-1198026299-172.18.0.5-1715867784546

Summary

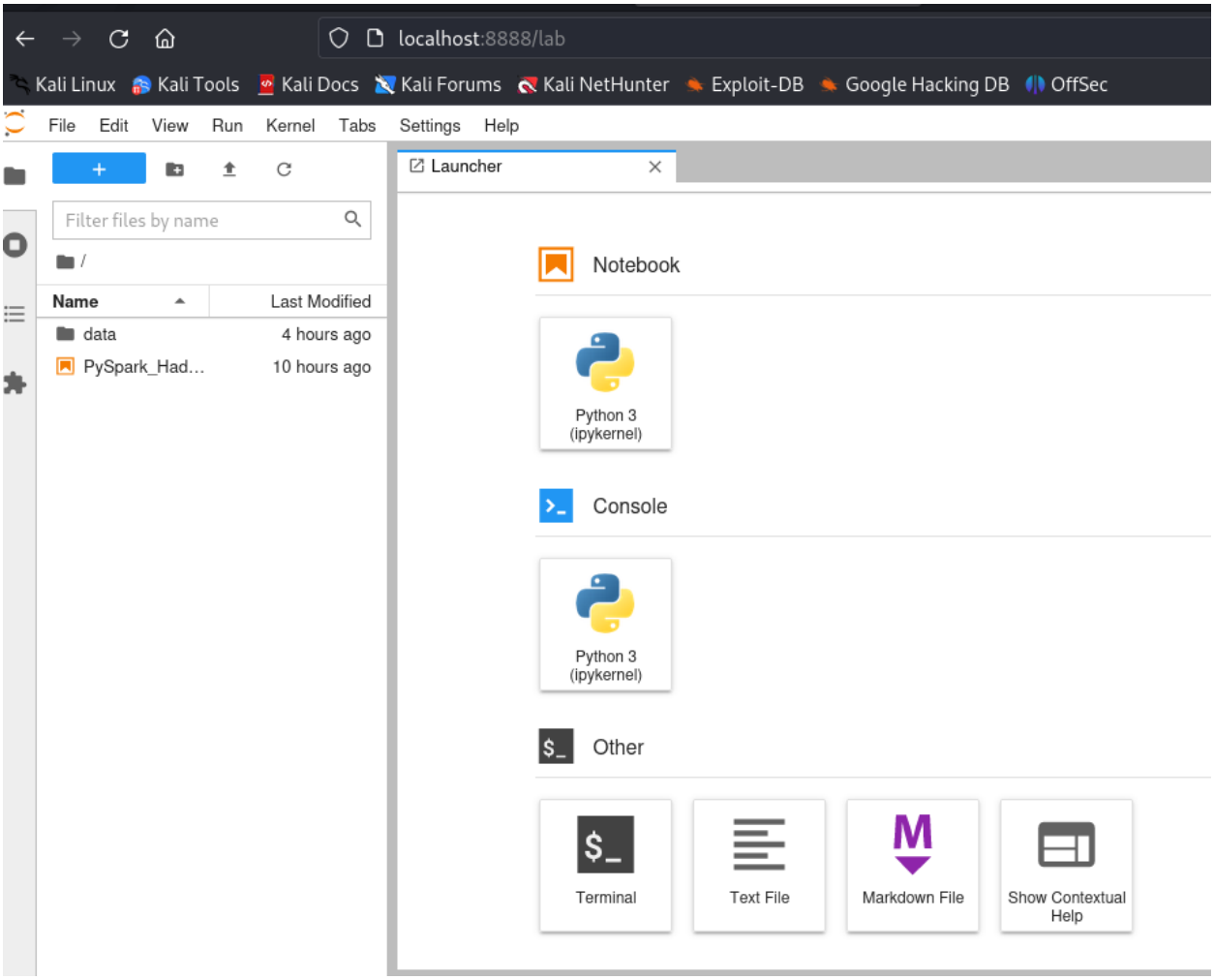
Security is off.

Safemode is off.

:Spark


localhost:8080
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Spark 3.0.0
Spark Master at spark://spark-master:7077
URL: spark://spark-master:7077
Alive Workers: 2
Cores in use: 2 Total, 0 Used
Memory in use: 1024.0 MiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE
Workers (2)
Worker Id
Address
worker-20240516135614-172.18.0.10-43489
172.18.0.10:43489
worker-20240516135614-172.18.0.9-43945
172.18.0.9:43945

:Jupiter notebook



توضیحات مربوط به تعداد نودهای اسپارک و منابع استفاده شده:

تعداد نود های اسپارک: ۳ عدد، یک master و ۲ worker
منابع استفاده شده: ۲ هسته پردازنده و ۱۰۲۴ مگابایت رم

توضیحات مربوط به اطلاعات NameNode و فایل سیستم آن:

۱۲۸ فایل و دایرکتوری، ۸۳ بلوک (۸۳ بلوک تکراری، ۰ گروه بلوک کدگذاری شده پاک شده) = ۲۱۱ شیء سیستم فایل کل.

در دایرکتوری : /hadoop/dfs/name

Type = IMAGE_AND_EDITS

StorageType: DISK

با ظرفیت ۵۸,۸۲ گیگابایت

فاز ۱:

بخش اول:

نمایش و توضیح کد mapper بخش اول:

```
root@81846a6b9729:/home# cat mapper.py
#!/usr/bin/env python
import sys

for line in sys.stdin:
    doc_id, text = line.strip().split(",", 1)
    words = text.split()
    for word in words:
        print(f'{word}\t{doc_id}')
```

توضیح: ابتدا خود Hadoop فایل input.txt در sys.stdout میریزد. سپس در اینجا از فایل خوانده شده از sys.stdin خط به خط خوانده شده و بین doc id و جمله روبروی آن که با کاما از هم جدا شده اند با دستور ()split جدا میشوند و اسپیس های اضافی حذف میشوند و در نهایت با دستور()print در sys.stdout ریخته میشوند.

نمایش و توضیح کد reducer.py :

```python
#!/usr/bin/env python
import sys

current_word = None
current_docs = set()

for line in sys.stdin:
    word, doc_id = line.strip().split("\t", 1)
    doc_id = doc_id.strip()  # This removes the newline character from doc_id
    if current_word == word:
        current_docs.add(doc_id)
    else:
        if current_word:
            print(f'{current_word}\t{set(current_docs)}')
        current_word = word
        current_docs = {doc_id}

if current_word == word:
    print(f'{current_word}\t{set(current_docs)}')
```

توضیح:در اینجا فایل ایجاد شده در قسمت قبل خط به خط از sys.stdin خوانده شدن و اگر کلمه دوباره در یک doc id دیگر تکرار شده بود آن را به مجموعه doc id هایش اضافه میکند و اگر کلمه جدید بود آن را به مجموعه doc id های جدید اضافه میکند.

خروجی :

توضیح:همانطور که مشاهده میشود کلمات در کنار لیست (Set) از doc id های خودش قابل تفکیک و مشاهده است.

بخش دوم:

نمایش و توضیح کد mapper بخش دوم:

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    doc_id, text = line.strip().split(",", 1)
    words = text.split()
    for word in words:
        print(f'{doc_id},{word}\t1')
```

توضیح: ابتدا خود Hadoop فایل input.txt در sys.stdout میریزد. سپس در اینجا از فایل خوانده شده از sys.stdin خط به خط خوانده شده و بین doc id و جمله روبروی آن که با کاما از هم جدا شده اند با دستور ()split جدا میشوند و اسپیس های اضافی حذف میشوند و در نهایت با دستور()print در sys.stdout ریخته میشوند.

نمایش و توضیح کد reducer.py :

```
#!/usr/bin/env python
import sys
from collections import Counter

current_doc = None
word_counter = Counter()

for line in sys.stdin:
    doc_word, count = line.strip().split("\t", 1)
    doc_id, word = doc_word.split(",", 1)
    count = int(count)

    if current_doc == doc_id:
        word_counter[word] += count
    else:
        if current_doc:
            for word, count in word_counter.most_common(3):  # Top 3 words
                print(f'{current_doc},{word}\t{count}')
        current_doc = doc_id
        word_counter = Counter({word: count})

if current_doc == doc_id:
    for word, count in word_counter.most_common(3):  # Top 3 words
        print(f'{current_doc},{word}\t{count}')
```

توضیح:

ابتدا یک counter() برای شمارش ۳ = k برترین کلمه پر تکرار ایجاد شده ایجاد میکنیم و خط به خط از sys.stdin ، doc id ها و کلمه روبرویش را میخوانیم و به تعداد تکرار آن اضافه میکنیم و سپس زمانی که برای یک doc id این کار تمام شد با دستور most_common(3) که در کلاس counter وجود داشت ۳ برترین کلمه را پیدا کرده و در خروجی چاپ میکنیم.

خروجی برای k=3:

```
┌──(mohammad㉿kali)-[~/Desktop/hw3_hadoop_phase0_1/hind]
└─$ docker exec -it 81846a6b9729 bash -c "hdfs dfs -cat /output2/*"
doc1,in 3
doc1,technology 3
doc1,innovation 2
doc10,innovation        2
doc10,and       2
doc10,a 1
doc2,is 3
doc2,sustainability     3
doc2,a  1
doc3,automation 2
doc3,intelligence       2
doc3,a  1
doc4,security   2
doc4,a  1
doc4,and        1
```

توضیح: همانطور که مشاهده میشود در خروجی doc id و شماره تکرار هر کلمه روبروی آن به تفکیک مشخص شده است.