HDFS, YARN – zadania

Tym razem, wyjątkowo, nie będziemy niczego analizowali, chcemy rozglądnąć się po platformie Hadoop, a w szczególności po jej dwóch składowych:

- Systemie plików HDFS
- Systemie YARN

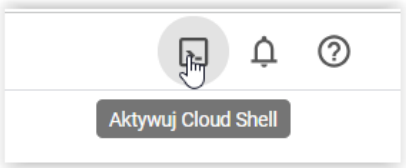
Przygotowanie środowiska

1. Utwórz klaster Dataproc korzystając z następujących kroków:

   a. Zaloguj się do platformy `https://cloud.google.com/gcp/`

   b. Przejdź do konsoli tej platformy

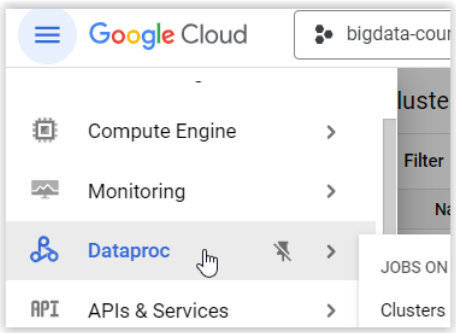
   c. Aktywuj *Cloud Shell* pozwalający zarządzać tę platformą za pomocą poleceń linii komend.

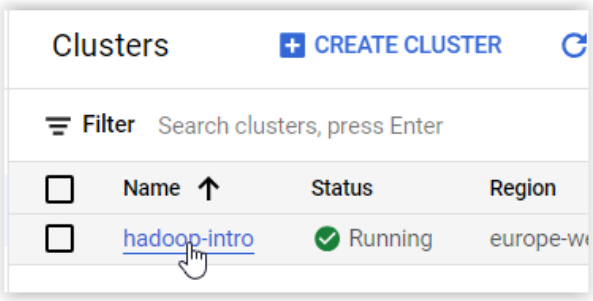


   d. Uruchom klaster Dataproc korzystając z poniższych poleceń *Cloud Shell*. Ustaw właściwe wartości zmiennych.

```
gcloud dataproc clusters create ${CLUSTER_NAME} \
 --enable-component-gateway \
 --region ${REGION} \
 --master-machine-type n1-standard-2 --master-boot-disk-size 50 \
 --num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 \
 --image-version  2.1-debian11 \
 --project ${PROJECT_ID} --max-age=3h
```

2. Wybierając z lewego menu pozycję *Dataproc* wyświetl listę klastrów. Zaczekaj aż klaster zostanie utworzony. Odśwież stronę i wybierz klaster utworzony przed chwilą

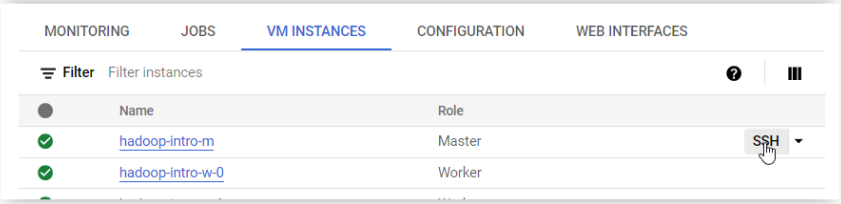




Przesłanie danych

Do naszych zadań potrzebujemy przykładowego zbioru danych. Każdorazowo taki nowy zbiór danych warto umieścić w powiązanym z klastrem zasobniku, a następnie ładować go bezpośrednio z zasobnika. Naszym zbiorem danych będzie ten sam, który wykorzystywany był w zestawie zadań z MapReduce. Jeśli masz go już umieszonego w zasobniku, skoryguj odpowiednio zadania z tej sekcji.

3. Ze strony zawierającej szczegóły dotyczące klastra wybierz zakładkę *VM Instances*, a następnie podłącz się za pomocą terminala SSH do maszyny master klastra



4. Pobierz na lokalny dysk maszyny master `cycle-share-dataset.zip`

```
wget https://jankiewicz.pl/bigdata/hadoop-intro/cycle-share-dataset.zip
```

5. A następnie go rozpakuj oraz usuń niepotrzebne już archiwum.

```
unzip cycle-share-dataset.zip

rm cycle-share-dataset.zip
```

```
jankiewicz_krzysztof@hadoop-intro-m:~$ unzip cycle-share-dataset.zip
Archive:  cycle-share-dataset.zip
  inflating: station.csv
   creating: trips/
  inflating: trips/trip1.csv
  inflating: trips/trip2.csv
  inflating: trips/trip3.csv
  inflating: weather.csv
```

6. Dane analizowane w ramach modelu *MapReduce*, a także przy wykorzystaniu wielu innych silników, dobrze jeśli znajdują się w systemie plików HDFS, a nie lokalnym systemie plików.
   Utwórz w systemie plików HDFS katalog `input`.

```
hadoop fs -mkdir -p input
```

7. Przekopiuj pliki tworzące nasz zestaw danych z lokalnego systemu plików do systemu plików HDFS do katalogu `input`.

```
hadoop fs -put * input/
```

8. Sprawdź czy pliki znajdują się na swoim miejscu docelowym

```
hadoop fs -ls input
```

```
jankiewicz_krzysztof@hadoop-intro-m:~$ hadoop fs -ls input
Found 3 items
-rw-r--r--   2 jankiewicz_krzysztof hadoop       5316 2023-10-02 08:31 input/station.csv
drwxr-xr-x   - jankiewicz_krzysztof hadoop          0 2023-10-02 08:31 input/trips
-rw-r--r--   2 jankiewicz_krzysztof hadoop      56516 2023-10-02 08:31 input/weather.csv
```

HDFS

W kilku kolejnych zadaniach przeegzaminujemy system plików HDFS, do którego wprowadziliśmy nasze dane.

9. Na początek, wykorzystując terminal SSH i poniższe polecenie, sprawdź miary związane z całym systemem plików HDFS

```
hdfs fsck /
```

   a. Jaki jest całkowity rozmiar zawartych w nim danych (*Total size*)?
   b. Ile te dane wykorzystują bloków (*Total block*)?
   c. Ile cały system zawiera plików (*Total files*)?
   d. Jaka jest średnia wielkość bloku (*avg. block size*)?
   e. Jaki jest domyślny poziom replikacji (*Default replication factor*)?
   f. Czy poziom replikacji mógłby być większy przy obecnej architekturze (liczba węzłów danych wykorzystywanych przez HDFS)?

10. Alternatywnym dla replikacji sposobem zabezpieczania danych w HDFS (i wielu innych rozproszonych systemach plików) jest *erasure coding*, który dla określonej liczby jednostek danych wylicza określoną liczbę jednostek parzystości (sumy kontrolne). Sprawdź jakie reguły (*policy*) są dostępne w naszej konfiguracji

```
hdfs ec -listPolicies
```

11. Sprawdź jaka reguła jest ustawiona dla naszego katalogu `input/trips`

```
hdfs ec -getPolicy -path input/trips
```

12. Utwórz nowy katalog. Tym razem skorzystamy z polecenia `hdfs dfs`, które przeznaczone jest tylko dla HDFS i nie pozwala na jego wykorzystanie (przez brak dodatkowej warstwy abstrakcji) w kontekście innych rozproszonych systemów plików (np. GFS)

```
hdfs dfs -mkdir -p input/trips-ec
```

13. Przypisz naszą aktywną regułę EC do tego katalogu

```
hdfs ec -setPolicy -policy RS-6-3-1024k -path input/trips-ec
```

14. Spróbuj przekopiować nasze dane dotyczące przejazdów do tego katalogu

```
hadoop fs -cp input/trips/* input/trips-ec
```

15. Wyjaśnij dlaczego ta operacja się nie powiodła

16. Sprawdź ile bloków zajmuje nasz katalog `input`.

```
hdfs fsck input -files -blocks -racks
```

17. I jeszcze kilka pytań dotyczących pliku `station.csv`. Skorzystaj z poniższego polecenia.
   a. Sprawdź ile bloków zajmuje?
   b. W ilu szafkach rezydują bloki tego pliku?
   c. W ilu węzłach danych znajdują się bloki tego pliku?
   d. Jaki jest średni poziom replikacji bloków tego pliku?

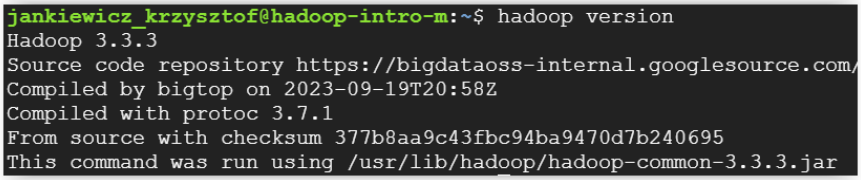
```
hdfs fsck input/station.csv -files -blocks -racks
```

18. Rozwiązując kolejne zadania skorzystaj z możliwości poniższych poleceń. Nazwy potrzebnych kluczy znajdziesz na poniższej stronie (a także w nawiasach przy poszczególnych pytaniach).
    `http://hadoop.apache.org/docs/r3.3.3/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml`

```
hdfs dfsadmin -report
hdfs getconf -confKey <klucz>
```

Wykorzystywaną wersję platformy Hadoop możesz poznać za pomocą polecenia

```
hadoop version
```



19. Korzystając z pierwszego z poleceń uzyskaj odpowiedzi na pytania:
    a. Ile wykorzystywanych jest węzłów danych (*Live datanodes*)?
    b. Ile dostępnej jest całkowitej przestrzeni (*DFS Remaining*)?
    c. Ile dostępnej jest przestrzeni na każdym z węzłów danych?

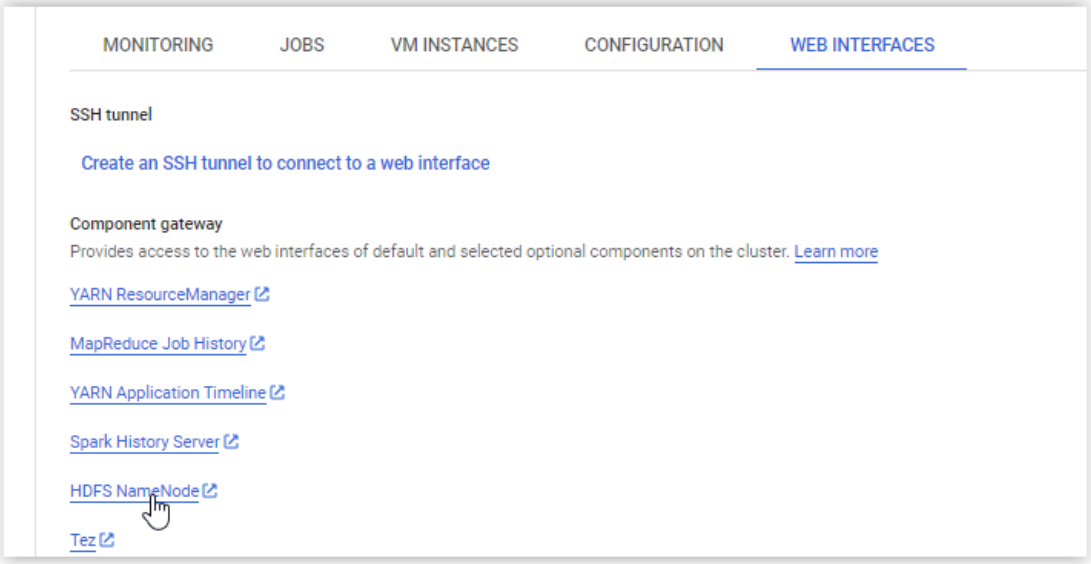
20. Korzystając z drugiego z poleceń uzyskaj odpowiedź na pytania.
    a. W jakim katalogu przechowywane są fizycznie bloki zarządzane przez węzeł danych (`dfs.datanode.data.dir`)?
    b. Jaka maksymalna liczba wątków obsługuje transfer danych pomiędzy klientami a węzłem danych (`dfs.datanode.max.transfer.threads`)?

```
hdfs getconf -confKey dfs.datanode.max.transfer.threads
```

    c. Na obsługę jakiej maksymalnej liczby plików przygotowany jest węzeł nazw, zakładając, że maksymalna wielkość *NameNode Java heap size* to 1 GB a poziom replikacji to 1.
    Wskazówki dotyczące tego jak to wyliczyć znajdziesz na stronie:

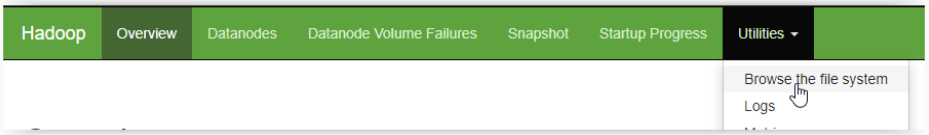
`https://www.cloudera.com/documentation/enterprise/5-8-x/topics/admin_nn_memory_config.html`

21. Korzystając z konsoli Google Cloud Platform, ze strony prezentującej nasz klaster Dataproc, przejdź na zakładkę *Web Interfaces* (Interfejsy sieciowe), a następnie korzystając z linku *HDFS NameNode* uruchom stronę z interfejsem węzła nazw.



Korzystając z jego interfejsu odpowiedz na następujące pytania

a. Ile węzłów danych jest aktywnych?
b. Jaki jest faktyczny rozmiar *Max Heap Memory*? Uwzględniając rzeczywisty poziom replikacji oblicz na obsługę jakiej maksymalnej liczby plików przygotowany jest ten węzeł nazw.
c. Zwróć uwagę na możliwość przeglądania (i nie tylko) zawartości HDFS dostępną w ramach tego interfejsu.



YARN

(10 minut)

Korzystając z otwartego terminala, zobaczmy teraz jak wygląda konfiguracja menadżera zasobów YARN

22. Sprawdź czy w pliku `/etc/hadoop/conf/yarn-site.xml` znajduje się parametr `yarn.resourcemanager.scheduler.class` określający typ programu szeregującego.

```
cat /etc/hadoop/conf/yarn-site.xml | grep scheduler.class -B 1 -A 3
```

```
jankiewicz_krzysztof@hadoop-intro-m:~$ cat /etc/hadoop/conf/yarn-site.xml | grep scheduler.class -B 1 -A 3
jankiewicz_krzysztof@hadoop-intro-m:~$
```

Jak widać nie ma go tam ustawionego. Na stronie `https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-common/yarn-default.xml` możesz zawsze znaleźć domyślne wartości parametrów.

yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler

Ponadto, zamiast przeglądać zawartość plików konfiguracyjnych możesz, mimo iż jest zapewne mało intuicyjne, wykorzystać użytą już wcześniej metodę

```
hdfs getconf -confKey yarn.resourcemanager.scheduler.class
```

"`hdfs getconf -confKey`" można wykorzystać do uzyskania wartości parametrów nie tylko HDFS, ale także YARN lub MapReduce. Warto to zapamiętać – dużo to ułatwia.

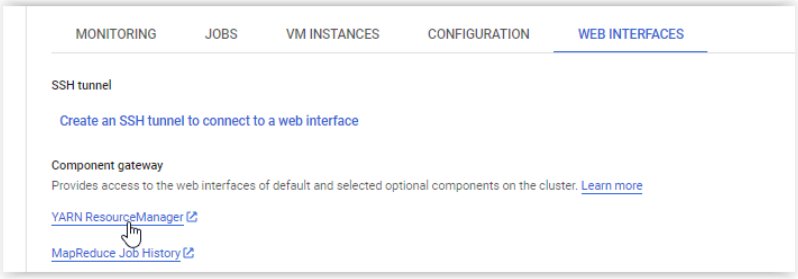
Zastanów się co oznacza ten parametr. Jeśli pierwsza z aplikacji YARN zajmie wszystkie zasoby w ramach kolejki, to czy przy uruchomieniu kolejnej aplikacji ta pierwsza zostanie wywłaszczona?

23. Ponownie korzystając z powyższego polecenia odpowiedz na poniższe pytania. Nazwy kluczy znajdziesz na poniższej stronie, a także w nawiasach przy poszczególnych pytaniach:
`https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-common/yarn-default.xml`
lub
`https://hadoop.apache.org/docs/r3.3.3/hadoop-yarn/hadoop-yarn-common/yarn-default.xml`
dla określonej wersji Hadoopa

24. Odpowiedz na następujące pytania
    a. Ile pamięci przeznaczono na wszystkie kontenery (`yarn.nodemanager.resource.memory-mb`)?
    b. Jaka najmniejsza ilość pamięci może zaalokowana dla jednego kontenera (`yarn.scheduler.minimum-allocation-mb`)?
    c. Jaki procent fizycznych procesorów na węźle roboczym będzie przeznaczona na obsługę kontenerów (`yarn.nodemanager.resource.percentage-physical-cpu-limit`)?
    d. Ile wirtualnych rdzeni procesora zostało zadeklarowanych dla każdego z Node Managerów (`yarn.nodemanager.resource.cpu-vcores`)?

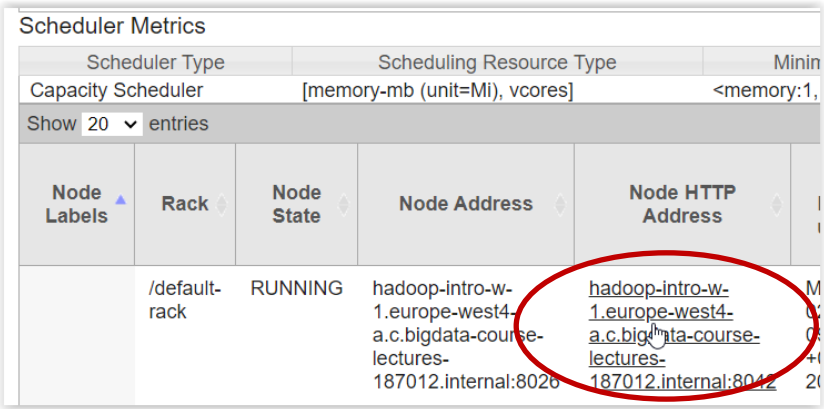
YARN Resource Manager interfejs sieciowy

25. Korzystając z konsoli Google Cloud Platform, ze strony prezentującej nasz klaster Dataproc, przejdź na zakładkę *Web Interfaces* (Interfejsy sieciowe), a następnie korzystając z linku *YARN ResourceManager* wejdź na stronę managera zasobów systemu YARN



26. Na początku sprawdź kolejki programu szeregującego (w menu nawigacji link *Scheduler*):
    a. Czy typ programu szeregującego zgadza się z poprzednimi wnioskami?
    b. Jak nazywają się dostępne kolejki?

27. Wybierz z menu nawigacji *Nodes*. Wybierz link pozwalający nam na przejście do informacji na temat jednego z aktywnych węzłów wchodzącego w skład naszego klastra.



28. Odpowiedz na następujące pytania:
    a. Ile obecnie używanych jest tam kontenerów?
    b. Ile fizycznej pamięci (PMem) jest zarezerwowanych dla kontenerów?
    c. Ile wirtualnych rdzeni (VCore) jest dostępnych dla kontenerów?

29. Wybierz z menu nawigacji w interfejsie *Resource Managera* link *Applications*. Trochę cicho w tym naszym klastrze. Trochę to teraz zmienimy.

30. Spróbujemy uruchomić kanoniczny program MapReduce zliczający słowa (patrz np.: `https://www.cloudera.com/documentation/other/tutorial/CDH5/topics/ht_wordcount1.html`). Pobierz go za pomocą poniższego polecenia.

```
wget https://jankiewicz.pl/bigdata/hadoop-intro/WordCount.java
```

31. Skompiluj program korzystając z poniższych poleceń

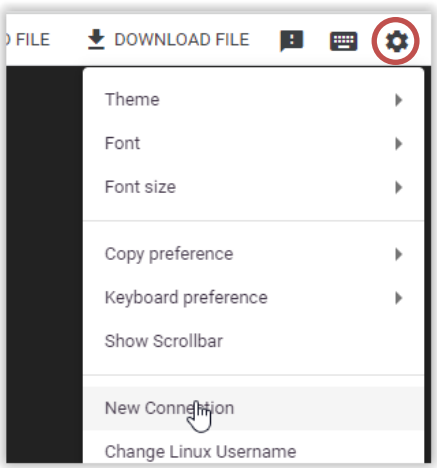
```
mkdir -p build
javac -cp `hadoop classpath` WordCount.java -d build
jar -cvf wordcount.jar -C build/ .
```

32. Przeanalizuj jeszcze raz dostępne zasoby dla aplikacji YARN. Przyglądnij się poniższym parametrom polecenia uruchamiającego nasz program. Jak będzie wyglądała zajętość zasobów naszego klastra po uruchomieniu tego programu? Jeśli nie znasz mechanizmu działania programu MapReduce skonsultuj się z prowadzącym.

```
hadoop jar wordcount.jar WordCount \
    -Dyarn.app.mapreduce.am.resource.mb=3072 \
    -Dyarn.app.mapreduce.am.resource.cpu-vcores=1 \
    -Dmapreduce.map.memory.mb=3072 \
    -Dmapreduce.map.cpu.vcores=1 \
    -Dmapreduce.job.maps=3 \
    -Dmapreduce.reduce.memory.mb=3072 \
    -Dmapreduce.reduce.cpu.vcores=1 \
    -Dmapreduce.job.reduces=3 \
    input/trips \
    output1
```

33. Korzystając z parametrów polecenia, Twojej wiedzy na temat przetwarzania zadań MapReduce oraz sieciowego interfejsu YARN, odpowiedz na pytania:
    a. Czy aplikacja MapReduce rzeczywiście uruchomiła aplikację YARN?
    b. W ramach jakiej kolejki zaalokowała ona zasoby?
    c. Ile wykorzystała ona kontenerów w pierwszej fazie (mapowania)?
    d. Ile pamięci zaalokowała dla wszystkich kontenerów w pierwszej fazie (mapowania)?
    e. Ile wirtualnych procesorów zaalokowała dla wszystkich kontenerów w pierwszej fazie (mapowania)?
    f. Czy zawłaszczyła wszystkie zasoby z kolejki?

34. Uruchom nowy terminal SSH do maszyny master. Nazwijmy go **dodatkowym**, podczas gdy wcześniejszy nazwiemy **głównym**.



35. W terminalu **głównym** ponownie uruchom nasz program MapReduce i od razu nie czekając na jego zakończenie przejdź do następnego punktu

```
hadoop jar wordcount.jar WordCount \
    -Dyarn.app.mapreduce.am.resource.mb=3072 \
    -Dyarn.app.mapreduce.am.resource.cpu-vcores=1 \
    -Dmapreduce.map.memory.mb=3072 \
    -Dmapreduce.map.cpu.vcores=1 \
    -Dmapreduce.job.maps=3 \
    -Dmapreduce.reduce.memory.mb=3072 \
    -Dmapreduce.reduce.cpu.vcores=1 \
    -Dmapreduce.job.reduces=3 \
    input/trips \
    output2
```

36. W terminalu **dodatkowym** uruchom jeszcze jedną instancję naszego programu MapReduce tym razem opierając się na wartościach domyślnych i od razu przejdź do następnego punktu

```
hadoop jar wordcount.jar WordCount \
    input/trips \
    output3
```

37. Odśwież stronę interfejsu zarządcy zasobów (*Resource Managera*) YARN, tak aby uchwycić fakt zablokowania jednego zadania przez drugie.

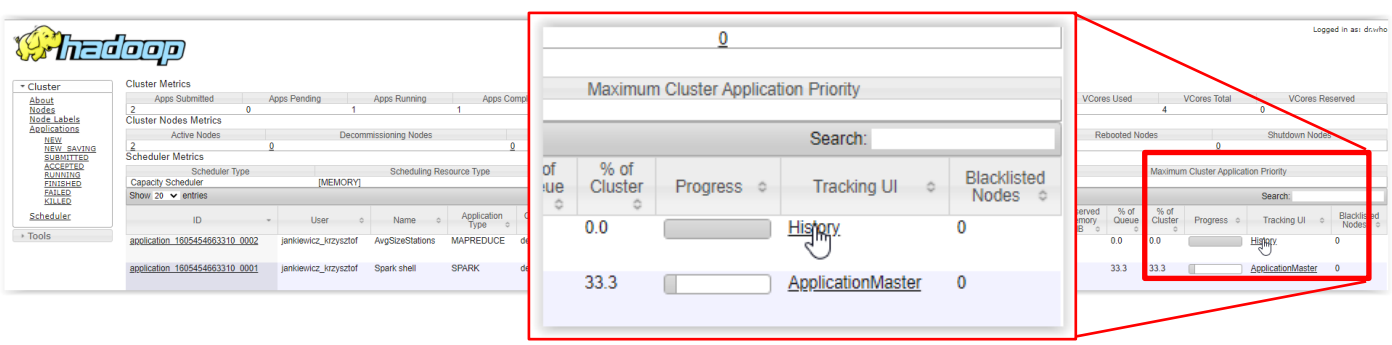
ne	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores	Reserved Memory MB	Reserved GPUs	% of Queue	% of Cluster
	ACCEPTED	UNDEFINED	0	0	0	-1	0	0	-1	0.0	0.0
	RUNNING	UNDEFINED	4	4	12288	-1	0	0	-1	100.0	100.0

Ze względu na zaalokowane zasoby przez pierwsze z zadań, to drugie nie jest w stanie przejść z fazy ACCEPTED do fazy RUNNING.
Jaki mechanizm szeregowania zadań dostępny w YARN pozwoliłby na wywłaszczenie części zasobów aplikacji pierwszej i przydzielenie ich od razu aplikacji drugiej, aby ta nie musiała czekać na uruchomienie?

38. Podczas uruchamiania **ostatniego** z zadań *MapReduce* wygenerowany został link pozwalający na poznanie jego szczegółów. Znajdź ten link w terminalu.

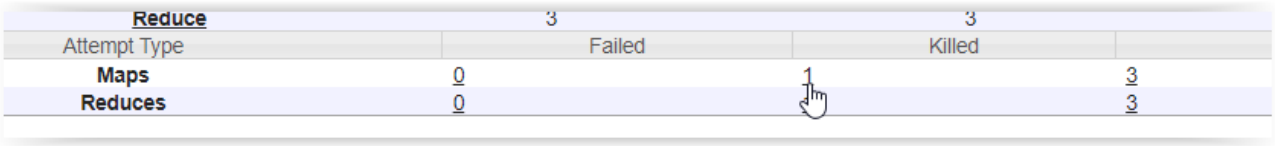
```
2023-10-02 10:19:02,854 INFO impl.YarnClientImpl: Submitted application application_1696235164860_0003
2023-10-02 10:19:02,909 INFO mapreduce.Job: The url to track the job: http://hadoop-intro-m.c.bigdata-course-lectures-187012.internal.:8088/proxy/application_1696235164860_0003/
```

Ze względu na charakter środowiska musimy skorzystać z interfejsu *Resource Managera* YARN.
On także dostarcza odpowiedni link (w naszym przypadku właściwy).



39. Odpowiedz na następujące pytania:
    a. Jaka była nazwa tej aplikacji (*Job Name*)? Skąd się ona wzięła? Gdzie można było ją zmienić?
    b. W ramach której kolejki była wykonana ta aplikacja?
    c. Jaki jest obecny stan ten aplikacji?
    d. Ile jednostek zadań *Map* zostało wykorzystanych?
    e. Ile jednostek zadań *Reduce* zostało wykorzystanych?
    f. Co oznacza termin *uberized?*

40. Czy w Twoim przypadku jedna z jednostek zadań została także zabita? Jeśli tak dowiedz się dlaczego. Jeśli czegoś nie rozumiesz, skonsultuj to z prowadzącym.



41. Przejdź na stronę z licznikami (*Job-> Counters*) dotyczącymi tego zadania
    a. Ile MB odczytano z systemu plików HDFS?
    b. Ile MB zapisano w systemie plików HDFS?
    c. Ile MB zapisano w lokalnych systemach plików?

    Podobne linki jak ten udostępniający informacje o aplikacji MapReduce powinny być (i są) dostępne są dla każdego typu aplikacji działających w ramach klastra YARN.

YARN interfejs wiersza poleceń

YARN oprócz interfejsu sieciowego *Resource Managera* dostarcza także interfejs dostępny z linii komend.

Szczegóły możesz znaleźć na poniższej stronie

`https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YarnCommands.html`

42. Uruchom w terminalu **dodatkowym** sesję Apache Spark. Będzie ona działała tak długo jak tylko będziemy chcieli. Ułatwi to nam eksplorację interfejsu linii komend.

```
spark-shell --master yarn
```

Użyj terminala **głównego** i interfejsu wiersza poleceń YARN, aby wykonać następujące zadania.

43. Pobierz listę węzłów roboczych (*NodeManagers*).

44. Pobierz listę aplikacji YARN.

45. Pobierz listę prób uruchomienia aplikacji Spark.

46. Pobierz listę kontenerów przydzielonych do pomyślnej próby uruchomienia.

47. Pobierz stan jednego z kontenerów.

48. Pobierz dzienniki jednego z kontenerów

49. Zabij aplikację Spark.