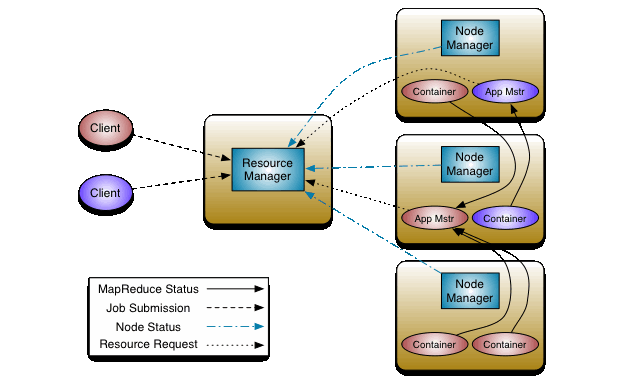
**HADOOP & YARN**

1. **APACHE HADOOP YARN**

**Docs:** [**https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html**](https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html)

[](file:///T:\Hadoop-yarn-html\Lab3-Hadoop-yarn-html\Hadoop%20&%20Yarn%200710e74db3724817a6e5c5035ec0f577\image.png)

**Apache Hadoop YARN (Yet Another Resource Negotiator)** là một trong những thành phần chính của hệ sinh thái Hadoop. Nó được thiết kế để quản lý tài nguyên và điều phối các ứng dụng trên cụm (cluster). YARN giúp phân chia trách nhiệm xử lý và quản lý tài nguyên, giúp tăng khả năng mở rộng và hiệu suất của các hệ thống phân tán lớn.

**Cấu trúc của YARN:**

1. **Resource Manager (RM)**:
   * Là thành phần trung tâm quản lý tài nguyên của toàn bộ hệ thống. RM chịu trách nhiệm cấp phát tài nguyên cho các ứng dụng đang chạy trên Hadoop.
   * RM gồm 2 thành phần con:
     + **Scheduler**: Phân phối tài nguyên dựa trên các chính sách lập lịch (ví dụ: Fair Scheduling, Capacity Scheduling).
     + **Application Manager**: Quản lý vòng đời của các ứng dụng trên cụm, theo dõi các yêu cầu tài nguyên từ các ứng dụng.
2. **Node Manager (NM)**:
   * Được cài đặt trên mỗi node (máy tính) trong cụm, NM giám sát các container (một đơn vị tài nguyên tính toán như CPU, bộ nhớ) và báo cáo lại trạng thái tài nguyên cho RM.
   * NM thực hiện các tác vụ như theo dõi sử dụng tài nguyên, khởi chạy, dừng, và quản lý vòng đời của container trên node.
3. **Application Master (AM)**:
   * Là một quy trình duy nhất chạy cho mỗi ứng dụng, chịu trách nhiệm đàm phán tài nguyên với RM và làm việc với các NM để khởi chạy và giám sát các container.
4. **Container**:
   * Là đơn vị tài nguyên cơ bản được YARN phân phối cho các ứng dụng, bao gồm một lượng CPU và bộ nhớ nhất định để thực thi tác vụ.

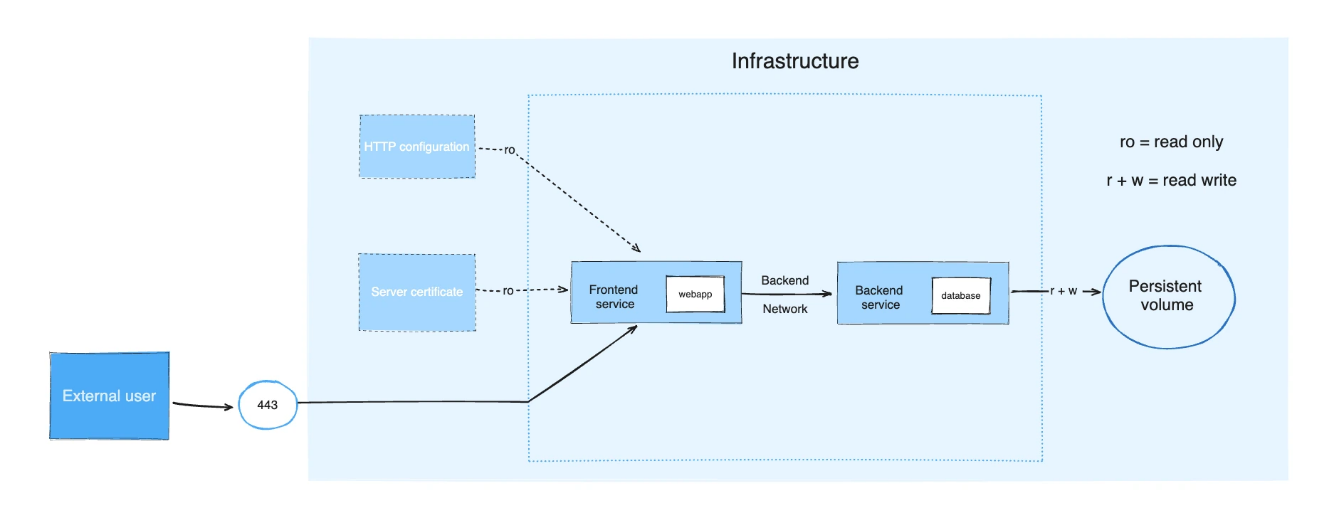
**Lợi ích của YARN:**

* **Quản lý tài nguyên hiệu quả**: YARN tách việc quản lý tài nguyên ra khỏi xử lý MapReduce, cho phép nhiều mô hình lập trình (beyond MapReduce) chạy trên cùng một cụm.
* **Khả năng mở rộng cao**: YARN hỗ trợ phân phối tài nguyên linh hoạt, phù hợp cho các hệ thống lớn với hàng nghìn node.
* **Hỗ trợ nhiều mô hình tính toán**: YARN cho phép các ứng dụng khác nhau như Apache Spark, Apache Flink, hoặc ứng dụng streaming hoạt động trên cùng một cụm Hadoop.

**Quy trình hoạt động cơ bản của YARN:**

1. Người dùng nộp một ứng dụng tới RM.
2. RM khởi động một AM cho ứng dụng.
3. AM yêu cầu tài nguyên từ RM.
4. RM phân phối tài nguyên dưới dạng container tới các NM.
5. NM khởi chạy các container với tác vụ mà AM yêu cầu.
6. AM theo dõi và điều phối việc thực thi các container.
7. **RUN HADOOP YARN VỚI DOCKER COMPOSE**
8. **Docker compose**

* **Docs:** [**https://docs.docker.com/compose/**](https://docs.docker.com/compose/)

**[](file:///T:\Hadoop-yarn-html\Lab3-Hadoop-yarn-html\Hadoop%20&%20Yarn%200710e74db3724817a6e5c5035ec0f577\image%201.png)**

**Docker Compose** là một công cụ giúp định nghĩa và quản lý nhiều container Docker như một nhóm dịch vụ, tất cả được xác định trong một file duy nhất gọi là docker-compose.yml. Với Docker Compose, bạn có thể dễ dàng khởi tạo, cấu hình và chạy nhiều container với các dịch vụ khác nhau trên cùng một máy mà không cần phải khởi chạy chúng riêng lẻ.

**Các thành phần chính của Docker Compose:**

1. Dockerfile (tùy chọn):
   * Là file chứa các hướng dẫn để xây dựng image Docker. Mỗi service trong docker-compose.yml có thể sử dụng image đã được tạo sẵn hoặc tự tạo từ Dockerfile.
2. docker-compose.yml:
   * Đây là file định nghĩa cấu hình của các container, bao gồm các service, volumes, networks, và các thông số cấu hình khác. File này sử dụng cú pháp YAML.

**Quy trình hoạt động của Docker Compose:**

1. **Định nghĩa các dịch vụ** : Trong file docker-compose.yml, bạn định nghĩa các service khác nhau mà bạn muốn chạy, chẳng hạn như web, database, cache.
2. **Tạo và khởi động :** Sử dụng lệnh docker-compose up, Docker Compose sẽ tự động xây dựng và khởi chạy tất cả các container dựa trên định nghĩa trong file.
3. **Quản lý vòng đời :** Bạn có thể dừng tất cả các dịch vụ bằng docker-compose down, quản lý logs của từng container bằng docker-compose logs, hoặc kiểm tra trạng thái container với docker-compose ps.

**Cấu trúc file docker-compose.yml:**

Một ví dụ file docker-compose.yml đơn giản cho ứng dụng web với cơ sở dữ liệu MySQL có thể trông như sau:

A computer screen with text on it

Description automatically generated

**Giải thích cấu trúc trên:**

* **version**: Phiên bản của Docker Compose. Phiên bản 3 là phiên bản phổ biến và ổn định nhất.
* **services**: Định nghĩa các dịch vụ, mỗi dịch vụ tương ứng với một container.
  + **web**: Dịch vụ chính, sử dụng image được xây dựng từ Dockerfile trong thư mục hiện tại (thông qua build: .), và sẽ chạy trên port 5000.
  + **db**: Dịch vụ cơ sở dữ liệu, sử dụng image MySQL 5.7. Dữ liệu của MySQL được lưu trữ trên volume db\_data.
* **depends\_on**: Xác định rằng web sẽ khởi động sau khi db đã sẵn sàng.
* **volumes**: Định nghĩa nơi lưu trữ dữ liệu để giữ nguyên khi container bị xóa, ở đây là lưu trữ dữ liệu của MySQL.

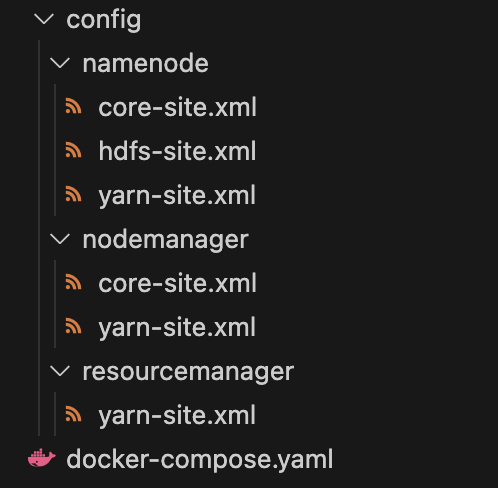
**Lệnh Docker Compose cơ bản:**

* **docker-compose up**: Khởi động tất cả các dịch vụ đã định nghĩa trong docker-compose.yml.
* **docker-compose down**: Dừng tất cả các dịch vụ và xóa các container.
* **docker-compose build**: Xây dựng lại image Docker từ Dockerfile.
* **docker-compose ps**: Hiển thị trạng thái của các container.
* **docker-compose logs**: Xem logs từ các container.

**Lợi ích của Docker Compose:**

* Dễ dàng quản lý nhiều container: Thay vì phải khởi động các container riêng lẻ, bạn có thể định nghĩa tất cả trong một file và chạy tất cả chúng với một lệnh duy nhất.
* Tự động hóa cấu hình: Các container được cấu hình tự động qua file YAML, giúp đồng bộ hóa cấu hình môi trường giữa các máy phát triển.
* Tích hợp dễ dàng: Docker Compose rất hữu ích khi làm việc với các ứng dụng đa tầng như microservices, nơi mỗi dịch vụ chạy trong một container riêng biệt (ví dụ: web, API, cơ sở dữ liệu).

1. **Hadoop Yarn với Docker Compose**
2. **Cấu trúc thư mục lab3:**

[](file:///T:\Hadoop-yarn-html\Lab3-Hadoop-yarn-html\Hadoop%20&%20Yarn%200710e74db3724817a6e5c5035ec0f577\dir.png)

#### Docker-compose.yaml file

services:

namenode:

image: bde2020/hadoop-namenode:2.0.0-hadoop2.7.4-java8

container\_name: namenode

environment:

- CLUSTER\_NAME=test-cluster

volumes:

- namenode:/hadoop/dfs/name

- ./config/namenode/core-site.xml:/etc/hadoop/core-site.xml

- ./config/namenode/hdfs-site.xml:/etc/hadoop/hdfs-site.xml

- ./config/namenode/yarn-site.xml:/etc/hadoop/yarn-site.xml

ports:

- 9870:9870

- 9000:9000

networks:

- hadoop

datanode:

image: bde2020/hadoop-datanode:2.0.0-hadoop2.7.4-java8

container\_name: datanode

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

- CLUSTER\_NAME=test-cluster

volumes:

- datanode:/hadoop/dfs/data

networks:

- hadoop

resourcemanager:

image: bde2020/hadoop-resourcemanager:2.0.0-hadoop2.7.4-java8

container\_name: resourcemanager

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

volumes:

- ./config/resourcemanager/yarn-site.xml:/etc/hadoop/yarn-site.xml

ports:

- 8088:8088

networks:

- hadoop

nodemanager:

image: bde2020/hadoop-nodemanager:2.0.0-hadoop2.7.4-java8

container\_name: nodemanager

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

volumes:

- ./config/nodemanager/yarn-site.xml:/etc/hadoop/yarn-site.xml

networks:

- hadoop

depends\_on:

- resourcemanager

historyserver:

image: bde2020/hadoop-historyserver:2.0.0-hadoop2.7.4-java8

container\_name: historyserver

environment:

- CORE\_CONF\_fs\_defaultFS=hdfs://namenode:9000

- YARN\_CONF\_yarn\_resourcemanager\_hostname=resourcemanager

- YARN\_CONF\_yarn\_resourcemanager\_address=resourcemanager:8050

networks:

- hadoop

volumes:

namenode:

datanode:

networks:

hadoop:

#### Khởi **động docker compose**

docker compose up -d --build

Output

🍎 ❯ docker compose up -d --build

[+] Running 5/5

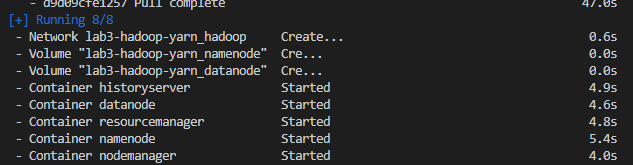
✔ Container resourcemanager Started 0.2s

✔ Container namenode Started 0.2s

✔ Container historyserver Started 0.2s

✔ Container datanode Started 0.2s

✔ Container nodemanager Started 0.4s

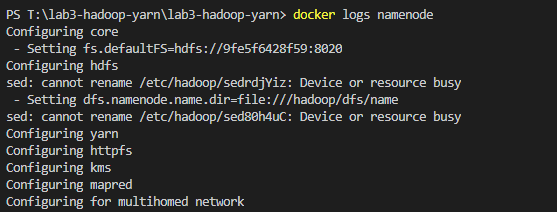


#### Kiểm tra log từng service bằng lệnh

docker logs <container\_name>

ví dụ xem log ở container *namenode*

docker logs namenode



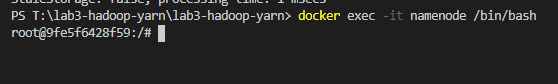
#### Kiểm tra cluster hadoop

* + **NameNode UI:** [http://localhost:9870](http://localhost:9870/)
  + **ResourceManager UI:** http://localhost:8088

1. **Một số lệnh yarn cơ bản**

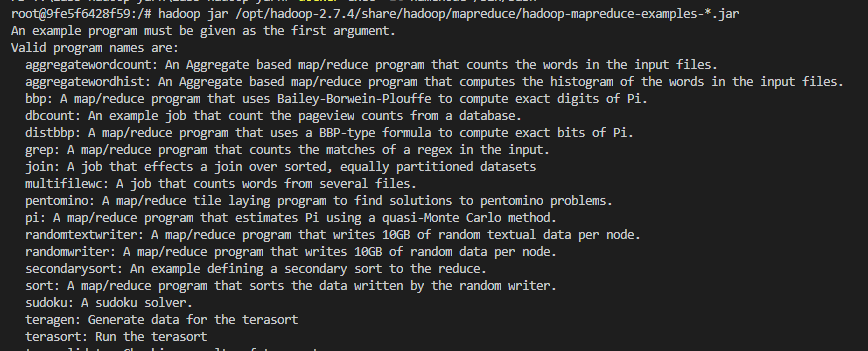
#### Truy cập vào cluser bằng lệnh:

docker exec -it namenode /bin/bash



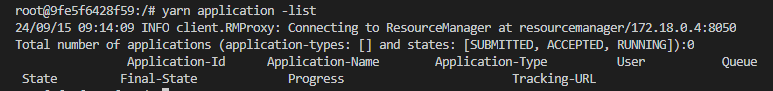
#### Danh sách các ứng dụng được cài đặt sẵn:

hadoop jar /opt/hadoop-2.7.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar



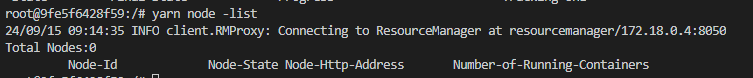
#### Hiển thị các ứng dụng đang chạy và đã hoàn thành.

yarn application –list



#### Xem danh sách các node

yarn node –list



1. **WordCount**

#### Tạo thư mục lab2

mkdir lab3 && cd lab3



#### Tạo file input như sau và lưu với tên iuh.txt

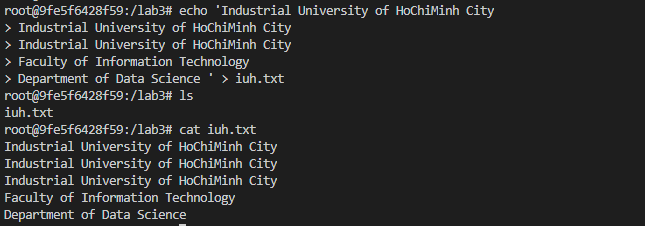
echo 'Industrial University of HoChiMinh City

Industrial University of HoChiMinh City

Industrial University of HoChiMinh City

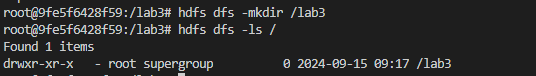
Faculty of Information Technology

Department of Data Science ' > iuh.txt



#### Bài tập:

#### Trên dfs tạo 1 thư mục tên lab3 và tải file iuh.txt vào thư mục đó

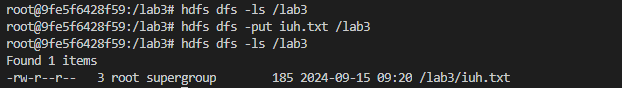


#### Kiểm tra file trên dfs

root@2bb70ff7de7f:/lab2# hdfs dfs -ls /lab2

Found 1 items

-rw-r--r-- 3 root supergroup 190 2024-09-03 07:41 /lab2/iuh.txt

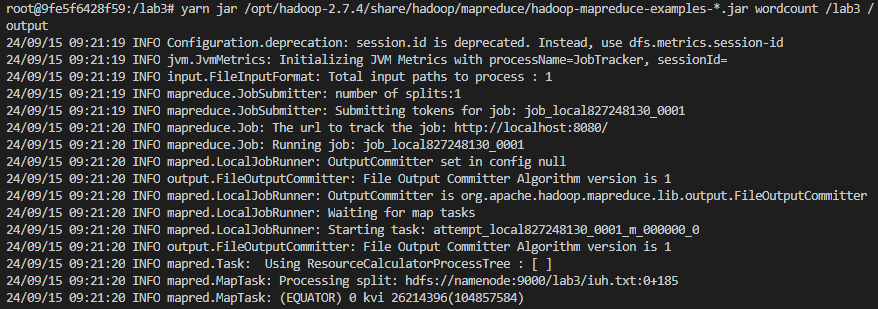
****

#### Chạy ứng dụng workcout được cài sẵn và xem kết quả

yarn jar <JARs> <application\_name> <input\_dir> <output\_dir>

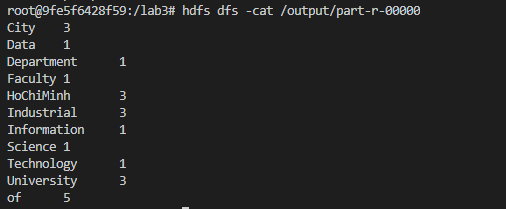
ví dụ run với ứng dụng workcout

yarn jar /opt/hadoop-2.7.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar wordcount /lab2 /output



#### Kiểm tra kết quả tại thư mục output

root@2bb70ff7de7f:/lab2# hdfs dfs -cat /output/part-r-00000



1. **Sudoku**

#### Chuẩn bị input có tên ‘sudoku\_input.txt' với nội dung như sau

echo '8 5 ? 3 9 ? ? ? ?

? ? 2 ? ? ? ? ? ?

? ? 6 ? 1 ? ? ? 2

? ? 4 ? ? 3 ? 5 9

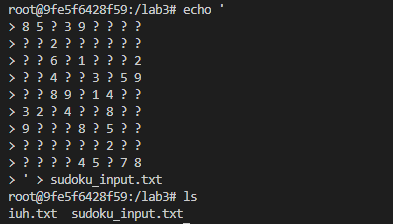
? ? 8 9 ? 1 4 ? ?

3 2 ? 4 ? ? 8 ? ?

9 ? ? ? 8 ? 5 ? ?

? ? ? ? ? ? 2 ? ?

? ? ? ? 4 5 ? 7 8' > sudoku\_input.txt



#### Hãy dùng ứng dụng sudoku để giải bài toán trên.cat

yarn jar /opt/hadoop-2.7.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar sudoku /lab3/sudoku\_input.txt

Output

