**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐH TÀI NGUYÊN & MÔI TRƯỜNG TPHCM**

**KHOA HỆ THỐNG THÔNG TIN & VIỄN THÁM**



**BIG DATA**

**ĐỒ ÁN: BÀI TOÁN MOVIES RECOMMENDATION SỬ DỤNG ELASTICSEARCH VÀ APACHE SPARK**

**NHÓM:** 1

**LỚP:** 10ĐH\_CNTT2

***TP.HCM, Ngày 5 Tháng 11 Năm 2023***

MỤC LỤC

[LỜI MỞ ĐẦU 4](#_Toc150120678)

[CHƯƠNG 1: GIỚI THIỆU 6](#_Toc150120679)

[1. Thành viên nhóm 6](#_Toc150120680)

[2. Lý do lựa chọn đề tài 7](#_Toc150120681)

[CHƯƠNG 2: CÁC CÔNG NGHỆ 8](#_Toc150120682)

[1. Tổng quan về Big Data 8](#_Toc150120683)

[1.1. Khái niệm 8](#_Toc150120684)

[1.2. Đặc trưng 9](#_Toc150120685)

[1.3. Ứng dụng 10](#_Toc150120686)

[2. APACHE HADOOP 10](#_Toc150120687)

[2.1. Khái niệm 10](#_Toc150120688)

[2.2. Kiến trúc 11](#_Toc150120689)

[2.3. Vai trò của Hadoop trong Big Data 12](#_Toc150120690)

[3. APACHE SPARK 13](#_Toc150120691)

[3.1. Khái niệm 13](#_Toc150120692)

[3.2. Các thành phần của Apache Spark 13](#_Toc150120693)

[3.3. Kiến trúc của Apache Spark 15](#_Toc150120694)

[3.4. Vai trò của Apache Spark trong Bài toán Movies Recommendation 16](#_Toc150120695)

[4. ELASTICSEARCH 17](#_Toc150120696)

[4.1. Khái niệm 17](#_Toc150120697)

[4.2. Các thành phần cần biết về Elasticsearch 17](#_Toc150120698)

[2.4.1. Document 17](#_Toc150120699)

[2.4.2. Index 18](#_Toc150120700)

[2.4.3. Shard 18](#_Toc150120701)

[2.4.4. Node 18](#_Toc150120702)

[2.4.5. Cluster 18](#_Toc150120703)

[4.3. Elasticsearch hoạt động như thế nào 19](#_Toc150120704)

[4.4. Vai trò trong bài toán Movies Recommendation 19](#_Toc150120705)

[CHƯƠNG 3: CÀI ĐẶT VÀ THỰC NGHIỆM 20](#_Toc150120706)

[1. CÀI ĐẶT CÁC CÔNG NGHỆ 20](#_Toc150120707)

[1.1. Cài đặt Java JDK 1.8 20](#_Toc150120708)

[1.2. Cài đặt Hadoop 22](#_Toc150120709)

[1.3. Cài đặt Apache Spark 26](#_Toc150120710)

[1.4. Cài đặt Elasticsearch 28](#_Toc150120711)

[1.5. Cài đặt Python 29](#_Toc150120712)

[2. THỰC NGHIỆM 30](#_Toc150120713)

[2.1. Thu thập dữ liệu và Tiền xử lý dữ liệu 30](#_Toc150120714)

[2.2. Kết nối và tải dữ liệu vào Elasticsearch 40](#_Toc150120715)

[2.3. Đào tạo một mô hình đề xuất về dữ liệu xếp hạng 47](#_Toc150120716)

[2.4. Sử dụng recommend của Elasticsearch 49](#_Toc150120717)

[3. ĐÁNH GIÁ VÀ NHẬN XÉT 56](#_Toc150120718)

[CHƯƠNG 4: KẾT LUẬN 57](#_Toc150120719)

[TÀI LIỆU THAM KHẢO 58](#_Toc150120720)

LỜI MỞ ĐẦU

Trong thời đại số hóa, ngành công nghiệp điện ảnh đã trải qua một cuộc cách mạng đáng kể. Khả năng truy cập đa dạng về phim, chương trình truyền hình và nội dung trực tuyến đã khiến cho việc tìm kiếm và gợi ý phim trở nên quan trọng hơn bao giờ hết. Người dùng mong muốn có trải nghiệm xem phim cá nhân hóa và dựa trên sở thích của họ.

Trong bài toán gợi ý phim (movies recommendation), chúng ta đã chứng kiến sự phát triển đáng kể của các công nghệ như Apache Spark và Elasticsearch. Apache Spark, một hệ thống xử lý dữ liệu phân tán mạnh mẽ, cho phép xử lý và phân tích dữ liệu lớn một cách hiệu quả. Elasticsearch, một hệ thống tìm kiếm và phân loại dữ liệu, giúp chúng ta tìm kiếm thông tin một cách nhanh chóng và hiệu quả.

Bài toán gợi ý phim sử dụng Apache Spark và Elasticsearch là một nỗ lực để cung cấp cho người dùng trải nghiệm xem phim tốt nhất dựa trên sở thích cá nhân của họ. Chúng ta sẽ kết hợp sức mạnh của Apache Spark để xử lý và phân tích dữ liệu xem phim lớn và phức tạp. Sau đó, chúng ta sẽ sử dụng Elasticsearch để tạo một hệ thống gợi ý phim mạnh mẽ, có khả năng tìm kiếm nhanh và phản hồi dựa trên sở thích của người dùng.

Trong bài toán này, chúng ta sẽ khám phá quá trình thu thập, xử lý dữ liệu, xây dựng mô hình gợi ý và triển khai hệ thống gợi ý phim sử dụng Apache Spark và Elasticsearch. Chúng ta hy vọng rằng thông qua việc sử dụng các công nghệ này, chúng ta sẽ tạo ra một hệ thống gợi ý phim mạnh mẽ và mang lại trải nghiệm xem phim tốt nhất cho người dùng.

Bài tiểu luận của nhóm sẽ mô tả lại chi tiết từng bước của quá trình, bắt đầu từ việc thu thập và xử lý dữ liệu phim, đến việc xây dựng mô hình gợi ý, và cuối cùng triển khai hệ thống gợi ý phim dựa trên Apache Spark và Elasticsearch.

1. GIỚI THIỆU
   1. Thành viên nhóm

|  |  |  |
| --- | --- | --- |
| **Tên đề tài** | **Tích hợp Elasticsearch, Apache Spark cho bài toán Movie Recommendation**  [**https://www.youtube.com/watch?v=iy4TX\_xLSNQ&feature=youtu.be**](https://www.youtube.com/watch?v=iy4TX_xLSNQ&feature=youtu.be)  [**https://github.com/truonghb29/movie\_recommend**](https://github.com/truonghb29/movie_recommend) | **Đóng góp của thành viên** |
| **Họ và tên thành viên 1** | **Nguyễn Thanh Trường – 1050080082** | Đóng góp:  Điểm tự đánh giá quá trình:8  Điểm tự đánh giá cuối kỳ: |
| **Họ và tên thành viên 2** | **Cao Đức Minh – 1050080062** | Đóng góp:  Điểm tự đánh giá quá trình:  Điểm tự đánh giá cuối kỳ: |
| **Họ và tên thành viên 3** | **Phạm Thế Anh – 1050080043** | Đóng góp:  Điểm tự đánh giá quá trình:  Điểm tự đánh giá cuối kỳ: |

* 1. Lý do lựa chọn đề tài

Tính ứng dụng cao: Gợi ý phim có tính ứng dụng cao trong cuộc sống hàng ngày. Nó không chỉ giúp người dùng tìm kiếm các bộ phim mới mà họ có thể thích mà còn có thể được sử dụng trong các nền tảng xem phim trực tuyến và ứng dụng di động.

Sự phát triển của ngành công nghiệp phim: Ngành công nghiệp phim đang phát triển nhanh chóng với sự gia tăng của nền tảng xem phim trực tuyến. Các dịch vụ như Netflix, Amazon Prime, và Disney+ đều đang tập trung vào cung cấp trải nghiệm xem phim cá nhân hóa.

Sở thích cá nhân: Nếu bạn đam mê xem phim và muốn cải thiện khả năng gợi ý phim dựa trên sở thích cá nhân, đề tài này có thể đem lại niềm vui trong quá trình nghiên cứu.

Sự kết hợp của khoa học dữ liệu và máy học: Gợi ý phim thường kết hợp khoa học dữ liệu và máy học để xác định sở thích của người dùng và đưa ra các đề xuất. Nghiên cứu về movies recommendation có thể giúp bạn học và áp dụng những kỹ năng này.

Dữ liệu phong phú: Có sẵn một lượng lớn dữ liệu về phim, xếp hạng, lượt xem và sở thích của người dùng. Điều này cung cấp một cơ hội tốt để thực hiện nghiên cứu thực tế và kiểm tra các mô hình gợi ý.

Áp dụng trong thực tế: Các hệ thống gợi ý phim đã được triển khai rộng rãi trong thực tế, từ các dịch vụ xem phim trực tuyến đến các ứng dụng di động. Nghiên cứu về movies recommendation có thể mang lại đóng góp thiết thực cho cộng đồng người dùng.

Phù hợp với hướng nghiên cứu: Nếu bạn đang theo đuổi hướng nghiên cứu trong lĩnh vực khoa học dữ liệu, trí tuệ nhân tạo hoặc máy học, đề tài movies recommendation có thể phù hợp với hướng đi của bạn.

1. CÁC CÔNG NGHỆ
   1. Tổng quan về Big Data
      1. Khái niệm

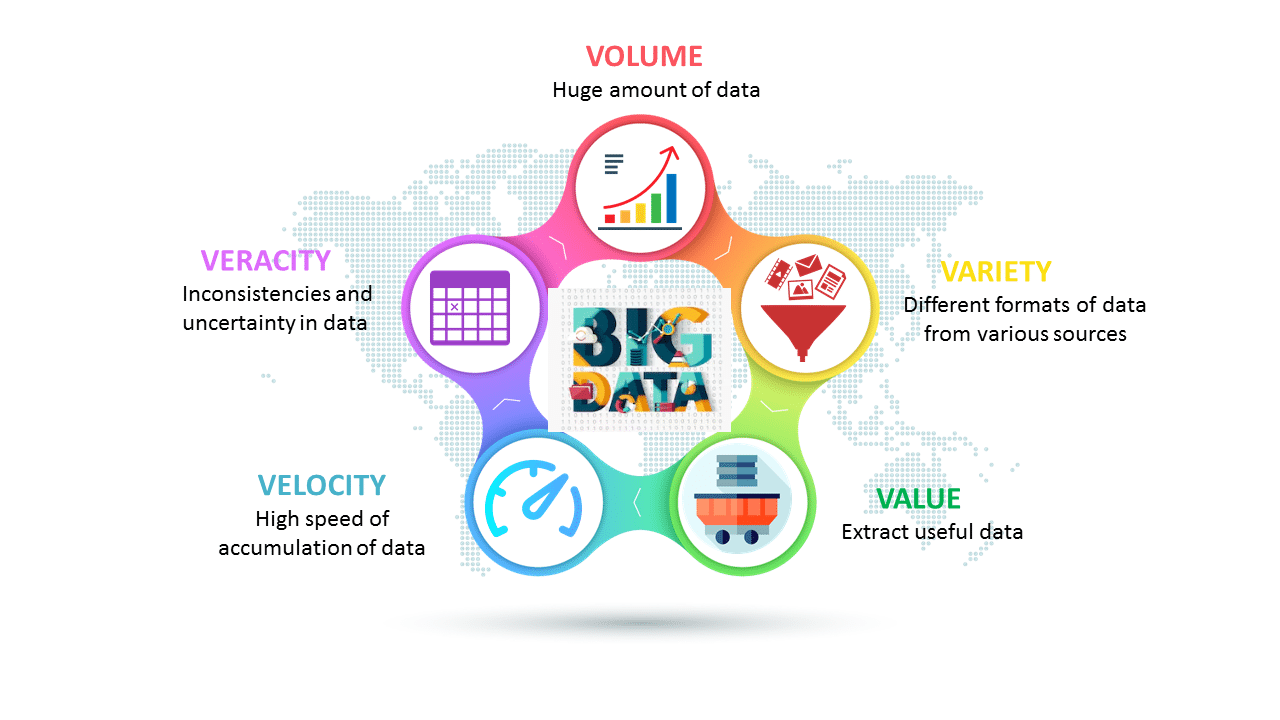
Dữ liệu lớn là một thuật ngữ cho việc xử lý một tập hợp dữ liệu rất lớn và phức tạp mà các ứng dụng xử lý dữ liệu truyền thống không xử lý được. Dữ liệu lớn bao gồm các thách thức như phân tích, thu thập, giám sát dữ liệu, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư. Thuật ngữ này thường chỉ đơn giản đề cập đến việc việc sử dụng các phân tích dự báo, phân tích hành vi người dùng, hoặc một số phương pháp phân tích dữ liệu tiên tiến khác trích xuất giá trị từ dữ liệu mà ít khi đề cập đến kích thước của bộ dữ liệu.



Big Data - Dữ liệu lớn thường bao gồm tập hợp dữ liệu với kích thước vượt xa khả năng của các công cụ phần mềm thông thường để thu thập, hiển thị, quản lý và xử lý dữ liệu trong một thời gian có thể chấp nhận được. Kích thước dữ liệu lớn là một mục tiêu liên tục thay đổi. Như năm 2012 thì phạm vi một vài tá terabytes tới nhiều petabytes dữ liệu. Dữ liệu lớn yêu cầu một tập các kỹ thuật và công nghệ được tích hợp theo hình thức mới để khai phá từ tập dữ liệu đa dạng, phức tạp, và có quy mô lớn.

* + 1. Đặc trưng

Về sự đặc trưng, Big Data không chỉ đặc trưng bởi khối lượng dữ liệu lớn mà còn bởi tính đa dạng của dữ liệu, tốc độ nhanh, giá trị tiềm năng và thách thức về tính đáng tin cậy và xử lý. Được thể hiện qua 5 yếu tố chính - gọi là “5Vs”:



Volume (Khối lượng): Đề cập đến lượng lớn dữ liệu được tạo ra hoặc tích luỹ. Big Data thường xuất phát từ sự tích hợp của nhiều nguồn khác nhau và có thể nằm trong khoảng từ terabytes (TB) đến petabytes (PB) hoặc thậm chí exabytes (EB) dữ liệu.

Velocity (Tốc độ): Liên quan đến tốc độ mà dữ liệu được tạo ra, truyền tải và cập nhật. Big Data thường được tạo ra liên tục và có thể cần xử lý dữ liệu trong thời gian thực hoặc gần thời gian thực.

Variety (Đa dạng): Variety đề cập đến tính đa dạng của dữ liệu trong Big Data. Dữ liệu có thể tồn tại dưới nhiều định dạng và có thể là dữ liệu có cấu trúc (ví dụ: cơ sở dữ liệu SQL), dữ liệu không cấu trúc (ví dụ: văn bản), hoặc dữ liệu bán cấu trúc (ví dụ: JSON).

Veracity (Sự xác thực): Nói đến sự xác thực và độ tin cậy của dữ liệu. Big Data có thể chứa dữ liệu không chính xác, không đầy đủ hoặc có lỗi. Việc kiểm tra và đảm bảo sự xác thực của dữ liệu là quan trọng.

Value (Giá trị): Là về giá trị tiềm năng của dữ liệu. Sử dụng phân tích dữ liệu, người ta có thể trích xuất thông tin giá trị và kiến thức từ dữ liệu để đưa ra quyết định thông minh hoặc tạo ra giá trị kinh doanh.

* + 1. Ứng dụng

Big Data có sự hiện diện trong hầu hết các khía cạnh của cuộc sống hiện đại, từ công nghiệp đến giáo dục, giải trí và nhiều lĩnh vực khác và có nhiều ứng dụng quan trọng trong đời sống hàng ngày của chúng ta. Dưới đây là một số ví dụ về cách Big Data được áp dụng để cải thiện và tối ưu hóa nhiều mặt của cuộc sống:

Dự báo thời tiết: Big Data giúp các dự đoán thời tiết trở nên chính xác hơn thông qua việc thu thập và phân tích dữ liệu từ hàng ngàn cảm biến và nguồn dữ liệu khác nhau. Điều này giúp cảnh báo kịch bản thời tiết xấu và cung cấp thông tin dự đoán thời tiết chi tiết.

Y tế và chăm sóc sức khỏe: Big Data đóng vai trò quan trọng trong phát triển các ứng dụng y tế và dịch vụ chăm sóc sức khỏe thông minh. Nó giúp phân tích dữ liệu y tế để đưa ra chẩn đoán, phát hiện bệnh và theo dõi sức khỏe cá nhân.

Giao thông thông minh: Big Data được sử dụng để cải thiện quản lý giao thông, dự đoán tắc nghẽn và cung cấp thông tin về tình trạng đường và tuyến giao thông.

Dự đoán nguồn cung cấp và cầu cảng: Big Data giúp các công ty và cơ quan quản lý dự đoán nhu cầu tiêu dùng và quản lý nguồn cung cấp một cách hiệu quả.

Quảng cáo và tiếp thị: Các công ty tiếp thị sử dụng dữ liệu lớn để hiểu khách hàng hơn và tạo các chiến dịch quảng cáo được tùy chỉnh.

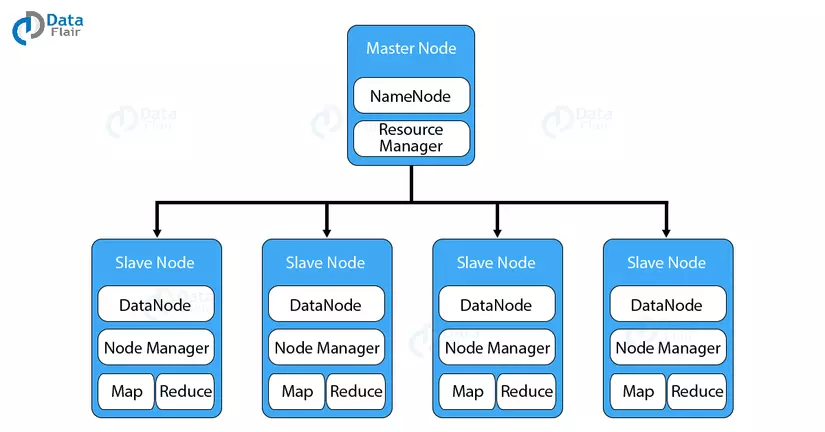
Chăm sóc khách hàng: Big Data giúp các doanh nghiệp hiểu rõ nhu cầu của khách hàng và cung cấp dịch vụ hỗ trợ khách hàng tốt hơn.

Dự đoán thị trường tài chính: Big Data được sử dụng để dự đoán biến động thị trường tài chính và quản lý rủi ro đầu tư.

An ninh và phòng ngừa tội phạm: Big Data hỗ trợ việc theo dõi và phân tích dữ liệu về an ninh để phát hiện và ngăn ngừa tội phạm.

* 1. APACHE HADOOP
     1. Khái niệm

Hadoop là một Apache framework mã nguồn mở cho phép phát triển các ứng dụng phân tán (distributed processing) để lưu trữ và quản lý các tập dữ liệu lớn. Hadoop hiện thực mô hình MapReduce, mô hình mà ứng dụng sẽ được chia nhỏ ra thành nhiều phân đoạn khác nhau được chạy song song trên nhiều node khác nhau. Hadoop được viết bằng Java tuy nhiên vẫn hỗ trợ C++, Python, Perl bằng cơ chế streaming.

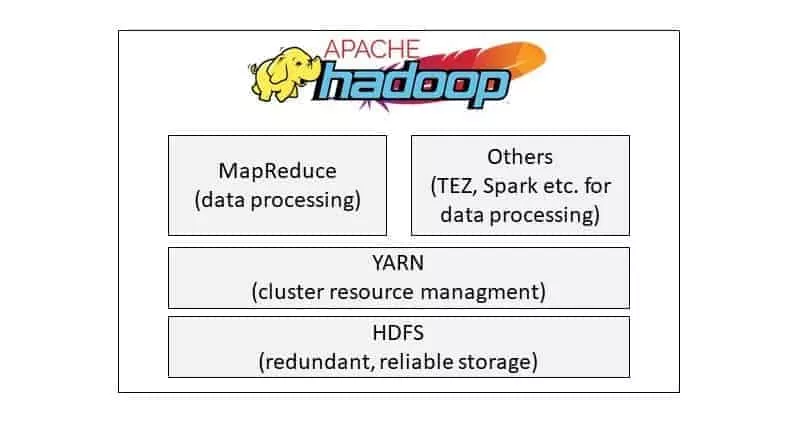


Hadoop có một cấu trúc liên kết master-slave. Trong cấu trúc này, chúng ta có một node master và nhiều node slave . Chức năng của node master là gán một tác vụ cho các node slave khác nhau và quản lý tài nguyên. Các node slave là máy tính thực tế có thể không mạnh lắm. Các node slave lưu trữ dữ liệu thực trong khi trên master chúng ta có metadata.

* + 1. Kiến trúc

Kiến trúc Hadoop gồm có ba lớp chính đó là:

HDFS (Hadoop Distributed File System): HDFS là hệ thống tệp phân tán của Hadoop, được sử dụng để lưu trữ dữ liệu lớn trên nhiều máy chủ. Dữ liệu được chia thành các khối nhỏ và phân phát trên nhiều máy chủ để đảm bảo khả năng mở rộng và độ tin cậy. HDFS là một phần quan trọng của cấu trúc Hadoop và được sử dụng để lưu trữ dữ liệu nguyên bản.



Map-Reduce: MapReduce là mô hình lập trình và framework xử lý dữ liệu lớn trong Hadoop. Nó chia công việc xử lý thành hai phần: phần Map, nơi dữ liệu được xử lý và ánh xạ thành cặp key-value, và phần Reduce, nơi các cặp key-value được tổng hợp và phân tích. Điều này cho phép xử lý song song và phân tán dữ liệu.

Yarn: YARN là một bộ phận quản lý tài nguyên trong Hadoop. Nó quản lý và cấp phát tài nguyên cho các ứng dụng Hadoop chạy trên cluster. YARN giúp cải thiện hiệu suất và quản lý tài nguyên hiệu quả hơn.

* + 1. Vai trò của Hadoop trong Big Data

Cấu trúc của Hadoop cho phép xử lý và lưu trữ dữ liệu lớn một cách hiệu quả, đáp ứng nhu cầu ngày càng tăng của việc quản lý và phân tích dữ liệu lớn. Các thành phần này làm cho Hadoop trở thành một framework mạnh mẽ và đóng nhiều vai trò quan trọng trong lĩnh vực Big Data như là:

Lưu trữ Dữ liệu Lớn: Hadoop Distributed File System (HDFS) cho phép lưu trữ và quản lý dữ liệu lớn một cách hiệu quả. Dữ liệu được chia thành các khối nhỏ và phân tán trên nhiều máy chủ, giúp đảm bảo khả năng mở rộng và độ tin cậy của hệ thống.

Xử lý Dữ liệu Lớn: Hadoop cung cấp mô hình lập trình MapReduce, cho phép xử lý và phân tích dữ liệu lớn một cách phân tán và song song. Điều này giúp giải quyết các công việc tính toán phức tạp trên dữ liệu lớn.

Khả năng Mở Rộng: Hadoop dễ dàng mở rộng theo nhu cầu bằng cách thêm máy chủ vào cluster. Điều này cho phép xử lý dữ liệu lớn với khối lượng dữ liệu ngày càng tăng mà không cần thay đổi kiến trúc hệ thống.

Hệ sinh thái Hadoop (Hadoop Ecosystem): Hadoop có một hệ sinh thái với nhiều công cụ và framework bổ sung như Hive, Pig, Spark, HBase, Kafka và nhiều thành phần khác. Các công cụ này mở rộng khả năng xử lý và phân tích dữ liệu lớn, cho phép ứng dụng đa dạng và đáp ứng các nhu cầu cụ thể.

Trí Tuệ Nhân Tạo và Machine Learning: Hadoop kết hợp với các framework như Apache Spark để xử lý và phân tích dữ liệu lớn cho các ứng dụng trí tuệ nhân tạo và machine learning. Điều này bao gồm việc xây dựng mô hình dự đoán, học máy và xử lý ngôn ngữ tự nhiên.

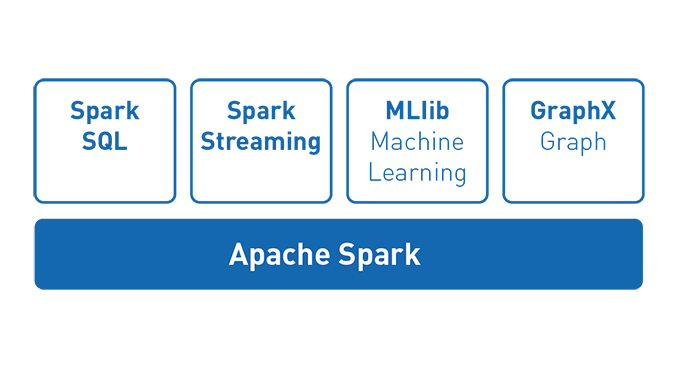
* 1. APACHE SPARK
     1. Khái niệm

Apache Spark là framework mã nguồn mở tính toán cụm được thiết kế để cung cấp tốc độ tính toán, khả năng mở rộng (scalability) và khả năng lập trình cho Big data. Được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Đặc biệt, trong quá trình truyền một luồng dữ liệu liên tục (data streaming) với tốc độ cao, cơ sở dữ liệu đồ thị (graph data), máy học (machine learning) và ứng dụng trí tuệ nhân tạo (AI) .

Apache Spark được xem là công cụ xử lý dữ liệu nhanh gấp 10 lần so với các công cụ khác. Nó cung cấp một cơ chế tính toán phân tán trên một cụm máy tính (Computer Cluster) với hiệu suất cao hơn so với Apache Hadoop đặc biệt MapReduce, đồng thời hỗ trợ nhiều ngôn ngữ lập trình như Scala, Java, Python,...

* + 1. Các thành phần của Apache Spark

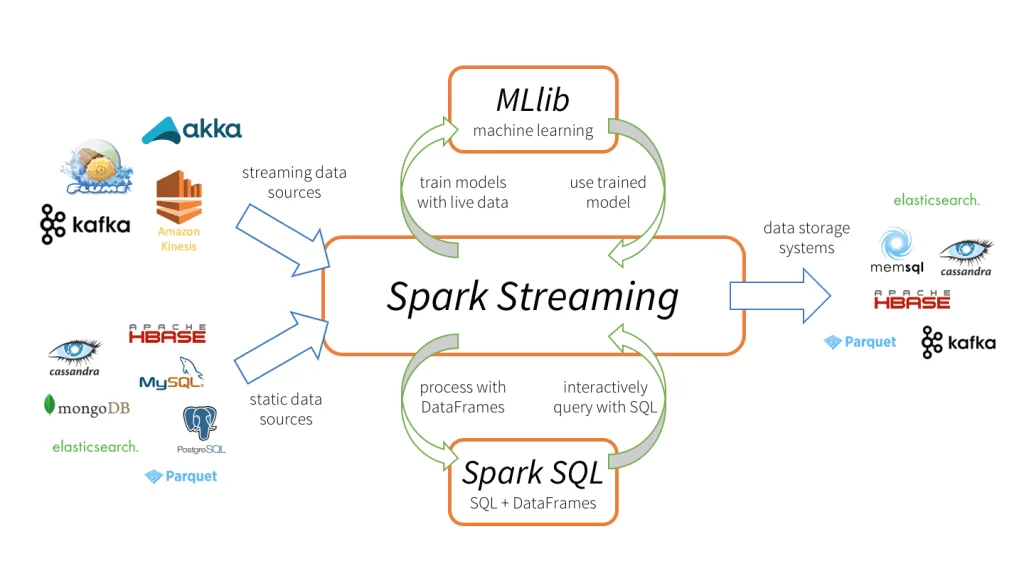
Apache Spark là một framework phân tán cho xử lý dữ liệu lớn và phân tích dữ liệu, và nó bao gồm nhiều thành phần quan trọng để hỗ trợ quá trình xử lý và tính toán dữ liệu. Dưới đây là các thành phần chính của Apache Spark:

****

Spark Core: là thành phần cốt lõi của Apache Spark, các thành phần khác muốn hoạt động đều cần thông qua Spark Core. Cung cấp các chức năng cơ bản để xử lý dữ liệu phân tán (distributed data processing - DDP) bao gồm quản lý bộ nhớ, thiết lập lịch tác vụ và khôi phục lỗi.

Spark SQL: cho phép người dùng truy vấn dữ liệu bằng ngôn ngữ SQL trên các tập dữ liệu lớn. Spark SQL cung cấp một cơ chế để tích hợp với các cơ sở dữ liệu quan hệ như MySQL, Oracle hoặc PostgreSQL và xử lý dữ liệu bằng cách sử dụng các tính năng phân tán của Spark. Ngoài ra, Spark SQL còn tích hợp tốt với các công cụ ETL và Spark Streaming, điều đó giúp nó trở thành một phần quan trọng trong hệ thống phân tích dữ liệu phân tán và các ứng dụng trực tuyến trong thời gian thực.

Spark Streaming: là một module cho phép xử lý dữ liệu trực tiếp trong thời gian thực (real-time). Spark Streaming cung cấp một cơ chế xử lý dữ liệu liên tục (stream processing) bằng cách chia nhỏ dữ liệu thành một chuỗi các microbatch nhỏ hơn và xử lý chúng thông qua API Apache Spark. Bên cạnh đó, nó còn tích hợp với các module khác của Apache Spark như Spark SQL và MLlib. Spark Streaming được sử dụng rộng rãi cho các ứng dụng như phân tích dữ liệu trực tuyến, phát hiện sự cố hệ thống và phân tích dữ liệu giám sát trong thời gian thực.



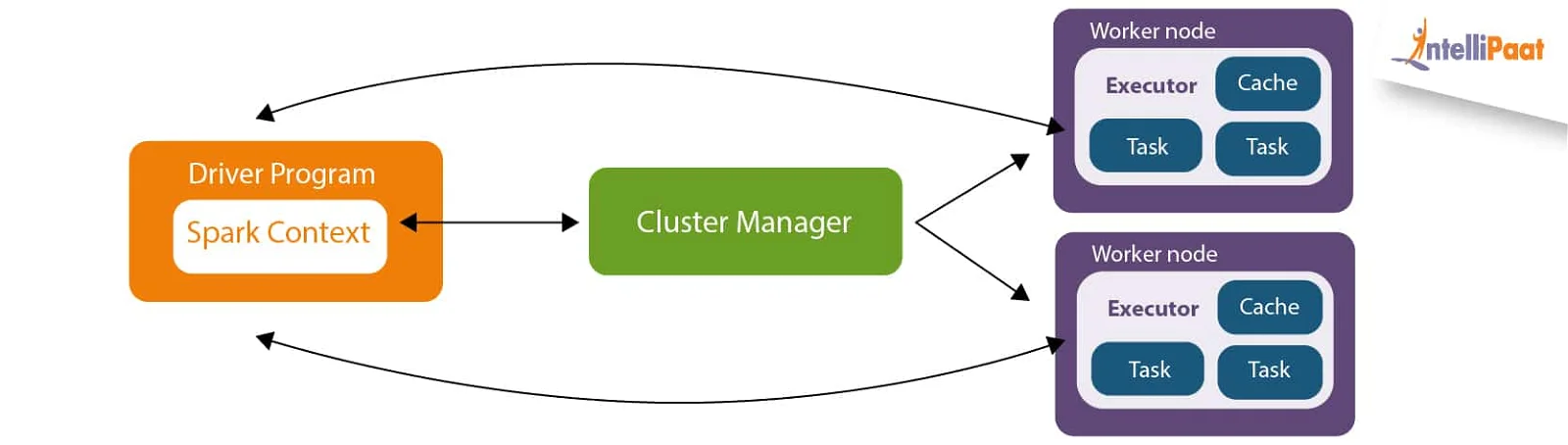
Spark MLlib: là thư viện Machine Learning được tích hợp sẵn trong Apache Spark, cung cấp các thuật toán Machine Learning phổ biến giúp ích trong việc xử lý big data. MLlib cung cấp các thuật toán ML phổ biến như: Regression, Classification, Clustering, Collaborative Filtering, Dimensionality Reduction, Feature Extraction and Transformation,... Spark MLlib cũng tích hợp với các module khác của Apache Spark nhằm đáp ứng nhu cầu phân tích và xử lý dữ liệu lớn.

GraphX: đi kèm với lựa chọn các thuật toán phân tán để xử lý cấu trúc đồ thị. Để thực hiện các tính toán trên đồ thị, GraphX cung cấp RDD, VertexRDD và EdgeRDD trong đó Vertex (đỉnh) và Edge (cạnh).

* + 1. Kiến trúc của Apache Spark

Driver Program trong kiến ​​trúc Apache Spark sẽ quản lý, điều khiển quá trình xử lý trên cluster và tạo Spark Context. Spark Context sẽ bao gồm tất cả các chức năng cơ bản.

Spark Driver là một thành phần quan trọng trong kiến trúc của Apache Spark. Nó là chương trình chính của ứng dụng Spark, chạy trên một node trong cluster, quản lý và điều khiển quá trình xử lý trên cluster. Spark Driver và Spark Context cùng giám sát quá trình thực thi cluster. Spark Driver hoạt động với Cluster Manage để quản lý nhiều công việc khác.

****

Cluster Manage giúp quản lý và phân phối tài nguyên trên các node của cluster, sau đó sẽ phân phối và giám sát tiến trình xử lý trên các node này để đảm bảo chúng hoạt động hiệu quả.

Bất cứ khi nào cấu trúc dữ liệu bất biến và phân tán (Resilient Distributed Datasets - RDD) được tạo trong Spark Context, thì nó sẽ được phân phối trên nhiều worker nodes và được cached ở đó. Các worker nodes sẽ thực thi nhiệm vụ được giao bởi Cluster Manager và sau đó đưa nó trở lại Spark Context.

Executors sẽ lên tiến trình chạy trên các node khác trong cluster, được quản lý bởi driver program để thực hiện các nhiệm vụ xử lý dữ liệu. Mỗi Executor có thể chứa nhiều task (công việc) được giao để xử lý.

* + 1. Vai trò của Apache Spark trong Bài toán Movies Recommendation

Trong bài toán Movies Recommendation sử dụng Apache Spark và Elasticsearch, Apache Spark đóng một vai trò quan trọng trong việc xử lý dữ liệu, phân tích và xây dựng hệ thống gợi ý phim.

Xử Lý Dữ Liệu Lớn: Dữ liệu về phim, đánh giá của người dùng và các thông tin liên quan đến bài toán gợi ý có thể lớn và phức tạp. Apache Spark có khả năng xử lý dữ liệu lớn một cách hiệu quả trên nền tảng phân tán, cho phép nắm bắt, tiền xử lý và chuẩn bị dữ liệu cho việc gợi ý.

Xử Lý Dữ Liệu Cấu Trúc và Bán Cấu Trúc: Dữ liệu về phim và người dùng thường là dạng cấu trúc hoặc bán cấu trúc. Spark SQL cho phép thực hiện truy vấn, tổng hợp và biến đổi dữ liệu theo cách quen thuộc với SQL.

Machine Learning (MLlib): Để xây dựng mô hình gợi ý, Apache Spark MLlib cung cấp các thuật toán machine learning cho việc phân loại và học máy. Spark MLlib có khả năng xây dựng mô hình dựa trên lịch sử đánh giá của người dùng và thông tin về phim.

Xử Lý Dữ Liệu Thời Gian Thực: Apache Spark Streaming có thể sử dụng để xử lý dữ liệu thời gian thực, ví dụ, theo dõi hoạt động của người dùng để cung cấp gợi ý ngay lập tức.

Tích Hợp Với Elasticsearch: Elasticsearch là một hệ thống tìm kiếm và lưu trữ dữ liệu phân tán. Spark có khả năng tích hợp với Elasticsearch để truy vấn và trích xuất dữ liệu một cách hiệu quả. Dữ liệu về phim và thông tin gợi ý có thể được lưu trữ và truy vấn từ Elasticsearch.

* 1. ELASTICSEARCH
     1. Khái niệm

Elasticsearch là một công cụ tìm kiếm dựa trên nền tảng Apache Lucene. Nó cung cấp một bộ máy tìm kiếm dạng phân tán, có đầy đủ công cụ với một giao diện web HTTP có hỗ trợ dữ liệu JSON. Elasticsearch được phát triển bằng Java và được phát hành dạng nguồn mở theo giấy phép Apache.



Elasticsearch cung cấp cho người dùng một hệ thống tìm kiếm dạng phân tán, phù hợp với tệp dữ liệu lớn. Elasticsearch thường được sử dụng trong việc tìm kiếm đơn giản trên trang web, chỉ mục và thu thập dữ liệu, cho đến phân tích và trực quan hóa dữ liệu kinh doanh. Elasticsearch thường hoạt động với các thành phần khác trong ELK Stack, Logstash và Kibana, trong đó đóng vai trò lập chỉ mục dữ liệu.

* + 1. Các thành phần cần biết về Elasticsearch
       1. Document

Document là thông tin cơ bản được lập chỉ mục trong Elasticsearch, là đơn vị lưu trữ dữ liệu nhỏ nhất trong Elasticsearch. Document có thể là văn bản, hoặc có thể là bất kỳ dạng cấu trúc dữ liệu nào được mã hóa bằng JSON như số, chuỗi, ngày tháng,…

Mỗi Document có một ID duy nhất cho một kiểu dữ liệu duy nhất. Ví dụ, một Document có thể đại diện cho một bài báo bách khoa toàn thư.

* + - 1. Index

Index (Chỉ mục) là tập hợp các tài liệu có đặc điểm liên quan về mặt logic. Ví dụ: Trong trang web của một sàn thương mại điện tử, bạn sẽ tìm thấy một Chỉ mục cho Khách hàng, một Chỉ mục cho Sản phẩm,… Chỉ mục được sử dụng để tìm kiếm, thêm mới hoặc xóa các tài liệu trong đó.

Một khái niệm trong Index nên biết đó là Inverted Index (Chỉ mục đảo ngược). Chỉ mục đảo ngược không lưu trữ trực tiếp các chuỗi mà chia từng tài liệu thành các cụm từ tìm kiếm riêng lẻ. Nhờ đó, người dùng có thể tìm thấy các kết quả phù hợp nhanh chóng, kể cả trong các tệp dữ liệu với khối lượng lớn.

* + - 1. Shard

Shard là tập con các Document của 1 Index, là đơn vị lưu trữ dữ liệu nhỏ nhất, hoạt động ở mức thấp nhất. Bằng cách phân phối Documents trong một Index trên nhiều Shard, Elasticsearch có thể đảm bảo tính dự phòng, bảo vệ hệ thống khỏi lỗi phần cứng và giúp tăng khả năng truy vấn khi hoạt động.

Có 2 loại Shard là : Primary Shard và Replica Shard.

* Primary Shard được lưu trữ dữ liệu và đánh Index, sau đó được vận chuyển tới các Replica Shard.
* Replica Shard là nơi lưu trữ dữ liệu nhân bản của Primary Shard, giúp đảm bảo dữ liệu Primary Shard được toàn vẹn ngay cả khi hệ thống xảy ra vấn đề.
  + - 1. Node

Node được coi là bộ phận “đầu não” của Elasticsearch. Đây là nơi lưu trữ dữ liệu trực tiếp, đánh Index của Cluster và thực hiện các thao tác tìm kiếm dữ liệu. Mỗi Node thường được định danh bằng 1 một cái tên duy nhất.

* + - 1. Cluster

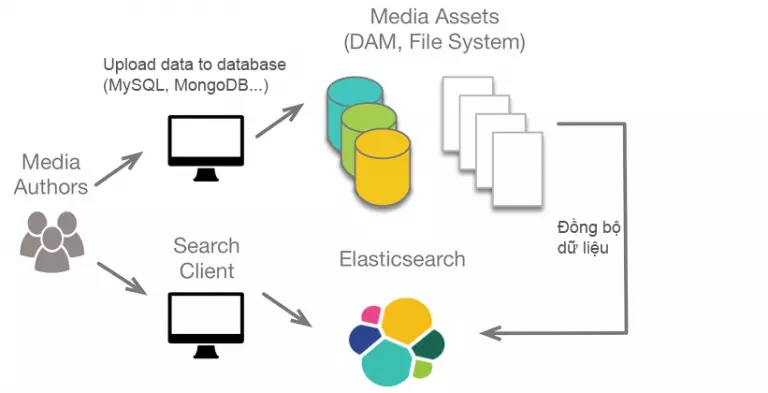
Tập hợp các nodes hoạt động cùng với nhau, chia sẽ cùng thuộc tính cluster.name. Chính vì thế Cluster sẽ được xác định bằng 1 'unique name'. Việc định danh các cluster trùng tên sẽ gây nên lỗi cho các node vì vậy khi setup các bạn cần hết sức chú ý điểm này

Mỗi cluster có một node chính (master), được lựa chọn một cách tự động và có thể thay thế nếu sự cố xảy ra. Một cluster có thể gồm 1 hoặc nhiều nodes. Các nodes có thể hoạt động trên cùng 1 server . Tuy nhiên trong thực tế , một cluster sẽ gồm nhiều nodes hoạt động trên các server khác nhau để đảm bảo nếu 1 server gặp sự cố thì server khác (node khác) có thể hoạt động đầy đủ chức năng so với khi có 2 servers. Các node có thể tìm thấy nhau để hoạt động trên cùng 1 cluster qua giao thức unicast.

Chức năng chính của Cluster đó chính là quyết định xem shards nào được phân bổ cho node nào và khi nào thì di chuyển các Cluster để cân bằng lại Cluster

* + 1. Elasticsearch hoạt động như thế nào

Về cơ bản, Elasticsearch hoạt động như một Cloud Server có khả năng tìm kiếm thông qua cơ chế RESTful. Trong đó, người dùng tạo ra các HTTP Request và dữ liệu dạng JSON, sau đó nhập vào Elasticsearch. Các dữ liệu này đều được đánh Index (Chỉ mục), giúp đem lại hiệu quả tìm kiếm rất cao.



Đầu tiên, dữ liệu được đưa vào Elasticsearch từ nhiều nguồn khác nhau, sau đó được phân tích, xử lý, trong quá trình nhập liệu. Tiếp theo, dữ liệu được phân loại và chỉ mục, được đẩy lên Server Elasticsearch. Cuối cùng, người dùng có thể tạo các truy vấn phức tạp và lấy dữ liệu được trả về từ server Elasticsearch.

* + 1. Vai trò trong bài toán Movies Recommendation

Elasticsearch được sử dụng kết hợp với Apache Spark đóng vai trò quan trọng trong việc lưu trữ, tìm kiếm, và gợi ý phim trong hệ thống Movies Recommendation, giúp cung cấp trải nghiệm tốt cho người dùng và cung cấp hiệu suất tìm kiếm cao.Cụ thể với các vai trò sau:

Lưu Trữ và Tìm Kiếm Dữ Liệu Phim và Người Dùng: Elasticsearch có thể được sử dụng để lưu trữ thông tin về các phim và người dùng. Dữ liệu về phim bao gồm thông tin chi tiết, ví dụ: tên phim, thể loại, diễn viên, đạo diễn, và điểm đánh giá. Dữ liệu về người dùng bao gồm lịch sử xem phim, đánh giá, sở thích và thông tin cá nhân, cho phép tìm kiếm dữ liệu này một cách hiệu quả.

Tạo Chỉ Số Tìm Kiếm Nhanh: Elasticsearch sử dụng cơ chế chỉ số (indexing) để tạo chỉ mục từ dữ liệu. Điều này giúp tìm kiếm trên dữ liệu nhanh chóng. Chẳng hạn, khi người dùng muốn tìm kiếm phim theo tiêu chí cụ thể như thể loại hoặc diễn viên, Elasticsearch có khả năng trả về kết quả một cách nhanh chóng.

Thực Hiện Gợi Ý Tìm Kiếm: Elasticsearch có khả năng thực hiện tìm kiếm gợi ý (recommendation search), dựa trên dữ liệu về lịch sử xem phim và đánh giá của người dùng, Elasticsearch có thể đề xuất các phim mà người dùng có thể quan tâm. Điều này tạo nên phần quan trọng của hệ thống Movies Recommendation.

Phản Hồi Nhanh và Trả Kết Quả Đáng Tin Cậy: Elasticsearch cung cấp hiệu suất tìm kiếm cao và cho phép hệ thống trả kết quả gợi ý một cách nhanh chóng và đáng tin cậy, cung cấp trải nghiệm người dùng tốt.

Kết Hợp Với Apache Spark: Elasticsearch có tích hợp với Apache Spark, cho phép trích xuất và cập nhật dữ liệu một cách dễ dàng từ Elasticsearch thông qua Spark. Điều này cho phép Spark xử lý dữ liệu, tính toán chỉ số gợi ý, và cập nhật dữ liệu trong Elasticsearch.

Đảm Bảo Tính Mở Rộng và Sẵn Sàng: Elasticsearch là một hệ thống phân tán, cho phép bạn mở rộng tài nguyên lưu trữ và tìm kiếm theo nhu cầu. Đồng thời, việc sử dụng các shard và replica shard giúp đảm bảo tính sẵn sàng và an toàn của dữ liệu.

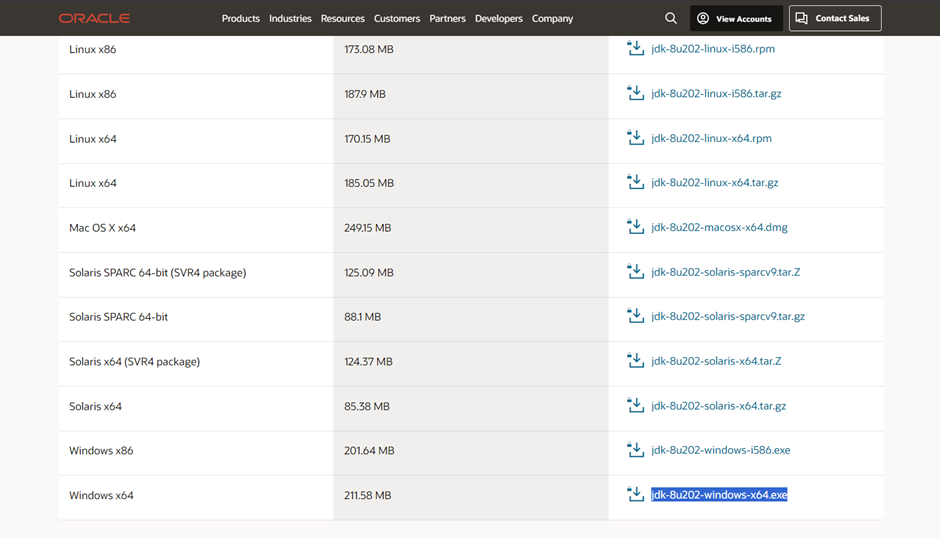
1. CÀI ĐẶT VÀ THỰC NGHIỆM
   1. CÀI ĐẶT CÁC CÔNG NGHỆ

<https://github.com/ptheanh412/BigData/tree/main/C%C3%A0i%20%C4%91%E1%BA%B7t%20c%C3%A1c%20c%C3%B4ng%20ngh%E1%BB%87>

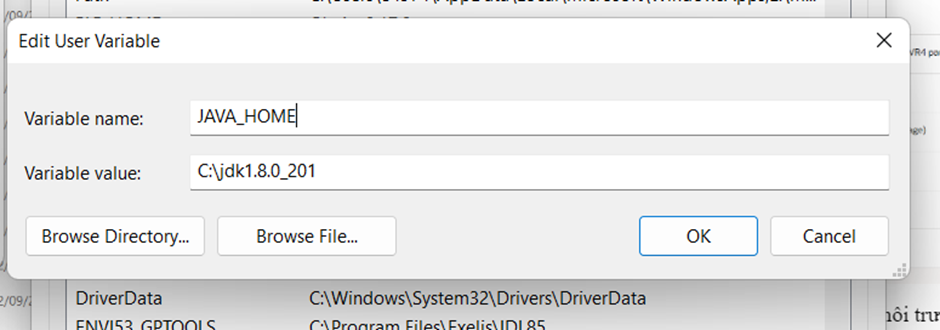
Cài đặt JDK, Hadoop, Apache Spark, Python, Elasticsearch trên Window

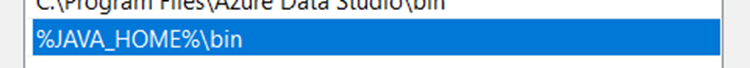
* + 1. Cài đặt Java JDK 1.8

https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html

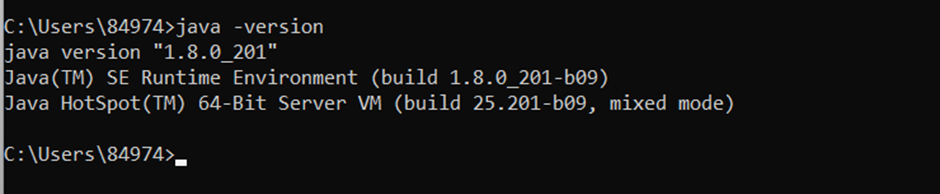


Thiết lập biến môi trường



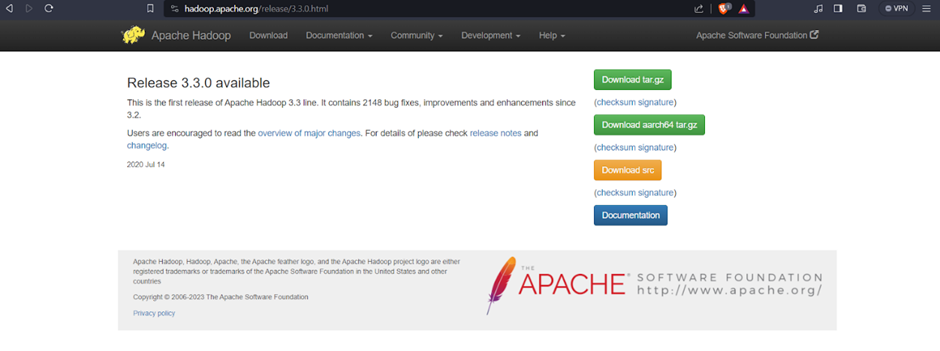


Gõ lệnh trên cmd để kiểm tra cài đặt có thành công không

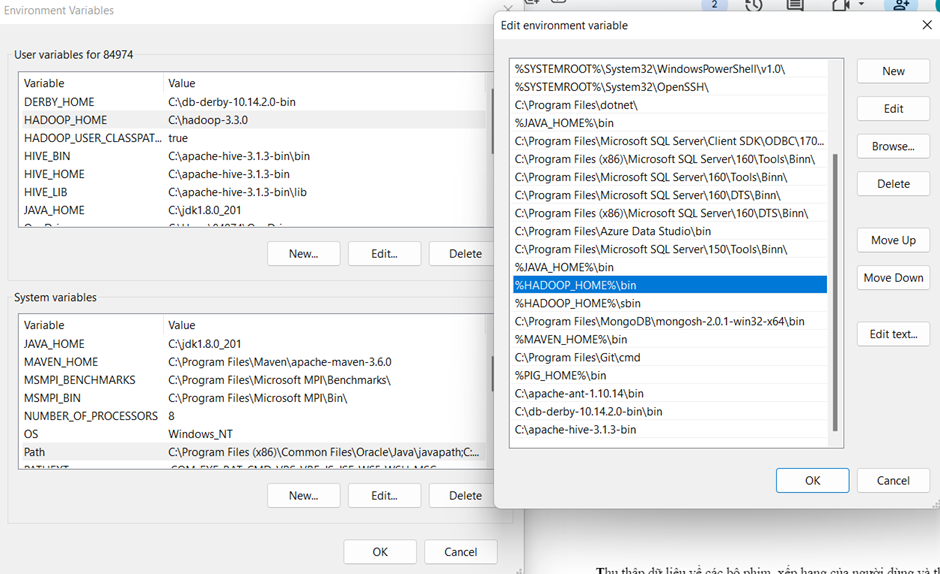


* + 1. Cài đặt Hadoop

https://hadoop.apache.org/releases.html



Thiết lập biến môi trường



Cấu hình các tập tin Hadoop

**Trong thư mục /etc/hadoop chỉnh sửa các file:**

core-site.xml

mapred-site.xml

hdfs-site.xml

yarn-site.xml

hadoop-env.cmd

**Cấu hình core-site.xml như dưới đây:**

configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

**Cấu hình mapred-site.xml như dưới đây:**

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

Tạo thư mục **“data”** trong “C:/Hadoop-3.3.0”

Tạo thư mục con **“datanode”** trong “C:/Hadoop-3.3.0/data”

Tạo thư mục con **“namenode”** trong “C:/Hadoop-3.3.0/data”

**Sau đó cấu hình hdfs-site.xml như sau:**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>/hadoop-3.3.0/data/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/hadoop-3.3.0/data/datanode</value>

</property>

</configuration>

**Cấu hình yarn-site.xml như dưới đây:**

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name> <value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

</configuration>

**Mở file hadoop-env.cmd và cấu hình:**

Tìm lệnh:

* set JAVA\_HOME=%JAVA\_HOME%

Sửa %JAVA\_HOME% thành đường dẫn cài JDK

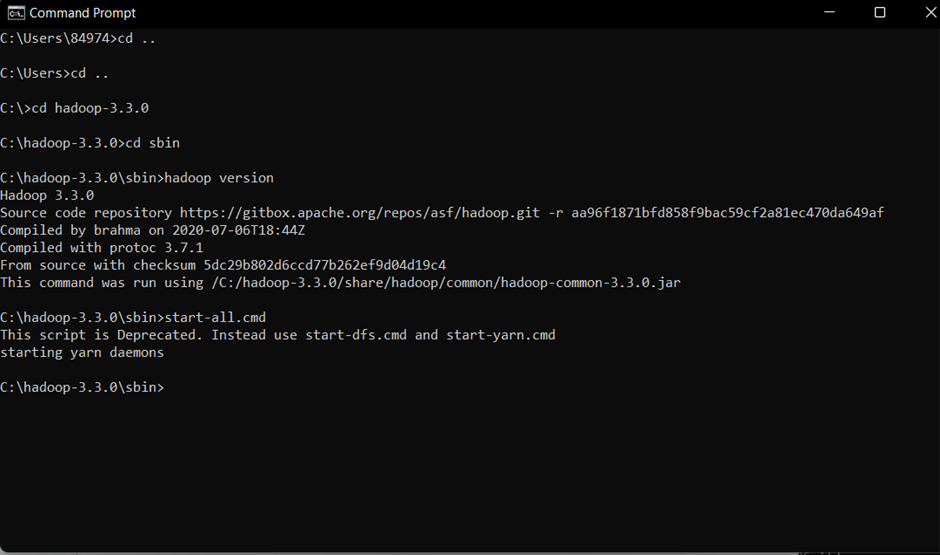
* **set JAVA\_HOME= C:/jdk1.8.0\_201**

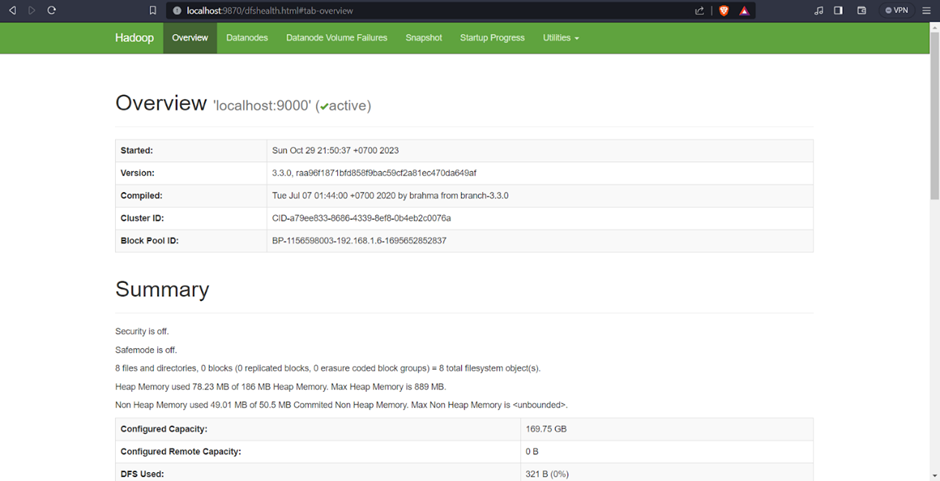
Vào cmd gõ lệnh:

**·hdfs namenode -format**

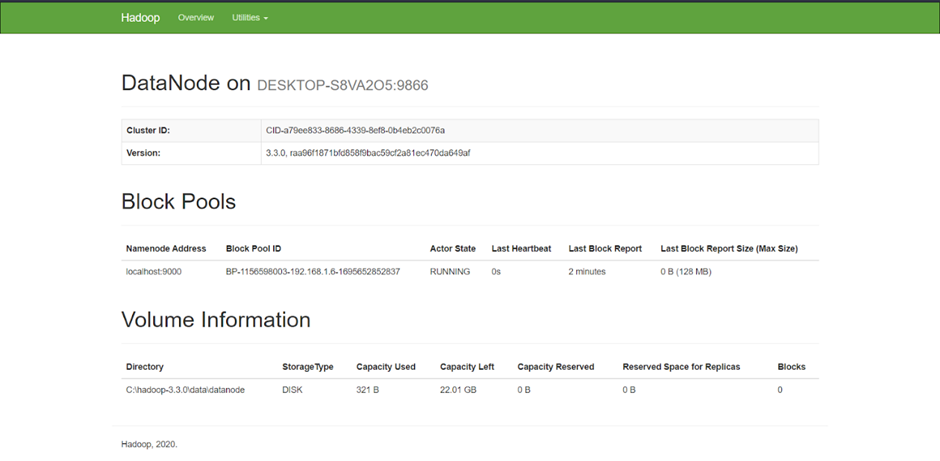
**·hdfs datanode -forma**t

**Sau đó kiểm tra Hadoop cài đặt thành công chưa và truy cập các localhost**

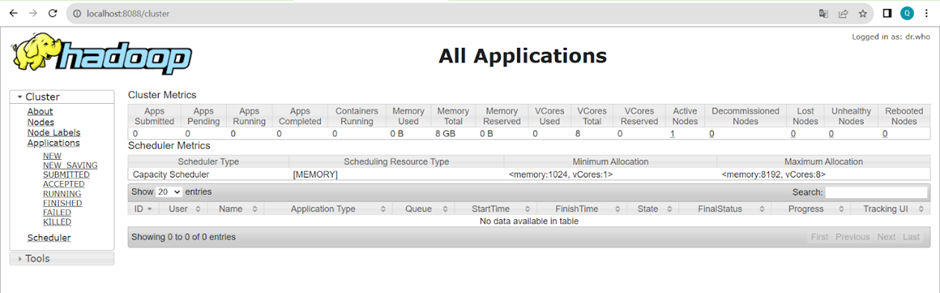
**Truy cập namenode port 9870**



**Truy cập datanode port 9864**

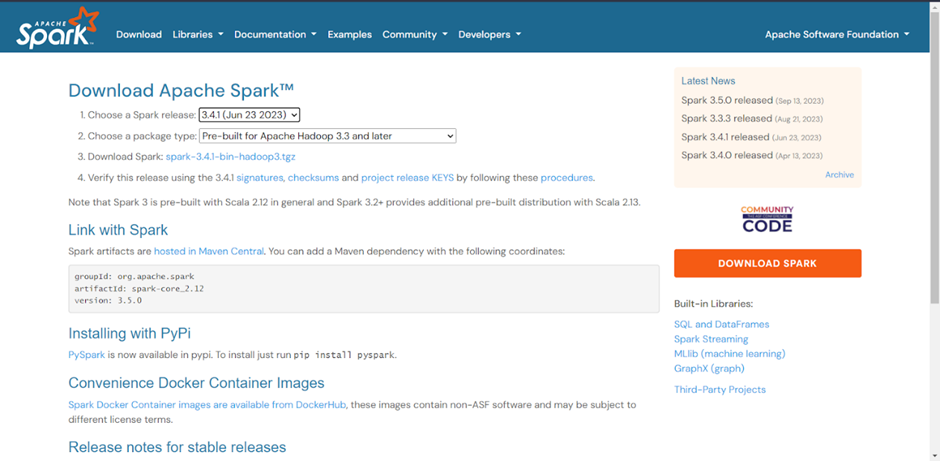


**Truy cập port 8088**

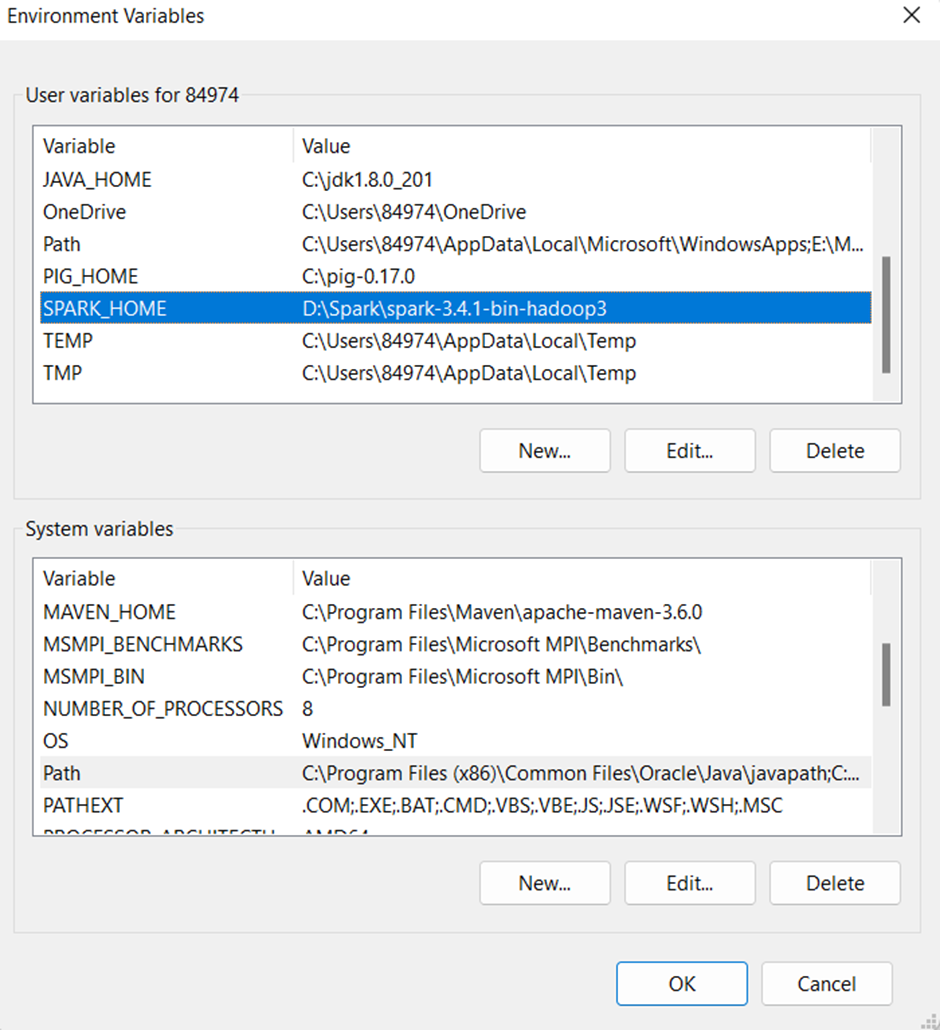


* + 1. Cài đặt Apache Spark

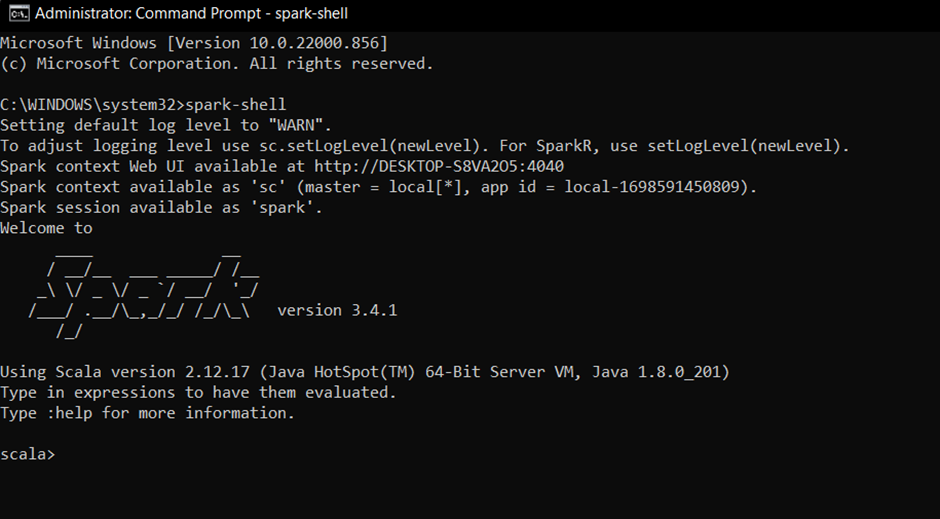
<https://spark.apache.org/downloads.html>



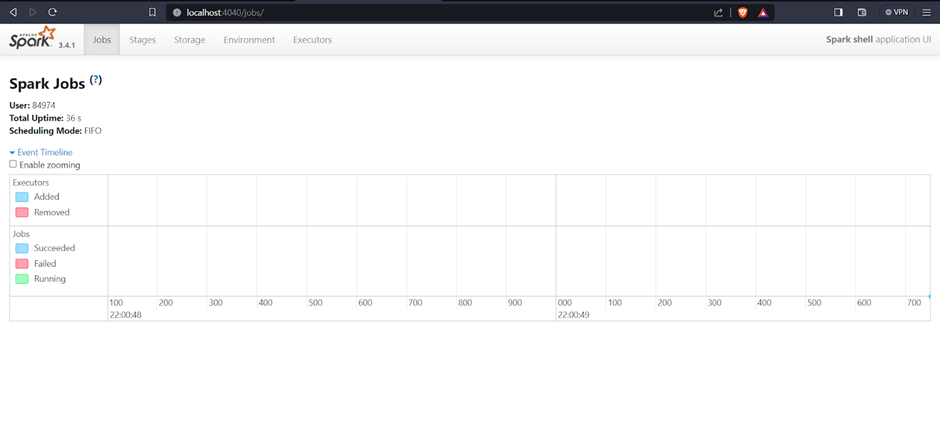
**Thiết lập biến môi trường**



**Kiểm tra cài đặt thành công trên cmd bằng lệnh spark-shell**

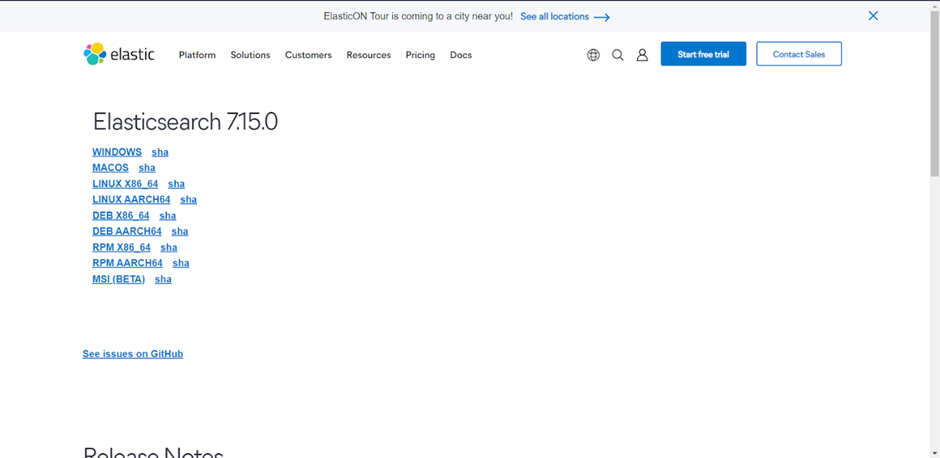


**Truy cập port 4040**

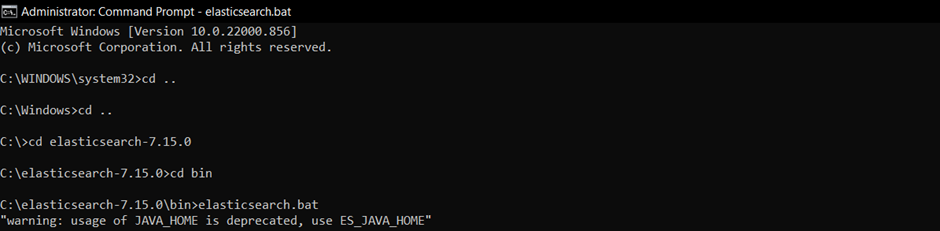


* + 1. Cài đặt Elasticsearch

<https://www.elastic.co/downloads/past-releases/elasticsearch-7-15-0>



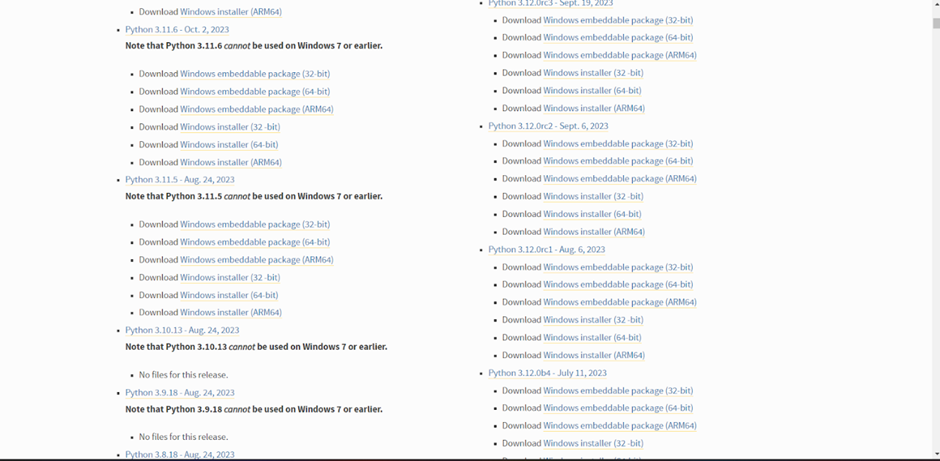
**Vào cmd gõ elasticsearch.bat để chạy port 9200**

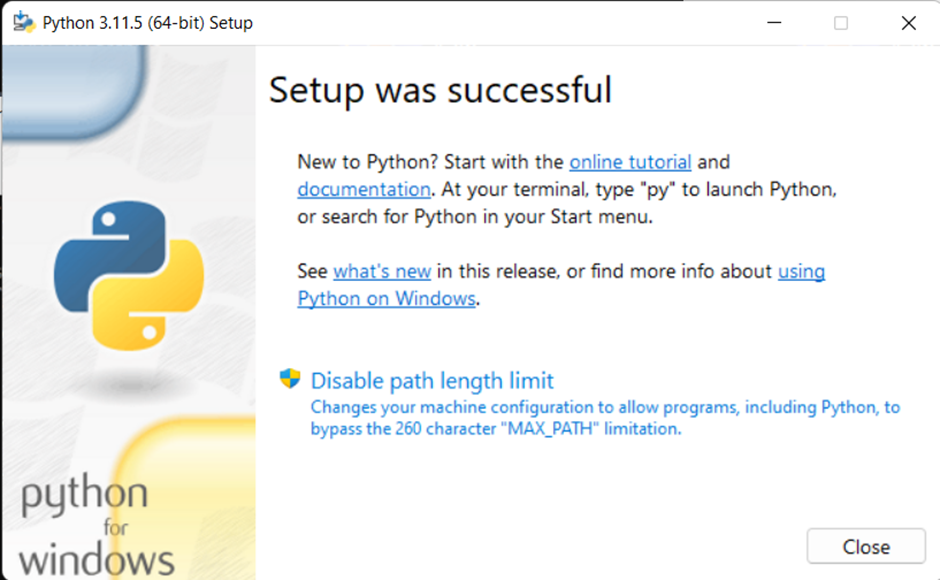




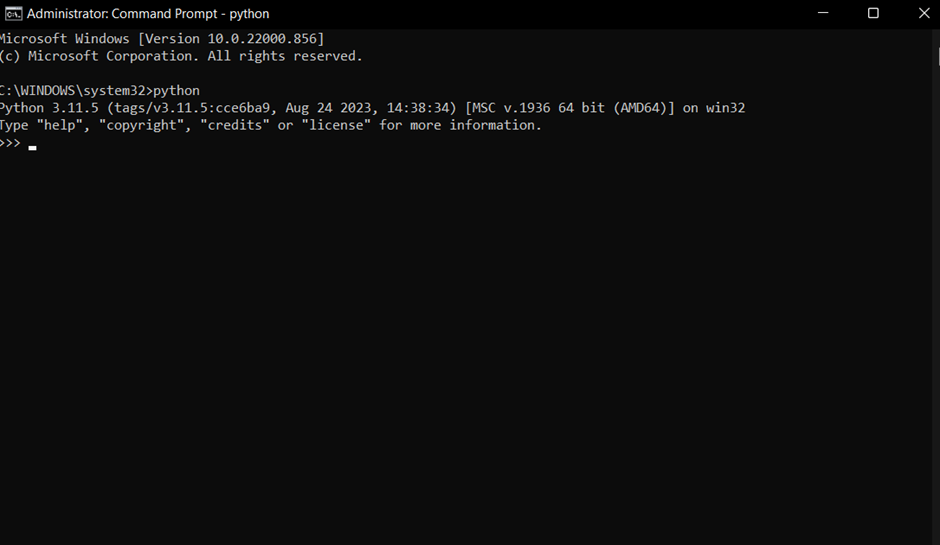
* + 1. Cài đặt Python

<https://www.python.org/downloads/windows/>





**Gõ lệnh python vào cmd để kiểm tra cài đặt thành công**



* 1. THỰC NGHIỆM
     1. Thu thập dữ liệu và Tiền xử lý dữ liệu

**T**hu thập dữ liệu về các bộ phim, xếp hạng của người dùng và thông tin về người dùng. Dữ liệu này có thể lấy từ các nguồn như Kaggle, MovieLens, hoặc cơ sở dữ liệu của riêng bạn.

Thu thập dữ liệu về phim từ các trang web:imdb,tmdb,phimmoi.net…

Sử dụng python và thư viện requests,BeautifulSoup để crawl dữ liệu:

Các bước crawl với trang: <https://www.imdb.com/chart/top/>

Crawl dữ liệu

Bước 1: Import các thư viện cần thiết

***import requests***

***from bs4 import BeautifulSoup***

Bước 2: Gửi yêu cầu HTTP đến trang web

***url = 'https://www.example.com' response = requests.get(url)***

Có 2 trạng thái sau khi gửi yêu cầu:

[200] OK: Yêu cầu thành công. Đây là mã trạng thái bạn muốn nhìn thấy, nó có nghĩa là yêu cầu của bạn đã được chấp nhận và trang web đã trả về dữ liệu theo yêu cầu.



[403] Forbidden: Trang web từ chối yêu cầu của bạn. Điều này thường xuyên xảy ra khi bạn không có quyền truy cập vào một trang hoặc một phần của trang web. Có thể do bạn không có quyền truy cập hoặc trang web có hệ thống chống crawler và ngăn chặn việc truy cập tự động.



Nếu gặp lỗi này,có thể thử sử dụng cách sau:

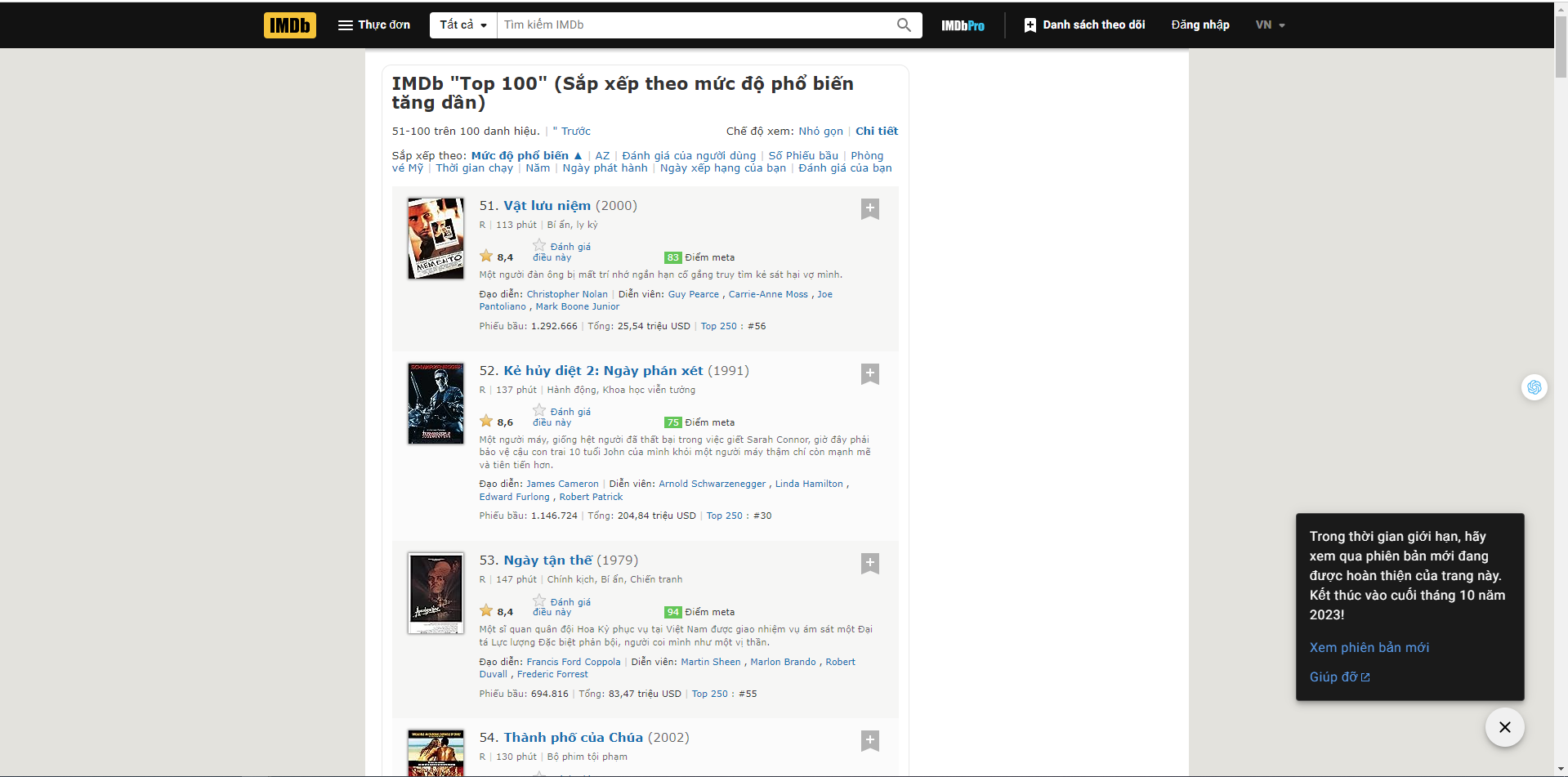
User-Agent Header: Thêm User-Agent header vào yêu cầu của bạn để giả mạo trình duyệt web.Có thể đặt giá trị User-Agent là một trình duyệt phổ biến như Chrome hoặc Firefox để tránh việc bị từ chối truy cập.

***headers = {***

***'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'***

***}***

***response = requests.get('https://www.imdb.com/search/title/?groups=top\_100&start=51&ref\_=adv\_nxt')***

******

Bước 3: Phân tích HTML để lấy dữ liệu bằng BeautifulSoup

***soup = BeautifulSoup(response.content, 'html.parser')***

Bước 4: Lấy dữ liệu từ các phần tử HTML

Dùng các phương thức của thư viện BeautifulSoup để lấy dữ liệu từ các thẻ HTML.

#Kiểm tra trạng thái response

if response.status\_code == 200:

# Sử dụng BeautifulSoup để phân tích cú pháp HTML của trang web

soup = BeautifulSoup(response.content, 'html.parser')

# Tạo hoặc mở một tệp văn bản để ghi thông tin

with open('phim\_top\_100\_imdb.txt', 'w', encoding='utf-8') as file:

# Tìm tất cả các phần tử có class là 'lister-item mode-advanced'

movies = soup.find\_all('div', class\_='lister-item mode-advanced')

# Lặp qua từng phim và trích xuất thông tin cần thiết

for movie in movies:

# Trích xuất tiêu đề phim

title = movie.h3.a.text.strip() if movie.h3.a else "N/A"

# Trích xuất năm sản xuất

year = movie.h3.find('span', class\_='lister-item-year').text.strip() if movie.h3.find('span', class\_='lister-item-year') else "N/A"

# Trích xuất thể loại phim

genre = movie.p.find('span', class\_='genre').text.strip() if movie.p.find('span', class\_='genre') else "N/A"

# Trích xuất điểm đánh giá từ IMDb

rating = movie.find('div', class\_='ratings-imdb-rating').strong.text.strip() if movie.find('div', class\_='ratings-imdb-rating') else "N/A"

# Trích xuất thông tin thời lượng từ thẻ <span class="runtime">

runtime\_span = movie.p.find('span', class\_='runtime')

runtime = runtime\_span.text.strip() if runtime\_span else "N/A"

# Trích xuất giá trị "Votes"

votes\_tag = movie.find('span', string='Votes:')

votes = votes\_tag.find\_next('span', {'name': 'nv', 'data-value': True}).text.strip() if votes\_tag else "N/A"

# Trích xuất giá trị "Gross"

gross\_tag = movie.find('span', string='Gross:')

gross = gross\_tag.find\_next('span', {'name': 'nv', 'data-value': True}).text.strip() if gross\_tag else "N/A"

# Viết thông tin của phim vào tệp văn bản

print(f'Tiêu đề: {title}\n')

print(f'Năm sản xuất: {year}\n')

print(f'Thể loại: {genre}\n')

print(f'Điểm đánh giá: {rating}\n')

print(f'Thời lượng: {runtime}\n')

print(f'Votes: {votes}\n')

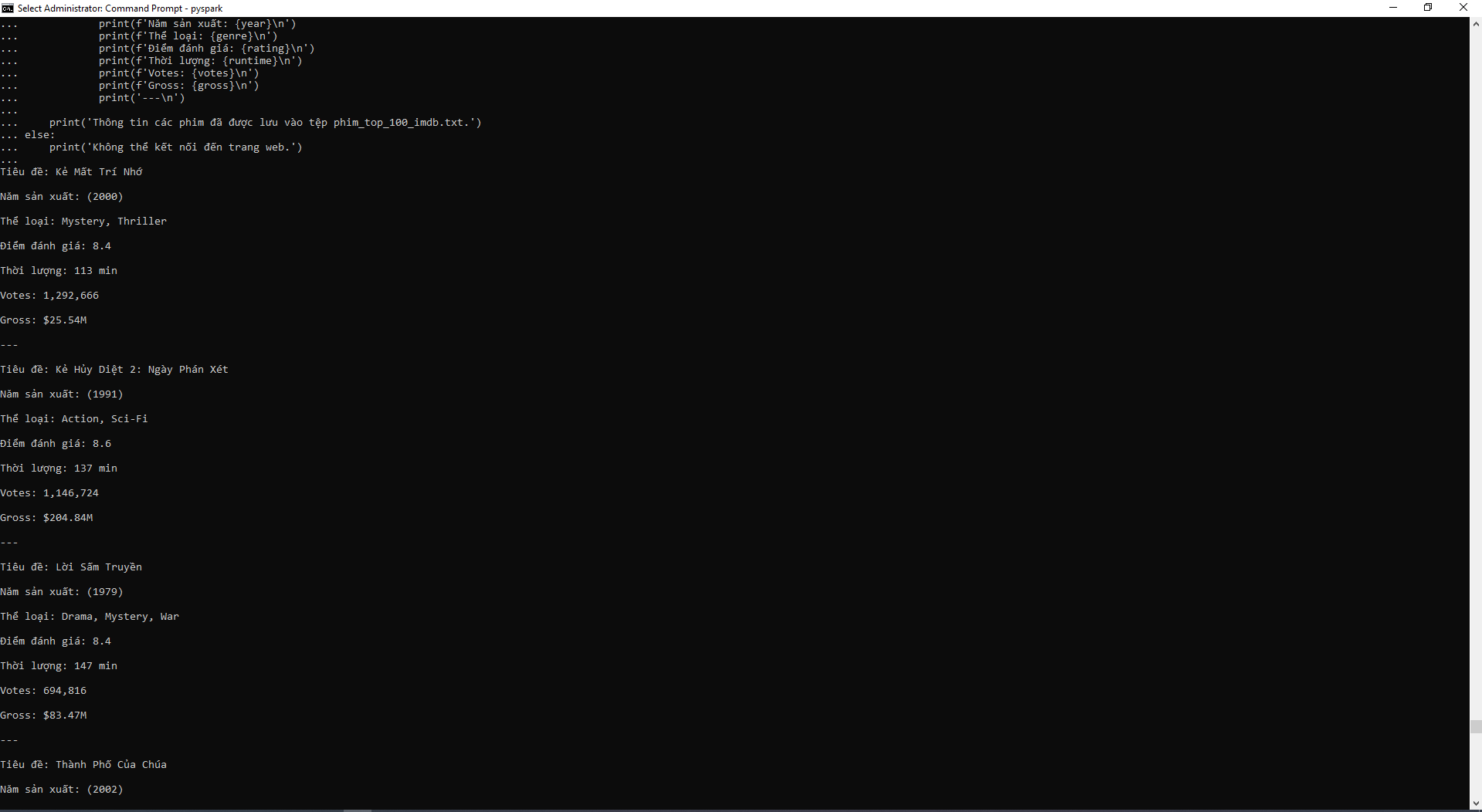
print(f'Gross: {gross}\n')

print('---\n')

print('Thông tin các phim đã được lưu vào tệp phim\_top\_100\_imdb.txt.')

else:

print('Không thể kết nối đến trang web.')

****

Dữ liệu public

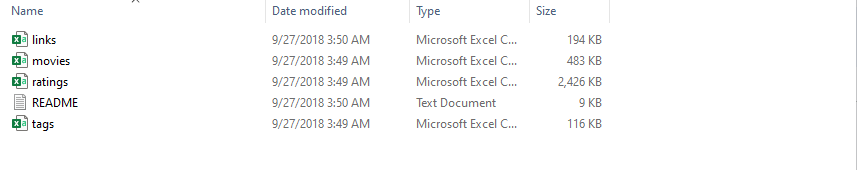
Sử dụng dữ liệu public trên trang: <https://grouplens.org/datasets/movielens/>

Tải dữ liệu và xử lý dữ liệu

Sau khi tải dữ liệu ví dụ dữ liệu phim vào 9/2018 bằng cách click vào ml-lastest-small.zip



Sau khi tải về và giải nén ra sẽ thu được các file csv:



chứa khoảng 100.000 xếp hạng, 9.000 phim và 700 người dùng, sử dụng các tập tin sau ratings.csv - movie rating data

links.csv - external database ids for each movie

movies.csv - movie title and genres

Khởi tạo spark

*from IPython.display import Image, HTML, display*

*from pyspark.sql import SparkSession*

*spark = SparkSession.builder.appName('thanhtruong').master('local[\*]').getOrCreate()*

Đọc file rating.csv và in ra  
[IN]

*PATH\_TO\_DATA = "../data/ml-latest-small"*

*ratings = spark.read.csv(PATH\_TO\_DATA + "/ratings.csv", header=True, inferSchema=True)*

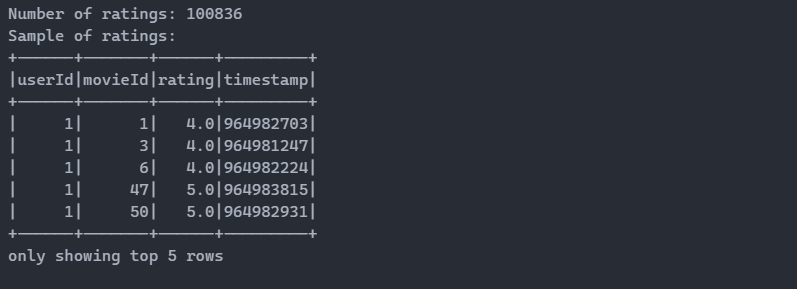
*ratings.cache()*

*print("Number of ratings: {}".format(ratings.count()))*

*print("Sample of ratings:")*

*ratings.show(5)*

[OUT]



Chuyển đổi timestamps từ giây sang mili giây

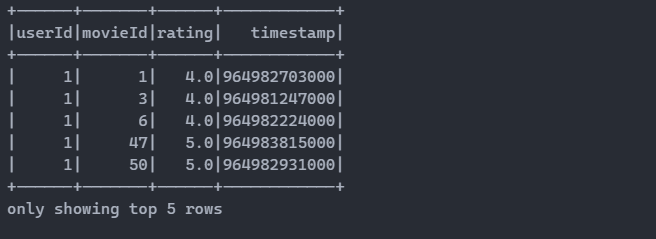
[IN]

*ratings = ratings.select(*

*ratings.userId, ratings.movieId, ratings.rating, (ratings.timestamp.cast("long") \* 1000).alias("timestamp"))*

*ratings.show(5)*

[OUT]



Tệp `movies.csv` chứa `movieId`, `title` và `genres` cho mỗi phim. Như bạn có thể thấy, trường `genres` hơi khó sử dụng, vì các thể loại ở dạng một chuỗi được phân cách bằng ký tự `|`: `Adventure|Animation|Children|Comedy| Fantasy`.

[IN]

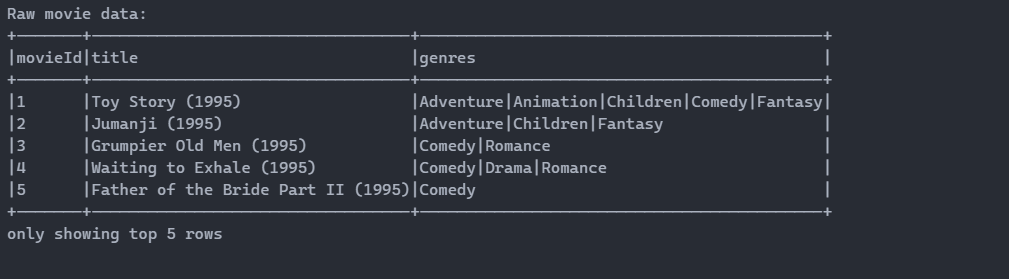
*# load raw data from CSV*

*raw\_movies = spark.read.csv(PATH\_TO\_DATA + "/movies.csv", header=True, inferSchema=True)*

*print("Raw movie data:")*

*raw\_movies.show(5, truncate=False)*

[OUT]



Vì thế chúng ta sẽ chuyển đổi chuỗi genres thành một mảng genres viết thường

[IN]  
*from pyspark.sql.functions import udf*

*from pyspark.sql.types import ArrayType, StringType*

*import os*

*import sys*

*from pyspark import SparkContext*

*os.environ['PYSPARK\_PYTHON'] = sys.executable*

*os.environ['PYSPARK\_DRIVER\_PYTHON'] = sys.executable*

*# Định nghĩa một UDF để chuyển đổi chuỗi genres thành một mảng genres viết thường*

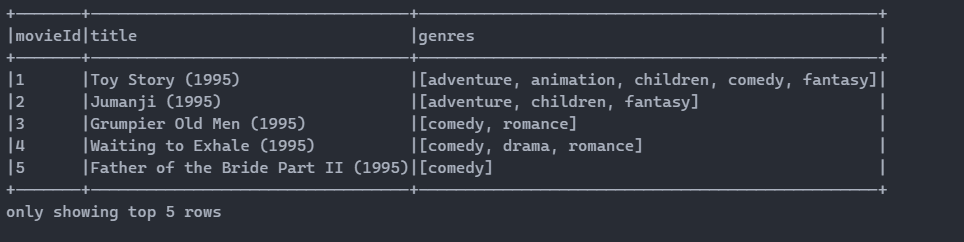
*extract\_genres = udf(lambda x: x.lower().split("|"), ArrayType(StringType()))*

*# Sử dụng UDF để chuyển đổi cột "genres" và chọn cột "movieId" và "title"*

*# Đặt tên cột mới là "genres"*

*raw\_movies.select("movieId", "title", extract\_genres("genres").alias("genres")).show(5, False)*

[OUT]



Bạn cũng có thể nhận thấy rằng tên phim có năm phát hành vì thế chúng ta sẽ tách nó ra thành hai phần riêng

[IN]

*import re*

*from pyspark.sql.types import StringType, StructType, StructField*

*# define a UDF to extract the release year from the title, and return the new title and year in a struct type*

*def extract\_year\_fn(title):*

*result = re.search("\(\d{4}\)", title)*

*try:*

*if result:*

*group = result.group()*

*year = group[1:-1]*

*start\_pos = result.start()*

*title = title[:start\_pos-1]*

*return (title, year)*

*else:*

*return (title, 1970)*

*except:*

*print(title)*

*extract\_year = udf(extract\_year\_fn, StructType([StructField("title", StringType(), True),StructField("release\_date", StringType(), True)]))*

*movies = raw\_movies.select(*

*"movieId", extract\_year("title").title.alias("title"),\*

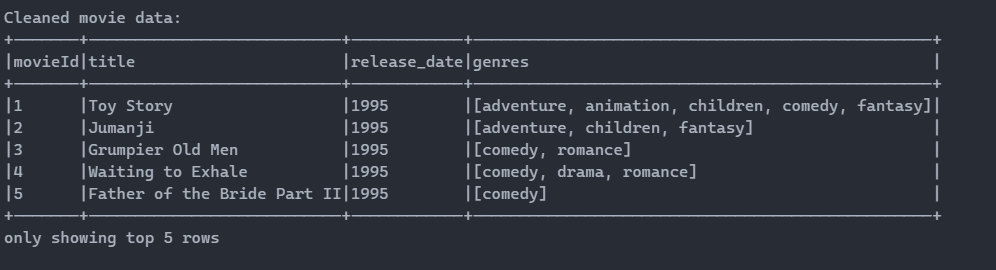
*extract\_year("title").release\_date.alias("release\_date"),\*

*extract\_genres("genres").alias("genres"))*

*print("Cleaned movie data:")*

*movies.show(5, truncate=False)*

[OUT]



Tiếp theo, nối dữ liệu `links.csv` với `phim` để có id cho Cơ sở dữ liệu phim\_ tương ứng với từng phim. Bạn có thể sử dụng id này để truy xuất poster phim khi hiển thị đề xuất của mình sau này.

[IN]

*link\_data = spark.read.csv(PATH\_TO\_DATA + "/links.csv", header=True, inferSchema=True)*

*# join movies with links to get TMDB id*

*movie\_data = movies.join(link\_data, movies.movieId == link\_data.movieId)\*

*.select(movies.movieId, movies.title, movies.release\_date, movies.genres, link\_data.tmdbId)*

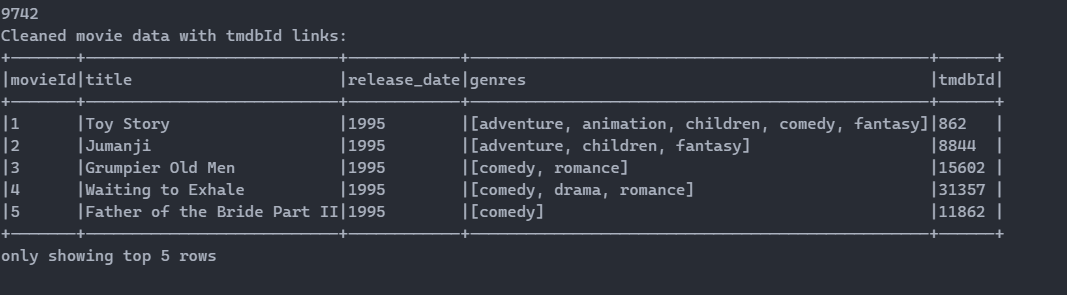
*num\_movies = movie\_data.count()*

*print(num\_movies)*

*print("Cleaned movie data with tmdbId links:")*

*movie\_data.show(5, truncate=False)*

[OUT]



Cài đặt gói tmdbsimple bằng dòng lệnh pip install tmdbsimple trong console

Sau khi cài đặt xong thì chạy dòng lệnh sao để lấy ảnh poster phim từ tmdb

[IN]

*try:*

*import tmdbsimple as tmdb*

*import json*

*from requests.exceptions import HTTPError*

*# replace this variable with your actual TMdb API key*

*tmdb.API\_KEY = 'd991c0846837eea6cb0709bd3a8440bb'*

*print("Successfully imported tmdbsimple!")*

*# base URL for TMDB poster images*

*IMAGE\_URL = 'https://image.tmdb.org/t/p/w500'*

*movie\_id = movie\_data.first().tmdbId*

*movie\_info = tmdb.Movies(movie\_id).info()*

*movie\_poster\_url = IMAGE\_URL + movie\_info['poster\_path']*

*display(Image(movie\_poster\_url, width=200))*

*except ImportError:*

*print("Cannot import tmdbsimple as it is not installed, no movie posters will be displayed!")*

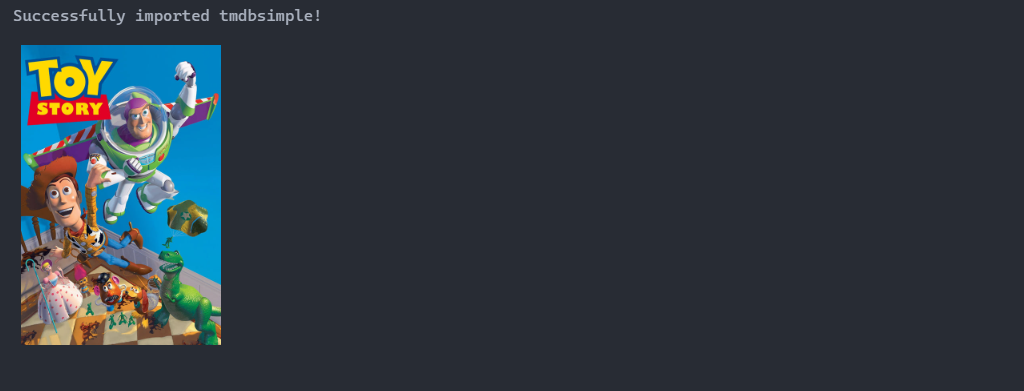
*except HTTPError as e:*

*if e.response.status\_code == 401:*

*j = json.loads(e.response.text)*

*print("TMdb API call failed: {}".format(j['status\_message']))*

[OUT]



* + 1. Kết nối và tải dữ liệu vào Elasticsearch

[IN]

*from elasticsearch import Elasticsearch*

*# Cung cấp đầy đủ thông tin cấu hình, bao gồm scheme, host và port*

*es = Elasticsearch([{'host': 'localhost', 'port': 9200, 'scheme': 'http'}])*

*# Test kết nối*

*es.info(pretty=True)*

[OUT]

**ObjectApiResponse({'name': 'DESKTOP-TQ54SGR', 'cluster\_name': 'elasticsearch', 'cluster\_uuid': 'Ql1sS42ORTevLiDTLR5PoQ', 'version': {'number': '7.15.2', 'build\_flavor': 'default', 'build\_type': 'zip', 'build\_hash': '93d5a7f6192e8a1a12e154a2b81bf6fa7309da0c', 'build\_date': '2021-11-04T14:04:42.515624022Z', 'build\_snapshot': False, 'lucene\_version': '8.9.0', 'minimum\_wire\_compatibility\_version': '6.8.0', 'minimum\_index\_compatibility\_version': '6.0.0-beta1'}, 'tagline': 'You Know, for Search'})**

Create Elasticsearch indices, with mappings for users, movies and rating events

[IN]

*# set the factor vector dimension for the recommendation model*

*VECTOR\_DIM = 20*

*create\_ratings = {*

*# this mapping definition sets up the fields for the rating events*

*"mappings": {*

*"properties": {*

*"timestamp": {*

*"type": "date"*

*},*

*"userId": {*

*"type": "integer"*

*},*

*"movieId": {*

*"type": "integer"*

*},*

*"rating": {*

*"type": "double"*

*}*

*}*

*}*

*}*

*create\_users = {*

*# this mapping definition sets up the metadata fields for the users*

*"mappings": {*

*"properties": {*

*"userId": {*

*"type": "integer"*

*},*

*# the following fields define our model factor vectors and metadata*

*"model\_factor": {*

*"type": "dense\_vector",*

*"dims" : VECTOR\_DIM*

*},*

*"model\_version": {*

*"type": "keyword"*

*},*

*"model\_timestamp": {*

*"type": "date"*

*}*

*}*

*}*

*}*

*create\_movies = {*

*# this mapping definition sets up the metadata fields for the movies*

*"mappings": {*

*"properties": {*

*"movieId": {*

*"type": "integer"*

*},*

*"tmdbId": {*

*"type": "keyword"*

*},*

*"genres": {*

*"type": "keyword"*

*},*

*"release\_date": {*

*"type": "date",*

*"format": "year"*

*},*

*# the following fields define our model factor vectors and metadata*

*"model\_factor": {*

*"type": "dense\_vector",*

*"dims" : VECTOR\_DIM*

*},*

*"model\_version": {*

*"type": "keyword"*

*},*

*"model\_timestamp": {*

*"type": "date"*

*}*

*}*

*}*

*}*

*# create indices with the settings and mappings above*

*res\_ratings = es.indices.create(index="ratings", body=create\_ratings)*

*res\_users = es.indices.create(index="users", body=create\_users)*

*res\_movies = es.indices.create(index="movies", body=create\_movies)*

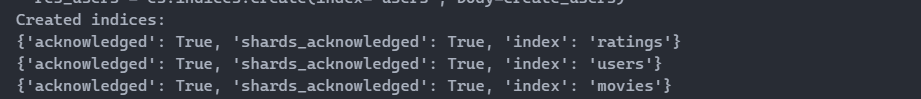
*print("Created indices:")*

*print(res\_ratings)*

*print(res\_users)*

*print(res\_movies)*

[OUT]



Tiến hành đưa dữ liệu lên Elasticsearch

Hàm json\_formatter được viết để chuyển đổi dữ liệu từ một Spark DataFrame thành danh sách các mục dữ liệu dạng JSON

[IN]

*def json\_formatter(dataset, index\_name, index\_type='\_doc'):*

*try:*

*List = []*

*columns = dataset.columns*

*for row in dataset.rdd.collect():*

*dic = {}*

*dic['\_index'] = index\_name*

*dic['\_type'] = index\_type*

*source = {}*

*for i, col in enumerate(columns):*

*source[col] = row[i]*

*dic['\_source'] = source*

*List.append(dic)*

*return List*

*except Exception as e:*

*print("There is a problem: {}".format(e))*

*json\_Formatted\_ratings = json\_formatter(dataset=ratings, index\_name='ratings', index\_type='\_doc')*

*json\_Formatted\_ratings[0]*

[OUT]



Sử dụng thư viện elasticsearch.helpers để lưu trữ danh sách các mục dữ liệu JSON (đã được chuẩn bị bằng hàm json\_formatter) vào Elasticsearch

[IN]

*from elasticsearch import helpers*

*try:*

*res = helpers.bulk(es, json\_Formatted\_ratings)*

*print("successfully imported to elasticsearch.")*

*except Exception as e:*

*print(f"error: {e}")*

[OUT]



Kiểm tra

[IN]

*es.search(index="ratings", q="\*", size=3)*

[OUT]

**ObjectApiResponse({'took': 3709, 'timed\_out': False, '\_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 10000, 'relation': 'gte'}, 'max\_score': 1.0, 'hits': [{'\_index': 'ratings', '\_type': '\_doc', '\_id': '2sIQn4sB5q04Oul4AlfG', '\_score': 1.0, '\_source': {'userId': 1, 'movieId': 1, 'rating': 4.0, 'timestamp': 964982703000}},**

**{'\_index': 'ratings', '\_type': '\_doc', '\_id': '28IQn4sB5q04Oul4AlfK', '\_score': 1.0, '\_source': {'userId': 1, 'movieId': 3, 'rating': 4.0, 'timestamp': 964981247000}},**

**{'\_index': 'ratings', '\_type': '\_doc', '\_id': '3MIQn4sB5q04Oul4AlfK', '\_score': 1.0, '\_source': {'userId': 1, 'movieId': 6, 'rating': 4.0, 'timestamp': 964982224000}}]}})**

Tiếp theo là movie data

[IN]

*def json\_formatter\_movies(dataset, index\_name):*

*try:*

*List = []*

*for row in dataset.rdd.collect():*

*dic = {}*

*dic['\_index'] = index\_name*

*dic['\_id'] = str(row["movieId"]) # Set trường "movieId" làm trường id*

*dic['\_source'] = row.asDict()*

*List.append(dic)*

*return List*

*except Exception as e:*

*print("There is a problem: {}".format(e))*

*json\_Formatted\_movies = json\_formatter\_movies(dataset=movie\_data, index\_name='movies')*

*json\_Formatted\_movies[0]*

*from elasticsearch import helpers*

*try:*

*res = helpers.bulk(es, json\_Formatted\_movies)*

*print("successfully imported to elasticsearch.")*

*except Exception as e:*

*print(f"error: {e}")*

[OUT]  


Kiểm tra

[IN]

*es.search(index="movies", q="title:matrix", size=3)*

[OUT]

**ObjectApiResponse({'took': 1674, 'timed\_out': False, '\_shards': {'total': 1, 'successful': 1, 'skipped': 0, 'failed': 0}, 'hits': {'total': {'value': 3, 'relation': 'eq'}, 'max\_score': 9.550224, 'hits': [**

**{'\_index': 'movies', '\_type': '\_doc', '\_id': '2571', '\_score': 9.550224, '\_source': {'movieId': 2571, 'title': 'Matrix, The', 'release\_date': '1999', 'genres': ['action', 'sci-fi', 'thriller'], 'tmdbId': 603}},**

**{'\_index': 'movies', '\_type': '\_doc', '\_id': '6365', '\_score': 8.322349, '\_source': {'movieId': 6365, 'title': 'Matrix Reloaded, The', 'release\_date': '2003', 'genres': ['action', 'adventure', 'sci-fi', 'thriller', 'imax'], 'tmdbId': 604}},**

**{'\_index': 'movies', '\_type': '\_doc', '\_id': '6934', '\_score': 8.322349, '\_source': {'movieId': 6934, 'title': 'Matrix Revolutions, The', 'release\_date': '2003', 'genres': ['action', 'adventure', 'sci-fi', 'thriller', 'imax'], 'tmdbId': 605}}]}})**

* + 1. Đào tạo một mô hình đề xuất về dữ liệu xếp hạng

[IN]

*from pyspark.ml.recommendation import ALS*

*from pyspark.sql.functions import col*

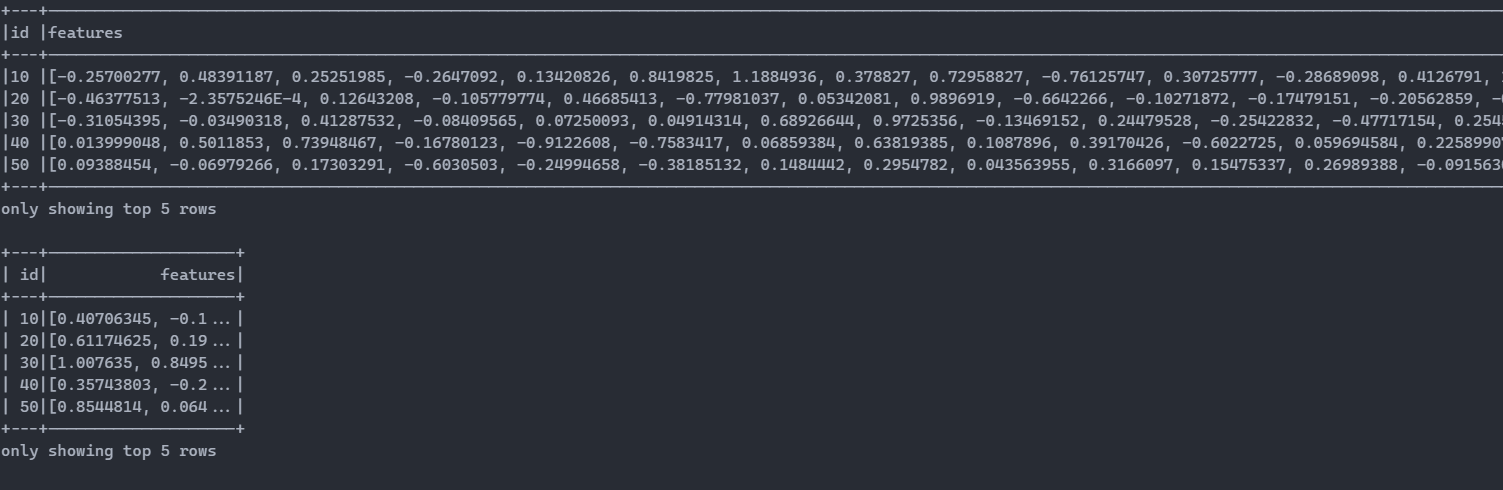
*als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating", regParam=0.02, rank=20, seed=54)*

*model = als.fit(ratings)*

*model.userFactors.show(5, truncate=False)*

*model.itemFactors.show(5)*

[OUT]



[IN]

*from pyspark.sql.functions import lit, current\_timestamp, unix\_timestamp*

*ver = model.uid*

*ts = unix\_timestamp(current\_timestamp())*

*movie\_vectors = model.itemFactors.select("id",\*

*col("features").alias("model\_factor"),\*

*lit(ver).alias("model\_version"),\*

*ts.alias("model\_timestamp"))*

*movie\_vectors.show(5)*

*user\_vectors = model.userFactors.select("id",\*

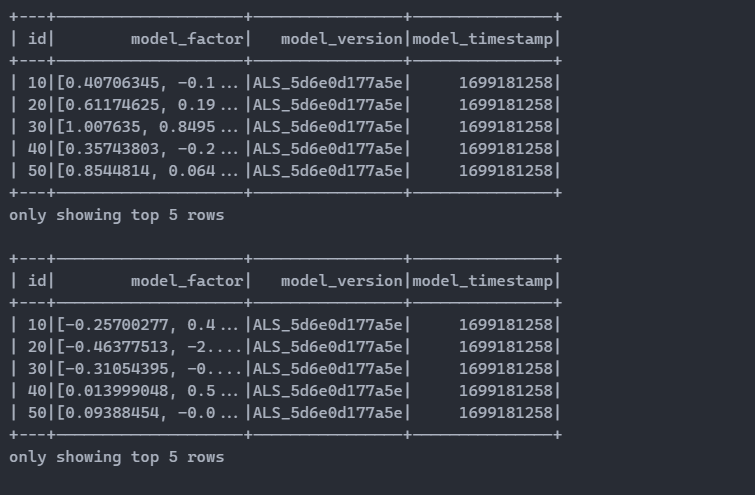
*col("features").alias("model\_factor"),\*

*lit(ver).alias("model\_version"),\*

*ts.alias("model\_timestamp"))*

*user\_vectors.show(5)*

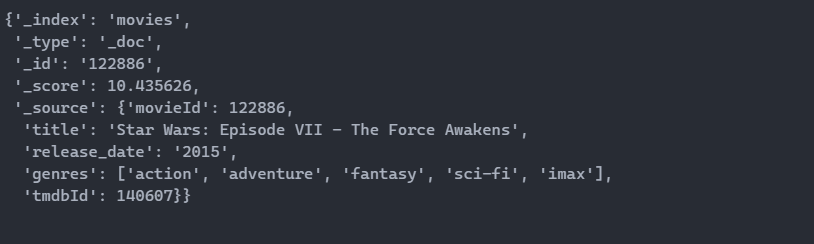
[OUT]



[IN]

*es.search(index="movies", q="force awakens")['hits']['hits'][0]*

[OUT]



* + 1. Sử dụng recommend của Elasticsearch

**Khởi tạo một vài hàm recommend**

[IN]

*from IPython.display import Image, display*

*def recommendSimilarMovies(movie\_id, num\_recs=10, q=None):*

*# Lấy thông tin chi tiết của bộ phim nguồn*

*movie\_info = es.get(index="movies", id=movie\_id)*

*source\_movie = movie\_info['\_source']*

*# Lấy thể loại của bộ phim nguồn*

*source\_genres = source\_movie.get('genres', [])*

*# Tạo truy vấn Elasticsearch để tìm các bộ phim cùng thể loại hoặc theo truy vấn tùy chỉnh (q)*

*bool\_queries = [{"match": {"genres": genre}} for genre in source\_genres]*

*if q:*

*custom\_query = {"query\_string": {"query": q}}*

*bool\_queries.append(custom\_query)*

*query = {*

*"size": num\_recs,*

*"query": {*

*"bool": {*

*"should": bool\_queries*

*}*

*}*

*}*

*# Thực hiện truy vấn Elasticsearch*

*result = es.search(index="movies", body=query)*

*hits = result['hits']['hits']*

*# Lấy thông tin các bộ phim tương tự*

*similar\_movies = [hit['\_source'] for hit in hits if hit['\_id'] != movie\_id]*

*return similar\_movies*

*def get\_poster\_url(tmdb\_id):*

*# Hàm này sẽ trả về URL ảnh poster từ TMDb API dựa trên tmdb\_id*

*IMAGE\_URL = 'https://image.tmdb.org/t/p/w500'*

*try:*

*import tmdbsimple as tmdb*

*from tmdbsimple import APIKeyError*

*try:*

*movie = tmdb.Movies(tmdb\_id).info()*

*poster\_url = IMAGE\_URL + movie['poster\_path'] if 'poster\_path' in movie and movie['poster\_path'] is not None else ""*

*return poster\_url*

*except APIKeyError as ae:*

*return "KEY\_ERR"*

*except Exception as me:*

*return "NA"*

*def recommendSimilarMoviesWithDetails(movie\_id, num\_recs=10, q=None):*

*# Lấy danh sách các bộ phim tương tự*

*similar\_movies = recommendSimilarMovies(movie\_id, num\_recs, q)*

*# Hiển thị thông tin và ảnh poster cho từng bộ phim tương tự*

*for movie in similar\_movies:*

*print(f"Movie: {movie['title']} (tmdbId: {movie['tmdbId']})")*

*# Lấy URL ảnh poster từ TMDb API*

*poster\_url = get\_poster\_url(movie['tmdbId'])*

*if poster\_url != "NA":*

*display(Image(poster\_url, width=150))*

*else:*

*print("No poster available for this movie.")*

*def getMoviesRatedByUser(user\_id, num\_movies=10):*

*user\_movies = get\_movies\_for\_user(user\_id, num=num\_movies)*

*if user\_movies:*

*for movie in user\_movies:*

*print(f"Movie: {movie['title']} (tmdbId: {movie['tmdbId']})")*

*poster\_url = get\_poster\_url(movie['tmdbId'])*

*if poster\_url != "NA":*

*display(Image(poster\_url, width=150))*

*else:*

*print("No poster available for this movie.")*

*else:*

*print(f"No highly-rated movies found for user {user\_id}.")*

*def recommendMoviesForUser(user\_id, num\_recs=10):*

*query = {*

*"size": num\_recs,*

*"query": {*

*"bool": {*

*"should": [*

*{"term": {"userId": user\_id}}*

*]*

*}*

*}*

*}*

*result = es.search(index="ratings", body=query)*

*hits = result['hits']['hits']*

*recommended\_movie\_ids = [hit['\_source']['movieId'] for hit in hits]*

*# Lấy thông tin chi tiết của các bộ phim đã được recommend và hiển thị ảnh poster*

*for movie\_id in recommended\_movie\_ids:*

*movie\_info = es.get(index="movies", id=movie\_id)*

*movie = movie\_info['\_source']*

*# Hiển thị tên bộ phim*

*print(f"Movie: {movie['title']} (tmdbId: {movie['tmdbId']})")*

*# Hiển thị ảnh poster nếu có*

*poster\_url = get\_poster\_url(movie['tmdbId'])*

*if poster\_url != "NA":*

*display(Image(poster\_url, width=150))*

*else:*

*print("No poster available for this movie.")*

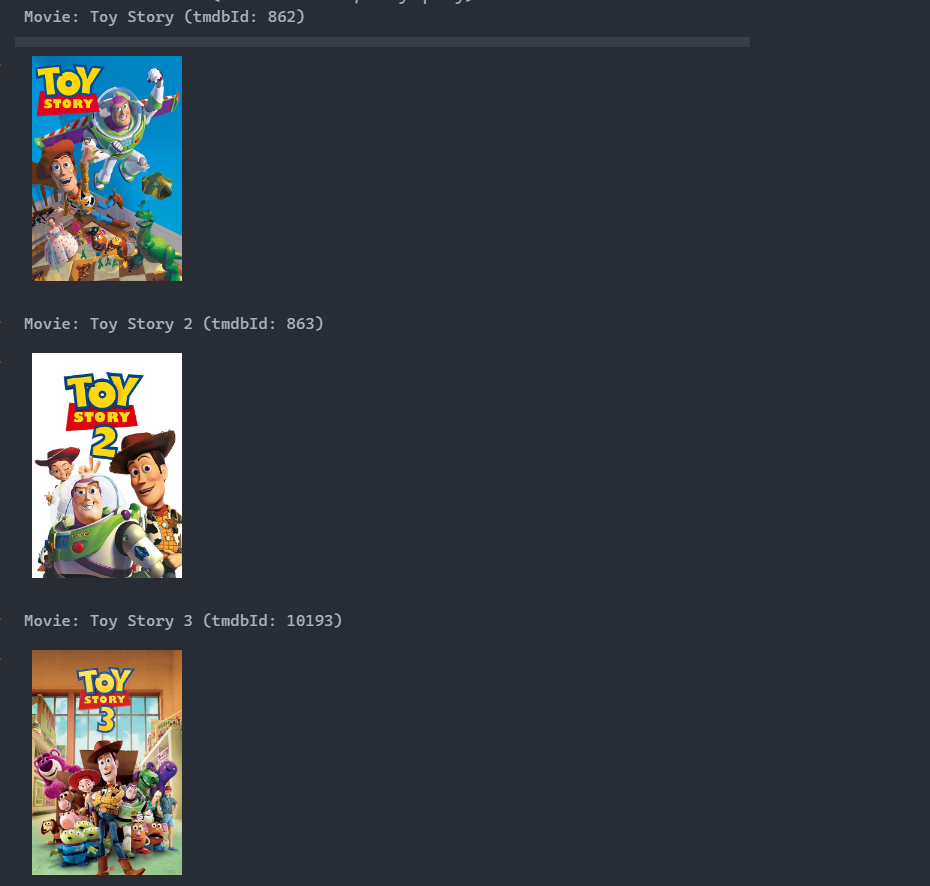
Sử dụng những hàm trên

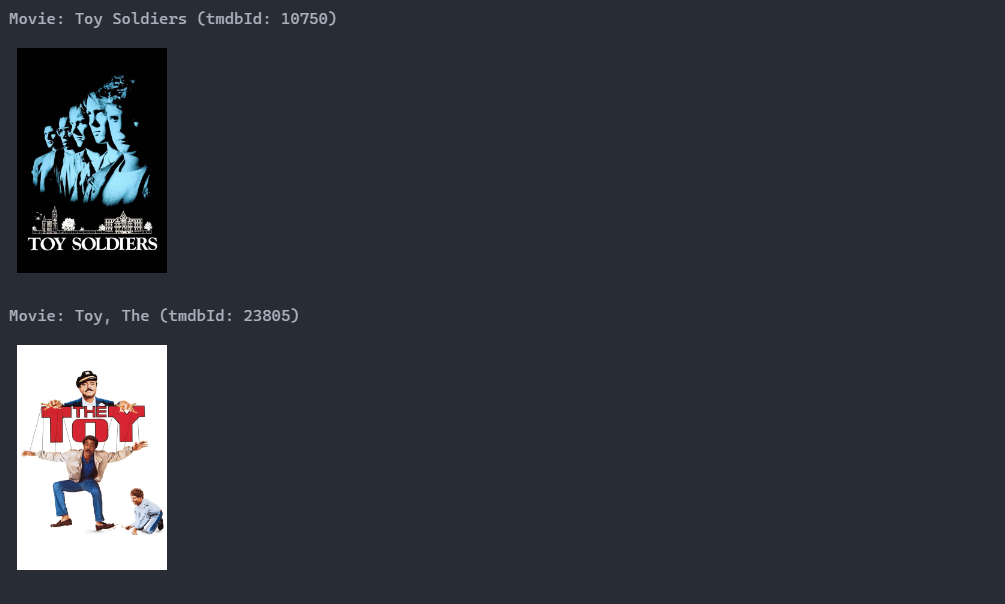
Recommend danh sách các bộ phim tương tự

[IN]

*recommendSimilarMoviesWithDetails(10, num\_recs=5, q="toy story")*

[OUT]



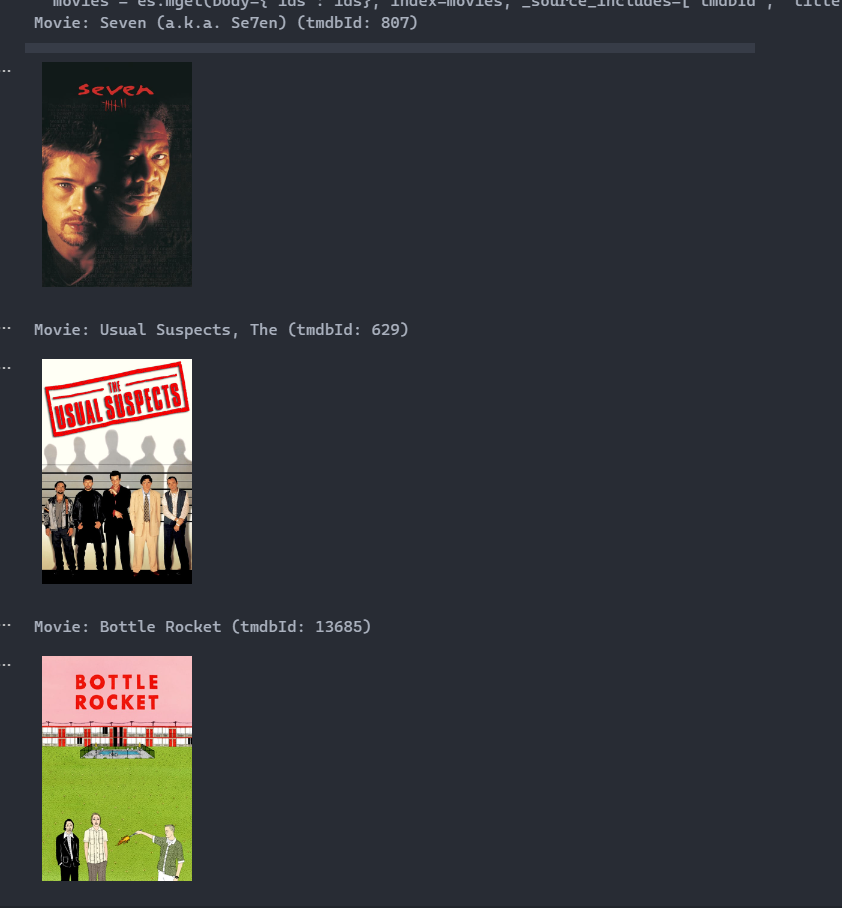


Recommend danh sách các bộ phim được đánh giá cao bởi người dùng

[IN]

*getMoviesRatedByUser(1, num\_movies=5)*

[OUT]



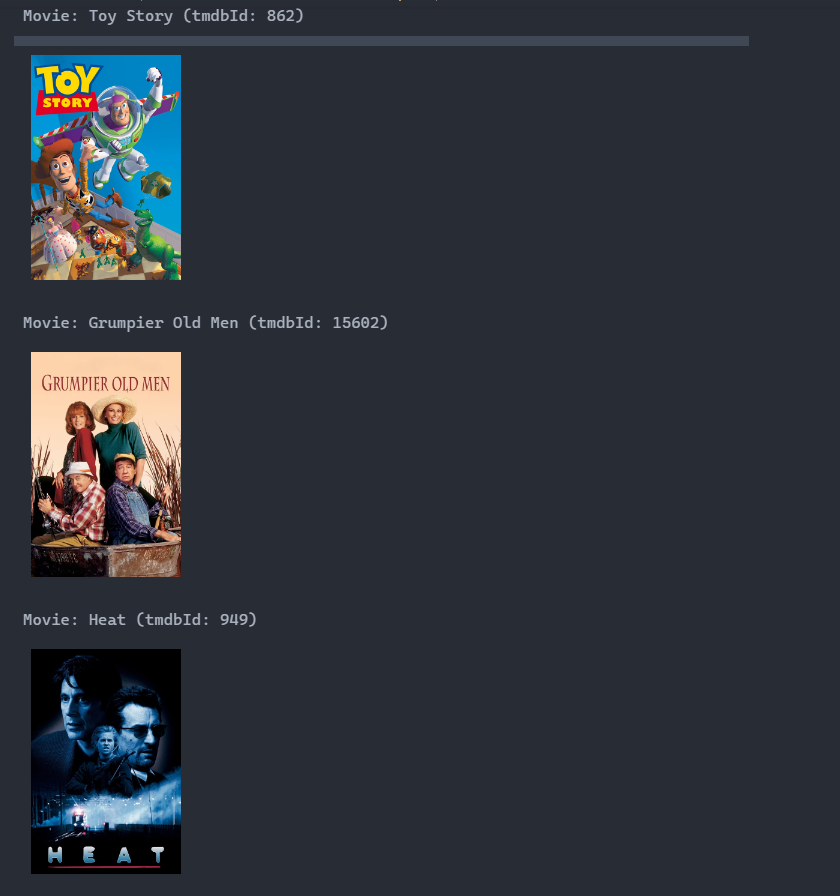


Gợi ý cho người dùng đó dựa trên thông tin đánh giá của họ

[IN]

*recommendMoviesForUser(1, num\_recs=5)*

[OUT]





* 1. ĐÁNH GIÁ VÀ NHẬN XÉT

**Ưu điểm:**

Tìm kiếm nhanh chóng: Elasticsearch được thiết kế để tìm kiếm dữ liệu nhanh chóng, đặc biệt là với tập dữ liệu lớn. Nó sử dụng cơ chế index và tìm kiếm ngược (inverted index) để cải thiện hiệu suất tìm kiếm.

Tích hợp với nhiều ngôn ngữ và ứng dụng: Elasticsearch cung cấp các API và thư viện client cho nhiều ngôn ngữ lập trình, điều này làm cho nó dễ dàng tích hợp vào các ứng dụng sử dụng các ngôn ngữ khác nhau.

Phân tán và mở rộng dễ dàng: Elasticsearch cho phép bạn mở rộng hệ thống một cách dễ dàng bằng cách thêm nhiều nút vào cụm (cluster). Điều này giúp cải thiện khả năng mở rộng và khả năng chịu lỗi.

Hỗ trợ tìm kiếm đa dạng: Elasticsearch hỗ trợ nhiều loại truy vấn tìm kiếm, bao gồm tìm kiếm full-text, tìm kiếm phù hợp (phrase matching), tìm kiếm đa trường (multi-field search), và nhiều tính năng khác.

Xây dựng trên Lucene: Elasticsearch dựa trên Lucene, một thư viện tìm kiếm mạnh mẽ. Điều này đảm bảo rằng nó có nhiều tính năng mạnh mẽ và ổn định.

**Nhược điểm:**

Học cách sử dụng: Elasticsearch có một học hỏi đôi chút ngưỡng cao, đặc biệt đối với người mới bắt đầu. Nó yêu cầu người dùng nắm vững các khái niệm và cách sử dụng các API của nó.

Yêu cầu tài nguyên hệ thống cao: Elasticsearch đòi hỏi tài nguyên phần cứng và lưu trữ đáng kể để hoạt động hiệu quả. Một cụm Elasticsearch phân tán có thể yêu cầu nhiều máy chủ để đảm bảo hiệu suất.

Khả năng bảo mật mặc định hạn chế: Elasticsearch có khả năng bảo mật mặc định hạn chế và cần phải cấu hình một số biện pháp bảo mật thêm để đảm bảo an toàn dữ liệu.

Phiên bản không tương thích: Elasticsearch thường phát triển và cập nhật liên tục, điều này có thể dẫn đến sự không tương thích giữa các phiên bản khác nhau của Elasticsearch và các plugin của nó.

Quản lý dữ liệu phức tạp: Quản lý và duy trì dữ liệu trong Elasticsearch có thể phức tạp, đặc biệt là với các cụm lớn hoặc trong các mô hình tải công việc cao.

1. KẾT LUẬN

Trong quá trình nghiên cứu và phát triển hệ thống Recommend Phim, nhóm em đã tiến hành các bước quan trọng để tạo ra một giải pháp recommendation linh hoạt và hiệu quả. Từ việc thu thập dữ liệu phim đến xây dựng mô hình recommendation sử dụng Apache Spark, mỗi giai đoạn đóng góp vào việc cung cấp trải nghiệm giải trí tối ưu cho người dùng.

Việc tích hợp Elasticsearch vào hệ thống đã mang lại một bước tiến quan trọng. Nhóm đã xây dựng và cấu hình một cơ sở dữ liệu Elasticsearch để lưu trữ thông tin về các bộ phim cũng như các chỉ mục tìm kiếm. Elasticsearch đóng vai trò quan trọng trong việc nâng cao khả năng tìm kiếm phim và cung cấp thông tin chi tiết một cách dễ dàng, từ đó giúp người dùng tìm thấy nội dung mong muốn một cách thuận tiện.

Tích hợp Elasticsearch cùng với Apache Spark tạo nên một sự kết hợp mạnh mẽ, mở ra không gian cho việc cải thiện hệ thống recommendation theo sở thích cá nhân của người dùng. Không chỉ cung cấp các gợi ý phim chính xác, mà còn tăng cường khả năng tương tác và khám phá trong hệ thống.

Và nhóm tin tưởng rằng việc tích hợp Elasticsearch và Spark đã mang lại một bước tiến quan trọng cho hệ thống recommend phim và hy vọng rằng nó sẽ mang lại giá trị lớn cho người dùng. Chân thành cảm ơn sự đóng góp và hỗ trợ từ tất cả các thành viên trong nhóm.

Nhóm em cũng cảm ơn thầy đã theo dõi bài báo cáo của chúng em.

TÀI LIỆU THAM KHẢO

CÁC GiÁO TRÌNH THAM KHẢO

Fast Data Processing with Spark, 2nd Edition

Advanced Analytics with Spark, 2nd Edition

CÁC TRANG WEB THAM KHẢO

<https://longcnttbkhn.github.io/huong-dan-spark-co-ban-cho-nguoi-moi/>

<https://www.tutorialspoint.com/elasticsearch/elasticsearch_search_apis.htm>