**BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG**

**TRƯỜNG ĐH TÀI NGUYÊN VÀ MÔI TRƯỜNG TP. HCM**

**KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**  
A logo with a person holding a book

Description automatically generated

**Xử Lý Video Streaming Phát Hiện Chuyển Động Sử Dụng OpenCV, KafKa, Spark**

Giảng viên hướng dẫn: Phạm Trọng Huynh

Môn học: Big Data

Thành viên của nhóm 13:

Huỳnh Nhật Huy | 1050080136

Trần Duy Kha | 1050080139

Trần Quốc Bảo | 1050080129

Lớp: 10ĐH\_CNTT4

***TP. Hồ Chí Minh, 6 tháng 11 năm 2023***

**BỘ TÀI NGUYÊN VÀ MÔI TRƯỜNG**

**TRƯỜNG ĐH TÀI NGUYÊN VÀ MÔI TRƯỜNG TP. HCM**

**KHOA HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**  
A logo with a person holding a book

Description automatically generated

**Xử Lý Video Streaming Phát Hiện Chuyển Động Sử Dụng OpenCV, KafKa, Spark**

Giảng viên hướng dẫn: Phạm Trọng Huynh

Môn học: Big Data

Thành viên của nhóm 3:

Huỳnh Nhật Huy | 1050080136

Trần Duy Kha | 1050080139

Trần Quốc Bảo | 1050080129

Lớp: 10ĐH\_CNTT4

***TP. Hồ Chí Minh, 6 tháng 11 năm 2023***

**THÔNG TIN NHÓM 3**

|  |  |  |
| --- | --- | --- |
| **Tên đề tài** | **Xử Lý Video Streaming Phát Hiện Chuyển Động Sử Dụng OpenCV, KafKa, Spark**  **Video Youtube:** [**Link**](https://www.youtube.com/watch?v=ODyoW6_dxKU) | **Đóng góp của thành viên** |
| **Họ và Tên thành viên 1** | Trần Quốc Bảo – 1050080139 | Đóng góp: Cài đặt thực nghiệm  Điểm tự đánh giá quá trình: 4  Điểm tự đánh giá cuối kì: 4 |
| **Họ và Tên thành viên 2** | Trần Duy Kha – 1050080139  Không có mô tả. | Đóng góp:  Điểm tự đánh giá quá trình: 4  Điểm tự đánh giá cuối kì: 4 |
| **Họ và Tên thành viên 3** | Huỳnh Nhật Huy -1050080136 | Đóng góp:  Điểm tự đánh giá quá trình: 4  Điểm tự đánh giá cuối kì: 4 |

**MỤC LỤC**

[1. Giới thiệu Big Data 4](#_Toc149514540)

[1.1. Định nghĩa 4](#_Toc149514541)

[1.2. Đặt trưng của Big data 6](#_Toc149514542)

[1.3. Những vấn đề liên quan 8](#_Toc149514543)

[2. Bài toán 15](#_Toc149514544)

[3. Giải pháp 16](#_Toc149514545)

[3.1. Apache Spark 16](#_Toc149514546)

[3.2. Kafka 18](#_Toc149514547)

[3.3. OpenCV 20](#_Toc149514548)

[3.4. Mô hình giải pháp 22](#_Toc149514549)

[4. Đề xuất 22](#_Toc149514550)

[5. Đánh giá 22](#_Toc149514551)

[6. Kết luận 22](#_Toc149514552)

**BÁO CÁO TÓM TẮT**

* Lý do chọn đề tài:

Công nghệ đã mang lại sự bùng nổ chưa từng có về dữ liệu phi cấu trúc. Các nguồn như thiết bị di động, trang web, phương tiện truyền thông xã hội, bộ máy khoa học, vệ tinh, thiết bị IoT và camera giám sát đang tạo ra số lượng lớn hình ảnh và video mỗi giây.

Quản lý và phân tích hiệu quả dữ liệu này là một thách thức. Hãy xem xét mạng lưới camera giám sát video của một thành phố. Việc giám sát luồng video của mọi camera để khám phá bất kỳ đối tượng hoặc sự kiện quan tâm nào là không thực tế và không hiệu quả. Thay vào đó, thư viện thị giác máy tính (CV) xử lý các luồng video này và cung cấp khả năng phân tích video và phát hiện đối tượng thông minh.

Tuy nhiên, hệ thống CV truyền thống có những hạn chế. Trong các hệ thống phân tích video truyền thống, máy chủ có thư viện CV thu thập và xử lý dữ liệu cùng một lúc và do đó lỗi máy chủ sẽ mất dữ liệu phát trực tuyến video. Phát hiện lỗi nút và chuyển xử lý sang nút khác có thể dẫn đến dữ liệu bị phân mảnh.

Nhiều tác vụ thúc đẩy việc sử dụng các công nghệ dữ liệu lớn trong phân tích luồng video: xử lý song song và theo yêu cầu các luồng video quy mô lớn, trích xuất một bộ thông tin khác từ khung video, phân tích dữ liệu với các thư viện học máy khác nhau, đưa dữ liệu được phân tích đến các thành phần khác nhau của ứng dụng để xử lý thêm và xuất dữ liệu được xử lý ở các định dạng khác nhau.

Nhóm 03 chọn đề tài **Xử Lý Video Streaming Phát Hiện Chuyển Động Sử Dụng OpenCV, KafKa, Spark** với mong muốn tìm hiểu về một trong số các kỹ thuật, công nghệ được sử dụng trong mô hình phát hiện chuyển động cho người dùng trên nền tảng dữ liệu lớn.

* Tầm quan trọng:

Đề tài phù hợp với nhu cầu xã hội, ứng dụng thực tế, hiện nay đang sử dụng rộng rãi trên các ứng dụng để phát hiện khuôn mặt, phát hiện xe, đếm số người đi bộ, hệ thống bảo mật và xe không người lái.

Các thành viên trong nhóm có cơ hội đi sâu tìm hiểu về một ứng dụng thực tế của lĩnh vực big data: cách thức setup, cài đặt một hệ thống cluster, các kỹ thuật, công nghệ được sử dụng trong bài toán bắt chuyển động.

Các nội dung và kết quả

* Mô tả bài toán
* Giải pháp, mô hình
* Các công nghệ sử dụng
* Cài đặt
* Demo
* Đánh giá

**BÁO CÁO CHI TIẾT**

1. **Giới thiệu Big Data**
   1. Định nghĩa

Dữ liệu lớn, hay Big Data, là một thuật ngữ được sử dụng để mô tả việc xử lý và quản lý một tập hợp dữ liệu rất lớn và phức tạp, mà các phương pháp xử lý dữ liệu truyền thống không thể đáp ứng được. Dữ liệu lớn đặt ra một số thách thức và yêu cầu sử dụng các công nghệ và phương pháp đặc biệt để hiểu và tận dụng giá trị từ dữ liệu này. Dưới đây là một số chi tiết thêm về dữ liệu lớn:

1. Phạm vi và Kích thước: Dữ liệu lớn thường bao gồm các tập dữ liệu có kích thước vượt xa khả năng của các công cụ phần mềm thông thường để xử lý. Kích thước này thay đổi theo thời gian và có thể bao gồm từ một vài terabytes đến hàng petabytes của dữ liệu.
2. Thách thức của Dữ liệu lớn: Các thách thức của dữ liệu lớn bao gồm phân tích, thu thập, giám sát, tìm kiếm, chia sẻ, lưu trữ, truyền nhận, trực quan, truy vấn và tính riêng tư. Dữ liệu lớn yêu cầu các phương pháp và công nghệ đặc biệt để khai phá giá trị từ các tập dữ liệu phức tạp và đa dạng.
3. Ứng dụng của Dữ liệu lớn: Dữ liệu lớn có ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm kinh doanh, y tế, quảng cáo, chính phủ và nghiên cứu khoa học. Nó giúp tìm ra các tương quan mới, dự đoán xu hướng kinh doanh, ngừa bệnh tật, chống tội phạm và nhiều ứng dụng khác.
4. Nguồn dữ liệu lớn: Dữ liệu lớn đang tăng nhanh do sự phát triển của các thiết bị kết nối internet, chẳng hạn như thiết bị di động, cảm biến, và thiết bị thu thập dữ liệu. Khả năng lưu trữ thông tin của thế giới đã tăng đáng kể trong những năm gần đây.
5. Hệ quản trị cơ sở dữ liệu: Công việc quản lý và xử lý dữ liệu lớn đôi khi yêu cầu hàng trăm hoặc hàng nghìn máy chủ. Hệ quản trị cơ sở dữ liệu truyền thống không đủ khả năng xử lý dữ liệu lớn, vì vậy cần sử dụng các hệ thống và công nghệ mới.
6. 3Vs và bổ sung: Gartner đã định nghĩa dữ liệu lớn bằng mô hình "3Vs" - Khối lượng (Volume), Tốc độ (Velocity), và Đa dạng (Variety). Ngoài ra, một số tổ chức đã bổ sung thêm tính xác thực (Veracity) để mô tả về dữ liệu lớn.
7. Machine Learning và Digital Footprint: Dữ liệu lớn thường liên quan đến sử dụng máy học để phân tích và rút ra thông tin từ dữ liệu. Nó cũng thường xuất phát từ dấu chân kỹ thuật số của các hoạt động trực tuyến và tương tác kỹ thuật số.
8. Tóm lại, dữ liệu lớn đóng vai trò quan trọng trong nhiều khía cạnh của cuộc sống hiện đại và yêu cầu sử dụng các công nghệ và phương pháp đặc biệt để xử lý và hiểu rõ giá trị của nó.
   1. Đặt trưng của Big data

Dữ liệu lớn (big data) có những đặc trưng cơ bản sau:

1. Lượng dữ liệu lớn: Big data thường bao gồm lượng dữ liệu lớn, thậm chí là cả petabytes hoặc exabytes. Dữ liệu này có thể bao gồm các loại dữ liệu như văn bản, hình ảnh, âm thanh, video, dữ liệu máy móc, dữ liệu xã hội, và nhiều loại dữ liệu khác.
2. Tốc độ nhanh: Dữ liệu lớn thường được tạo ra hoặc thu thập với tốc độ nhanh. Ví dụ, dữ liệu từ các thiết bị IoT, trạng thái giao dịch tài chính, hoặc dữ liệu mạng xã hội có thể được tạo ra liên tục và cần xử lý ngay sau khi được tạo ra.
3. Đa dạng về nguồn và định dạng: Big data thường xuất phát từ nhiều nguồn khác nhau và có nhiều định dạng khác nhau. Nó có thể bao gồm dữ liệu cấu trúc, bất cấu trúc, và bán cấu trúc, chẳng hạn như dữ liệu trong cơ sở dữ liệu SQL, dữ liệu JSON, dữ liệu hình ảnh, dữ liệu video, và nhiều loại dữ liệu khác.
4. Khả năng mở rộng (scalability): Hệ thống xử lý dữ liệu lớn phải có khả năng mở rộng để có thể xử lý lượng dữ liệu ngày càng lớn. Điều này thường đòi hỏi sự sử dụng các nền tảng phân tán và công cụ xử lý dữ liệu phân tán.
5. Phân tán và phân tầng (distributed and layered): Dữ liệu lớn thường được lưu trữ và xử lý trên các hệ thống phân tán và có kiến trúc phân tầng. Các hệ thống này có thể bao gồm nhiều máy chủ hoặc nút, và dữ liệu được chia nhỏ để phân tán trên chúng.
6. Tính thất thoát và không chắc chắn (variability and uncertainty): Dữ liệu lớn thường có tính thất thoát (variability) và không chắc chắn (uncertainty). Dữ liệu có thể bị thiếu, không chính xác, hoặc bị nhiễu, và đòi hỏi quá trình xử lý đặc biệt để xử lý các vấn đề này.
7. Khả năng trích xuất thông tin (information extraction): Mục tiêu của việc xử lý dữ liệu lớn thường là trích xuất thông tin hữu ích và hiểu biết từ dữ liệu. Điều này có thể bao gồm việc áp dụng các kỹ thuật như khai phá dữ liệu (data mining), học máy (machine learning), và thị giác máy tính (computer vision).
8. Thời gian thực (real-time): Nhiều ứng dụng big data đòi hỏi xử lý dữ liệu trong thời gian thực hoặc gần thời gian thực. Điều này đặt ra yêu cầu về tốc độ và hiệu suất của hệ thống xử lý dữ liệu.
9. Bảo mật và quyền riêng tư (security and privacy): Bảo mật và quyền riêng tư của dữ liệu lớn là một vấn đề quan trọng. Vì dữ liệu này thường là quý giá và nhạy cảm, việc bảo vệ nó đòi hỏi các biện pháp bảo mật cơ sở dữ liệu và quản lý quyền truy cập.
10. Khả năng tìm kiếm (searchability): Dữ liệu lớn thường đòi hỏi khả năng tìm kiếm hiệu quả để truy xuất thông tin cụ thể từ lượng dữ liệu khổng lồ.

Việc hiểu và quản lý các đặc trưng này là quan trọng khi làm việc với dữ liệu lớn để có thể tận dụng các cơ hội và giải quyết các thách thức liên quan đến nó.

* 1. Những vấn đề liên quan

1. Các tính năng của Big Data
2. Khả năng lưu trữ: Lượng dữ liệu lớn đòi hỏi hệ thống lưu trữ mạnh mẽ và khả năng mở rộng để chứa dữ liệu. Câu hỏi liên quan bao gồm việc chọn lựa giữa lưu trữ on-premises hoặc cloud, thiết kế cơ sở dữ liệu phù hợp, và cách quản lý lưu trữ dữ liệu dài hạn.
3. Tính toàn vẹn của dữ liệu: Dữ liệu lớn có thể có sự không chắc chắn và thất thoát. Điều này đặt ra vấn đề về tính toàn vẹn của dữ liệu và cách xử lý dữ liệu bị lỗi hoặc mất mát trong quá trình truyền tải và lưu trữ.
4. Tính an ninh và quyền riêng tư: Dữ liệu lớn thường chứa thông tin quý giá và nhạy cảm. Bảo mật và quyền riêng tư là một vấn đề quan trọng. Cần có biện pháp bảo vệ dữ liệu khỏi truy cập trái phép và đảm bảo tuân thủ các quy định về quyền riêng tư.
5. Khả năng xử lý dữ liệu: Dữ liệu lớn đòi hỏi khả năng xử lý nhanh chóng và hiệu quả. Cần phải chọn lựa công cụ và nền tảng phù hợp để xử lý dữ liệu, bao gồm việc sử dụng hệ thống phân tán và công cụ xử lý dữ liệu.
6. Tìm kiếm và truy xuất dữ liệu: Dữ liệu lớn thường khó để tìm kiếm và truy xuất thông tin. Cần phải xây dựng cơ sở dữ liệu hiệu quả và cơ chế tìm kiếm để có thể truy xuất dữ liệu một cách nhanh chóng và hiệu quả.
7. Quản lý meta dữ liệu: Metadữ liệu là thông tin về dữ liệu, bao gồm nguồn gốc, định dạng, quyền truy cập và nhiều thông tin khác. Quản lý metadữ liệu đóng vai trò quan trọng trong việc tổ chức và quản lý dữ liệu lớn.
8. Phân tích dữ liệu: Phân tích dữ liệu lớn đòi hỏi sử dụng các kỹ thuật và công cụ phân tích mạnh mẽ, bao gồm khai phá dữ liệu, học máy, và thị giác máy tính. Khả năng tìm ra thông tin hữu ích và kiến thức từ dữ liệu là một thách thức lớn.
9. Quản lý chi phí: Lưu trữ và xử lý dữ liệu lớn có thể đòi hỏi chi phí lớn. Việc quản lý chi phí và tối ưu hóa cơ sở dữ liệu và cơ sở hạ tầng là một vấn đề quan trọng.
10. Sự nhất quán dữ liệu: Khi dữ liệu được cập nhật từ nhiều nguồn, đảm bảo sự nhất quán của dữ liệu là một thách thức. Cần có các giải pháp để đồng bộ hóa và kiểm tra tính nhất quán của dữ liệu.
11. Xử lý dữ liệu thời gian thực: Một số ứng dụng big data đòi hỏi xử lý dữ liệu trong thời gian thực hoặc gần thời gian thực. Điều này đặt ra yêu cầu về tốc độ và hiệu suất của hệ thống.
12. Tối ưu hóa hiệu suất và mở rộng: Xây dựng hệ thống có khả năng mở rộng và tối ưu hóa hiệu suất là một vấn đề quan trọng để đảm bảo rằng hệ thống có thể xử lý lượng dữ liệu lớn một cách hiệu quả.

Những vấn đề này đòi hỏi sự chú tâm và kỹ thuật để xử lý và quản lý dữ liệu lớn một cách hiệu quả và hiệu quả.

1. Sử dụng phần mềm và công cụ hỗ trợ

Hệ sinh thái Hadoop:

Hadoop được coi là xương sống trong các dự án Big Data. Thư viện phần mềm Hadoop là một nền tảng cho phép các tập dữ liệu lớn được xử lý phân tán thông qua các cụm máy tính sử dụng mô hình lập trình đơn giản. Nó được thiết kế để mở rộng từ một máy chủ lên hàng nghìn máy, mỗi máy chủ có khả năng cung cấp năng lực tính toán và lưu trữ cục bộ.

Dự án bao gồm một số mô-đun:

* Hadoop Common, các tiện ích phổ biến hỗ trợ các mô-đun Hadoop khác
* Hadoop Distributed File System, cung cấp quyền truy cập thông lượng cao vào dữ liệu ứng dụng
* Hadoop YARN, một nền tảng cho việc lập kế hoạch công việc và quản lý tài nguyên cụm
* Hadoop MapReduce, một hệ thống dựa trên YARN để xử lý song song các tập dữ liệu lớn.
* Apache Spark

Apache Spark là một nền tảng hệ thống tính toán mã nguồn mở phục vụ như một công cụ để xử lý Big Data trong Hadoop.

Spark đã trở thành một trong những nền tảng xử lý phân tán dữ liệu quan trọng và có thể được triển khai theo nhiều cách khác nhau.

Nó cung cấp các ràng buộc nguyên gốc cho Java, Scala, Python (đặc biệt là Python Anaconda) và ngôn ngữ lập trình R (R đặc biệt phù hợp với dữ liệu lớn), đồng thời cũng hỗ trợ SQL, dữ liệu luồng, machine learning và xử lý đồ thị.

Cơ sở dữ liệu

Cơ sở dữ liệu NoSQL lưu trữ và quản lý dữ liệu theo cách linh hoạt và xử lý tốc độ cao. Không giống như cơ sở dữ liệu SQL, nhiều cơ sở dữ liệu NoSQL có thể được thu nhỏ theo chiều ngang trên hàng trăm hoặc hàng nghìn máy chủ.

Cơ sở dữ liệu bộ nhớ trong

Cơ sở dữ liệu bộ nhớ trong (IMDB) là một hệ thống quản lý cơ sở dữ liệu chủ yếu dựa vào bộ nhớ chính, thay vì đĩa, để lưu trữ dữ liệu. Cơ sở dữ liệu bộ nhớ trong nhanh hơn so với cơ sở dữ liệu trên đĩa được tối ưu hóa, một lựa chọn quan trọng cho việc sử dụng phân tích Big Data tạo kho dữ liệu.

1. Các trường hợp cần sử dụng big data

Big data có nhiều ứng dụng tiềm năng trong nhiều lĩnh vực khác nhau. Dưới đây là một số trường hợp cần sử dụng big data:

1. Tiếp thị và Quảng cáo: Sử dụng dữ liệu lớn để phân tích hành vi của khách hàng, dự đoán xu hướng mua sắm, tối ưu hóa chiến dịch quảng cáo, và cải thiện trải nghiệm khách hàng.
2. Dự đoán và Dự báo: Sử dụng dữ liệu lớn để dự đoán sự cố trong lĩnh vực như thời tiết, tài chính, y tế, và giao thông. Dữ liệu lớn có thể giúp cải thiện độ chính xác của các dự báo.
3. Y tế và Chăm sóc sức khỏe: Sử dụng dữ liệu lớn để theo dõi và đánh giá sức khỏe của bệnh nhân, phát hiện dấu hiệu sớm của bệnh, tối ưu hóa chăm sóc sức khỏe, và nghiên cứu y tế.
4. Học máy và Trí tuệ nhân tạo: Sử dụng dữ liệu lớn để phát triển mô hình học máy và trí tuệ nhân tạo, bao gồm việc xây dựng hệ thống nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, và dự đoán.
5. IoT (Internet of Things): Thiết bị IoT tạo ra lượng dữ liệu lớn từ các cảm biến và thiết bị kết nối. Dữ liệu này có thể được sử dụng để quản lý và kiểm soát các hệ thống tự động và thông minh.
6. Tài chính và Ngân hàng: Sử dụng dữ liệu lớn để phân tích giao dịch tài chính, phát hiện gian lận, quản lý rủi ro, và tối ưu hóa quản lý tài chính.
7. Dự đoán và Quản lý chuỗi cung ứng: Sử dụng dữ liệu lớn để dự đoán nhu cầu và biến đổi trong chuỗi cung ứng, quản lý hàng tồn kho, và tối ưu hóa quá trình vận chuyển.
8. Thị trường Tài chính: Sử dụng dữ liệu lớn để thực hiện phân tích tài chính, giao dịch tài chính, và quản lý rủi ro đầu tư.
9. Giáo dục: Sử dụng dữ liệu lớn để theo dõi tiến bộ học tập của học sinh, cung cấp giáo dục tùy chỉnh, và nâng cao chất lượng giáo dục.
10. Khoa học và Nghiên cứu: Sử dụng dữ liệu lớn để nghiên cứu và phân tích dữ liệu khoa học trong các lĩnh vực như thiên văn học, việc làm và kinh tế học.
11. An ninh và Quân sự: Sử dụng dữ liệu lớn để theo dõi và đánh giá sự kiện an ninh và quân sự, dự đoán hành vi đe dọa, và cải thiện an ninh quốc gia.
12. Tùy chỉnh sản phẩm và Dịch vụ: Sử dụng dữ liệu lớn để tạo ra sản phẩm và dịch vụ tùy chỉnh dựa trên hành vi và sở thích của khách hàng.

Những ứng dụng này chỉ là một phần nhỏ của những trường hợp mà big data có thể được áp dụng. Sử dụng big data có thể cải thiện hiệu suất và hiệu quả trong nhiều ngành và mang lại giá trị lớn cho doanh nghiệp và tổ chức.

# Bài toán

Để xử lý đáng tin cậy và xử lý hiệu quả quy trình video luồng dữ liệu, yêu cầu một hệ thống phân tích có thể mở rộng, chịu lỗi, kết thúc an toàn. Phân tích mô-đun video lớn của luồng video yêu cầu một hệ thống mạnh mẽ được hỗ trợ bởi các công nghệ dữ liệu lớn.

Các công nghệ mở nguồn như OpenCV, Kafka và Spark có thể được sử dụng để xây dựng hệ thống phân phối và chịu lỗi cho video luồng phân tích.

Phát triển chuyển động là quá trình tìm kiếm sự thay đổi vị trí của một vật thể (thường là con người) trong môi trường xung quanh.

Nó được sử dụng chủ yếu trong các video giám sát hệ thống liên tục theo dõi một khu vực cụ thể.

Một thuật toán được cung cấp bởi video dữ liệu nguồn phân tích CV thư viện được gửi bởi một máy ảnh tương tự và tìm kiếm bất kỳ chuyển động nào.

Phát hiện một sự kiện kích hoạt chuyển động có thể gửi tin nhắn đến ứng dụng hoặc cảnh báo người dùng.

Bộ thu thập luồng video nhận dữ liệu luồng video từ một cụm camera IP.

Thành phần này tuần tự hóa các khung hình video để truyền bộ đệm dữ liệu, đây là hàng đợi dữ liệu có khả năng chịu lỗi để truyền dữ liệu video.

Bộ xử lý luồng video sử dụng dữ liệu luồng từ bộ đệm và xử lý nó.

Thành phần này sẽ áp dụng các thuật toán xử lý video để phát hiện chuyển động trong dữ liệu luồng video. Cuối cùng, các tệp hình ảnh hoặc dữ liệu đã xử lý sẽ được lưu trữ trong bộ chứa S3 hoặc thư mục HDFS .

Hệ thống xử lý luồng video này đã được thiết kế bằng cách sử dụng các khung OpenCV, Apache Kafka và Apache Spark .

# Giải pháp

* 1. Các công cụ sử dụng

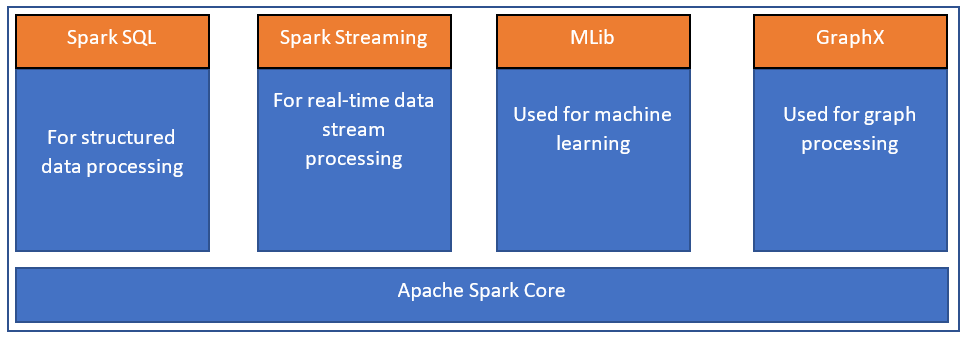
Giới thiệu Apache Spark

Apache Spark là một open source cluster computing framework được phát triển sơ khởi vào năm 2009 bởi AMPLab tại đại học California, Berkeley.

Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Spark cho phép xây dựng và phân tích nhanh các mô hình dự đoán.

Spark cung cấp khả năng truy xuất toàn bộ dữ liệu cùng lúc, và còn cung cấp tính năng streaming, được dùng để xây dựng các mô hình real-time bằng cách nạp toàn bộ dữ liệu vào bộ nhớ.

Spark cho phép phân chia tác vụ này thành những phần dễ quản lý hơn. Sau đó, Spark sẽ chạy các tác vụ này trong bộ nhớ, trên các cluster của nhiều server khác nhau để khai thác tốc độ truy xuất nhanh từ RAM. Spark sử dụng API Resilient Distributed Dataset (RDD) để xử lý dữ liệu.



Đặc điểm các thành phần Spark:

* Apache Spark Core: Spark Core là thành phần cốt lõi thực thi cho tác vụ cơ bản làm nền tảng cho các chức năng khác. Nó cung cấp khả năng tính toán trên bộ nhớ và datase trong bộ nhớ hệ thống lưu trữ ngoài.
* Spark SQL: Là một thành phần nằm trên Spark Core nó cung cấp một sự ảo hóa mới cho dữ liệu là SchemaRDD, hỗ trợ các dữ liệu có cấu trúc và bán cấu trúc. Spark SQL tập trung vào xử lý dữ liệu có cấu trúc, sử dụng cách tiếp cận khung dữ liệu tương tự như R và Python (trong Pandas). Spark SQL mang lại sức mạnh của Apache Spark cho các nhà phân tích cũng như nhà phát triển dữ liệu bằng cách cung cấp giao diện với cú pháp SQL để truy vấn
* Spark Streaming: Cho phép thực hiện phân tích xử lý trực tuyến xử lý theo lô.
* MLlib (Machine Learning Library): MLlib là một nền tảng học máy phân tán bên trên Spark do kiến trúc phân tán dựa trên bộ nhớ. Theo các so sánh benchmark Spark MLlib nhanh hơn chín lần so với phiên bản chạy trên Hadoop (Apache Mahout)
* GrapX: Grapx là nền tảng xử lý đồ thị dựa trên Spark. Nó cung cấp các Api để diễn tả các tính toán trong đồ thị

Apache Spark có các tính năng đặc trưng sau đây:

* Tốc độ: Spark có thể chạy trên cụm Hadoop và có thể chạy nhanh hơn 100 lần khi chạy trên bộ nhớ RAM, và nhanh hơn 10 lần khi chạy trên ổ cứng. Bằng việc giảm số thao tác đọc nghi lên đĩa cứng. Nó lưu trữ trực tiếp dữ liệu xử lý lên bộ nhớ
* Hỗ trợ đa ngôn ngữ: Spark cung cấp các API có sẵn cho các ngôn ngữ Java, Scala, hoặc Python. Do đó, bạn có thể viết các ứng dụng bằng nhiều các ngôn ngữ khác nhau. Spark đi kèm 80 truy vấn tương tác mức cao.
* Phân tích nâng cao: Spark không chỉ hỗ trợ ‘Map’ và ‘Reduce’.
* Nó còn hỗ trợ truy vấn SQL, xử lý theo Stream, học máy, và các thuật toán đồ thị (Graph)
* Bộ dữ liệu phân tán linh hoạt (RDD) là một cơ sở dữ liệu cấu trúc của Spark. Nó là một tập hợp phân tích bất kỳ biến thể của một đối số icon. Mỗi tập dữ liệu trong RDD được chia thành nhiều vùng logic. Có thể tính toán trên các nút khác nhau của một cụm máy chủ (cụm). RDD có thể chứa bất kỳ loại dữ liệu nào của Python, Java hoặc Scala đối tượng, bao gồm các loại dữ liệu do người dùng định nghĩa.
  1. Kafka

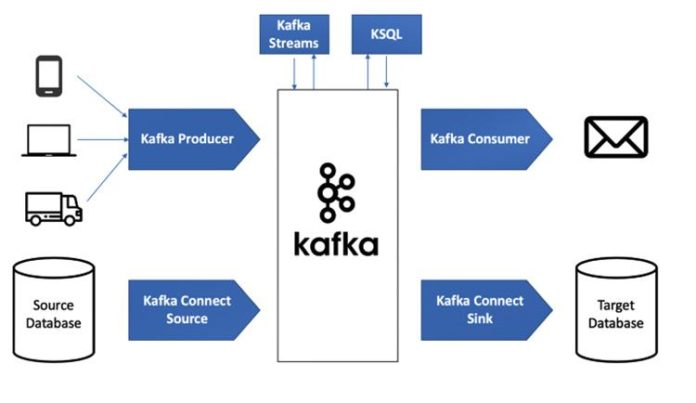
Giới thiệu Kafka

Apache Kafka là một nền tảng theo kiến trúc phân tán cho phép lưu trữ sự kiện và xử lý dữ liệu luồng (streaming) mã nguồn mở được phát triển bởi Apache Software Foundation được viết bằng Java và Scala.

Kafka ban đầu được phát triển bởi LinkedIn và sau đó được mở nguồn vào đầu năm 2011. Jay Kreps,  Neha Narkhede và Jun Raolà các đồng sáng lập nền tảng phần mềm Kafka.

Xuất phát từ dự án Apache Incubator vào ngày 23 tháng 10 năm 2012, phần mềm được Jay Kreps đặt tên theo tên nhà văn Franz Kafkavì theo anh nó là "một hệ thống được tối ưu hóa cho việc viết", và anh rất thích các tác phẩm của Kafka.

Nhiều hệ thống ứng dụng xuất bản dữ liệu số lượng lớn hoặc ứng dụng thời gian thực đã sử dụng Kafka, ví dụ quản lý đối sánh hành khách và tài xế tại Uber, nhiều dịch vụ thời gian thực trên toàn bộ Linkedln.



Kiến trúc Kafka lưu trữ các gói tin dạng cặp khóa-giá trị gửi đến từ nhiều tiến trình được gọi là *producer*. Dữ liệu có thể được phân hoạch thành các "phân vùng" (partition) trong các "chủ đề" (topic) khác nhau.

Trong một *phân vùng*, các gói tin được sắp xếp theo thứ tự nghiêm ngặt của chúng (vị trí của một gói tin trong một phân vùng), được lập chỉ mục và lưu trữ cùng với một mốc thời gian. Các tiến trình khác được gọi là *consumer* có thể đọc tin nhắn từ các *phân vùng*.

Để xử lý dữ liệu luồng stream, Kafka cung cấp Stream API cho phép viết các ứng dụng Java sử dụng dữ liệu từ Kafka và ghi kết quả trở lại Kafka. Apache Kafka cũng hoạt động với các hệ thống xử lý dữ liệu luồng (stream processing system) khác bên ngoài như Apache Apex, Apache Flink, Apache Spark, Apache Storm hay Apache NiFi.

Kafka chạy trên một cụm gồm một hoặc nhiều máy chủ (được gọi là *broker*) và các partition của tất cả các topic được phân phối trên các nút trong cụm máy chủ. Ngoài ra, các phân vùng được sao chép cho nhiều broker.

Kiến trúc này cho phép Kafka truyền tải các luồng thông điệp khổng lồ theo phương thức truyền tải an toàn, có khả năng chịu lỗi và cho phép nó thay thế một số hệ thống truyền tin thông thường như Java Message Service (JMS), Advanced Message Queueing Protocol (AMQP), v.v.

Kể từ phiên bản 0.11.0.0, Kafka cung cấp tính năng ghi phiên giao dịch (*transactional writes*), cho phép xử lý dữ liệu luồng chính xác một lần bằng cách sử dụng streams API.

Có năm API chính trong Kafka bao gồm:

* **Producer API** - Cho phép ứng dụng xuất bản các luồng bản ghi.
* **Consumer API** - Cho phép ứng dụng đăng ký các topic và xử lý các bản ghi luồng dữ liệu.
* **Connector API** - Thực thi các consumer và producer API cho phép liên kết các topic với các ứng dụng sẵn có.
* **Streams API** - API này chuyển đổi các dữ liệu luồng đầu vào thành đầu ra và tạo ra kết quả.
* **Admin API** - được sử dụng để quản lý các topic, broker, và các đối tượng Kafka khác.

Các consumer và producer API xây dựng dựa trên giao thức truyền tin của Kafka và cung cấp triển khai tham chiếu cho các ứng dụng khách consumer và producer client xây dựng trên Java.

Giao thức truyền tin cơ bản là một giao thức dạng nhị phân (binary protocol) mà các nhà phát triển có thể sử dụng để viết các ứng dụng consumer và producer client của riêng họ trên bất kỳ ngôn ngữ lập trình nào. Điều này cho phép Kafka thoát khỏi việc phụ thuộc vào hệ sinh thái Máy ảo Java (JVM).

* 1. OpenCV

OpenCV (Computer Vision) là một thư viện chứa các chức năng lập trình chủ yếu dành cho thị giác máy tính theo thời gian thực .Ban đầu được phát triển bởi Intel, sau đó được hỗ trợ bởi Willow Garage , sau đó là Itseez (sau này được Intel mua lại).

Thư viện đa nền tảng và được cấp phép dưới dạng phần mềm mã nguồn mở miễn phí theo Apache License 2. Bắt đầu từ năm 2011, OpenCV có tính năng tăng tốc GPU cho các hoạt động thời gian thực.

Được chính thức ra mắt vào năm 1999, dự án OpenCV ban đầu là một sáng kiến của Intel Research nhằm nâng cao các ứng dụng sử dụng nhiều CPU , một phần của một loạt các dự án bao gồm dò tia thời gian thực và tường hiển thị 3D.

Những người đóng góp chính cho dự án bao gồm một số chuyên gia tối ưu hóa tại Intel Nga, cũng như nhóm thư viện hiệu suất của Intel. Trong những ngày đầu của OpenCV, mục tiêu của dự án được mô tả là:

* Nghiên cứu thị giác nâng cao bằng cách không chỉ cung cấp mã mở mà còn tối ưu hóa cho cơ sở hạ tầng thị giác cơ bản. Không còn phát minh lại bánh xe.
* Phổ biến kiến thức về tầm nhìn bằng cách cung cấp cơ sở hạ tầng chung mà các nhà phát triển có thể xây dựng trên đó, để mã đó dễ đọc và dễ chuyển giao hơn.
* Nâng cao các ứng dụng thương mại dựa trên tầm nhìn bằng cách cung cấp miễn phí mã di động, tối ưu hóa hiệu suất - với giấy phép không yêu cầu mã phải được mở hoặc tự do.

Phiên bản alpha đầu tiên của OpenCV được phát hành ra công chúng tại Hội nghị IEEE về Thị giác máy tính và Nhận dạng mẫu vào năm 2000, và năm phiên bản beta được phát hành từ năm 2001 đến năm 2005.

Phiên bản 1.0 đầu tiên được phát hành vào năm 2006. Phiên bản tiền phát hành 1.1 được phát hành vào tháng 10 năm 2008.

Bản phát hành chính thứ hai của OpenCV là vào tháng 10 năm 2009. OpenCV 2 bao gồm những thay đổi lớn đối với giao diện C++, hướng đến các mẫu dễ dàng hơn, an toàn hơn, các chức năng mới và triển khai tốt hơn cho các giao diện hiện có về mặt hiệu suất (đặc biệt là trên nhiều nền tảng). hệ thống cốt lõi).

Các bản phát hành chính thức hiện được phát hành sáu tháng một lần và việc phát triển hiện được thực hiện bởi một nhóm độc lập của Nga được hỗ trợ bởi các tập đoàn thương mại.

Vào tháng 8 năm 2012, việc hỗ trợ cho OpenCV đã được tiếp quản bởi một tổ chức phi lợi nhuận OpenCV.org, nơi duy trì trang web của nhà phát triển và người dùng.

Vào tháng 5 năm 2016, Intel đã ký thỏa thuận mua lại Itseez, nhà phát triển OpenCV hàng đầu.

Vào tháng 7 năm 2020, OpenCV đã công bố và bắt đầu chiến dịch Kickstarter cho Bộ công cụ AI OpenCV, một loạt mô-đun phần cứng và các phần bổ sung cho OpenCV hỗ trợ AI không gian.

Vào tháng 8 năm 2020, OpenCV đã ra mắt OpenCV.ai – bộ phận tư vấn chuyên nghiệp. Nhóm các nhà phát triển cung cấp dịch vụ tư vấn và cung cấp các giải pháp Thị giác máy tính, Học máy và Trí tuệ nhân tạo.

OpenCV được viết bằng ngôn ngữ lập trình C++, cũng như giao diện chính của nó, nhưng nó vẫn giữ giao diện C cũ hơn nhưng kém toàn diện hơn. Tất cả các phát triển và thuật toán mới hơn đều xuất hiện trong giao diện C++. Có các ràng buộc ngôn ngữ trong Python, Java và MATLAB/Octave. Bạn có thể tìm thấy giao diện lập trình ứng dụng (API) cho các giao diện này trong tài liệu trực tuyến.

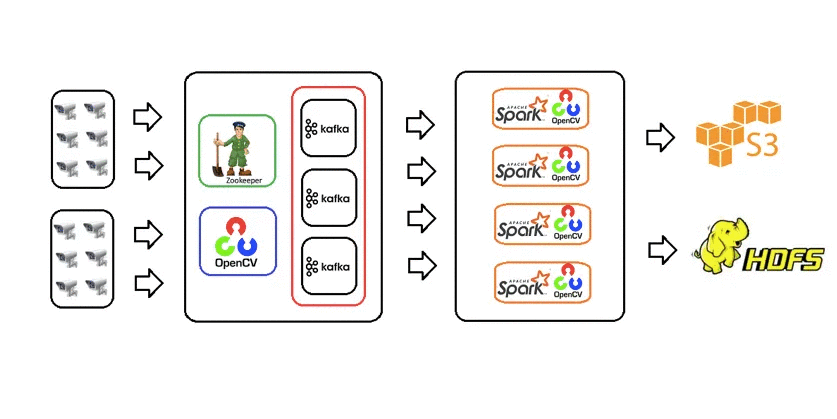
Thư viện Wrapper bằng một số ngôn ngữ đã được phát triển để khuyến khích nhiều đối tượng hơn áp dụng. Trong phiên bản 3.4, các liên kết JavaScript cho một tập hợp con các hàm OpenCV được chọn đã được phát hành dưới dạng OpenCV.js, để sử dụng cho các nền tảng web.

Nếu thư viện tìm thấy nguyên tắc hiệu năng tích hợp của Intel trên hệ thống, thư viện sẽ sử dụng các quy trình được tối ưu hóa độc quyền này để tự tăng tốc.

Giao diện đơn vị xử lý đồ họa (GPU) dựa trên Kiến trúc Thiết bị Thống nhất Điện toán (CUDA) đã được phát triển kể từ tháng 9 năm 2010.

Giao diện GPU dựa trên OpenCL đã được phát triển kể từ tháng 10 năm 2012

* 1. Mô hình giải pháp



Producer (Sản xuất viên):

Producer là thành phần tạo và gửi các bản tin (messages) đến các topic Kafka. Khi producer gửi một bản tin, nó cũng gắn một key với nó. Key này có thể được sử dụng để quyết định xem bản tin sẽ thuộc về partition nào.

Broker (Môi giới):

Kafka sử dụng một số broker (môi giới) để lưu trữ dữ liệu. Mỗi broker có thể nằm trên một máy chủ khác nhau.

Mỗi partition trong một topic được lưu trữ trên một broker. Các partition giúp tăng khả năng mở rộng và hiệu suất bằng cách chia nhỏ dữ liệu thành các phần nhỏ và phân phối chúng trên các broker.

Topic (Chủ đề):

Một topic là nơi lưu trữ các bản tin. Mỗi topic có thể được chia thành nhiều partition, mỗi partition có thể nằm trên một broker khác nhau.

Consumer (Người tiêu thụ):

Consumer là thành phần đọc và xử lý các bản tin từ các topic Kafka.

Consumer có thể đọc từ một hoặc nhiều partition trong một topic.

Consumer Group (Nhóm người tiêu thụ):

Các consumer có thể được tổ chức thành các nhóm (consumer group). Mỗi partition chỉ có thể được đọc bởi một consumer trong một nhóm. Điều này giúp phân chia công việc đọc dữ liệu giữa các consumer.

Zookeeper:

Zookeeper là một dịch vụ quản lý cấu hình và tình trạng của Kafka.

Nó giúp theo dõi trạng thái của các broker, partition, consumer, và nhóm consumer.

Cơ chế Gửi và Nhận (Publish-Subscribe):

Kafka sử dụng mô hình publish-subscribe, nơi producer gửi các bản tin đến các topic và các consumer đọc từ các topic mà họ quan tâm.

Các bản tin không bị mất ngay sau khi được đọc.

Thay vào đó, chúng sẽ được lưu trữ trong một khoảng thời gian được xác định (retention period), mà sau đó sẽ bị xóa.

Retention Period (Khoảng thời gian lưu trữ):

Kafka hỗ trợ việc lưu trữ các bản tin trong một khoảng thời gian xác định.

Điều này giúp consumer đọc các bản tin cũ nếu cần thiết.

Quy trình Gửi và Nhận (Produce and Consume Process):

Producer gửi các bản tin đến một topic Kafka. Các bản tin này được chia thành các partition dựa trên các key hoặc theo cách tự động.

Consumer nhóm đăng ký để đọc từ một hoặc nhiều topic. Mỗi consumer trong nhóm đọc từ một partition cụ thể.

Kafka theo dõi vị trí đọc của từng consumer trong mỗi partition, giúp đảm bảo mỗi bản tin chỉ được đọc một lần.

Như vậy, Kafka cung cấp một hệ thống truyền thông tin phân tán và có khả năng mở rộng, phù hợp cho việc xử lý dữ liệu lớn và các ứng dụng thời gian thực.

* 1. Cài đặt Thử nghiệm
* Dataset

Sử dụng video VIRAT\_S\_000002 (Video and Image Retrieval and Analysis Tool).

Chương trình Công cụ phân tích và truy xuất video và hình ảnh (VIRAT) là một dự án giám sát video được tài trợ bởi Văn phòng Công nghệ xử lý thông tin của Cơ quan Dự án nghiên cứu quốc phòng tiên tiến (DARPA).

Mục đích của chương trình là tạo ra một cơ sở dữ liệu có thể lưu trữ số lượng lớn video và giúp các nhân viên tình báo có thể dễ dàng tìm kiếm "nội dung video quan tâm" (ví dụ: "tìm tất cả các cảnh quay trong đó ba người trở lên đang đứng cùng nhau." trong một nhóm") -- đây được gọi là "tìm kiếm dựa trên nội dung".

Hiện đã có các hệ thống phát hiện đối tượng được phát triển cao (ví dụ: các chương trình có thể xác định xem đối tượng trong đoạn phim video là "ô tô" hay "người đeo ba lô").

VIRAT sẽ sử dụng những gì hiện có để phát hiện đối tượng. Việc tài trợ cho nghiên cứu phát hiện đối tượng không nằm trong phạm vi của VIRAT, trừ khi nó liên quan đến việc xác định một số loại hoạt động nhất định, như những hoạt động được đề cập ở trên.

Nguồn: [**https://data.kitware.com/#item/60a8b06f2fa25629b98b486f**](https://data.kitware.com/#item/60a8b06f2fa25629b98b486f)

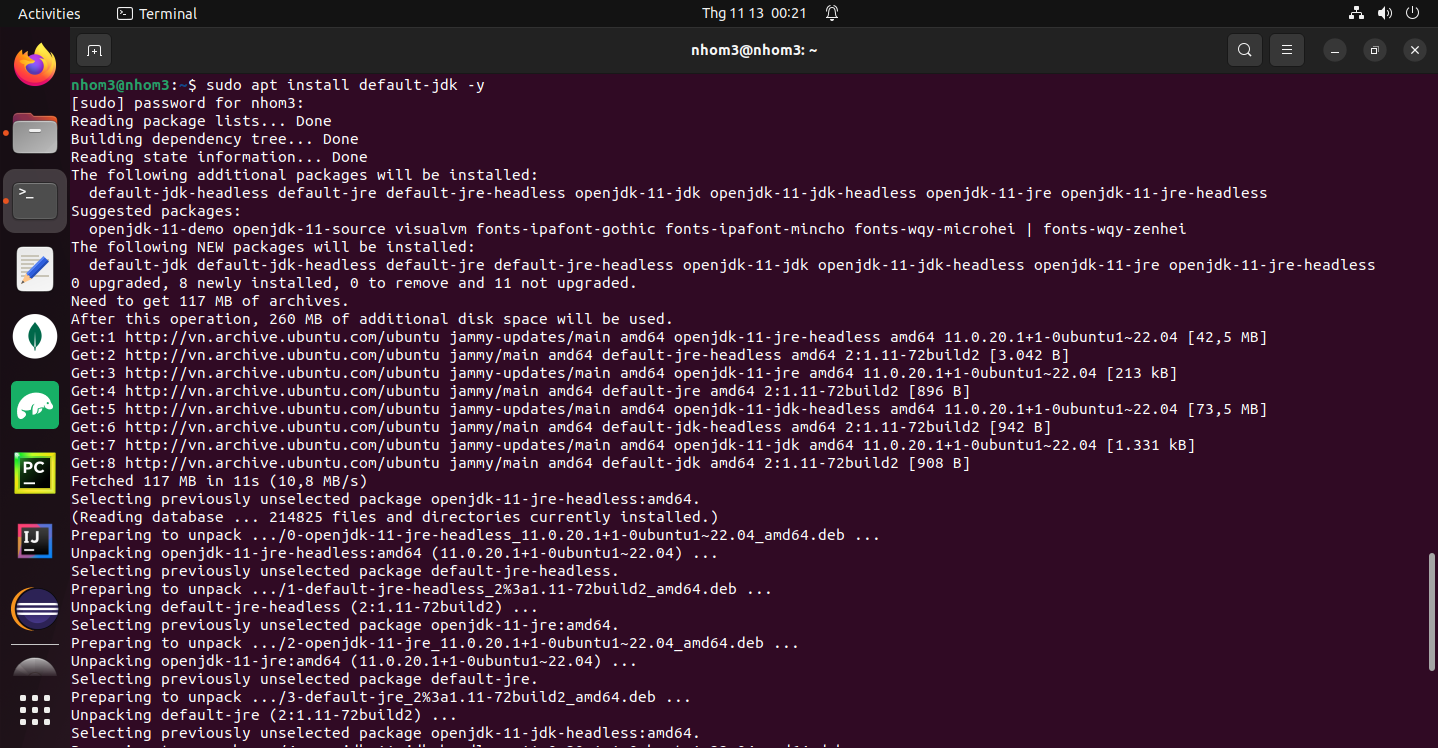
* Chuẩn bị

Cài đặt các công cụ sau:

|  |  |
| --- | --- |
| **OpenCV** | [**http://opencv.org/releases.html**](http://opencv.org/releases.html) |
| **Spark** | [**http://spark.apache.org/downloads.html**](http://spark.apache.org/downloads.html) |
| **Kafka** | [**http://kafka.apache.org/downloads.html**](http://kafka.apache.org/downloads.html) |
| **JDK** | [**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) |
| **Pycharm** | [**https://www.jetbrains.com/pycharm/download/?section=windows**](https://www.jetbrains.com/pycharm/download/?section=windows) |
| **Jupyter** | [**https://jupyter.org/install**](https://jupyter.org/install) |
| **Python** | [**https://www.python.org/downloads/**](https://www.python.org/downloads/) |

* Các bước cài đặt
  1. Cài đặt JAVA

apt install default-jdk -y



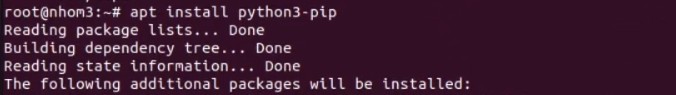
Kiểm tra đã cài đặt hay chưa

java -version



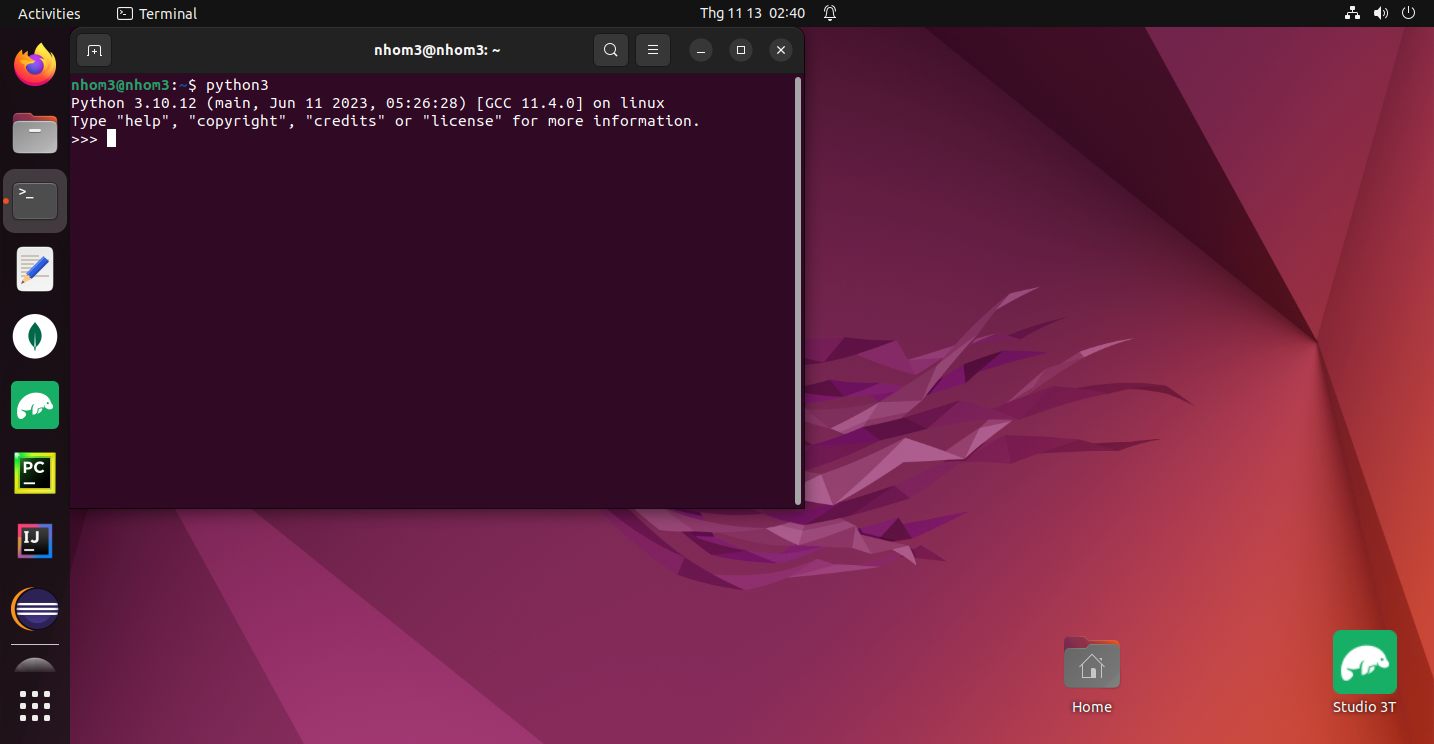
* 1. Tải Python

apt install python3-pip



Kiểm tra phiên bản

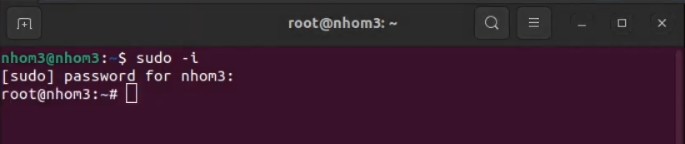
python3



* 1. Tải Apache Spark

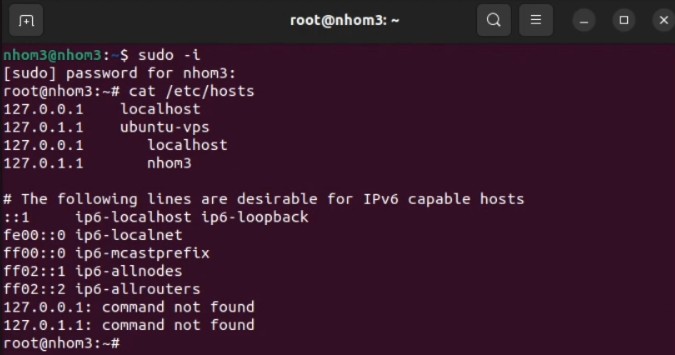
Đầu tiên sử dụng với user root bằng lệnh sudo -i hoặc sudo su -

Dùng lệnh sudo -i hoặc sudo su – để truy cập vào user root



Kiểm tra thông số

Kiểm tra hai thông số sau đã giống nhau chưa để khởi động spark, nếu khác nhau thì khi khởi động sẽ lỗi

cat /etc/hosts

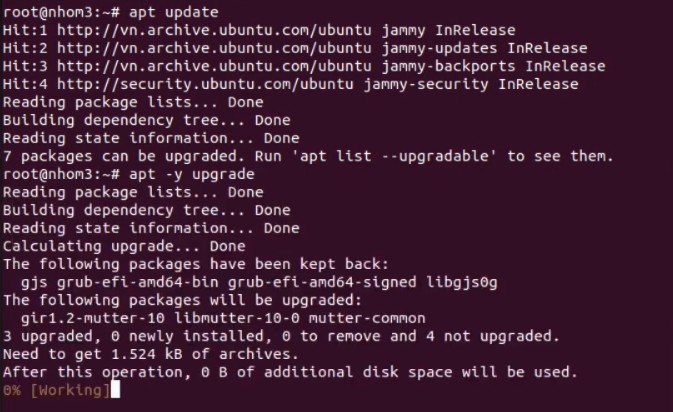
127.0.0.1 localhost

127.0.1.1 ubuntu-vps

Cập nhập để sử dụng phiên bản mới

apt update

apt -y upgrade



Cài đặt java

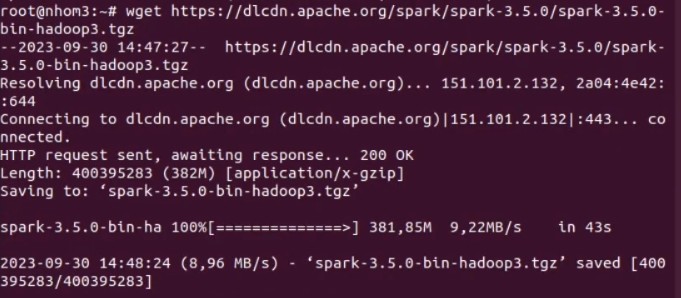
Nếu chưa thì hãy dùng lệnh sau để cài: apt install default-jdk -y

Kiểm tra java đã cài hay chưa java -version



Cài đặt apache spark

wget https://dlcdn.apache.org/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz



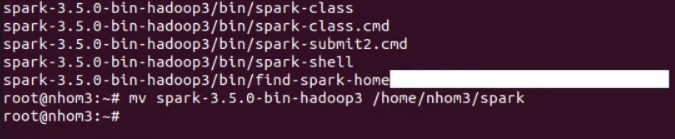
Giải nén file

tar xvf spark-3.5.0-bin-hadoop3.tgz



Chuyển thư mục vừa giải nén vào thư mục nào đó và đặt tên spark (nhớ kiểm tra lại đường dẫn)

mv spark-3.5.0-bin-hadoop3 /home/nhom3/spark



Kiểm tra đã chuyển thành công hay chưa



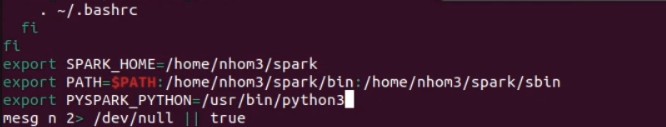
Khai báo biến môi trường để sử dụng spark

Mở file bằng lệnh sau: nano ~/.profile



Thêm nội dung sau vào (nhớ sửa lại đường dẫn) export SPARK\_HOME=/home/nhom3/spark

export PATH=$PATH:/home/nhom3/spark/bin:/home/nhom3/spark/sbin export PYSPARK\_PYTHON=/usr/bin/python3



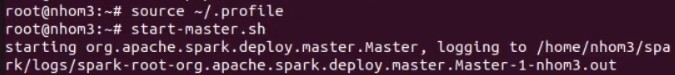
lưu lại

source ~/.profile



Tiến hành khởi động dịch vụ

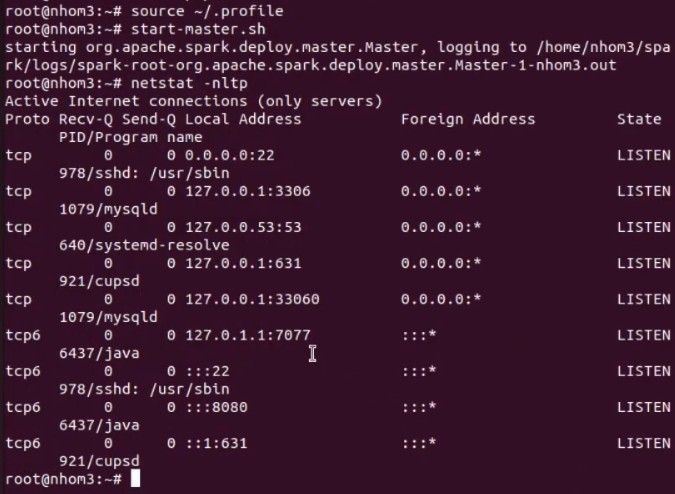
start-master.sh



Kiểm tra lại spark có trên port mặc định là 7077 chưa

(Lưu ý một số sẽ bị lỗi không tìm thấy lệnh này nên có thể tải nó bằng lệnh sau: sudo apt-get -y install net-tools)

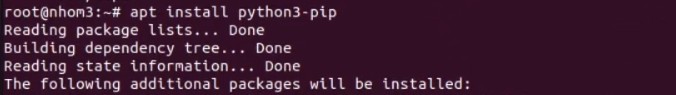
netstat -nltp



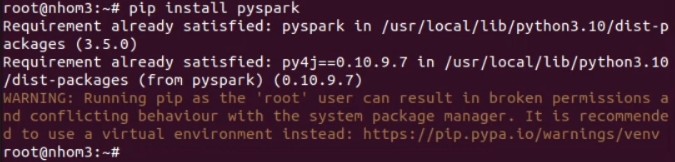
Tùy chọn thêm với PySpark

Mặc định ngôn ngữ của spark là Scala. Nếu muốn sử dụng ngôn ngữ Python trên spark thì phải cài thêm gói PySpark.

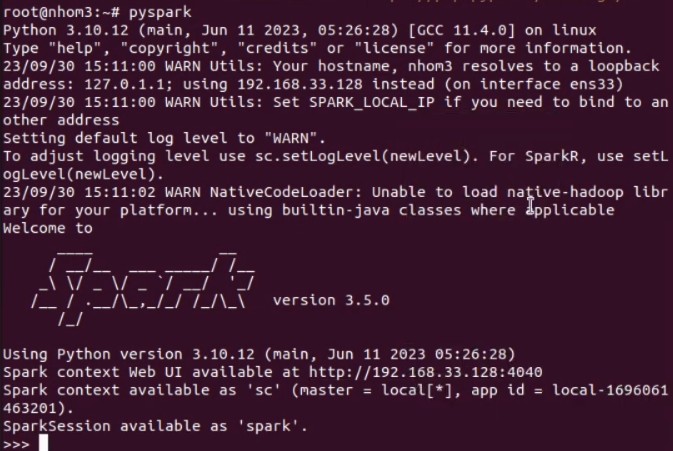
apt install python3-pip



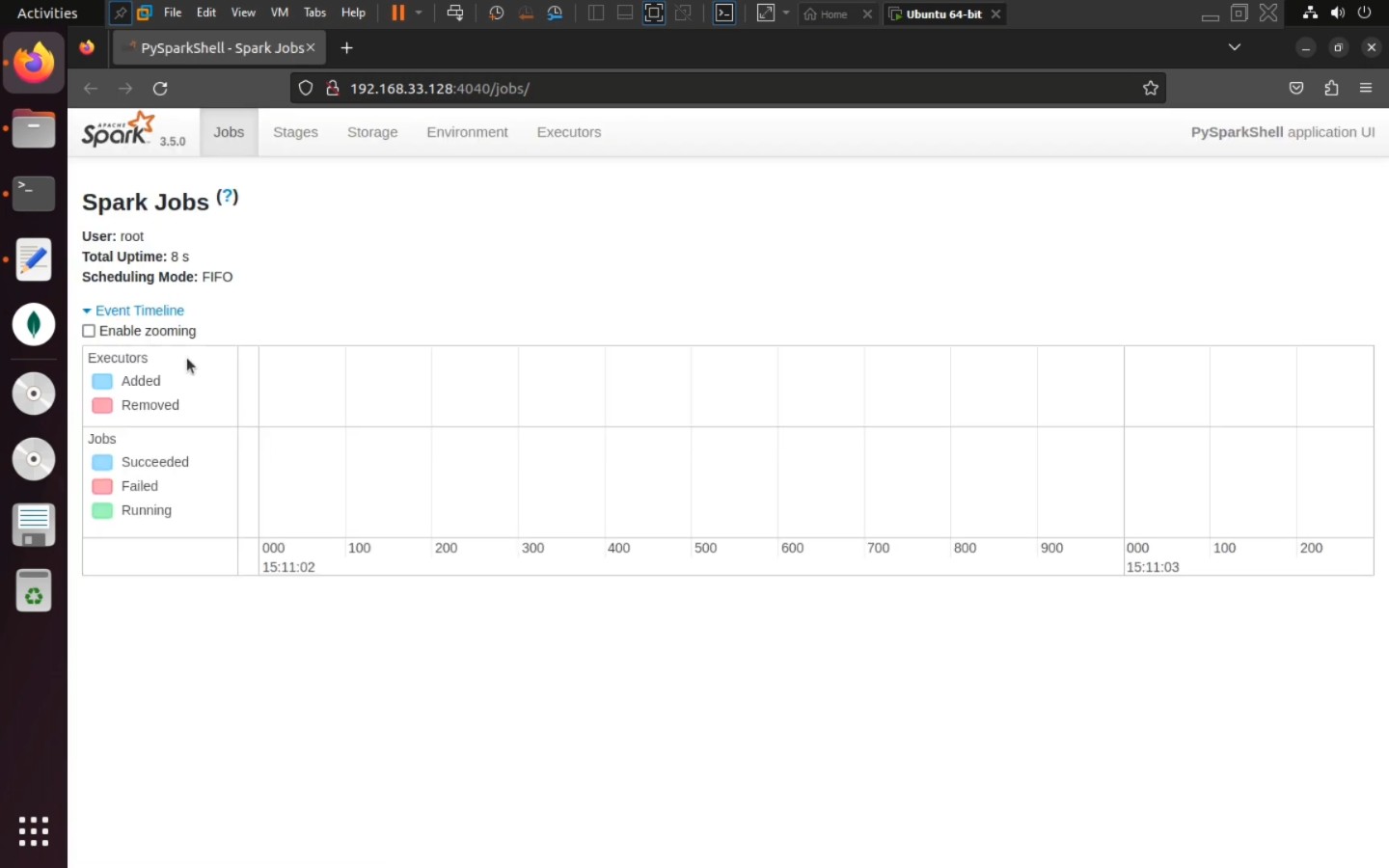
pip install pyspark



Kiểm tra lại đã kết nối được Python với spark chưa pyspark (ctrl + D để thoát)



Nếu kết nối được thì đã thành công cài đặt Apache Spark kèm với Python



* 1. Tải Apache Kafka

Cài đặt Java (đã cài đặt)

Nếu chưa thì hãy dùng lệnh sau để cài: apt install default-jdk -y

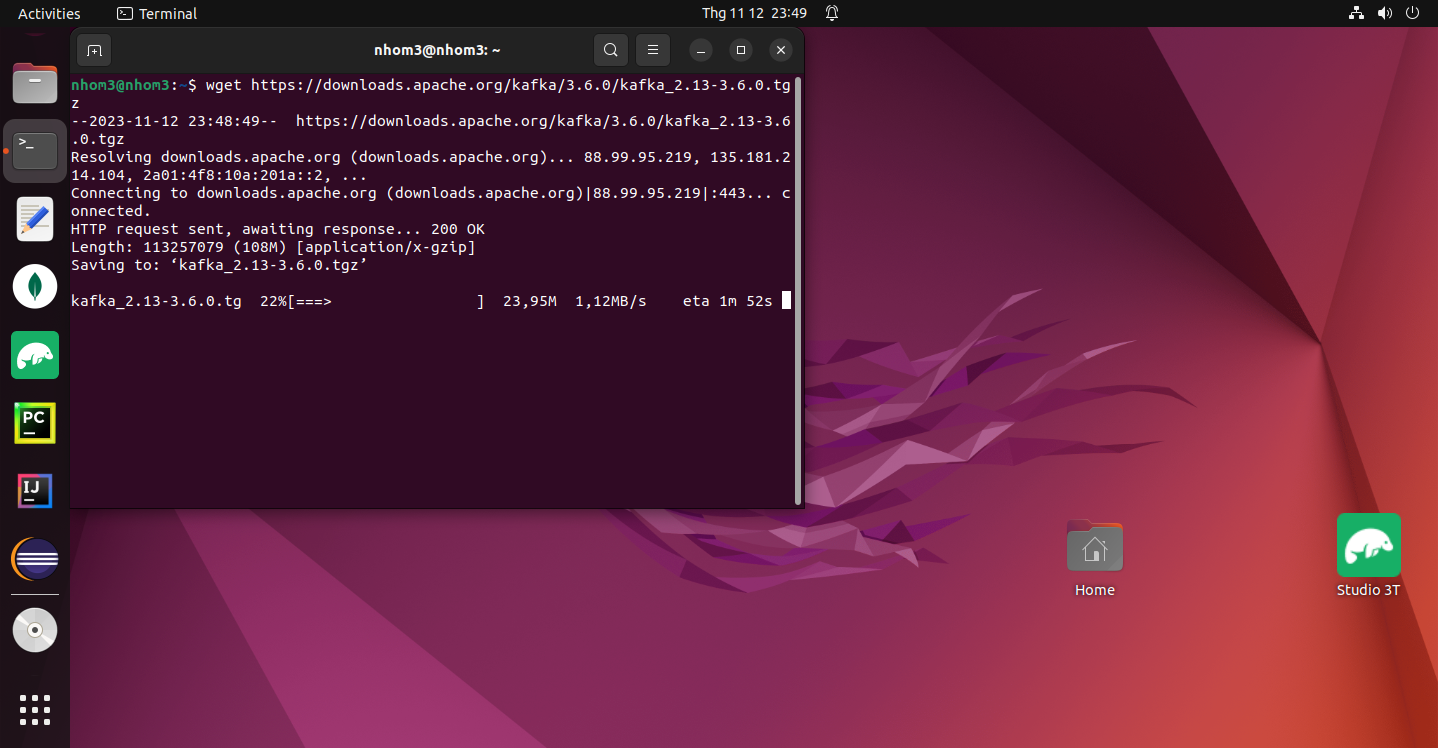
Kiểm tra java đã cài đặt

java -version



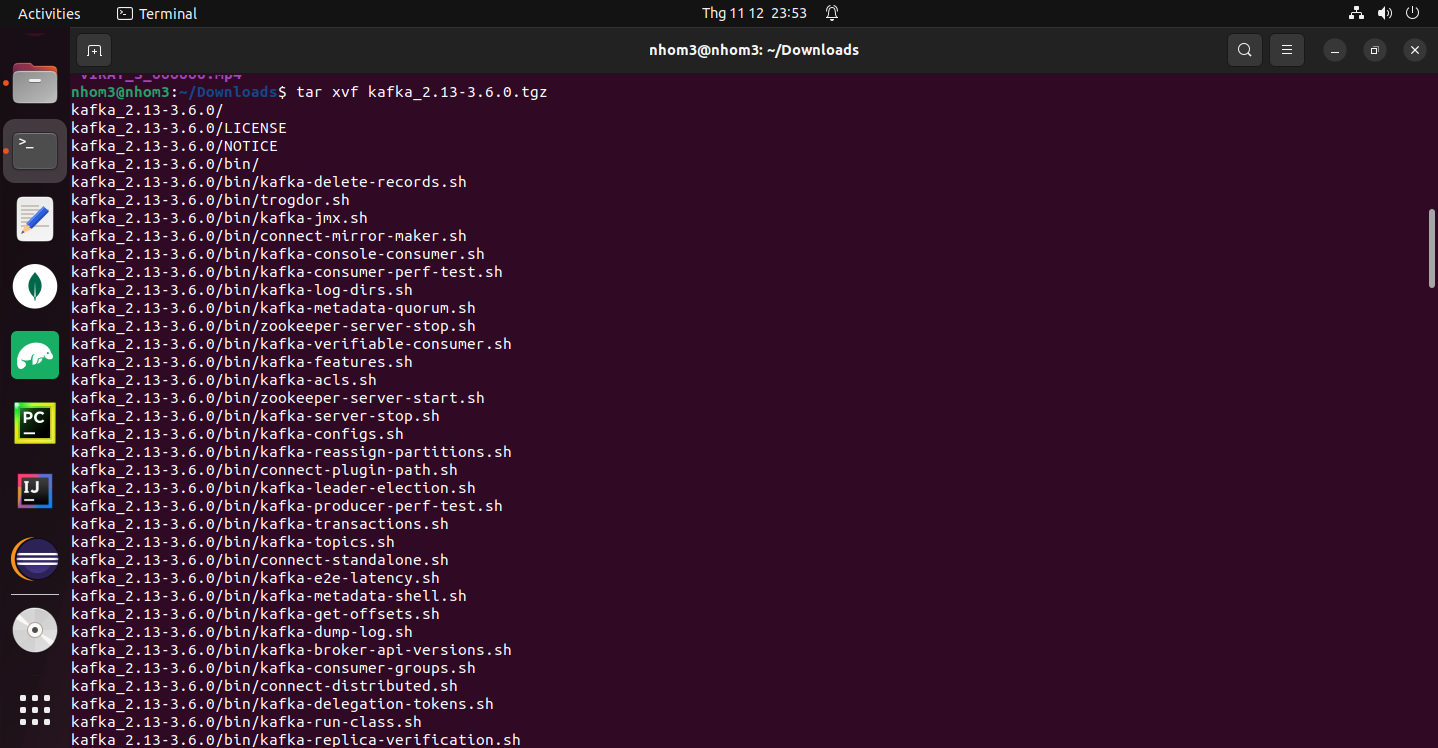
Cài đặt Kafka

Wget <https://downloads.apache.org/kafka/3.6.0/kafka_2.13-3.6.0.tgz>



Giải nén file

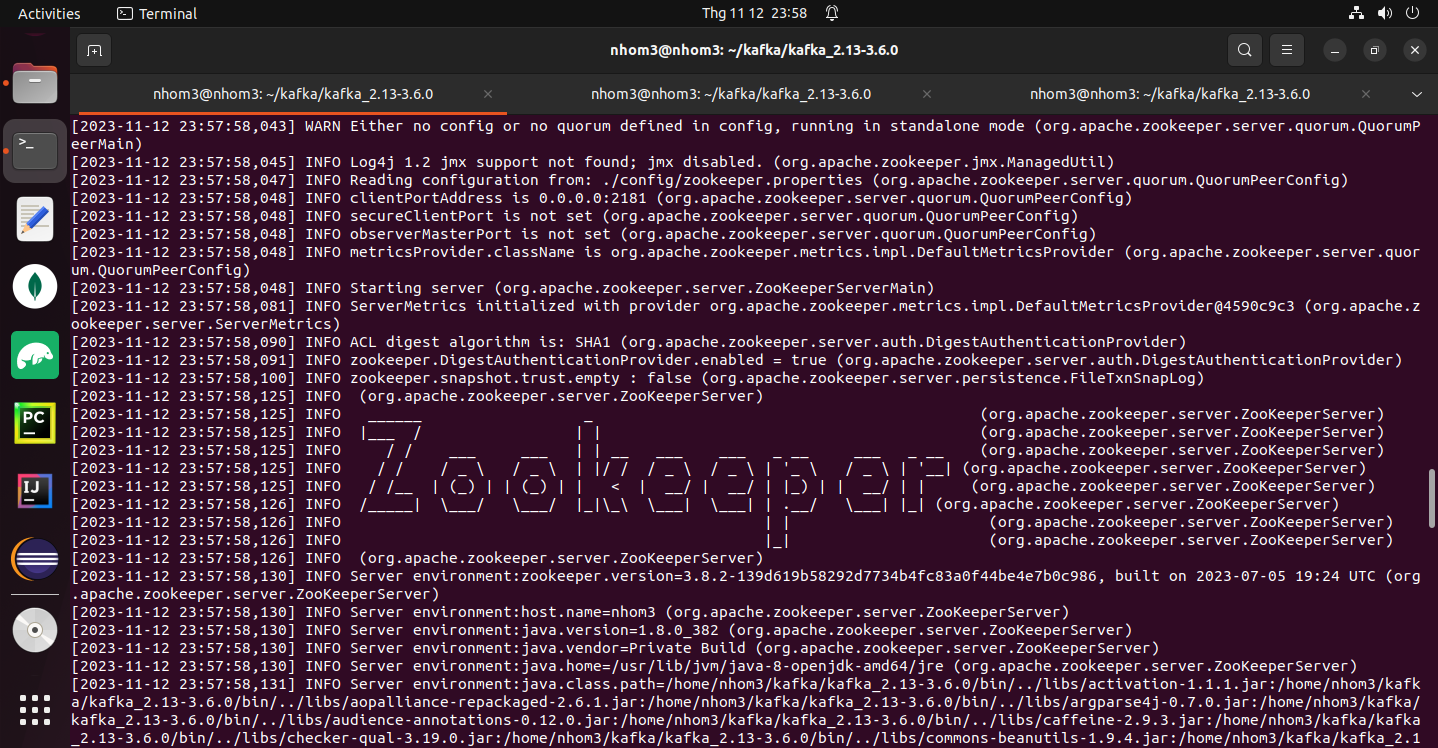
tar xvf kafka\_2.13-3.6.0.tgz



Khởi động Kafka

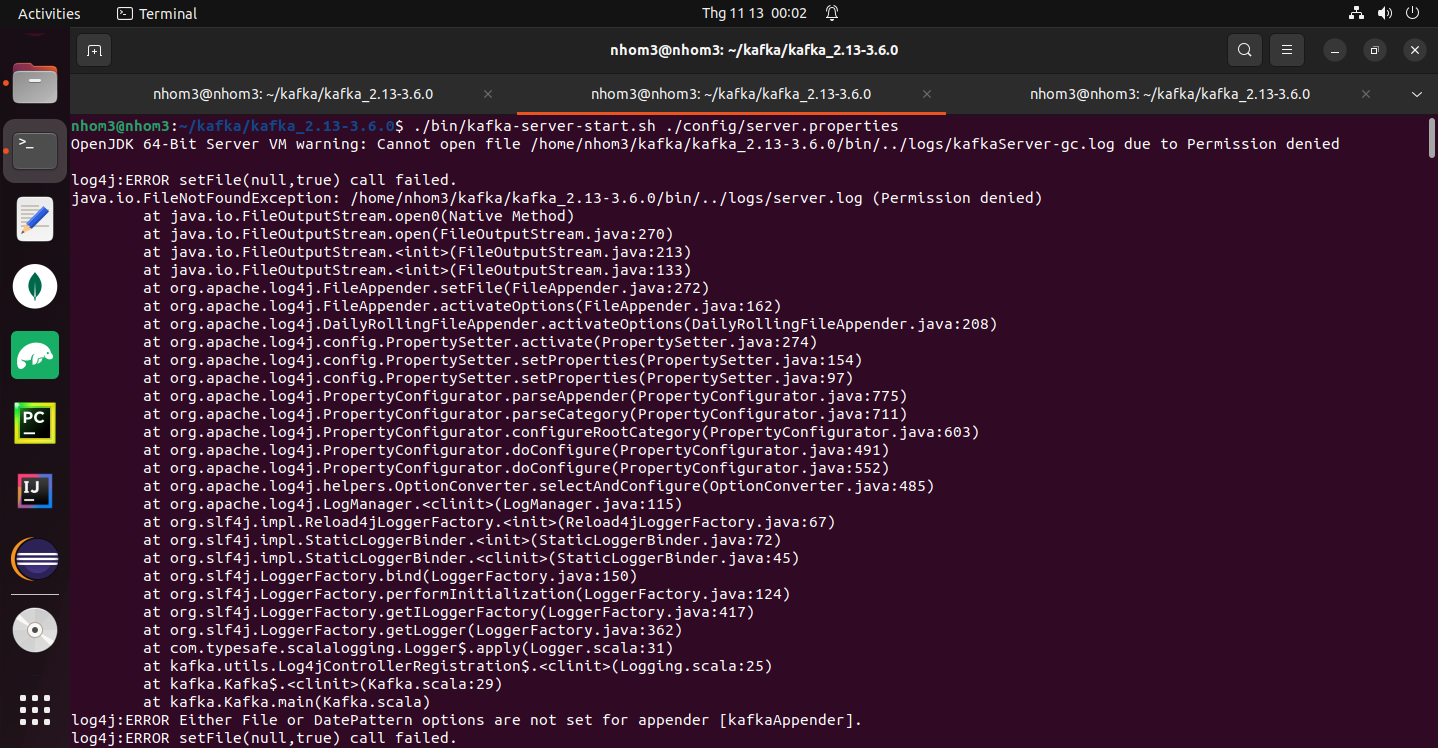
Khởi động zookeeper

zookeeper-server-start.sh ./config/zookeeper.properties



Khởi động Kafka server

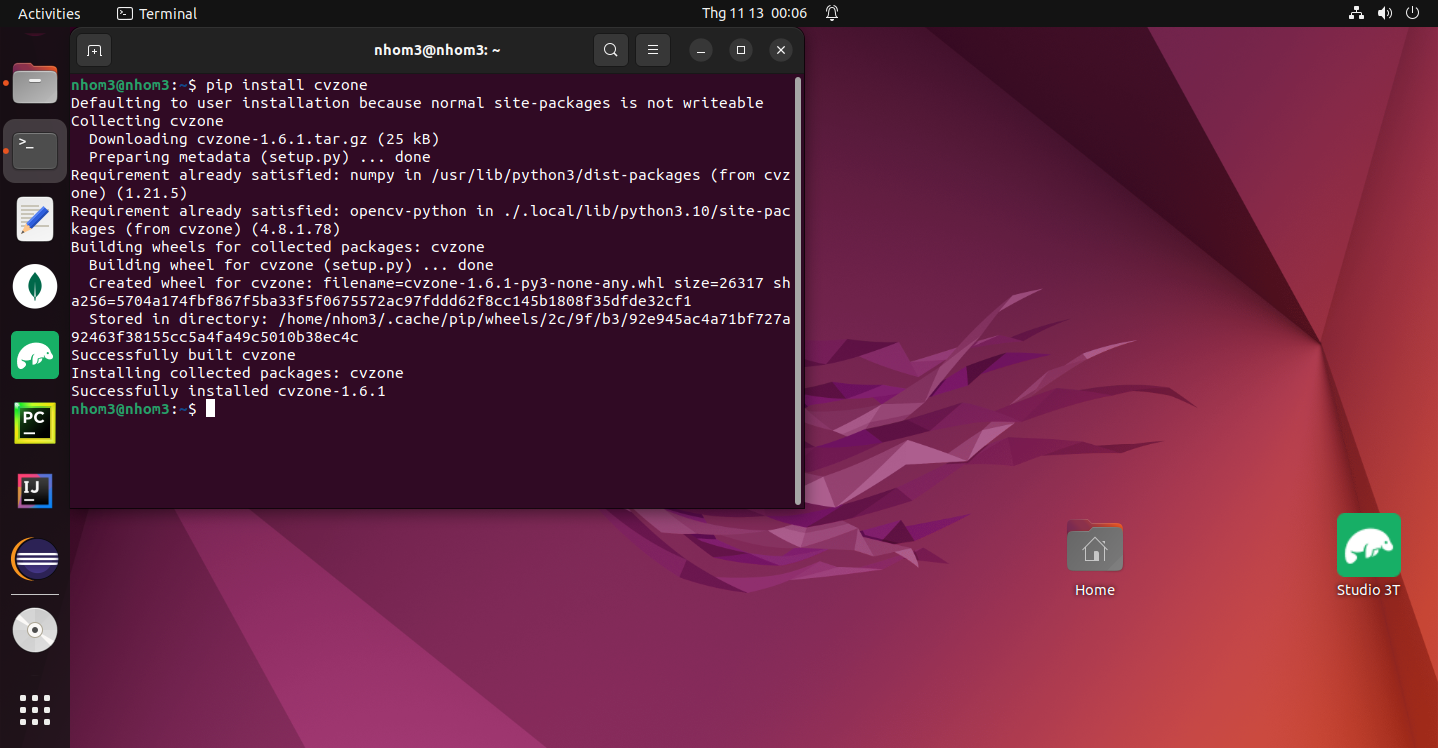
kafka-server-start.sh ./config/server.properties



Nếu chạy được 2 cái trên thì đã hoàn tất việc tải Kafka

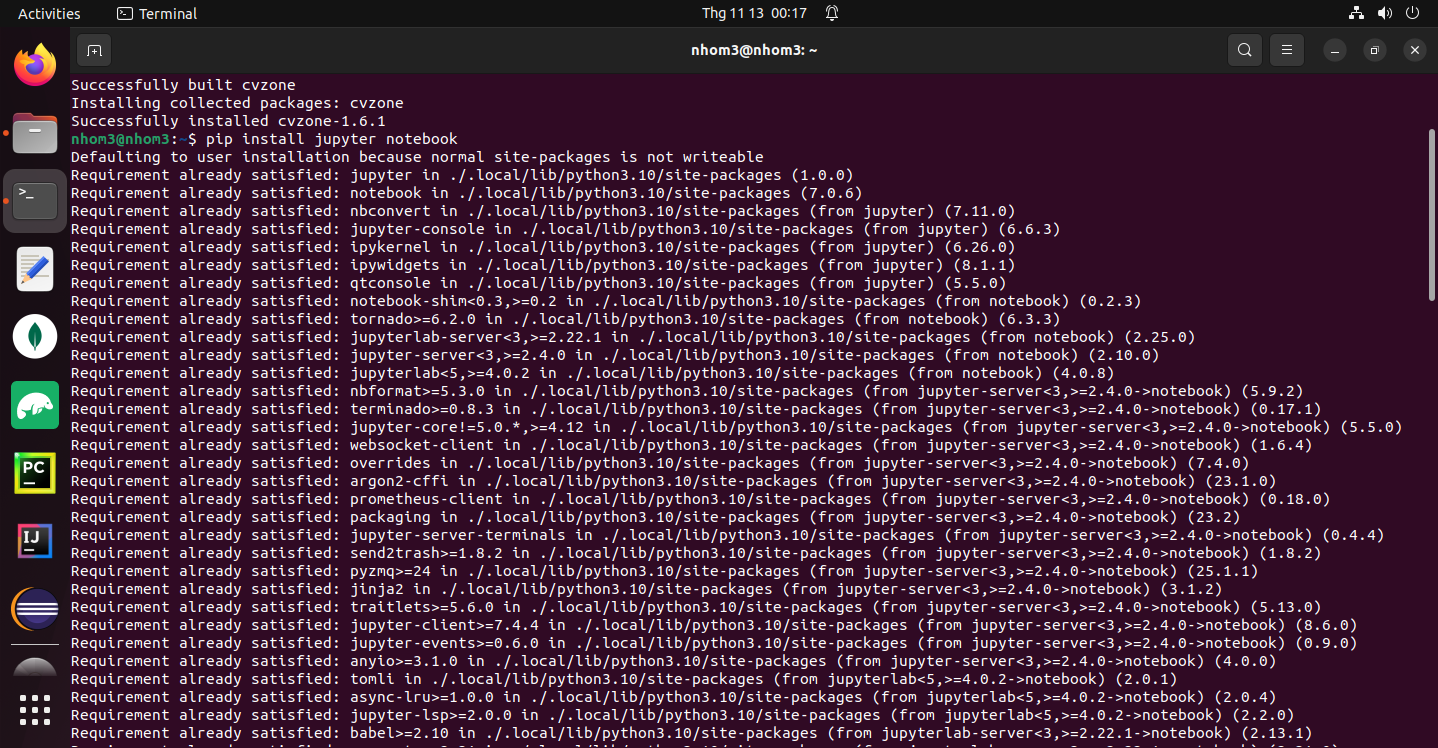
* 1. Cài đặt OpenCV

pip install cvzone



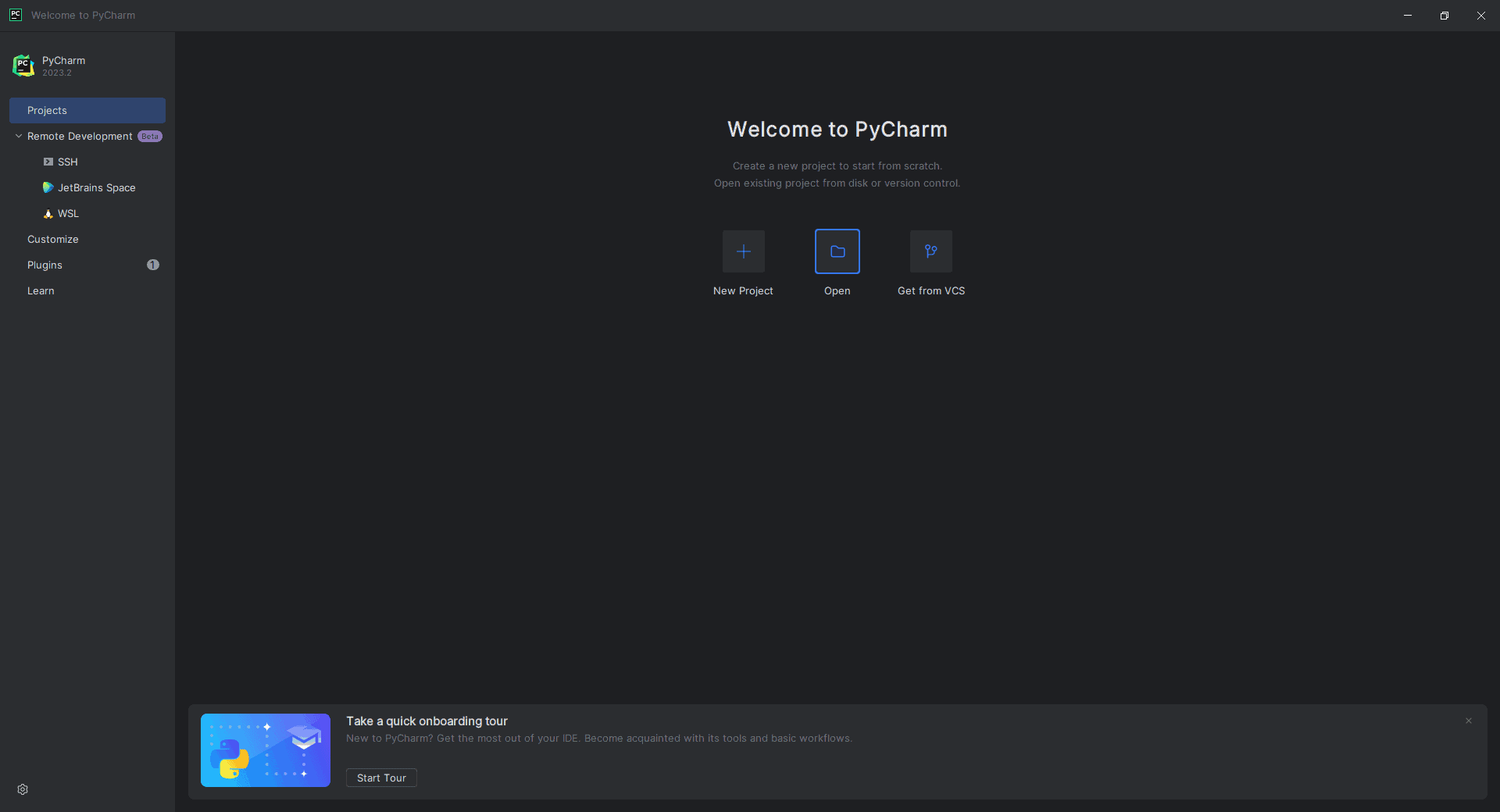
* 1. Cài đặt Jupyter notebook

pip install jupyter notebook



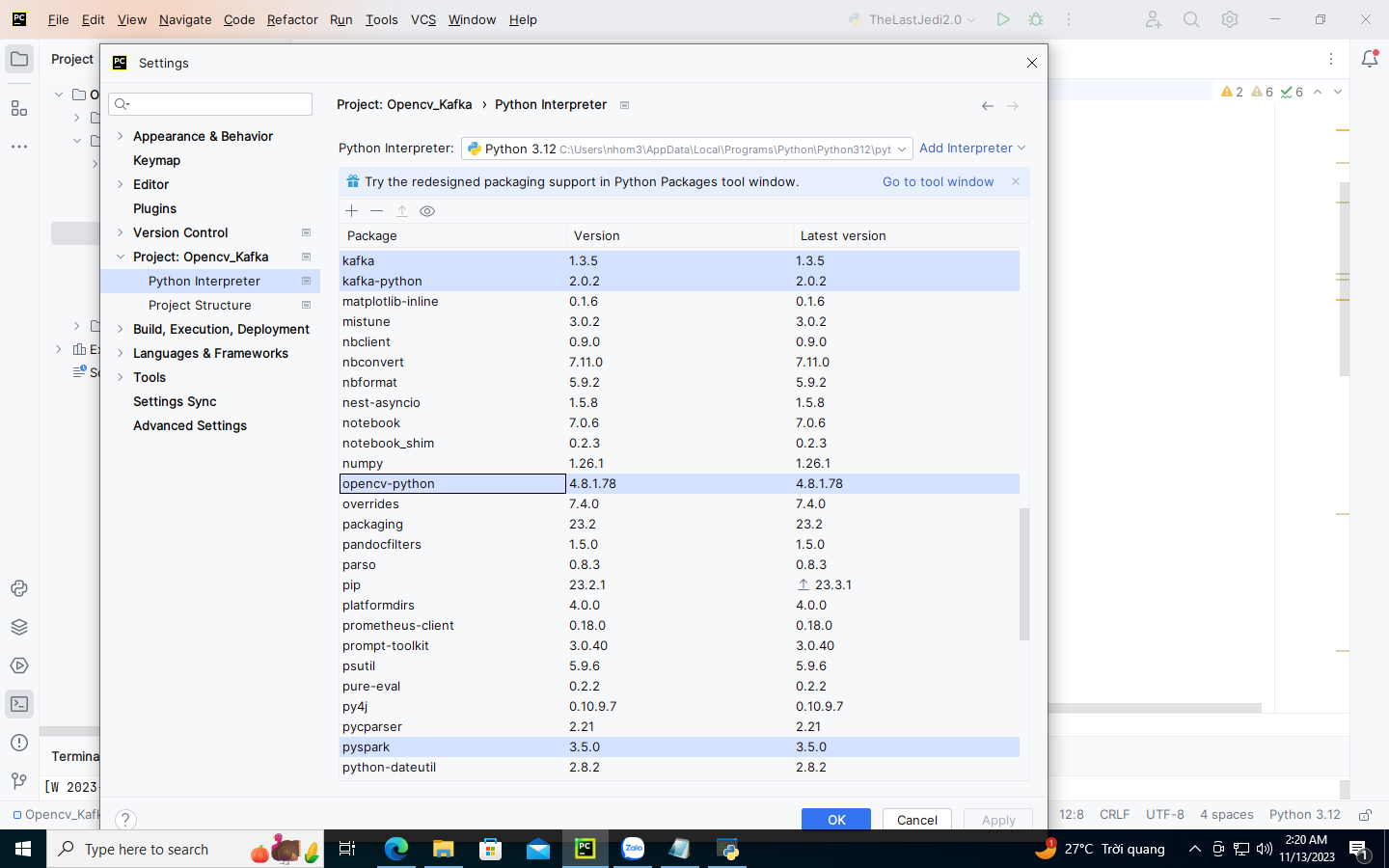
* 1. Cài đặt Pycharm

Sử dụng đường dẫn trên để cài đặt



Đảm bảo rằng đã cài đặt Python và các gói cần thiết

Nếu thiếu các gói đó có thể vào Settings – Python Interpreter và chọn tải những gói cần thiết



* Cách dùng

Xử lý video streaming để phát hiện chuyển động thường liên quan đến việc sử dụng một loạt công nghệ và công cụ khác nhau. OpenCV là một thư viện mạnh mẽ cho xử lý ảnh và video, thường được sử dụng để phân tích, xử lý hình ảnh và video.

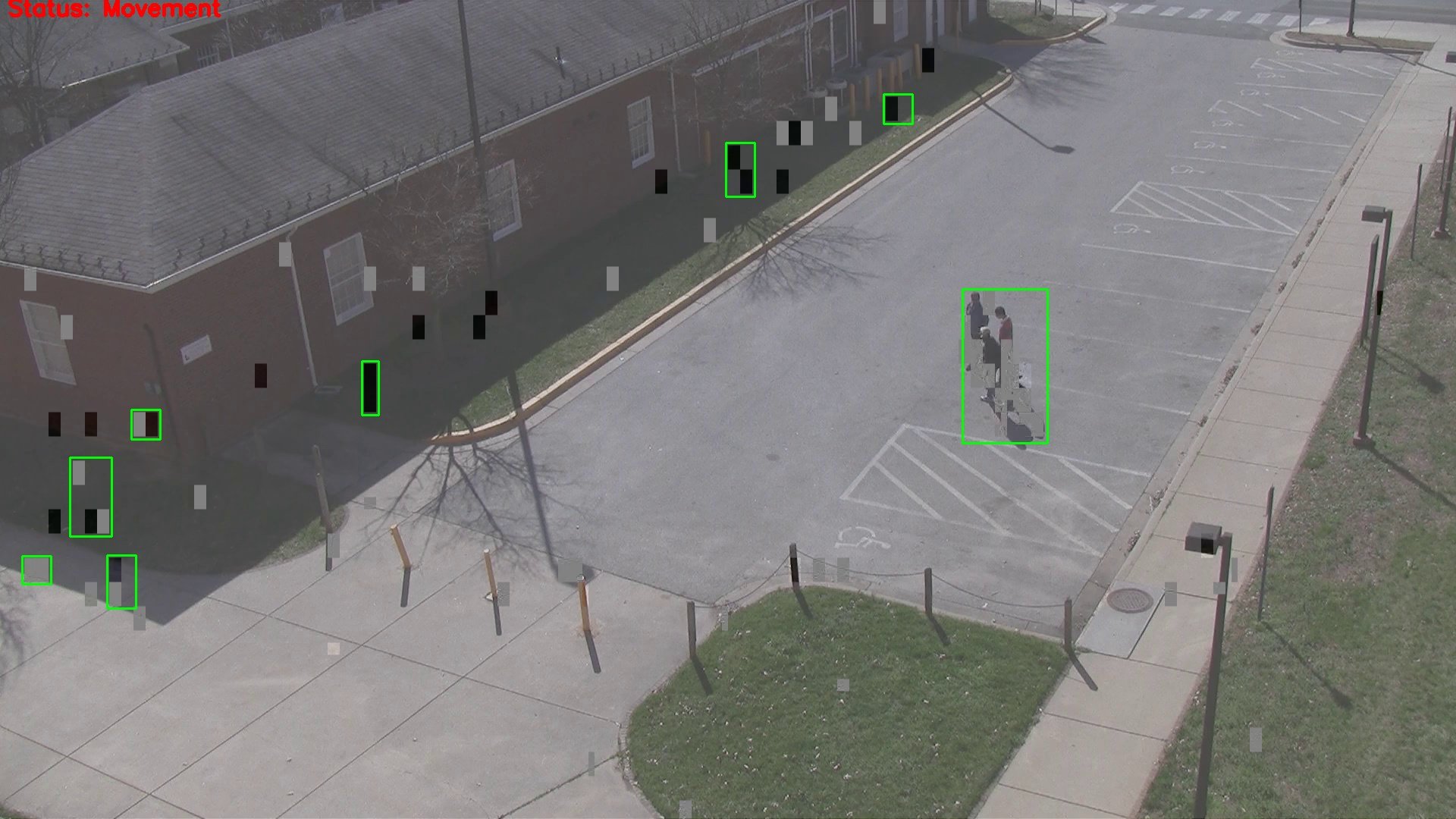
Kafka là một hệ thống hàng đợi và phân phối dữ liệu, thường được sử dụng để truyền dữ liệu giữa các ứng dụng và hệ thống. Nó có thể được tích hợp vào các quy trình xử lý dữ liệu liên tục.

Spark là một framework xử lý dữ liệu phân tán, cung cấp khả năng xử lý dữ liệu lớn và xử lý stream. Spark Streaming cung cấp các API cho việc xử lý dữ liệu video streaming.

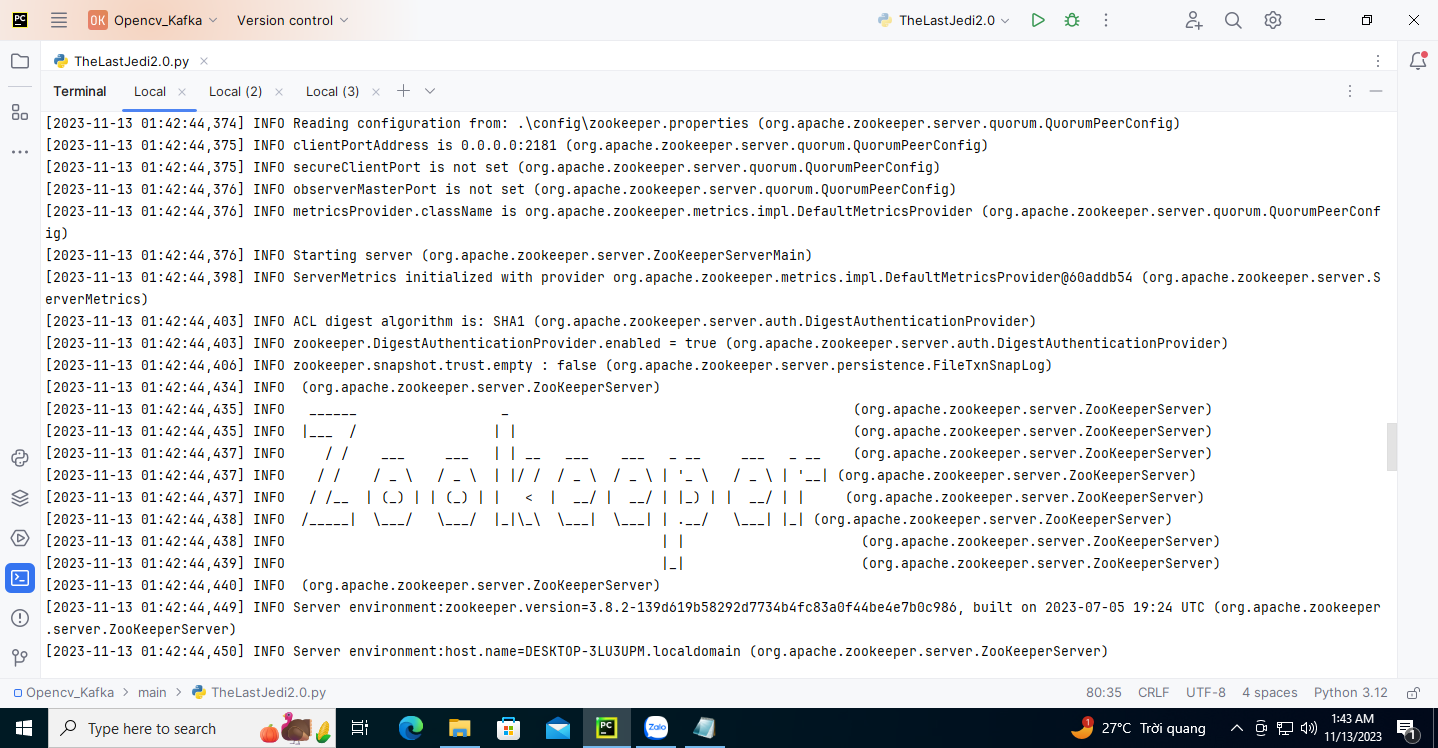
Để kết hợp chúng lại để phát hiện chuyển động trong video streaming, bạn có thể thực hiện các bước sau:

Thu thập trực tuyến video: Truyền phát video có thể được thu thập từ các nguồn như camera, webcam hoặc bất kỳ nguồn phát video nào khác. Trong đây chúng em sử dụng clip VIRAT\_S\_000002.mp4

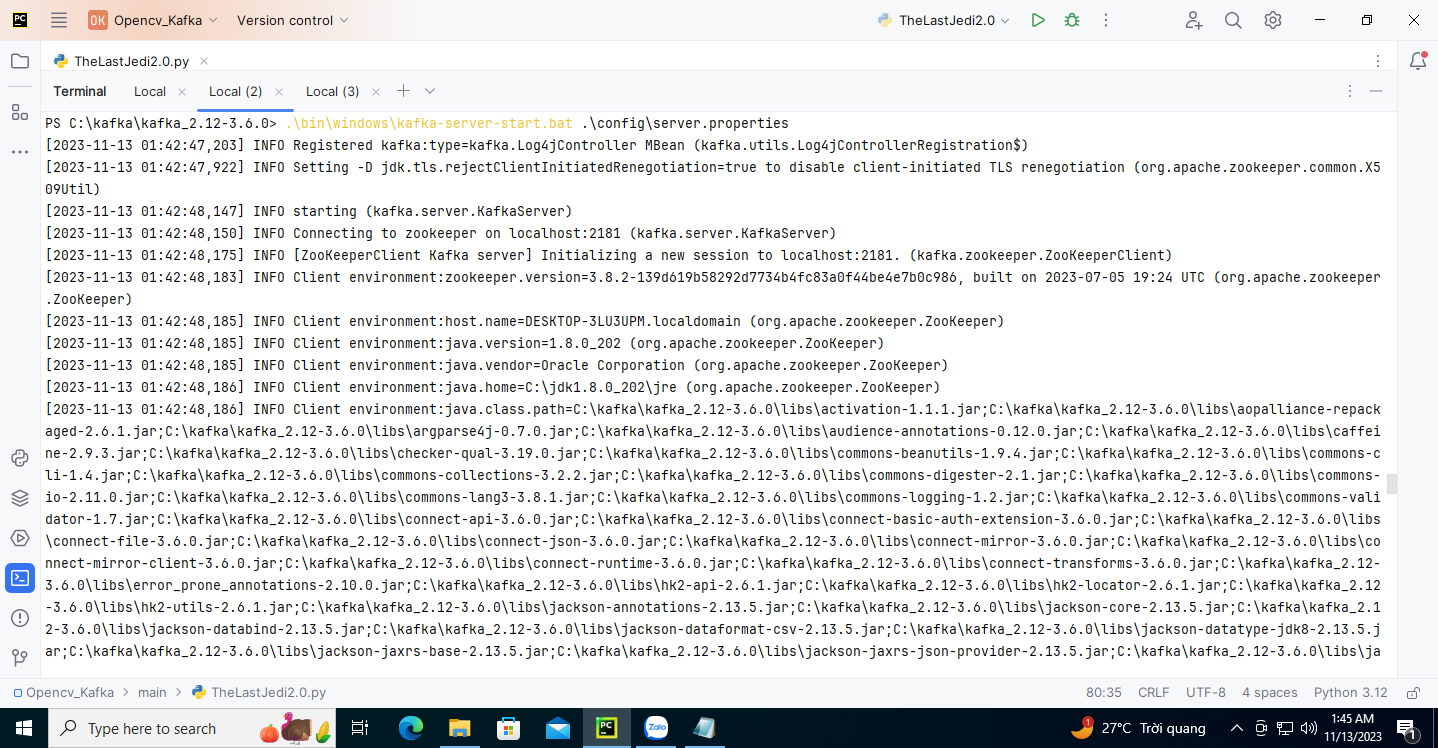
Xử lý video sử dụng OpenCV: Sử dụng OpenCV để bắt vật thể đang chuyển động trong video và trích xuất các khung hình từ video streaming và thực hiện việc phát hiện chuyển động thông qua việc so sánh các khung hình liên tiếp.



Khởi động Zookeeper

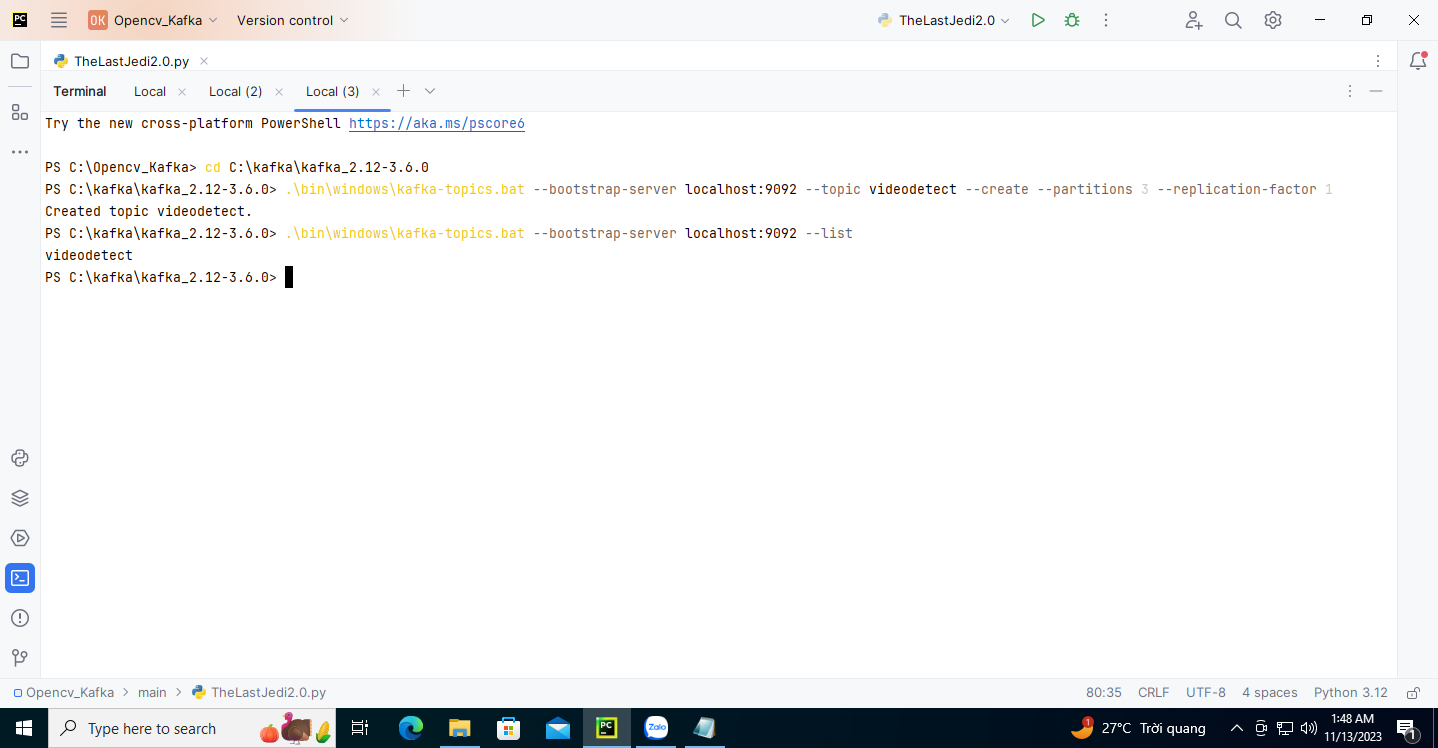


Khởi động Kafka server



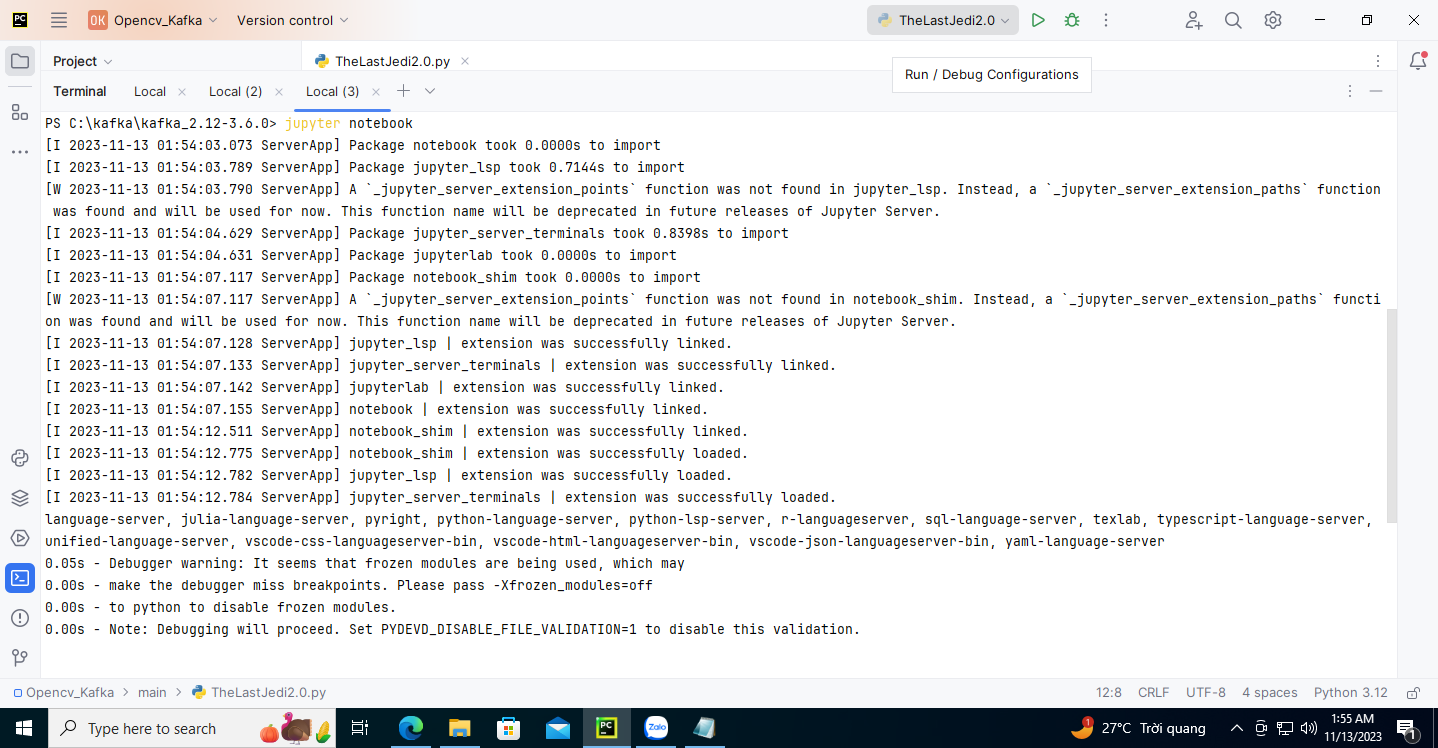
Tạo topics kafka tên videodetect

.\bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --topic videodetect --create --partitions 3 --replication-factor 1

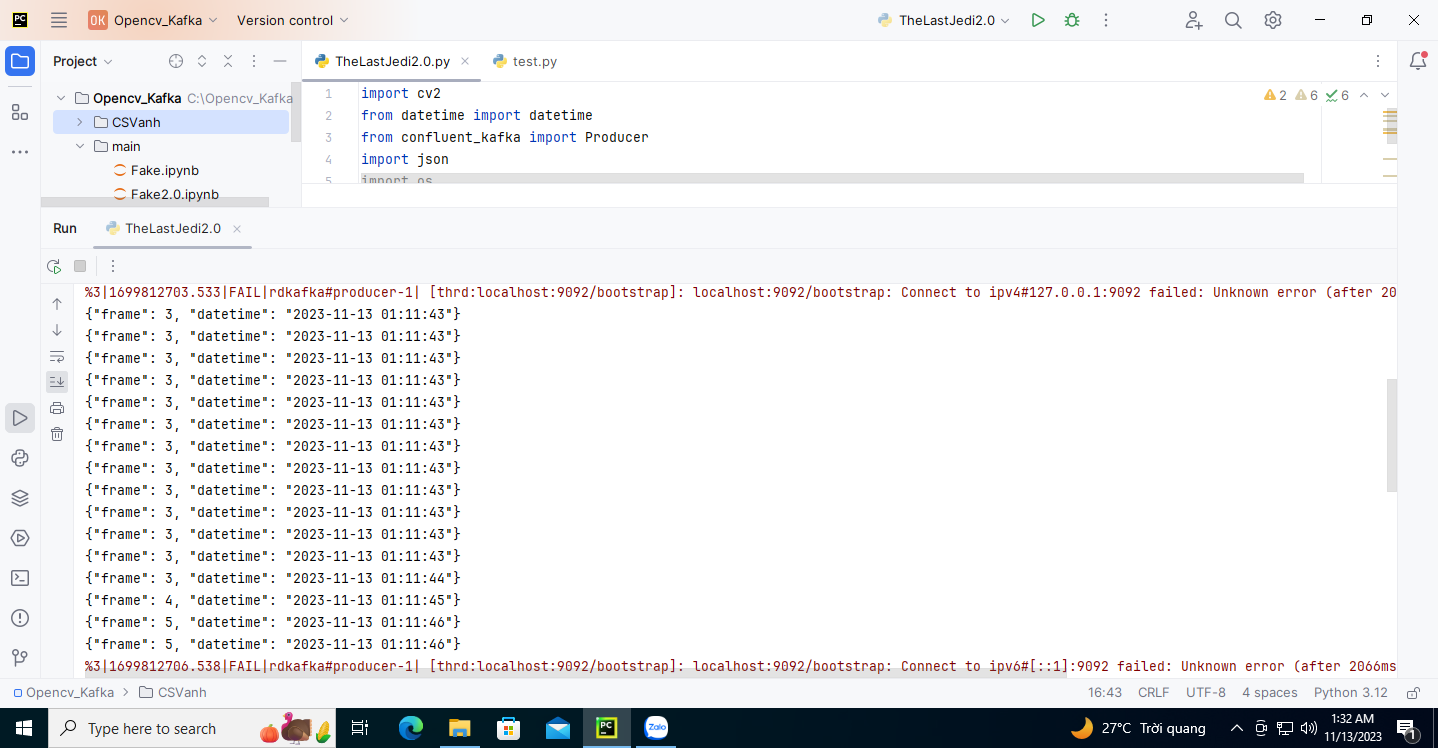


Khởi động jupyter notebook

jupyter notebook

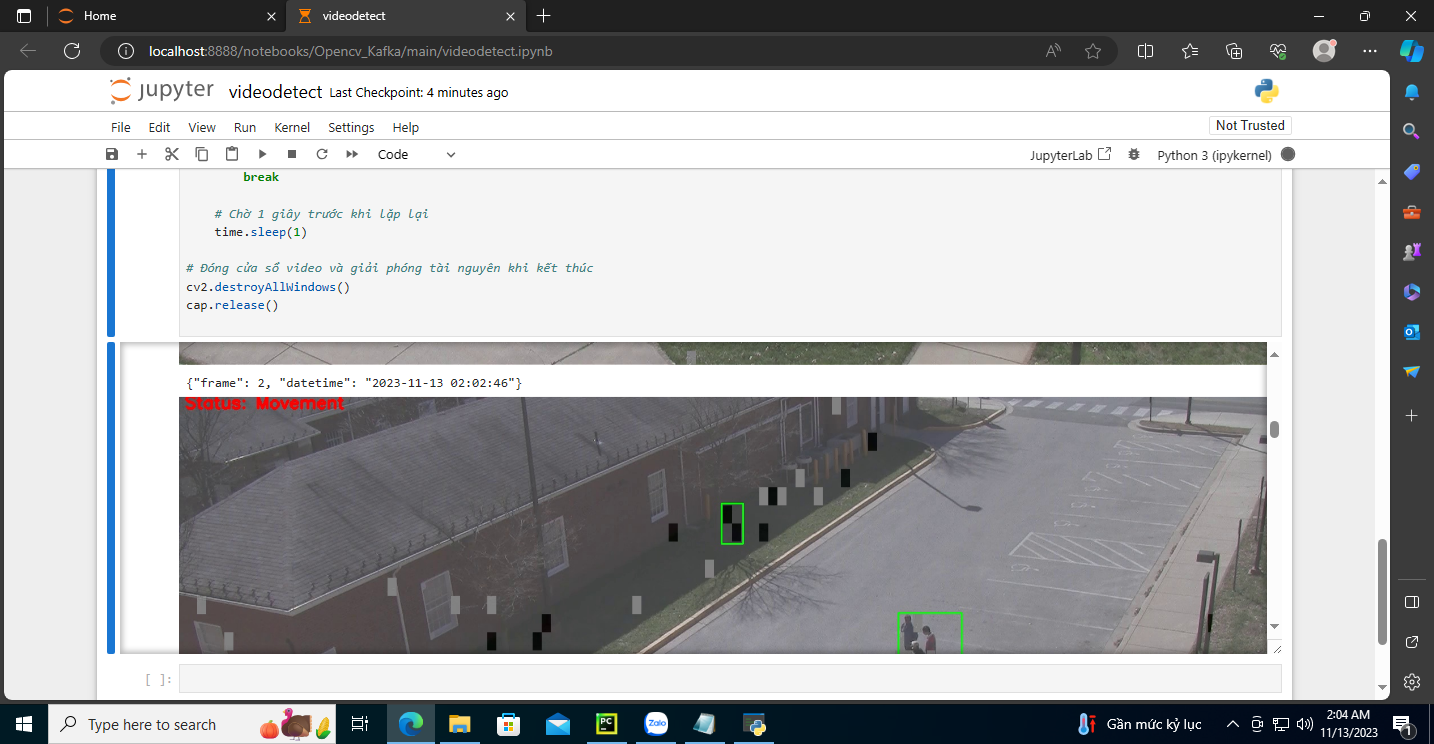


Gửi dữ liệu tới Kafka: Video dữ liệu hoặc các khung hình đã được xử lý có thể được gửi tới Apache Kafka để phân phối nhanh và hệ thống mở rộng.

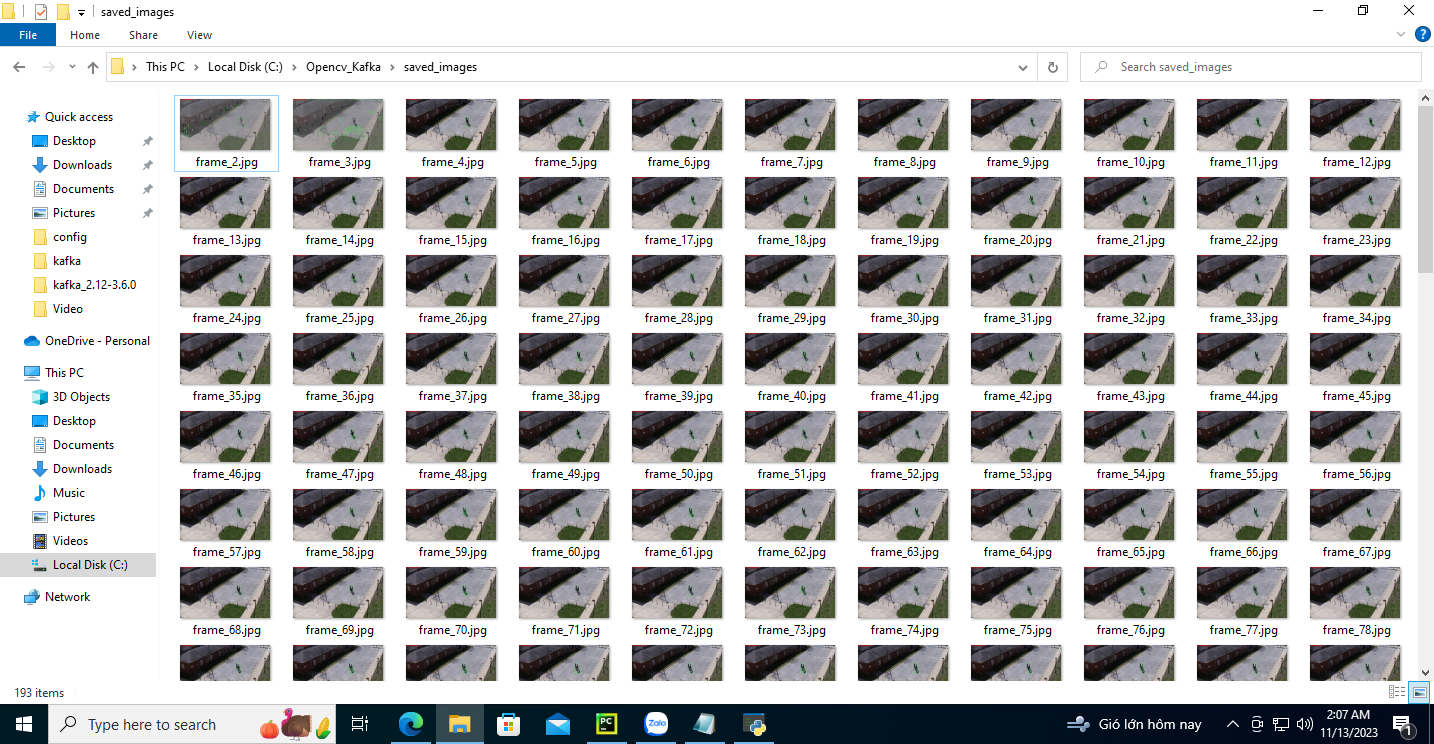


Xử lý dữ liệu bằng Apache Spark Streaming: Sử dụng Apache Spark Streaming để xử lý dữ liệu video đến từ Kafka.

Bạn có thể phát triển các công cụ xử lý phức tạp như phát hiện chuyển động, thống kê hoặc lọc cơ sở dữ liệu dựa trên các công cụ yếu tố yếu tố.



Hiển thị kết quả hoặc lưu trữ: Kết quả xử lý có thể được hiển thị ngay lập tức hoặc lưu trữ vào cơ sở dữ liệu hoặc kho lưu trữ khác của hệ thống.



* CODE:

from IPython.display import display, Image  
import cv2  
from datetime import datetime  
from confluent\_kafka import Producer  
import json  
import os  
import time  
  
*# Callback function được gọi khi Kafka Producer gửi tin nhắn thành công hoặc thất bại*def delivery\_report(err, msg):  
 if err is not None:  
 print('Message delivery failed: {}'.format(err))  
 else:  
 print('Message delivered to {} [{}]'.format(msg.topic(), msg.partition()))  
  
*# Khởi tạo cấu hình cho Kafka Producer*kafka\_bootstrap\_servers = 'localhost:9092'  
kafka\_topic = 'videodetect'  
producer\_config = {  
 'bootstrap.servers': kafka\_bootstrap\_servers,  
}  
producer = Producer(producer\_config)  
  
*# Đường dẫn của video đầu vào*cap = cv2.VideoCapture("C:\\Linhtinh\\Video\\VIRAT\_S\_000002.mp4")  
  
*# Thư mục để lưu trữ hình ảnh xuất*output\_folder = "C:\\Users\\admin\\OneDrive\\saved\_images"  
  
*# Tạo thư mục nếu nó chưa tồn tại*os.makedirs(output\_folder, exist\_ok=True)  
  
*# Đọc frame đầu tiên từ video*ret, frame1 = cap.read()  
ret, frame2 = cap.read()  
print(frame1.shape)  
  
*# Tạo cửa sổ mới để hiển thị video*cv2.namedWindow("Video", cv2.WINDOW\_NORMAL)  
cv2.resizeWindow("Video", 800, 600)  
  
*# Vòng lặp chính để xử lý video frame by frame*while cap.isOpened():  
 *# Tính toán sự thay đổi giữa hai frame liên tiếp* diff = cv2.absdiff(frame1, frame2)  
 gray = cv2.cvtColor(diff, cv2.COLOR\_BGR2GRAY)  
 blur = cv2.GaussianBlur(gray, (5, 5), 0)  
 \_, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH\_BINARY)  
 dilated = cv2.dilate(thresh, None, iterations=3)  
 contours, \_ = cv2.findContours(dilated, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
  
 *# Xử lý các contours để xác định vùng di chuyển* for contour in contours:  
 (x, y, w, h) = cv2.boundingRect(contour)  
  
 if cv2.contourArea(contour) < 900:  
 continue  
 *# Vẽ hộp giới hạn xung quanh vùng di chuyển* cv2.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)  
 cv2.putText(frame1, "Status: Movement", (10, 20), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
  
 *# Hiển thị video trong cửa sổ mới* cv2.imshow("Video", frame1)  
  
 *# Hiển thị hình ảnh trên console Jupyter* \_, img\_encoded = cv2.imencode('.jpg', frame1)  
 display(Image(data=img\_encoded.tobytes(), format='jpg'))  
  
 *# Tạo một dictionary chứa thông tin về sự kiện* event\_datetime = datetime.now()  
 event\_data = {  
 "frame": int(cap.get(cv2.CAP\_PROP\_POS\_FRAMES)),  
 "datetime": event\_datetime.strftime("%Y-%m-%d %H:%M:%S")  
 }  
  
 *# Chuyển đổi dictionary thành chuỗi JSON* event\_json = json.dumps(event\_data)  
  
 *# In dữ liệu JSON ra console* print(event\_json)  
  
 *# Gửi dữ liệu JSON đến Kafka* producer.produce(kafka\_topic, key=None, value=event\_json, callback=delivery\_report)  
  
 *# Lưu hình ảnh vào thư mục xuất* frame\_filename = f"{output\_folder}\\frame\_{int(cap.get(cv2.CAP\_PROP\_POS\_FRAMES))}.jpg"  
 cv2.imwrite(frame\_filename, frame1)  
  
 frame1 = frame2  
 ret, frame2 = cap.read()  
  
 *# Dừng vòng lặp nếu nhấn phím Esc* if cv2.waitKey(40) == 27:  
 break  
  
 *# Chờ 1 giây trước khi lặp lại* time.sleep(1)  
  
*# Đóng cửa sổ video và giải phóng tài nguyên khi kết thúc*cv2.destroyAllWindows()  
cap.release()

# Đề xuất

Để cải thiện hệ thống xử lý video streaming phát hiện chuyển động sử dụng OpenCV, Kafka và Spark, dưới đây là một số đề xuất:

Tối ưu hóa Xử Lý Video và Phát Hiện Chuyển Động:

* Thu gọn Dữ liệu: Thay vì xử lý mọi khung hình, có thể áp dụng kỹ thuật giảm thiểu, ví dụ như chỉ xử lý phần nào đó của video hoặc giảm độ phân giải nếu có thể.
* Thuật toán Tối ưu hóa: Nghiên cứu và áp dụng các thuật toán phát hiện chuyển động hiệu quả hơn để giảm tài nguyên tính toán và tăng độ chính xác.

Tăng Cường Hệ Thống Phân Tán:

* Mở Rộng Kafka: Sử dụng Kafka cluster để tăng khả năng mở rộng và đảm bảo khả năng chịu tải tốt hơn cho việc truyền tải dữ liệu.
* Optimize Spark Cluster: Tối ưu hoá cấu hình và quy mô của Spark cluster để đảm bảo hiệu suất xử lý dữ liệu tốt nhất có thể.

Tích Hợp Mô Hình Học Máy:

* Mô Hình Học Máy: Sử dụng học máy để cải thiện khả năng phân loại chuyển động.
* Có thể áp dụng mô hình như CNN (Convolutional Neural Networks) để tăng cường độ chính xác của việc phát hiện chuyển động.
* Học Có Giám Sát: Thu thập dữ liệu và huấn luyện mô hình học máy để phân loại chuyển động với độ chính xác cao hơn.

Giám Sát và Log:

* Giám Sát Hệ Thống: Sử dụng các công cụ giám sát để theo dõi hiệu suất hệ thống, xác định và khắc phục sự cố nhanh chóng.
* Ghi Log Thông Minh: Ghi log chi tiết về quá trình xử lý và phát hiện chuyển động để dễ dàng xác định lỗi hoặc cải thiện hiệu suất.

Bảo Mật và Quản Lý Dữ Liệu:

* Bảo Mật Dữ Liệu: Đảm bảo an toàn dữ liệu trong quá trình truyền tải từ Kafka và xử lý trong Spark bằng việc áp dụng các phương pháp mã hóa và kiểm soát truy cập.

Quản Lý Dữ Liệu:

* Xây dựng các cơ chế quản lý dữ liệu chính xác để lưu trữ và tiếp nhận dữ liệu từ video streaming.

Bằng việc kết hợp các đề xuất này, bạn có thể cải thiện hiệu suất và độ chính xác của việc phát hiện chuyển động trong video streaming và đồng thời đảm bảo tính bảo mật và quản lý tốt dữ liệu.

Xử lý video streaming để phát hiện chuyển động là một ứng dụng thú vị và hữu ích trong nhiều lĩnh vực như giám sát an ninh, quản lý chuỗi cung ứng, theo dõi sản xuất, và nhiều ứng dụng khác.

Sử dụng các công nghệ như OpenCV, Kafka và Spark, bạn có thể xây dựng một hệ thống phát hiện chuyển động trong thời gian thực.

Triển khai ứng dụng phát hiện chuyển động sử dụng OpenCV, Kafka và Spark sẽ giúp bạn tạo ra một hệ thống quản lý và phân tích video streaming hiệu quả và có khả năng mở rộng cho nhiều ứng dụng thực tế.

# Đánh giá

Xử lý video streaming để phát hiện chuyển động sử dụng OpenCV, Kafka và Spark có nhiều ưu điểm, nhưng cũng có một số khuyết điểm cần xem xét. Dưới đây là một đánh giá về ưu điểm và khuyết điểm của việc triển khai ứng dụng này:

Ưu Điểm:

1. Xử lý thời gian thực: Ứng dụng này có khả năng xử lý video streaming trong thời gian thực hoặc gần thời gian thực, cho phép phát hiện chuyển động một cách nhanh chóng và đáng tin cậy.

2. Phân phát dữ liệu phân tán: Sử dụng Kafka cho việc phân phát video streaming cho các ứng dụng Spark, giúp tối ưu hóa việc xử lý dữ liệu và quản lý tải.

3. Tích hợp với OpenCV: OpenCV là một thư viện mạnh mẽ cho xử lý hình ảnh và video. Kết hợp với Spark, nó tạo ra khả năng phát hiện chuyển động và xử lý hình ảnh phức tạp.

4. Mở rộng dễ dàng: Spark cho phép bạn mở rộng hệ thống một cách dễ dàng bằng cách thêm nhiều máy chủ xử lý dữ liệu, giúp xử lý lượng dữ liệu lớn mà không cần thay đổi nhiều về kiến trúc.

5. Tích hợp với các công nghệ khác:Ứng dụng này có thể tích hợp với nhiều công nghệ và các thành phần khác trong hệ thống của bạn, ví dụ: cơ sở dữ liệu, hệ thống bảo mật, hệ thống ghi nhật ký, và nhiều ứng dụng khác.

Khuyết Điểm:

1. Phức tạp trong triển khai: Triển khai và cấu hình các thành phần như Kafka, Spark và OpenCV có thể đòi hỏi kiến thức kỹ thuật sâu và thời gian.

2. Tài nguyên và hiệu suất: Xử lý video streaming có thể đòi hỏi tài nguyên lớn, đặc biệt khi bạn xử lý nhiều luồng dữ liệu video cùng một lúc. Cần đảm bảo rằng hệ thống có đủ tài nguyên để đảm bảo hiệu suất.

3. Quản lý dữ liệu lớn: Với lượng dữ liệu video lớn, quản lý và lưu trữ dữ liệu trở nên phức tạp. Cần thiết phải có một chiến lược quản lý dữ liệu cẩn thận.

4. Bảo mật và quyền riêng tư: Bảo mật dữ liệu video là một thách thức, đặc biệt khi bạn phát hiện chuyển động trong hệ thống an ninh. Cần thực hiện biện pháp bảo mật mạnh mẽ để đảm bảo an toàn và quyền riêng tư.

5. Khả năng giám sát: Cần có các công cụ giám sát và theo dõi để đảm bảo rằng hệ thống hoạt động một cách ổn định và không gặp sự cố.

Tóm lại, việc xử lý video streaming để phát hiện chuyển động bằng cách sử dụng OpenCV, Kafka và Spark có thể cung cấp giá trị lớn trong nhiều ứng dụng.

Sự kết hợp giữa OpenCV, Kafka và Spark trong việc xử lý video streaming phát hiện chuyển động mang lại tiềm năng lớn trong việc phân tích video thời gian thực.

Mặc dù có những thách thức về triển khai và quản lý tài nguyên, nhưng cung cấp khả năng xử lý lớn và hiệu suất cao đối với ứng dụng yêu cầu xử lý video streaming phức tạp.

Điều này mở ra cánh cửa cho việc áp dụng trong nhiều lĩnh vực từ an ninh đến theo dõi sản xuất.

Tuy nhiên, cần phải quản lý các thách thức liên quan đến triển khai, hiệu suất, bảo mật và quản lý dữ liệu để đảm bảo rằng hệ thống hoạt động hiệu quả và đáng tin cậy.

# Kết luận

Phân tích video quy mô lớn của các luồng video đòi hỏi một hệ thống mạnh mẽ được hỗ trợ bởi các công nghệ dữ liệu lớn. Các công nghệ nguồn mở như OpenCV, Kafka và Spark có thể được sử dụng để xây dựng một hệ thống phân tán và chịu lỗi cho phân tích luồng video.

Chúng tôi đã sử dụng OpenCV và Kafka để xây dựng thành phần bộ sưu tập luồng video nhận luồng video từ các nguồn khác nhau và gửi chúng đến thành phần bộ đệm dữ liệu luồng.

Kafka có vai trò là thành phần đệm dữ liệu luồng cung cấp khả năng lưu trữ dữ liệu phát trực tuyến bền vững.

Thành phần bộ xử lý luồng video được phát triển bằng OpenCV và Spark's Structured Streaming.

Thành phần này nhận trực tuyến dữ liệu từ luồng dữ liệu đệm và dữ liệu phân tích. Các tệp đã được xử lý đã được lưu trữ trong kho lưu trữ HDFS hoặc S3 đã được cấu hình sẵn.

Việc bắt chuyển động từ video bằng OpenCV, Kafka và Spark có nhiều ứng dụng thực tế, bao gồm an ninh, giám sát giao thông, theo dõi sản xuất, y tế, và nhiều ngữ cảnh khác. Nó giúp cải thiện quản lý, đảm bảo an ninh, và cung cấp thông tin theo thời gian thực cho quyết định quan trọng.

Bắt chuyển động trực tiếp từ video bằng OpenCV, Kafka và Spark là tạo ra một hệ thống mạnh mẽ cho việc xử lý và phân tích dữ liệu video trực tiếp trong thời gian thực. Dưới đây là một số ý nghĩa của quá trình này:

1. Giám sát và An ninh: Hệ thống này cho phép theo dõi và giám sát môi trường thông qua video theo thời gian thực. Nó có thể được ứng dụng trong an ninh, giám sát giao thông, và các lĩnh vực khác để phát hiện sự kiện bất thường hoặc đối tượng đáng chú ý.

2. Phân tích Dữ liệu Video: Quá trình này cho phép bạn phân tích dữ liệu từ video, ví dụ: phát hiện chuyển động, nhận dạng đối tượng, tính toán tốc độ, và nhiều tác vụ phân tích dữ liệu khác.

3. Xử lý Trực tiếp và Phân tán: Sử dụng Apache Kafka và Spark, bạn có thể xử lý dữ liệu video trực tiếp và phân tán nó trên nhiều máy chủ. Điều này giúp gia tăng hiệu suất và khả năng mở rộng của hệ thống.

4. Thông tin Theo Thời Gian Thực: Hệ thống này cung cấp thông tin và phản hồi theo thời gian thực, giúp bạn đưa ra quyết định nhanh chóng dựa trên dữ liệu video cập nhật liên tục.

5. Ứng dụng Rộng Rãi: Ý nghĩa của quá trình này rất đa dạng và có thể áp dụng trong nhiều lĩnh vực, bao gồm an ninh, giao thông, y tế, sản xuất, và nhiều ứng dụng khác. Hệ thống này có tiềm năng giúp cải thiện quản lý và an ninh trong nhiều ngữ cảnh.

Tóm lại, việc kết hợp OpenCV, Kafka và Spark trong việc bắt chuyển động trực tiếp từ video là một công cụ mạnh mẽ để giải quyết các vấn đề phát sinh từ công việc quản lý và phân tích dữ liệu video. Nó cung cấp khả năng theo dõi và phản hồi nhanh chóng các sự kiện và vấn đề quan trọng, mang lại lợi ích cho nhiều cảnh ứng dụng khác nhau.