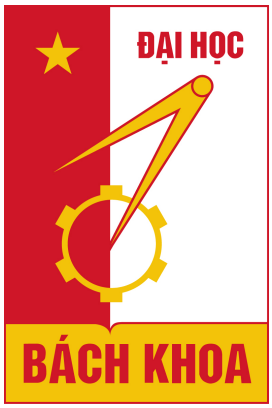
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

o0o



HỆ THỐNG DỮ LIỆU
PHÂN TÍCH NHU CẦU TUYỂN DỤNG

BÁO CÁO
IT4931 LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU LỚN

Giảng viên hướng dẫn: PGS.TS Nguyễn Bình Minh

Sinh viên: Nguyễn Tiến Long - 20180129
Phan Việt Hoàng - 20180086
Phạm Trần Anh - 20180018
Võ Hồng Sang - 20183973

Lớp: Tài năng Công nghệ thông tin K63

Hà Nội - 2022

Mục lục

1. Giới thiệu

Nắm bắt được nhu cầu thị trường lao động có thể giúp cho sinh viên cũng như những người đang tìm việc có được sự chuẩn bị tốt nhất khi đi ứng tuyển xin việc tại các công ty. Điều đó giúp họ trở thành những ứng viên phù hợp với yêu cầu doanh nghiệp cũng như giúp cho doanh nghiệp giảm bớt được thời gian đào tạo nhân viên. Để biết được thị trường lao động đang cần gì, một giải pháp đơn giản mà hiệu quả là thực hiện đánh giá, thống kê những kỹ năng, kiến thức được miêu tả trong các đơn tuyển dụng của các công ty trên các trang mạng tìm việc làm. Các công đoạn khi thực hiện giải pháp này cơ bản sẽ bao gồm thu thập dữ liệu, lọc dữ liệu và biểu diễn, thống kê dữ liệu. Nếu như chỉ cần đánh giá một vài lĩnh vực, ngành nghề trên một trang tuyển dụng, thì việc này có thể thực hiện nhanh chóng trên một chiếc máy laptop. Vấn đề dần dần xuất hiện khi ta thực hiện những đánh giá rộng hơn trên nhiều ngành nghề, nhiều các trang web khác nhau. Lượng dữ liệu thu thập sẽ ngày càng phình to khiến việc lưu trữ trên một máy tính trở nên khó khăn, kéo theo đó là việc xử lý đánh giá dữ liệu cũng không thể thực hiện được nhanh chóng. Bài toán đặt ra ở đây là thiết kế một hệ thống có khả năng lưu trữ dữ liệu tuyển dụng thu thập trên Internet, xử lý nhanh chóng và biểu diễn trực quan các thống kê về nhu cầu việc làm.

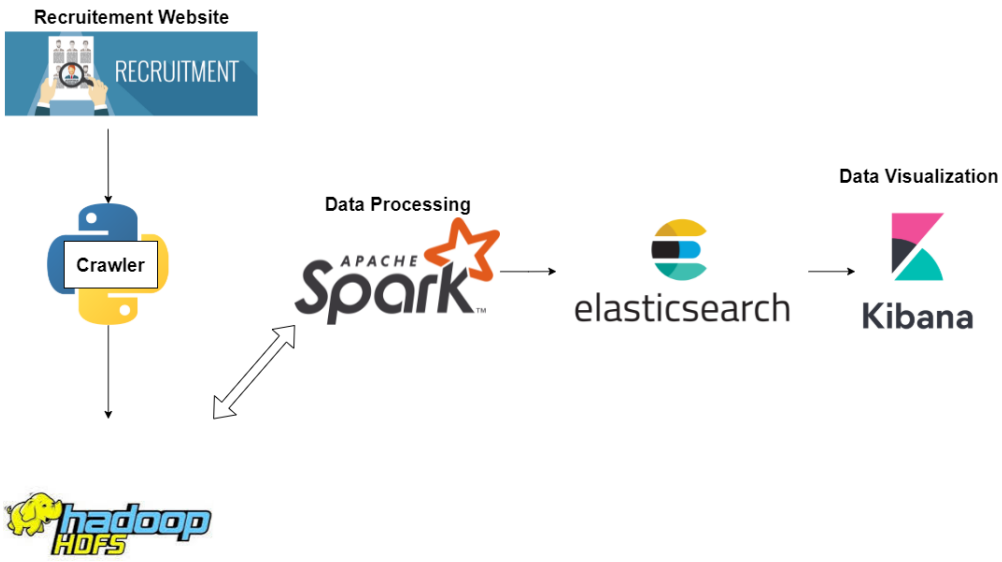
Trong báo cáo này, nhóm sinh viên đề xuất một hệ thống có khả năng lưu trữ và xử lý dữ liệu lớn, tên là RecruitmentInsight. Hệ thống được triển khai thử nghiệm bằng công cụ Docker Compose. Nguồn dữ liệu nhóm lựa chọn để nghiên cứu là dữ liệu liên quan đến việc làm trong lĩnh vực phần mềm, thu thập từ trang web TopCV.

Nhóm đã xây dựng RecruitmentInsight nhờ vận dụng các bài giảng về Hadoop, Spark, Elasticsearch Kibana cùng các bài giảng khác trong môn học Lưu trữ và xử lý dữ liệu lớn, dưới sự hướng dẫn nhiệt tình và các góp ý quý giá của giảng viên hướng dẫn, Phó giáo sư Tiến sĩ, thầy Nguyễn Bình Minh.

Nhóm sinh viên
Nguyễn Tiến Long
Phan Việt Hoàng
Phạm Trần Anh
Võ Hồng Sang

2. Hệ thống

2.1 Tổng quan hệ thống



Hình 2.1: Tổng quan hệ thống

Hệ thống được xây dựng gồm 4 phần với các chức năng nhằm thu thập, xử lý, lưu trữ và trực quan hoá dữ liệu tuyển dụng từ internet. Các thành phần của hệ thống bao gồm:

- Bộ phần thu thập dữ liệu: sử dụng BeautifulSoup, là một công cụ để phân tích cú pháp các văn bảng dạng HTML và XML, chuyên dụng trong việc thu thập dữ liệu từ các trang web
- Bộ phận lưu trữ : hệ thống lưu trữ dữ liệu vào Hadoop dưới dạng HDFS File System (HDFS) để có thể lưu dữ liệu phân tán và có chức năng sao lưu, đảm bảo truy cập được khi một số máy mất kết nối

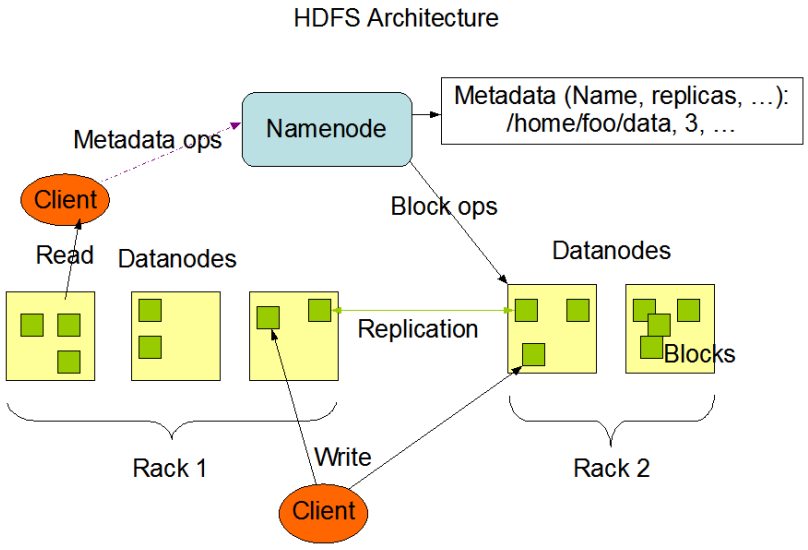
- Bộ phận xử lý dữ liệu: từ dữ liệu đã được lưu trong Hadoop, Spark được sử dụng để xử lý, làm sạch dữ liệu và thực hiện các truy vấn, giúp cho việc biểu diễn dữ liệu đơn giản hơn. Dữ liệu sau khi được làm sạch được lại được lưu về Hadoop và Elasticsearch
- Bộ phận biểu diễn dữ liệu: dữ liệu sau khi được xử lý bởi Spark được đưa vào Elasticsearch thông qua một thư viện mã nguồn mở là Elasticsearch for Apache Hadoop

2.2 Chi tiết hệ thống

2.2.1 Hadoop Cluster

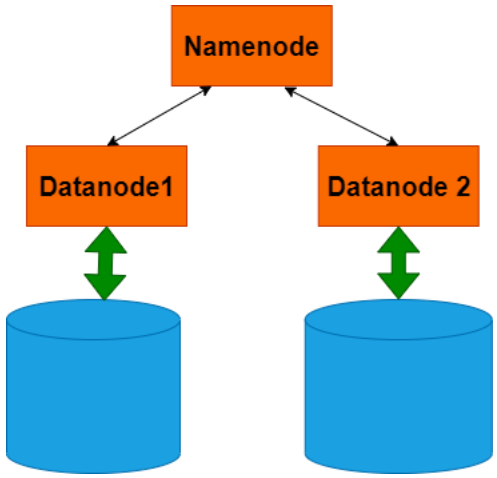
Một hệ thống dữ liệu lớn cần có một cụm các máy để lưu trữ dữ liệu phân tán. Cụm các máy này cần phải có khả năng chịu lỗi (ví dụ như một vài máy bị hỏng) và cho phép thực hiện các tác vụ cơ bản lên dữ liệu.

Hadoop Cluster ra đời để giải quyết nhu cầu đó. Một cụm Hadoop bao gồm nhiều các máy tính khác nhau, tuân theo kiến trúc Master-Slave, cho phép lưu trữ và phân tích hàng petabytes dữ liệu. Khi lưu trữ trên Hadoop, file dữ liệu được chia thành các chunk và được lưu thành nhiều bản sao, giúp cho cụm Hadoop có khả năng chịu lỗi.



Hình 2.2: Kiến trúc Master-Slave của Hadoop (nguồn : Apache Hadoop)

Đối với hệ thống RecruitmentInsight, dữ liệu thu thập được trên Internet sẽ được lưu trên cụm Hadoop. Cụm Hadoop của RecruitmentInsight bao gồm một Namenode và 2 Datanode. Khi lượng dữ liệu tăng lên, kiến trúc này có thể mở rộng thêm bằng cách bổ sung các Datanode để tăng cường dung lượng lưu trữ của hệ thống.



Hình 2.3: Hadoop Cluster trong hệ thống RecruitmentInsight

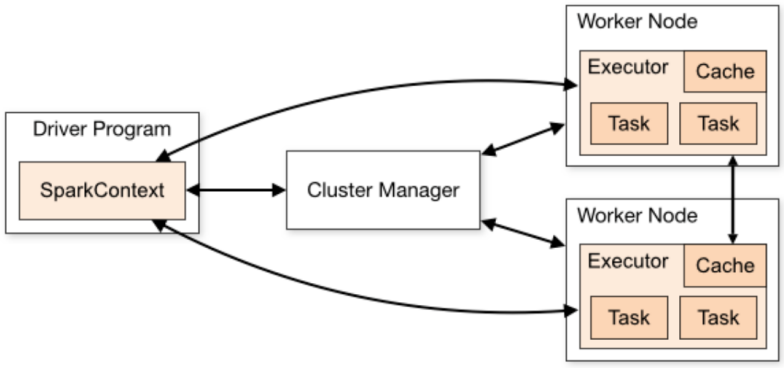
2.2.2 Spark Cluster

Apache Spark là công nghệ hỗ trợ tính toán nhanh với lượng dữ liệu lớn. Được xây dựng trên Hadoop MapReduce, Spark mở rộng mô hình này để thực hiện hiệu quả các bài toán học máy, các tác vụ truy vấn dữ liệu, xử lý luồng dữ liệu thời gian thực. Ưu điểm của Spark MapReduce so với Hadoop MapReduce là tốc độ tính toán nhanh chóng do được thực hiện tại bộ nhớ trong hay thực hiện hoàn toàn trên RAM. Một số thống kê về tính năng nổi bật của Spark có thể liệt kê ra như :

- Tốc độ: Spark giúp cho việc chạy các ứng dụng trên Hadoop Cluster nhanh gấp 100 lần khi thực hiện ở bộ nhớ trong, nhanh gấp 10 lần khi thực hiện trên ổ cứng.
- Hỗ trợ nhiều ngôn ngữ: Spark cung cấp các API có thể dễ dàng lập trình bằng Scaka, Java, Python, R, những ngôn ngữ chuyên dụng cho xử lý dữ liệu.
- Phân tích dữ liệu: bên cạnh MapReduce, các tác vụ phân tích dữ liệu như truy vấn SQL, học máy, streaming data hay các giải thuật đồ thị đều có được cung cấp bởi Spark.

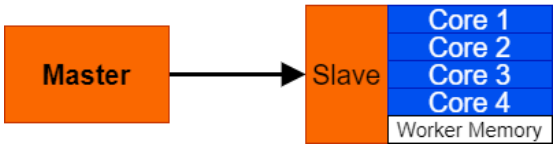
Các tính năng Spark mang lại hoàn toàn phù hợp với nhu cầu của hệ thống khi mà dữ liệu được lưu trong Hadoop Cluster và cần được tiền xử lý trước khi biểu diễn.

Cũng giống như Hadoop, kiến trúc của Spark tuân theo mô hình Master-Slave, trong đó gồm 1 driver (master node) và nhiều worker nodes. Khi thực thi, chương trình Driver (trên master node) tạo ra 1 SparkContext, sau đó giao tiếp với Cluster Manager để tính toán tài nguyên và phân chia các tác vụ đến cho các worker node.



Hình 2.4: Kiến trúc Spark (nguồn Apache Spark)

Spark Cluster trong hệ thống RecruitmentInsight được triển khai thành một Standalone Cluster gồm 1 master node(Resource Manager) và 1 Worker Node. Nếu khối lượng tính toán quá lớn, tính khả mở của hệ thống cho phép thiết lập thêm các Worker Node khác để chia sẻ tài nguyên với các Worker hiện tại.



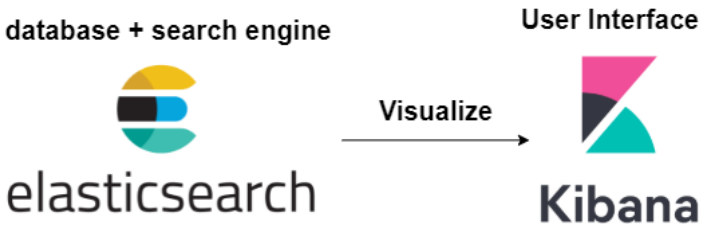
Hình 2.5: Standalone Spark Cluster trong hệ thống RecruitmentInsight

2.2.3 Elasticsearch và Kibana

Dữ liệu sau khi được làm sạch bởi Spark cần được biểu diễn dưới dạng bảng biểu, đồ thị để mang đến cho người dùng góc nhìn trực quan nhất. Elasticsearch và Kibana là những ứng dụng phù hợp để đảm nhận vai trò này.

Là một công cụ tìm kiếm (với tốc độ gần thời gian thực) và phân tích dữ liệu phân tán, Elasticsearch có thể lưu trữ và phân tích nhiều loại dữ liệu khác nhau như: giữ liệu có cấu trúc, giữ liệu phi cấu trúc, giữ liệu số, dữ liệu về không gian địa lý, đánh chỉ mục dữ liệu một cách hiệu quả nhằm hỗ trợ quá trình tìm kiếm được thực hiện nhanh chóng. Các truy vấn trên Elasticsearch được thực hiện thông qua API, curl , python, hoặc qua Kibana. Kibana cung cấp giao diện đồ hoạ để người dùng dễ dàng hơn trong việc khai phá, biểu diễn trực quan dữ liệu được lưu trên Elasticsearch.

Hệ thống RecruitmentInsight triển khai cụm Elasticsearch gồm một máy Elasticsearch đóng vai trò là cơ sở dữ liệu và 1 máy Kibana để người dùng tương tác trực tiếp.



Hình 2.6: Elasticsaerch Cluster

2.3 Xây dựng hệ thống bằng Docker Compose

Docker là một dự án mã nguồn mở giúp quá trình triển khai các ứng dụng, phần mềm trên các hệ điều hành Windows, Linux hoặc MacOS trở nên đơn giản thông qua các containers. Container được phát hành dưới dạng một bộ đóng gói chứa tất cả thành phần cần thiết để chạy ứng dụng: thư viện, file cấu hình,$\cdots$. Để chạy đồng thời nhiều containers, cấu hình của các container (images, ports, environemnt variables,$\cdots$) được khai báo chung trong một file dạng *YAML*, docker compose sẽ sử dụng file này để thiết lập các containers cùng lúc. Khi được triển khai bằng docker compose, các container có thể giao tiếp với nhau thông qua tên của các container (được khai báo trong file *YAML*) mà không cần phải chỉ rõ địa chỉ IP.

Hệ thống RecruitmentInsight được thử nghiệm bằng cách chạy các ứng dụng Hadoop, Spark, Elasticsearch trên các container khác nhau, mỗi container mô phỏng một máy tính khi được triển khai thực thế. Thông số cụ thể về các container được cho trong bảng 2.1

Cluster	Container	Provider	Ports	Version	Number
Hadoop	namenode	bde2020	9870, 9000	2.0.0-hadoop3.2.1-java8	1
	datanode	bde2020		2.0.0-hadoop3.2.1-java8	2
Spark	spark-master	bde2020	8080, 7070	3.0.0-hadoop3.2	1
	spark-worker	bde2020		3.0.0-hadoop3.2	1
Elastic	elasticsearch	docker.elastic.co	9200, 9300	7.15.1	1
	kibana	docker.elastic.co	5601	7.15.1	1

Bảng 2.1: Thông số các container

3. Thực nghiệm

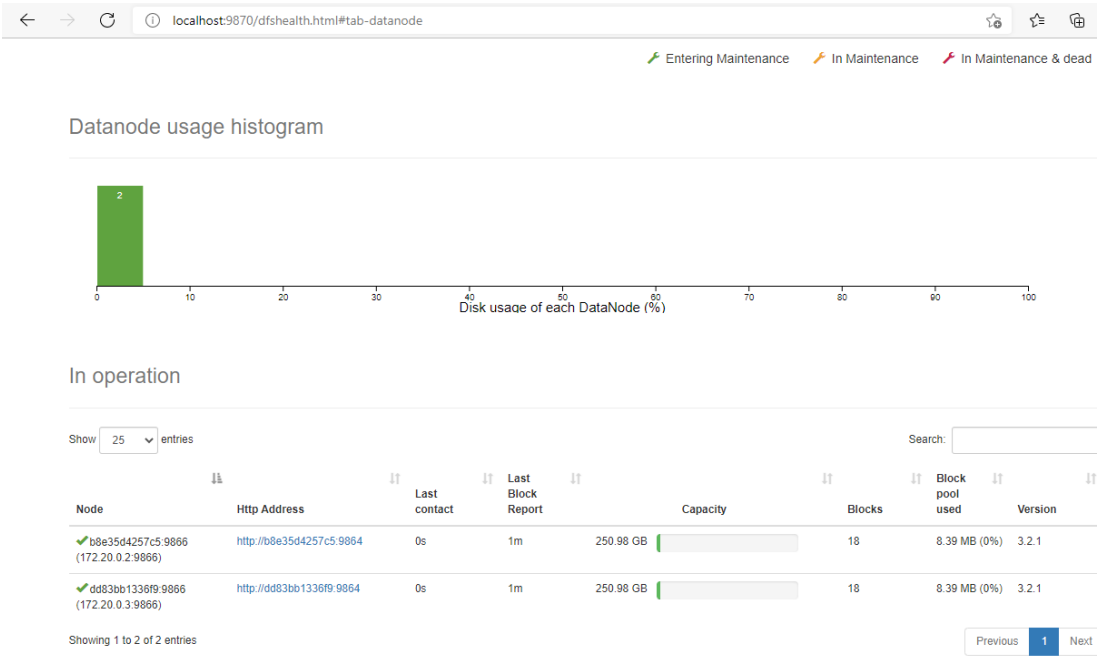
Mã nguồn dùng để thu thập dữ liệu có tại repo IT-Jobs-TopCV-Crawler. Mã nguồn để thử nghiệm hệ thống có tại repo BigData-HDFS-Spark-Elasticsearch-Kibana

3.1 Khởi chạy hệ thống

Do được xây dựng bằng Docker compose nên hệ thống có thể triển khai nhanh chóng bằng câu lệnh

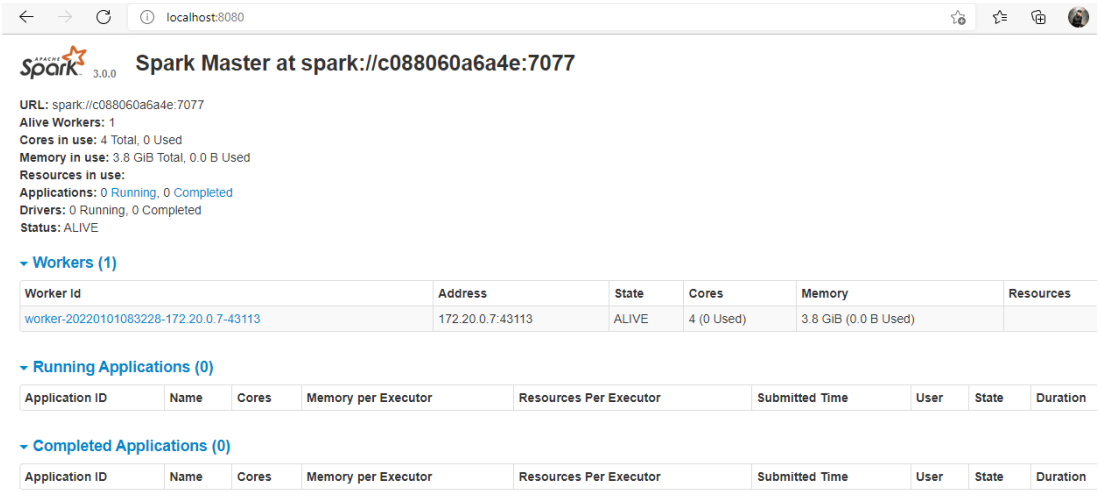
```
1    docker compose up -d
```

Hadoop Cluster



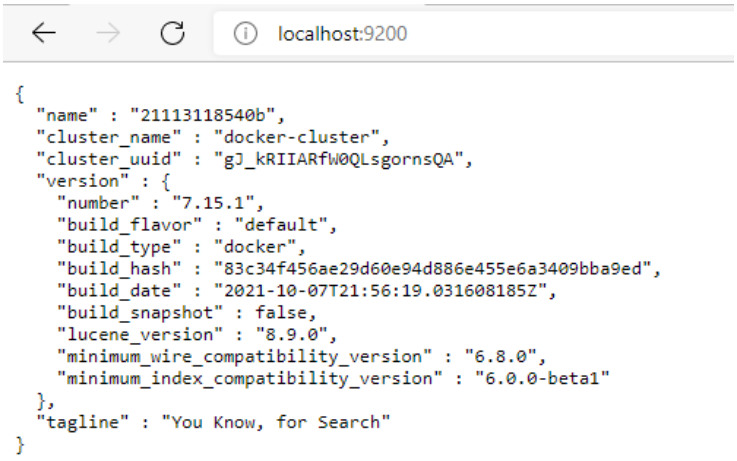
Hình 3.1: Hadoop Cluster đang hoạt động với một namenode và 2 datanode

Spark Cluster



Hình 3.2: Spark Cluster đang hoạt động với một Spark Master và 1 Spark Worker

Elasticsearch Cluster



Hình 3.3: Elasticsearch Container

3.2 Luồng dữ liệu của hệ thống

Luồng dữ liệu của hệ thống gồm 4 giai đoạn. Giai đoạn thứ nhất là thu thập trên Internet, sau đó được lưu vào Hadoop. Ứng dụng Spark sẽ làm sạch dữ liệu lưu trên Hadoop và lưu

thành 2 bản, 1 bản lưu vào Elasticsearch để biểu diễn, bản còn lại lưu về Hadoop. Dữ liệu trên Elasticsearch được biểu diễn dưới dạng bảng biểu, đồ thị bằng Kibana

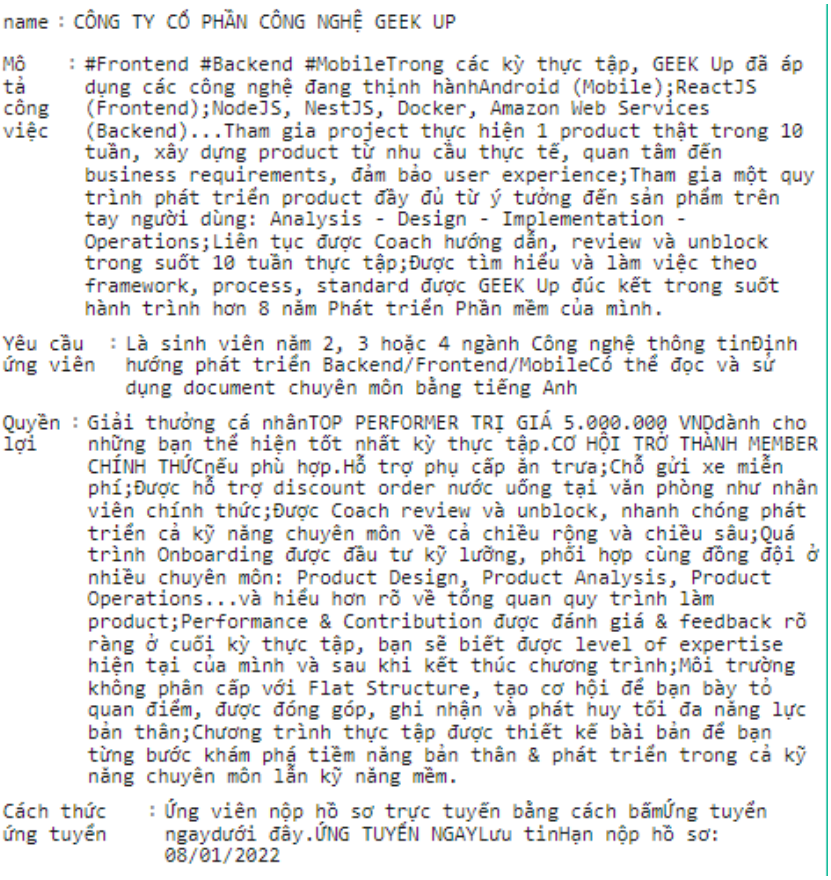
Thu thập dữ liệu

Dữ liệu của hệ thống là dữ liệu tuyển dụng liên quan đến lĩnh vực phần mềm, có thể thu thập được tại website TopCV. Tại thời điểm dữ liệu được thu thập, trên TopCV có tất cả 140 trang, file *html* của mỗi trang có chứa *link* đến đơn tuyển dụng của từng công ty. Hệ thống sẽ truy cập vào từng link và thu thập thông tin theo các thẻ. Mỗi đơn tuyển dụng sẽ được lưu thành một đối tượng $json$(một bản ghi), trong đó tên của các thẻ trong *html* và nội dung của các thẻ tương ứng sẽ tạo thành các cặp key-value. Một bản ghi sẽ bao gồm các trường sau :

- name (tên công ty tuyển dụng)
- Mô tả công việc
- Yêu cầu ứng viên
- Quyền lợi
- Cách thức ứng tuyển

Hình 3.4 là ví dụ về một bản ghi thu thập được từ một đơn tuyển dụng. Bản ghi đã được trình bày lại bằng phần mềm JsonFormatter.

Chương trình thu thập dữ liệu của hệ thống được lưu ở file *crawler.py*, sử dụng thư viện BeautifulSoup để parse các văn bản *html*. Để tăng tốc độ thực thi, hệ thống sử dụng một bash script để chạy song song 14 luồng cùng lúc, mỗi luồng thu thập dữ liệu trên 10 trang liên tiếp. Dữ liệu trả về được lưu ở 14 file $json$, tương ứng với kết quả chạy đồng thời của 14 luồng. Toàn bộ mã nguồn để thu thập dữ liệu được lưu trong repo IT-Jobs-TopCV-Crawler.



name : CÔNG TY CỔ PHẦN CÔNG NGHỆ GEEK UP

Mô tả công việc : #Frontend #Backend #MobileTrong các kỳ thực tập, GEEK Up đã áp dụng các công nghệ đang thịnh hànhAndroid (Mobile);ReactJS (Frontend);NodeJS, NestJS, Docker, Amazon Web Services (Backend)...Tham gia project thực hiện 1 product thật trong 10 tuần, xây dựng product từ nhu cầu thực tế, quan tâm đến business requirements, đảm bảo user experience;Tham gia một quy trình phát triển product đầy đủ từ ý tưởng đến sản phẩm trên tay người dùng: Analysis - Design - Implementation - Operations;Liên tục được Coach hướng dẫn, review và unblock trong suốt 10 tuần thực tập;Được tìm hiểu và làm việc theo framework, process, standard được GEEK Up đúc kết trong suốt hành trình hơn 8 năm Phát triển Phần mềm của mình.

Yêu cầu ứng viên : Là sinh viên năm 2, 3 hoặc 4 ngành Công nghệ thông tinĐịnh hướng phát triển Backend/Frontend/MobileCó thể đọc và sử dụng document chuyên môn bằng tiếng Anh

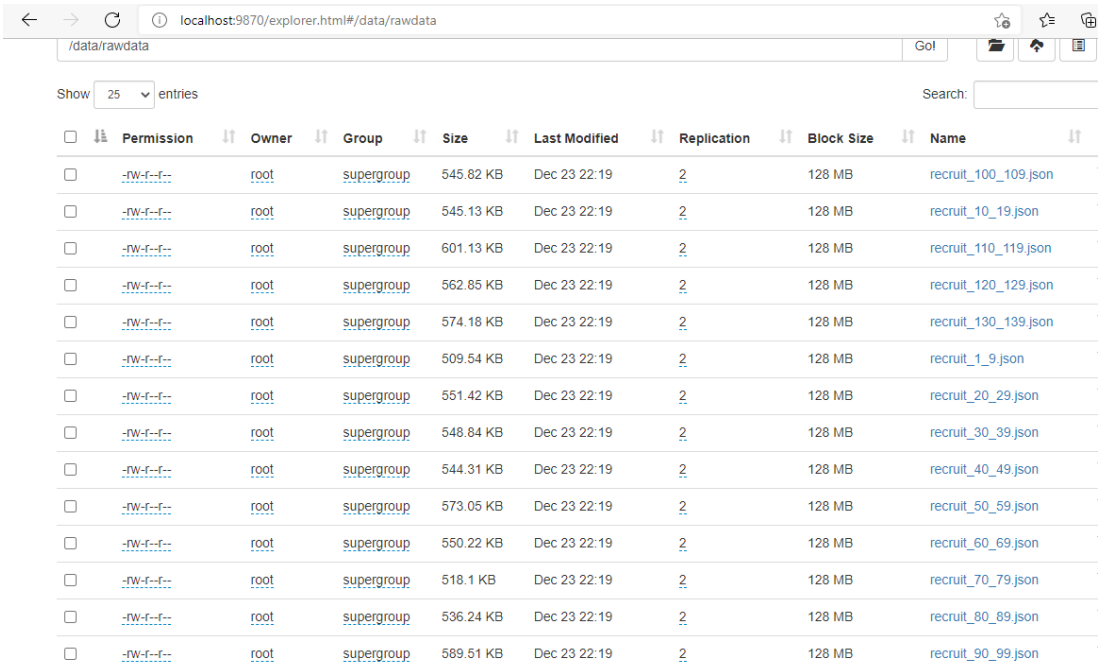
Quyền lợi : Giải thưởng cá nhânTOP PERFORMER TRỊ GIÁ 5.000.000 VNDdành cho những bạn thể hiện tốt nhất kỳ thực tập.CƠ HỘI TRỞ THÀNH MEMBER CHÍNH THỨCnếu phù hợp.Hỗ trợ phụ cấp ăn trưa;Chỗ gửi xe miễn phí;Được hỗ trợ discount order nước uống tại văn phòng như nhân viên chính thức;Được Coach review và unblock, nhanh chóng phát triển cả kỹ năng chuyên môn về cả chiều rộng và chiều sâu;Quá trình Onboarding được đầu tư kỹ lưỡng, phối hợp cùng đồng đội ở nhiều chuyên môn: Product Design, Product Analysis, Product Operations...và hiểu hơn rõ về tổng quan quy trình làm product;Performance & Contribution được đánh giá & feedback rõ ràng ở cuối kỳ thực tập, bạn sẽ biết được level of expertise hiện tại của mình và sau khi kết thúc chương trình;Môi trường không phân cấp với Flat Structure, tạo cơ hội để bạn bày tỏ quan điểm, được đóng góp, ghi nhận và phát huy tối đa năng lực bản thân;Chương trình thực tập được thiết kế bài bản để bạn từng bước khám phá tiềm năng bản thân & phát triển trong cả kỹ năng chuyên môn lẫn kỹ năng mềm.

Cách thức ứng tuyển : Ứng viên nộp hồ sơ trực tuyến bằng cách bấmỨng tuyển ngaydưới đây.ỨNG TUYỂN NGAYLưu tinHạn nộp hồ sơ: 08/01/2022

Hình 3.4: Ví dụ một bản ghi

Đẩy dữ liệu lên Hadoop

Các file dữ liệu nhận được sau khi được thu thập được đẩy lên và lưu trữ trong thư mục */data/rawdata/* tại Hadoop Cluster. Mỗi file sẽ được lưu thành 2 bản, để tăng cường tính chống lỗi của hệ thống.



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	545.82 KB	Dec 23 22:19	2	128 MB	recruit_100_109.json
-rw-r--r--	root	supergroup	545.13 KB	Dec 23 22:19	2	128 MB	recruit_10_19.json
-rw-r--r--	root	supergroup	601.13 KB	Dec 23 22:19	2	128 MB	recruit_110_119.json
-rw-r--r--	root	supergroup	562.85 KB	Dec 23 22:19	2	128 MB	recruit_120_129.json
-rw-r--r--	root	supergroup	574.18 KB	Dec 23 22:19	2	128 MB	recruit_130_139.json
-rw-r--r--	root	supergroup	509.54 KB	Dec 23 22:19	2	128 MB	recruit_1_9.json
-rw-r--r--	root	supergroup	551.42 KB	Dec 23 22:19	2	128 MB	recruit_20_29.json
-rw-r--r--	root	supergroup	548.84 KB	Dec 23 22:19	2	128 MB	recruit_30_39.json
-rw-r--r--	root	supergroup	544.31 KB	Dec 23 22:19	2	128 MB	recruit_40_49.json
-rw-r--r--	root	supergroup	573.05 KB	Dec 23 22:19	2	128 MB	recruit_50_59.json
-rw-r--r--	root	supergroup	550.22 KB	Dec 23 22:19	2	128 MB	recruit_60_69.json
-rw-r--r--	root	supergroup	518.1 KB	Dec 23 22:19	2	128 MB	recruit_70_79.json
-rw-r--r--	root	supergroup	536.24 KB	Dec 23 22:19	2	128 MB	recruit_80_89.json
-rw-r--r--	root	supergroup	589.51 KB	Dec 23 22:19	2	128 MB	recruit_90_99.json

Hình 3.5: Data được lưu trữ tại Hadoop Cluster

Xử lý dữ liệu tại Spark

Dữ liệu được lưu trữ tại Hadoop mới chỉ là dữ liệu thô, cần được trích xuất để mang loại bỏ thông tin dư thừa giúp tối ưu khả năng lưu trữ cũng như mang lại những tri thức, những góc nhìn có ý nghĩa về dữ liệu đối với người dùng. Việc khai phá tri thức từ dữ liệu sẽ do Spark đảm nhiệm. Trước tiên, Spark sẽ định nghĩa một schema để đọc dữ liệu tại Hadoop thành một dataframe:

```
schema = StructType([
      StructField("name",StringType(),True),
      StructField("Mô tả công việc",StringType(),True),
      StructField("Yêu cầu ứng viên",StringType(),True),
      StructField("Quyền lợi",StringType(),True),
      StructField("Cách thức ứng tuyển",StringType(),True)
  ])
```

Hình 3.6: schema

Một dataframe *raw_recruit_df* với schema đã được định nghĩa như trên được tạo ra từ dữ liệu lưu trong các file *json* được lưu trong Hadoop tại bước trước.

```
raw_recruit_df.show(5)

+--------------------+--------------------+--------------------+--------------------+--------------------+
|                name|     Mô tả công việc|    Yêu cầu ứng viên|          Quyền lợi| Cách thức ứng tuyển|
+--------------------+--------------------+--------------------+--------------------+--------------------+
|Công Ty Cổ Phần C...|- Làm việc trực t...|- Tốt nghiệp ĐH t...|- Thu nhập: Lên đ...|Ứng viên nộp hồ s...|
|     Viettel Digital|-\tPhát triển các...|- Có kinh nghiệm ...|-\tMức lương thu ...|Ứng viên nộp hồ s...|
|     Viettel Digital|Chịu trách nhiệm ...|Tốt nghiệp đại họ...|Mức lương thu hút...|Ứng viên nộp hồ s...|
|CÔNG TY CỔ PHẦN T...|- Cài đặt hệ điều...|- Tốt nghiệp trun...|- Lương từ 10 tri...|Ứng viên nộp hồ s...|
|     Viettel Digital|Phát triển các ứn...|Tốt nghiệp Đại họ...|Có thể phỏng vấn ...|Ứng viên nộp hồ s...|
+--------------------+--------------------+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Hình 3.7: 5 dòng đầu của dataframe tạo từ dữ liệu thô

Tuy nhiên, *raw_recruit_df* vẫn chỉ là 1 dataframe với dữ liệu thô. Từ *raw_recruit_df*, ứng dụng tại Spark sẽ trích xuất thông tin để tạo ra một dataframe, với các trường dữ liệu bao gồm :

- *CompanyName* : tên công ty tuyển dụng
- *FrameworksPlattforms* : một mảng gồm tên các frameworks, plattforms mà công ty tuyển dụng yêu cầu
- *Languages*: một mảng gồm tên các ngôn ngữ lập trình mà công ty tuyển dụng yêu cầu
- *DesignPatterns* : một mảng gồm tên các design patterns mà công ty tuyển dụng yêu cầu
- *Knowledges*: một mảng gồm tên các kiến thức, các kỹ năng mà công ty tuyển dụng yêu cầu
- *Salaries* : một mảng gồm các mức lương mà công ty tuyển dụng chi trả.

Các trường thông tin *FrameworksPlattforms*, *Languages*, *DesignPatterns*, *Knowledges* được trích xuất theo cùng một cách là tìm các xâu trong dữ liệu gốc mà khớp với các xâu được định nghĩa sẵn (gọi là các pattern) tương ứng với mỗi trường. Ví dụ, với trường *Languages*, các pattern được định nghĩa trước trong một mảng như ở hình 3.8

Mảng các pattern như vậy được định đặt trong file *patterns.py*. Các hàm user define function (đặt trong file *udfs.py*) sử dụng các pattern này để trích xuất thông tin từ dataframe. Ví dụ về một user define function được cho trong hình 3.9

Đối với trường *Salaries* thì việc làm sạch dữ liệu sẽ phức tạp hơn. Ý tưởng của nhóm là thống kê lương theo các khoảng 5 triệu VNĐ. Mức lương trong các đơn tuyển dụng sẽ được chia vào các khoảng tương ứng, biểu diễn bằng một mảng các số nguyên là chặn dưới của mỗi khoảng. Bảng 3.1 cho một số ví dụ về việc chuyển đổi mức lương.

```
languages = [' CHAIN ', ' ABAP ', 'Lingo', ' CPL', 'NPL', 'Xtend', ' Flex ', ' Io ', 'Erlang', 'Python', 'MSL', 'SAIL', ' Chef ', 'RPG', 'PHP',
             'XPath', 'Lava', ' Clojure', 'Mathematica', 'QPL', 'Oak', 'Objective-C', 'P#', 'css', ' Delphi', ' A+ ', 'XQuery',
             'CoffeeScript', ' SR ', 'ECMAScript', 'Nial', ' Red ', 'Mesa', 'LSL', 'T-SQL', 'E#', ' GAP ', 'Simula', 'Logo', ' Caml',
             'JavaScript', 'PeopleCode', ' UNITY ', ' Blue ', 'Span', 'Lucid', ' BeanShell', ' CSP', 'Scheme', 'Swift', 'TypeScript',
             ' Scala ', 'Scratch', 'Strand', 'XML', ' SAS', 'PortablE', 'JADE', ' C ', 'Processing', 'Pure', ' BASIC ', ' Bash', 'Pro*C',
             'ROOP', 'PL/SQL', 'Icon', ' Dart', ' Factor ', 'Java', 'LINC', ' Go ', ' TIE ', ' Cool', 'Kotlin', 'Rust', 'Opa', 'DYNAMO',
             ' Inform ', 'Mary', 'Ruby', 'YQL', 'Pike', ' rc ', ' html ', 'Oz', 'Groovy', 'PowerShell', ' CUDA', 'Hack', ' Self ',
             ' CFEngine', 'C#', 'SPS']
```

Hình 3.8: Mảng các xâu được định nghĩa trước dùng để trích xuất thông tin liên quan đến ngôn ngữ lập trình

```
@udf(returnType=ArrayType(StringType()))
def extract_framework_plattform(mo_ta_cong_viec,yeu_cau_ung_vien):
    return [framework for framework in patterns.framework_plattforms if re.search(framework, mo_ta_cong_viec + " " + yeu_cau_ung_vien, re.IGNORECASE)]
```

Hình 3.9: user define function để lọc các thông tin về frameworks và plattfornms

Mức lương	Mảng quy đổi
8 triệu VNĐ	[5]
26-38 triệu VNĐ	[25,30,35]
2000$	[45]

Bảng 3.1: Phím, dữ liệu truyền và nốt nhạc tương ứng

Riêng các mức lương trên 100 triệu VNĐ sẽ được biểu diễn bởi mảng một phần tử [100].

Để trích xuất thông tin về lương từ đơn tuyển dụng cần sử dụng các pattern là các xâu và các biểu thức chính quy

```
salary_patterns = ["lương(?:từ| )+ ((?:\d+|\.)+)", "((?:\d+|\.|-| )+(?:triệu| )+)đồng",
                   "(?:\d|\.|,)+.000.000", "(?:\d+| |-)+\d+ *(?:triệu|m)", "\$(?:\d+|\.)", "(?:\d+|\.)+ *(?:USD|\$)+",
                   "(?:\d|\.|,)+,000,000"]
```

Hình 3.10: Mảng các pattern để lọc thông tin liên quan đến *Salaries*

Các xâu nhận được sau khi lọc bằng các pattern này sẽ có các dạng ví dụ như sau : *' 08 triệu', '26.000.000', '$8000', ' 15 - 25 triệu', ' 2 m', '$300', '10-25M', ' 07-10 triệu', ' 18 - 30 m',* $\cdots$. Các xâu này vẫn còn phi cấu trúc, ta cần thêm một số bước nữa để chuẩn hoá dữ liệu như quy đổi các xâu có đơn vị dollar Mỹ ra đơn vị triệu VNĐ, chuyển các xâu có đơn vị VNĐ ra đơn vị triệu VNĐ. Mức lương sau khi được chuẩn hoá sẽ được chuyển thành mảng như trên.

Với các user define function được định nghĩa, một dataframe mới, $extracted\_recruit\_df$, được lọc từ $raw\_recruit\_df$

```
extracted_recruit_df=raw_recruit_df.select(raw_recruit_df["name"].alias("CompanyName"),
      udfs.extract_framework_plattform("Mô tả công việc","Yêu cầu ứng viên").alias("FrameworkPlattforms"),
      udfs.extract_language("Mô tả công việc","Yêu cầu ứng viên").alias("Languages"),
      udfs.extract_design_pattern("Mô tả công việc","Yêu cầu ứng viên").alias("DesignPatterns"),
      udfs.extract_knowledge("Mô tả công việc","Yêu cầu ứng viên").alias("Knowledges"),
      udfs.normalize_salary("Quyền lợi").alias("Salaries")
      )
extracted_recruit_df.cache()
```

Hình 3.11: Tạo dataframe với dữ liệu được lọc từ dataframe ban đầu

```
extracted_recruit_df.show(5)
```

```
+--------------------+--------------------+--------------------+----------------+--------------------+--------+
|         CompanyName| FrameworkPlattforms|           Languages|  DesignPatterns|          Knowledges|Salaries|
+--------------------+--------------------+--------------------+----------------+--------------------+--------+
|Công Ty Cổ Phần C...|     [MySQL, Oracle]|                  []|              []|          [kiểm thử]|    [25]|
|     Viettel Digital|         [Git, JSON]|[Objective-C, Swi...|[design pattern]|[UI/UX, Unit Test...]|     [5]|
|     Viettel Digital|[Reactjs, Vue, An...|[css, JavaScript,...|[design pattern]|[đồ họa, UI/UX, f...]|     [5]|
|CÔNG TY CỔ PHẦN T...|             [.NET]|                  []|              []| [Switch, phần cứng]| [0, 10]|
|     Viettel Digital|         [Git, JSON]|         [XML, Java]|              []|[UI/UX, Unit Test...| [5, 30]|
+--------------------+--------------------+--------------------+----------------+--------------------+--------+
only showing top 5 rows
```

Hình 3.12: 5 dòng đầu của dataframe lọc từ dataframe ban đầu

Tiền xử lý và lưu dữ liệu

Dataframe *extracted_recruit_df* về cơ bản là đã có thể tiến hành biểu diễn trên Kibana, tuy nhiên ta vẫn cần tiến hành tiền xử lý thêm một só bước để việc biểu diễn dễ dàng hơn. Khi người dùng quan tâm đến một nhóm các kiến thức mà thị trường tuyển dụng đang yêu cầu, thay vì các tri thức riêng rẽ, ví dụ như quan tâm đến một nhóm các kiến thức về*blockchain* và *bảo mật*, thay vì chỉ quan tâm đến các kiến thức cụ thể như *smart contract* hay *Defi*. Lúc này, chương trình cần gán nhãn trước các cho các kiến thức về một nhóm kiến thức. Với các nhãn này, từ dataframe *extracted_recruit_df* có thể đếm ra được các bản ghi chứa một nhóm tri thức cụ thể.

```
labeled_knowledges={' AI': 'AI', 'Machine Learning': 'AI', 'Data mining': 'AI', 'Chatbot': 'AI', 'data analys': 'AI',
                    'blockchains': 'blockchain_crypto', 'crypto': 'blockchain_crypto', 'NFT': 'blockchain_crypto',
                    'smart contract': 'blockchain_crypto', 'Solidity': 'blockchain_crypto', 'Defi': 'blockchain_crypto',
                    'XSS': 'blockchain_crypto', ' Security ': 'blockchain_crypto',
                    'lắp đặt': 'hardware', 'sửa chữa': 'hardware', 'phần cứng': 'hardware', 'router': 'hardware',
                    'Corel Draw': 'hardware', 'Switch': 'hardware',
                    'Word': 'office', 'Excel': 'office', 'Powerpoint': 'office', 'Office': 'office',
                    'Illustrator': 'photoshop', 'Photoshop': 'photoshop', 'Animate': 'photoshop',
                    'cấu trúc dữ liệu': 'programming_basic', 'thuật toán': 'programming_basic', 'OOP': 'programming_basic',
                    'hướng đối tượng': 'programming_basic',
                    'Black Box': 'tester', 'tester': 'tester', 'White Box': 'tester', 'Unit Test': 'tester',
                    'TestRail': 'tester', 'kiểm thử': 'tester',
                    'SVN': 'version_control', 'SCM': 'version_control', 'Git': 'version_control'}
```

Hình 3.13: Nhãn của một số kiến thức yêu cầu

Chương trình sử dụng 1 hàm udf để đánh nhãn các string trong cột *Knowledge* của dataframe *extracted_recruit_df*. Tuy nhiên, để hàm udf tìm được dictionary trong lúc đánh nhãn thì cần phải broadcast dictionary trước.

```
def broadcast_labeled_knowledges(sc,labeled_knowledges):
    '''
    broadcast the mapped of labeled_knowledges to group data in knowledge field
    '''
    global mapped_knowledge
    mapped_knowledge = sc.broadcast(labeled_knowledges)

@udf(returnType=StringType())
def labeling_knowledge(knowledge):
    try :
        return mapped_knowledge.value[knowledge]
    except :
        return None
```

Hình 3.14: Hàm broacast nhãn và udf để map các string trong cột *Knowledge* của dataframe *extracted_recruit_df*

Dữ liệu lúc này đã sẵn sàng để lưu về Hadoop và Elasticsearch, chương trình sử dụng 2 hàm *save_dataframes_to_hdfs()* và *save_dataframes_to_elasticsearch()* để tiến hành lưu trữ.

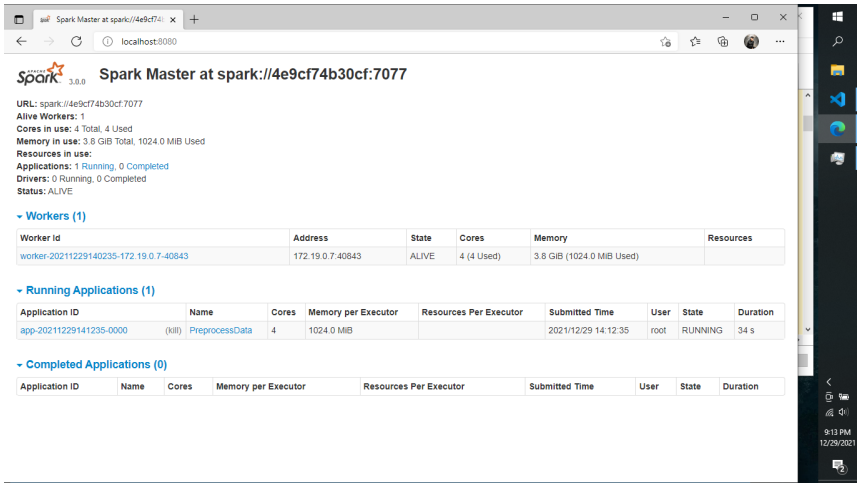
```
def save_dataframes_to_hdfs(path,config,data_dfs,target_file_names):
    for data_df,target_file_name in zip(data_dfs,target_file_names):
        print("Processing file: ",target_file_name)
        print("Processing dataframe of type ",type(data_df))
        data_df.write.format("json").mode("overwrite").save(config.
    get_hdfs_namenode()+"/"+path+"/"+target_file_name)

def save_dataframes_to_elasticsearch(dataframes,indices,es_write_config):
    for dataframe,index in zip(dataframes,indices):
        print("Processing index:",index)

        es_write_config['es.resource'] = index

        rdd_ = dataframe.rdd
        rdd_.map(lambda row: (None, \
                              json.dumps(row.asDict()))) \
                              .saveAsNewAPIHadoopFile(path='-', \
                              outputFormatClass="org.elasticsearch.hadoop.mr
    .EsOutputFormat", \
                              keyClass="org.apache.hadoop.io.NullWritable",
    \
                              valueClass="org.elasticsearch.hadoop.mr.
    LinkedMapWritable", \
                              conf=es_write_config)
```

Để Spark và Elasticsearch tương tác với nhau cần sử dụng thư viện Elasticsearch for Apache Hadoop. Thư viện có thể tải về từ Maven Repository dưới dạng file jar (ví dụ *elasticsearch-hadoop-7.15.1.jar*)

Mã nguồn của ứng dụng Spark được lưu trong folder src với hàm *main* ở file *main.py*. Sau khi upload folder *src* và file *elasticsearch-hadoop-7.15.1.jar* lên spark-master, chương trình có thể thực thi bằng *spark-submit* như sau:

```
spark/bin/spark-submit \
--master spark://spark-master:7077 \
--jars path_to_file/elasticsearch-hadoop-7.15.1.jar \
--driver-class-path path_to_file/elasticsearch-hadoop-7.15.1.jar \
path_to_folder/src/main.py
```

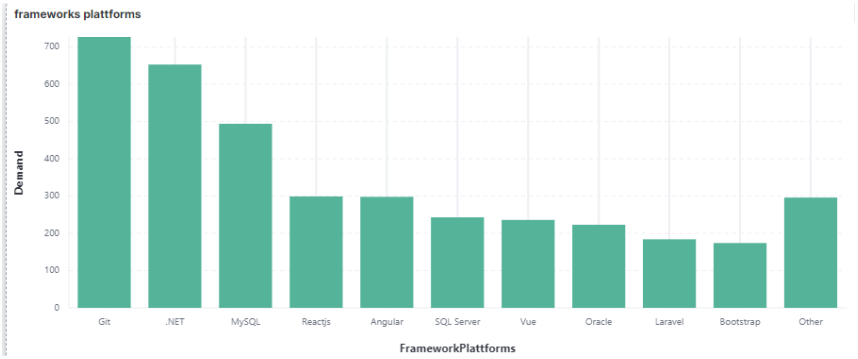
Spark-master sẽ tiến hành phân chia tác vụ và tài nguyên cho các spark-worker.



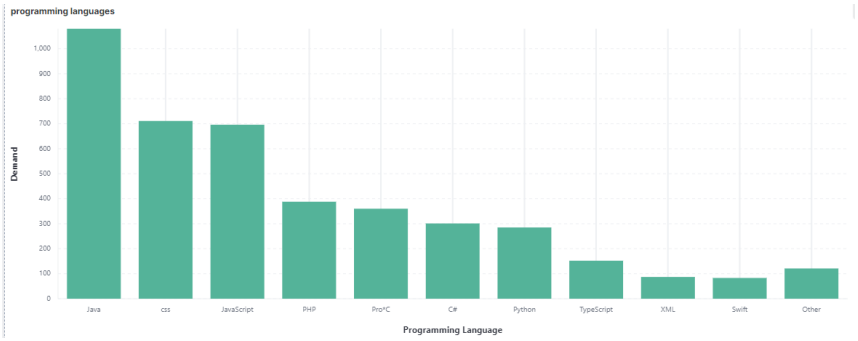
Hình 3.15: Spark worker đang xử lý các tác vụ được giao bởi spark-master, sử dụng toàn bộ 4 core

Biểu diễn dữ liệu trên Kibana

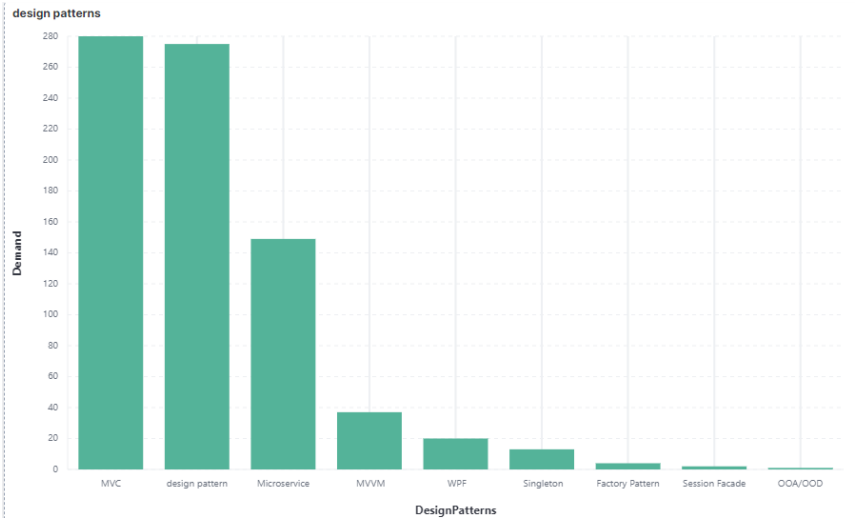
Dữ liệu lưu tại Elasticsearch sẽ được dùng Kibana để biểu diễn. Một số các thống kê về dữ liệu được cho trong các hình dưới đây.



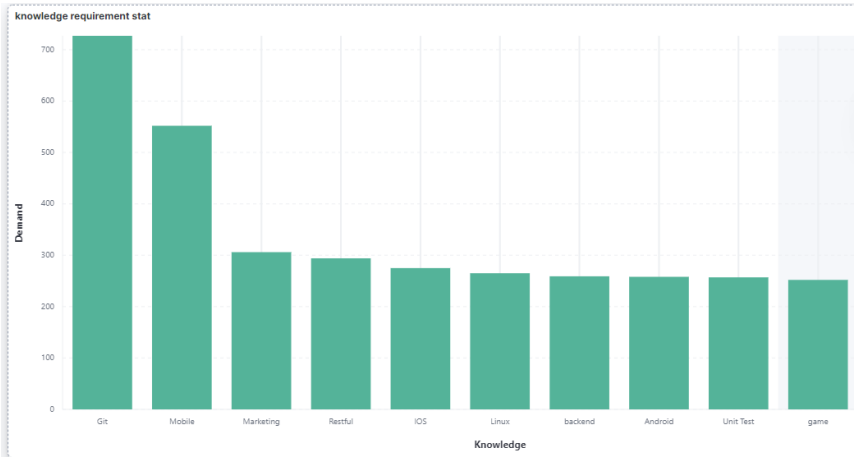
Hình 3.16: Thống kê yêu cầu thành thạo các frameworks, plattforms



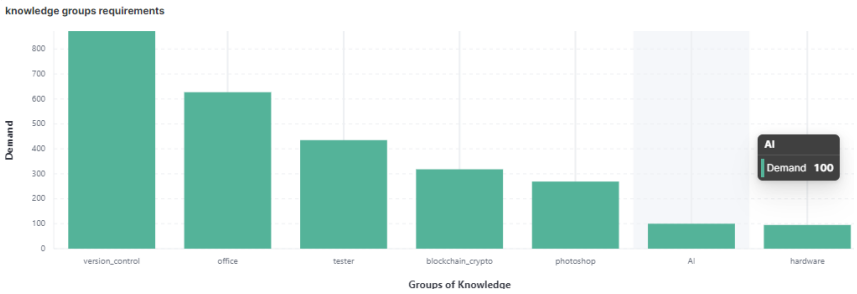
Hình 3.17: Thống kê yêu cầu thành thạo các ngôn ngữ lập trình



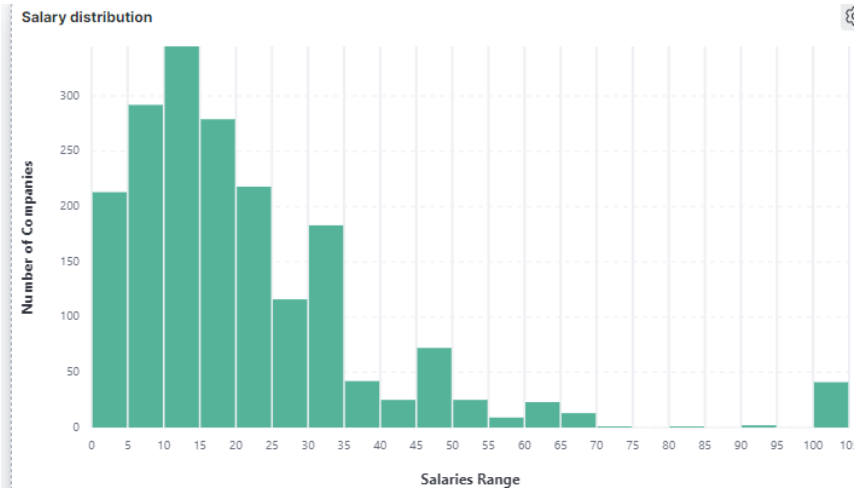
Hình 3.18: Thống kê yêu cầu thành thạo các design patterns



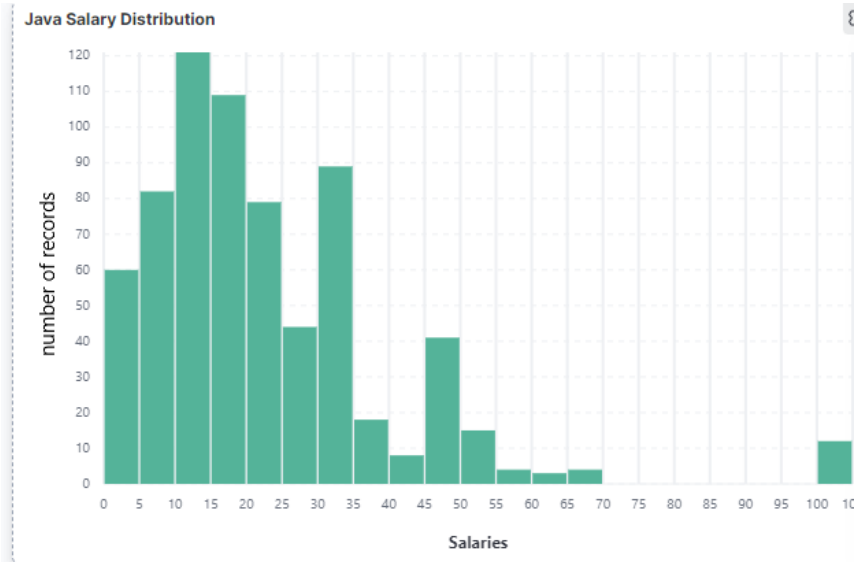
Hình 3.19: Thống kê yêu cầu về kiến thức



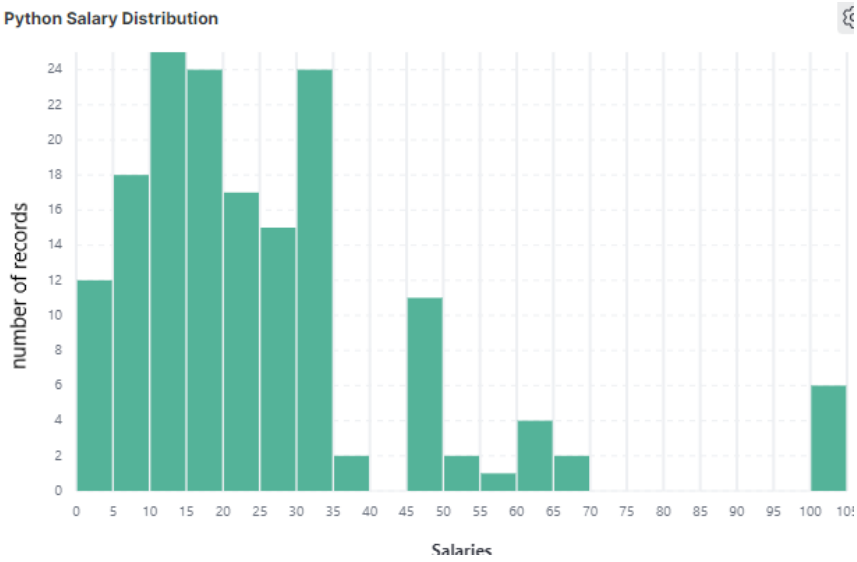
Hình 3.20: Thống kê yêu cầu về các nhóm kiến thức



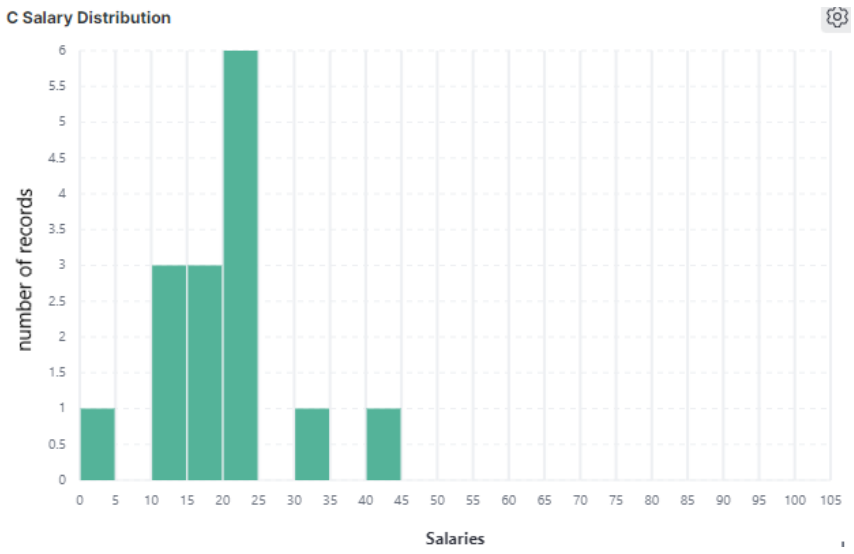
Hình 3.21: Phân phối lương



Hình 3.22: Phân phối lương của các công ty tuyển dụng lập trình viên Java



Hình 3.23: Phân phối lương của các công ty tuyển dụng lập trình viên Python

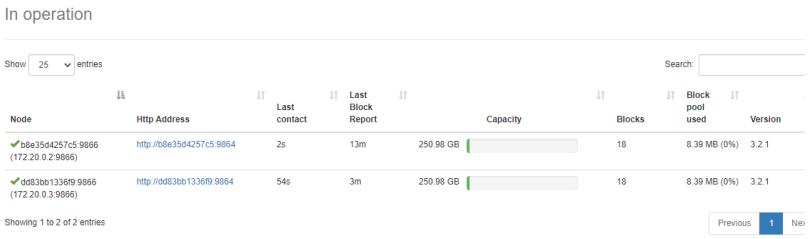


Hình 3.24: Phân phối lương của các công ty tuyển dụng lập trình viên C

3.3 Đánh giá khả năng chịu lỗi của Hadoop

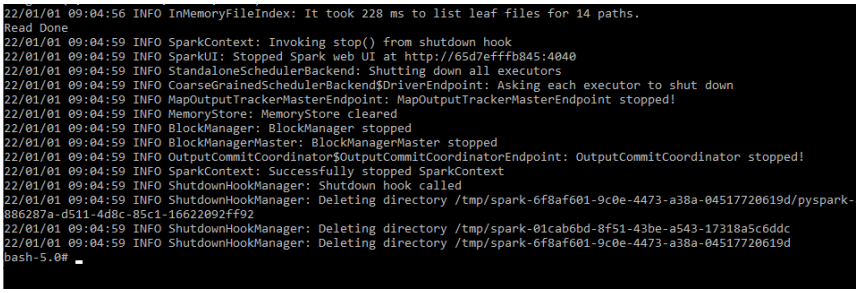
Khi triển khai hệ thống trong thực tế, việc một vài máy bị mất kết nối hoặc bị tắt nên không thể tham gia vào hệ thống là điều không thể tránh khỏi. Tại Hadoop Cluster, dữ liệu được lưu trữ phân tán và lưu thành các bản sao giúp cho hệ thống có khả năng chịu lỗi. Việc mô phỏng lỗi trong thực tế có thể tiến hành bằng cách tắt một container datanode và cho ứng dụng tại Spark đọc dữ liệu để đánh giá khả năng chịu lỗi của Hadoop.

Có thể kiểm tra xem một datanode đã bị tắt chưa bằng cách xem thời gian liên lạc gần nhất giữa namenode và datanode. Theo cài đặt, cứ 3s datanode sẽ phải liên lạc với namenode 1 lần, nếu thời gian liên lạc lớn hơn 3s, chứng tỏ đã có lỗi xảy ra tại datanode.



Hình 3.25: Có lỗi xảy ra tại Datanode 2

Ứng dụng tại Spark sẽ thử đọc lại các files dữ liệu thu thập lưu tại Hadoop, nếu đọc được cả 14 file, chương trình sẽ in ra *Read Done*, chứng tỏ Hadoop đã chịu lỗi thành công. Hình 3.26 cho thấy Hadoop vẫn hoạt động bình thường khi có một datanode gặp lỗi.



Hình 3.26: Hadoop chịu lỗi thành công

4. Kết luận

Hệ thống Recruitement Insight cho thấy những lợi ích mà một hệ thống Big Data đem lại như khả năng lưu trữ, tìm kiếm, biểu diễn lượng lớn dữ liệu, khả năng mở rộng khi lượng tài nguyên hiện tại không đủ, khả năng chịu lỗi trong một mạng phân tán khi có những thành phần trong mạng gặp trục trặc. Đây là những khả năng mà các hệ thống truyền thống không có hoặc khả năng đáp ứng còn hạn chế.

Tuy nhiên, do mới chỉ triển khai giả lập trên máy cá nhân bằng Docker, Recruitment Insight chưa thể chứng minh được khả năng chịu lỗi trong thực tế. Bên cạnh đó, sử dụng *spark-submit* để thực thi cũng làm giảm đi tính linh hoạt để hệ thống. Sử dụng thêm các container như *spark notebook* có thể giúp hệ thống khắc phục vấn đề này.

Mặc dù hệ thống đã được nghiên cứu và thử nghiệm kĩ lương, thiếu sót luôn là điều không tránh khỏi. Vì vậy, nhận xét của thầy và góp ý của bạn đọc sẽ là những ý kiến quý giá mà nhóm sinh viên tham khảo để hoàn thiện hệ thống.

Tài liệu tham khảo

[1] BeutifulSoup Documentation

[2] Spark Documentation

[3] PySpark Documentation

[4] Big Data Europe (`https://github.com/big-data-europe`)