**A) CONTENT**

[1. Jhipster 3](#_heading=h.gjdgxs)

[1.1. Jhipster là gì? 3](#_heading=h.30j0zll)

[1.2. Xây dựng app với Jhipster 3](#_heading=h.1fob9te)

[1.3. Microservice với Jhipster 10](#_heading=h.1fob9te)

[1.3.1. Microservice là gì 10](#_heading=h.3znysh7)

[1.3.2. So sánh với kiến trúc Monolith 11](#_heading=h.2et92p0)

[1.3.3. Cấu trúc Microservice 13](#_heading=h.tyjcwt)

[1.3.4. API Gateway 13](#_heading=h.3dy6vkm)

[1.3.5. Service Discovery với Jhipster Resgitry 16](#_heading=h.1t3h5sf)

[1.3.6. Service Discovery với Consul 16](#_heading=h.4d34og8)

1.3.7. Ưu và nhược điểm của Microservice 16

[2. Elasticsearch 17](#_heading=h.2s8eyo1)

[2.1. Elasticsearch là gì? 17](#_heading=h.17dp8vu)

[2.2. Cấu trúc của elasticsearch 17](#_heading=h.3rdcrjn)

[2.3. Cài đặt elastic stack và ví dụ với elasticsearch 18](#_heading=h.26in1rg)

[2.4. Elasticsearch APIs 23](#_heading=h.lnxbz9)

[2.4.1. Elasticsearch APIs 23](#_heading=h.35nkun2)

[2.4.2. ES Document APIs 23](#_heading=h.1ksv4uv)

[2.4.3. ES Search APIs 27](#_heading=h.2grqrue)

[2.4.4. ES Aggregation APIs 27](#_heading=h.44sinio)

[2.4.5. ES Index APIs 28](#_heading=h.2jxsxqh)

[2.4.6. ES Cluster APIs 29](#_heading=h.2jxsxqh)

[3. Kafka 30](#_heading=h.z337ya)

[3.1. Kafka là gì? 30](#_heading=h.3j2qqm3)

[3.2. Kafka installation -](#_heading=h.1y810tw)

[3.3. Kafka core concept -](#_heading=h.4i7ojhp)

[3.3.1. Kafka topics -](#_heading=h.2xcytpi)

[3.3.2. Topic replication -](#_heading=h.1ci93xb)

[3.3.3. Kafka producer -](#_heading=h.3whwml4)

[3.3.4. Consumer & Comsumer Groups -](#_heading=h.3whwml4)

[3.3.5. Kafka Use Cases -](#_heading=h.3whwml4)

[3.3.6. Kafka Application -](#_heading=h.3whwml4)

[3.3.7. Advantage & Disadvantage -](#_heading=h.3whwml4)

[3.3.8. Kafka Multiple Clusters -](#_heading=h.3whwml4)

[3.3.9. Kafka Architecture -](#_heading=h.3whwml4)

[3.4. Kafka với Spring boot -](#_heading=h.2bn6wsx)

[3.5.1. Kafka Template -](#_heading=h.3as4poj)

[3.5.2. Kafka Message Listener Container -](#_heading=h.1pxezwc)

[3.5.3. Kafka Listener -](#_heading=h.49x2ik5)

[3.5.4. Kafka Transaction Manager -](#_heading=h.49x2ik5)

[3.5.5. Spring Kafka Test -](#_heading=h.49x2ik5)

[4. Redis Cache -](#_heading=h.41mghml)

[5. Nguồn tham khảo -](#_heading=h.41mghml)

**B) STRUCTURE SYSTEM**

**A) CONTENT**

# Jhipster

## Jhipster là gì?

1. Giới thiệu

Jhipster là một trong những dự án Open-source được tạo bởi Julien Dubois vào 10/2013. Và Jhipter được public lần đầu ra công chúng (phiên bản 0.3.1) vào 7/12/2013. Hiện tại jhipster được phát triển bởi 19 lập trình viên và hơn 430 đóng góp viên.

Jhipster là sự kết hợp hoàn hảo giữa các frameworks đã thành công khi lập trình web như: Bootstrap, Angular, Spring boot, React JS.

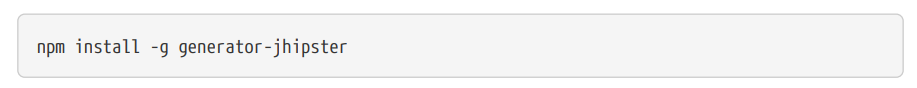
Jhipster là một công cụ tạo tự động core mã nguần giúp loại bỏ thời gian tạo ra các đoạn mã cơ bản từ đó tăng thời gian lập trình và cũng như tập trung vào logic của dự án

1. Cài đặt Genarator Jhipter

B1: Trước hết bạn cần cài đặt java, maven và git trước khi cài đặt Jhipster

B2: Sau đó cài Node.js từ Nodejs.org or Yarn

B3: Run dòng lệnh này trên command line:



## Xây dựng app với Jhipster

1. Genarating project

B1: Bạn cần tạo một folder để chứa dự án của bạn

B2:

Cách 1: Run dòng lệnh này trên command line:



Sau khi bạn chạy xong dòng code kia, thì bạn sẽ được hỏi các câu hỏi tương ứng với những gì bạn muốn cài trên dự án của bạn.



Giải thích các câu hỏi trên:

1: Which “type” of application would you like to create: Là kiểu kiến trúc bạn muốn trong dự án của bạn. Ở ví dụ này là Monolithic application

2: What is the base name of you application: Tên dự án của bạn. Ở ví dụ này là TwentyOnePoints

3: What is your default Java package name: Tên package của dụ án của bạn. Ở ví dụ này là org.jhipster.health

4: Do you want to use the Jhipster Registry to configure, monitor and scale your application: Bạn có muốn sử dụng Jhipter Registry để cài đặt và quản lý dự án của bạn hay không. Ở ví dụ này là No do nó là kiến trúc Monolith

5: Which “type” of authentication would you like to use: Kiểu xác thực mà bạn muốn sử dụng trong dự án là gì. Ở ví dụ này là JWT authentication (stateless, with a token)

6: Which “type” of database would you like to use: Loại Database mà bạn muốn sử dụng. Ở ví dụ này là SQL

7: Which ”production” database would you like to use: Database nào mà bạn muốn sử dụng trong chế độ sản phẩm. Ở ví dụ này là PostgreSQL

8: Which ”development” database would you like to use: Database nào mà bạn muốn sử dụng trong khi phát triển. Ở ví dụ này là H2 with disk-based persistence

9: Do you want to use the Spring cache abstraction: Bạn có muốn sử dụng cache cho dự án này không. Ở ví dụ này là Yes

10: Do you want to use Hiberate 2nd level cache: bạn có muốn sử dụng Hibernate 2nd không. Ở ví dụ này là Yes

11: Would you like to use Maven or Gradle for building the backend: Bạn thích sử dụng Maven hay Gradle cho xây dựng backend. Ở ví dụ này là Gradle

12: Which other technologies would you like to use: Công nghệ khác bạn muốn sử dụng trong dự án. Ở ví dụ này là Search engine using Elasticsearch

13: Which “framework” would you like to use for the client: Framework frontend bạn muốn sử dụng cho dự án. Ở ví dụ này là Anglar 6

14: Would you like to enable “SASS” support using the LibSass stylesheet preprocessor: Bạn có muốn kích hoạt hỗ trợ SASS cho tiền xử lý không. Ở ví dụ này là Yes.

15: Would you like to enable internationalization support: Bạn có muốn kích hoạt hỗ trợ đã ngôn ngữ không. Ở ví dụ này là Yes

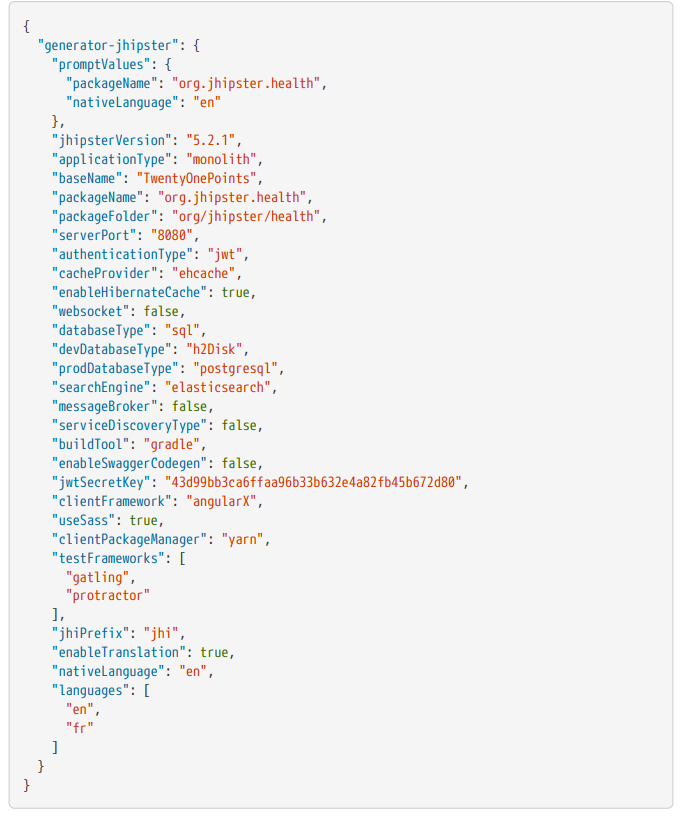
16: Please choose the native language of the application: Chọn ngôn ngữ mặc định cho dự án. Ở ví dụ này là English

17: Please choose additional languages to install: Chọn thêm ngôn ngữ. Ở ví dụ này là French

18: Resides Junit and jest, which testing frameworks would you like to use: Loại test nào bạn muốn sử dụng. Ở ví dụ này là Gatling, Protractor

19: Would you like to install other generators from the Jhipster Marketplace: Bạn có muốn cài đặt các trình tạo khác từ Jhipster Marketplace không. Ở ví dụ này là No

Cách 2: Cách khác để bạn tạo ra project đó là sử dụng file json (.yo-rc.json)



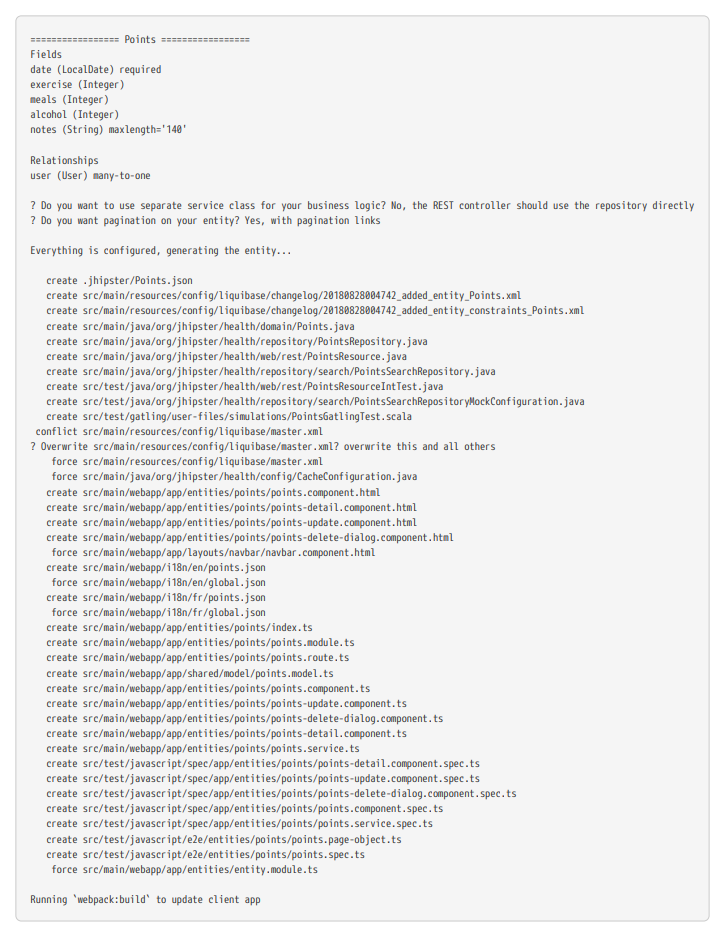
1. Genarating entities

Cho mỗi entity bạn thường muốn tạo ra như sau:

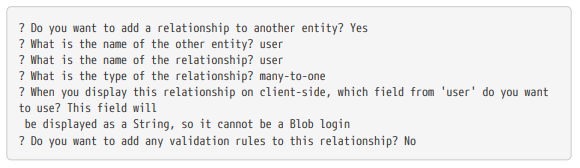
* A database table
* A liquibase change set
* A JPA entity class
* A Spring Data JpaRepository interface
* A Spring MVC RestController class
* An Angular router, controller, and service
* A HTML page

JDL-Studio là browser-based tool cho đĩnh nghĩa model của bạn với Jhipster Domain Language (JDL)

Cách tạo ra Entity: run command line: jhipster entiry [Tên entity]



Cách tạo relationship với bảng khác



## Microservice với Jhipster

### Microservice là gì?

1. Lịch sử của microservice

Theo Wikipedia, Microservice lần đầu tiên được sử dụng như một phong cách kiến trúc phổ biến tại một hội thảo của các kiến trúc sư phần mềm gần Venice vào tháng 2011 năm XNUMX. Vào tháng 2012 năm XNUMX, cùng một nhóm đã quyết định "Microservice" là một cái tên thích hợp hơn

1. Tại sao lại chọn microservice

Cho hầu hết các lập trình viên, nhóm dev và các tổ chức, thật dễ dàng để làm việc với project nhỏ và làm một dịch vụ rất là tốt. Không một service program đại diện cho cả ứng dụng vì vậy service có thể thay đổi frameworks với không mất giá cả cao. Các service khác nhau có thể viết bởi nhiều ngôn ngữ lập trình khác nhau như: Java, Ruby, Node….

Các nhà cung cấp Platform-as-a-service (PaaS) và chứa đã làm cho nó dễ dàng hơn để triển khai microservice . Tất cả các công nghệ cần để hỗ trợ cho kiến trúc monolith (Load balancing, discovery, process monitoring) được cung cấp bởi PaaS, bên ngoài container của bạn và nỗ lực triển khải trở nên dẽ dàng hơn.

1. Tự định nghĩa

Kiến trúc Microservice là kiểu kiến trúc ứng dụng mà ở đó ứng dụng được phát triển bằng một tập các service. Nó cung cấp các framework cho lập trình và triển khai và duy trì, bảo trì dự án. Và các service hoạt động một các độc lập với nhau

1. Nên sử dụng kiến trúc Microservice ở đâu?

Một số ví dụ có thể sử dụng kiến trúc microservice:

* **Website migration:** Một trang web phức tạp được lưu trữ trên nền tảng nguyên khối có thể được di chuyển sang nền tảng microservice dữa trên đám mây và dữa trên container
* **Media content:** Sử dụng kiến trúc Microservice, hình ảnh và tải sản video có thể được lưu trữ trong hệ thống lưu trữ đối tượng có thể mở rộng và được phục vụ trực tiếp cho web hoặc thiết bị di động.
* **Transactions and invoices:** Xử lý và đặt hàng thanh toán có thể được phân tách dưới dạng các đơn vị dịch vụ độc lập để các khoản thanh toán tiếp tục được chấp nhận và hóa đơn không hoạt động.
* **Data processing:** Nền tảng Microservice có thể mở rộng hỗ trợ đám mây cho các dịch vụ xử lý dữ liệu modul hiện có.

### Cấu trúc Microservice

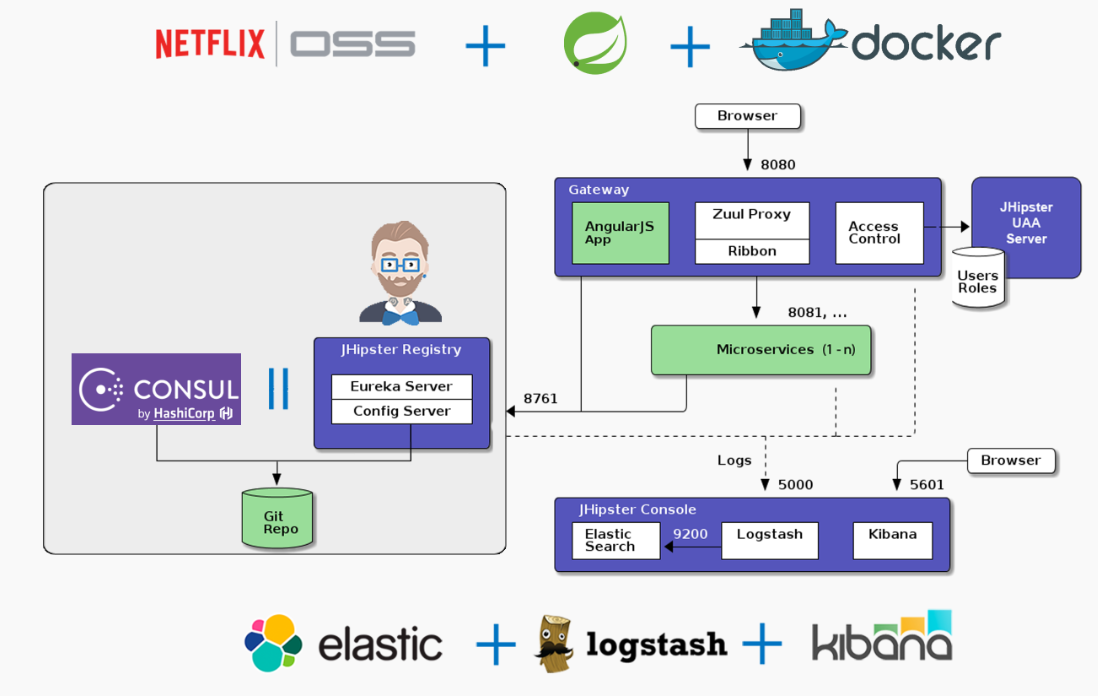
1. Cấu trúc chung



+ Management/orchestration: Thành phần này chịu trách nhiệm đặt các dịch vụ trên các nút, xác định lỗi, các dịch vụ cân bằng lại trên các nút… Thông thường thành phần này là một công nghệ ngoài luồng như Kubernetes, thay vì một cái gì đó được xây dựng tùy chỉnh.

+ API Gateway: API Gateway là điểm vào cho client. Thay vì gọi trực tiếp service, khách hàng gọi qua API Gateway, chuyển tiếp service đó đến các dịch vụ thích hợp ở phía sau.

1. Cấu trúc microservice trong jhipster



Kiến trúc Jhipster Microservice hoạt động gồm các thành phần sau:

+ **A Gateway**: Gateway là một ứng dụng do Jhipster tạo ra (Sử dụng microservice gateway) xử lý lưu lượng web và phục vụ ứng dụng Angular/React. Có thể có một số cộng khác nhau, nếu bạn muốn theo dõi các phụ trợ cho mẫu frontends, nhưng nó không phải bắt buộc. (Xem chi tiết trong phần Jhipster Gateway )

+ **The Jhipster Registry**: là một ứng dụng runtime trạy trên đó và tất cả các ứng dụng cần đăng ký và nhận cấu hình của chúng. Nó cũng cung cấp bảng điều kiển giám giát thời gian chạy

+ **Consul**: Consul là một dịch vụ khám phá dịch vụ, cũng như một của hàng key/value. Nó có thể được sử dụng như một thay thế cho **Jhipster Registry**

**+ Microservices:** Nó được tạo ra bằng Jhipster (microservice application) để xử lý các Rest request.

### So sánh với kiến trúc Monolith

Chart, bubble chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Key** | **Monolithic Architecture** | **Microservice Architecture** |
| Basic | Monolithic Architecture được xây dựng như một hệ thống lớn và thưởng có 1 code base | Mocroservices Architecture được xây dựng lên từ các module nhỏ độc lập với nhau dựa trên business functionality |
| Scale | Nó khó để mở rộng trên các lệnh | Dễ dàng mở rộng và maintain |
| Database | Nó chia sẻ database với nhau | Mỗi project và module có thể có các database của chính nó |
| Deployment | Một lượng lớn code base làm cho IDE và thời gian build tăng lên | Mỗi project hoạt động độc lập với nhau và với một lượng nhỏ code nên tổng thời gian build và deployment giảm đáng kể |
| Tightly coupled and loosely coupled | Nó cực kì khó để thay đổi công nghệ hoặc ngôn ngữ khác bởi vì nó có quan hệ chặt chẽ với nhau và phụ thuộc lẫn nhau | Dễ dàng thay đổi về công nghệ hoặc framework bời vì mỗi module và project hoạt động độc lập với nhau |

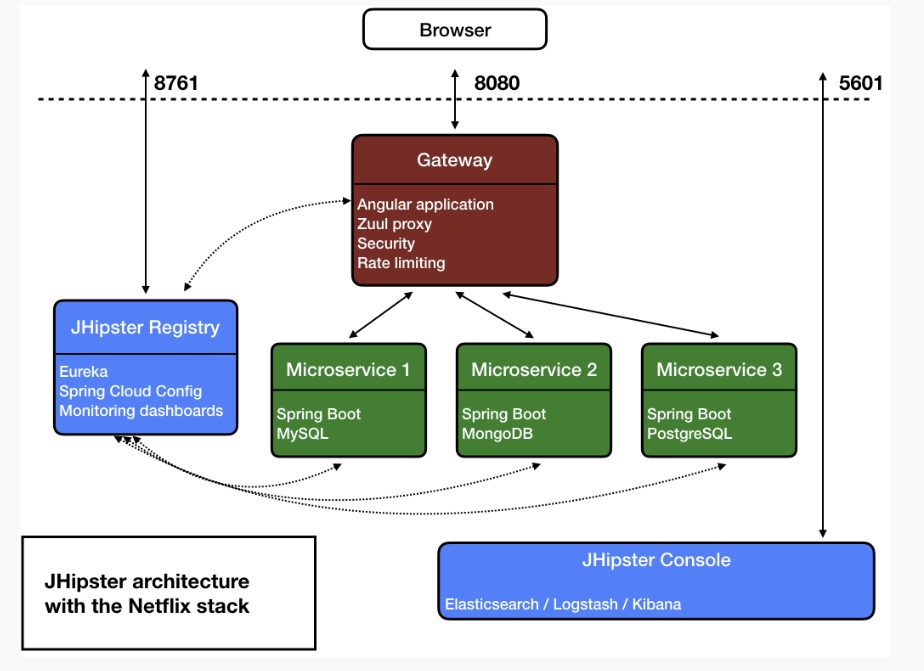
### API Gateway

1. Giới thiệu:

Jhipster có thể tạo ra API gateway. A gateway là một ứng dụng Jhipster thông thường, vì vậy bạn có thể sử dụng các lựa chọn của jhipster và phát triển workflows cho dự án nhưng nó cũng hoạt động như một cánh cổng để các request có thể đến được microservice của bạn.

Đặc biệt hơn, Gateway cung cấp HTTP routing và load balancing, bảo mật, tài liệu API ... cho tất cả microservice.

1. Architecture diagram



1. HTTP routing

Khi gateway và các ứng dụng microservice của bạn chạy, Nó sẽ đăng kí lên Registry (thông qua config eureka.client.serviceUrl.defaultZone) trong file application.yml or application.properties

Gateway sẽ tự động xử lý tất cả các request để chuyển đến các microservice tương ứng sử dụng tên của ứng dụng microservice của bạn.

Ví dụ:

+ Tên ứng dụng microservice là: app1

+ Gateway chạy ở port 8080

+ Truy nhập apit của app1: /api/foos

* Url: http://localhost:8080/services/app1/api/foos

1. Automatic documentation

Gateway sẽ hiển thị các định nghĩa API Swagger về các service mà nó hướng tới để bạn có tất cả các công cụ hưu ích để sử dụng

1. Rate limiting

Rate limiting là một chức năng nâng cao nó sử dụng Bucket4j và Hazelcast để cung cấp số lượng các serice trong microservice

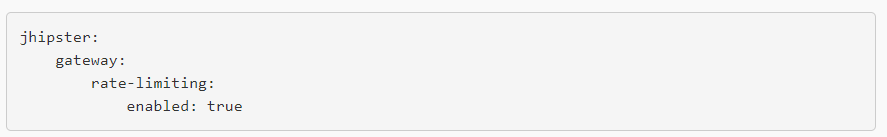
Gateway cung cấp chức năng rate-limiting, vì vậy Rest request có thể giới hạn:

* + - * Bởi IP address (cho user ẩn danh)
      * Bởi User login (cho user đã login)

Jhipster sẽ sử dụng Bucket4j và Hazelcast để tính toán số lượng các request và sẽ gửi HTTP 429 (QUÁ NHIỀU REQUEST) lỗi khi vượt quá giới hạn.

Mặc định giới hạn cho mỗi user là 100000 API calls cho mỗi giờ.

Kích hoạt nó bằng cách config trong file application.yml or application.properties



1. Access control policy

Mặc định tất cả các đăng kí bởi microservice sẽ tồn tại và có sẵn trong Gateway. Nhưng nếu bạn muốn chỉ một số API có thể truy cập bởi API Gateway thì bạn có thể config ở file file application.yml or application.properties



### Service Discovery với Jhipster Registry

* Jhipster Registry là một runtime application được cung cấp với nhóm của Jhipster.
* Nó là một mã nguần mở, Apache 2-licensed application và source code của nó có sẵn trên github: <https://github.com/jhipster/jhipster-registry>
* Jhipster Registry có 3 chức năng chính:
  + Eureka server: cung cấp như một máy chủ khám phá cho các ứng dụng. Máy chủ này duy trì và phân phối một danh sách động các trường hợp ứng dụng có sẵn sau đó được sử dụng bởi các dịch vụ microservice để thực hiện các yêu cầu HTTP định tuyến và cân bằng tải
  + Spring cloud config server: Nó cung cấp runtime config cho tất cả các ứng dụng
  + Đây là một máy chủ quản trị, với bẳng điều khiển để giám sát và quản lý các ứng dụng

### Service Discovery với Consul

### Ưu và nhược điểm của Microservice

1. Ưu điểm

* Microservice giúp dễ dàng continuos delivery và deployment các ứng dụng lớn phức tạp
  + Cải thiện về khả năng bảo trì, mỗi service tương đối nhỏ vì vậy dễ hiểu và thay đổi hơn
  + Khả năng testing dễ dàng hơn, vì những services nhỏ hơn và nhanh hơn để test
  + Khả năng triển khai tốt hơn, các service có thể được triển khai độc lập với nhau
* Giảm thiểu được rủi ro: Ví dụ như có lỗi trong một service thì chỉ có service đó bị ảnh hưởng. Các services khác sẽ vẫn tiếp tục xử lý các request khác
* Dễ dàng thay đổi để sử dụng các công nghệ mới, Khi thực hiện triển khai các services bạn có thể lựa chọn nhiều công nghệ mới phù hợp hơn.
* Agility (Nhanh nhẹn) Microservice hỗ trợ phát triển theo mô hình Agile

1. Nhược điểm

* Các nhà phát triển phần mềm phải đối mặt với sự phức tạp của việc tạo ra một hệ thống phân tán:
  + Cần phải implement việc communication giữa các inter-services
  + Việc Handle parial failture là rất phức tạp vì một luồng xử lý cần đi qua nhiều service
  + Việc thực hiện những requests ở nhiều services khó khăn hơn, nó cũng đòi hỏi sự phối hợp cẩn thận giữa các teams
  + Những khó khăn trong việc đảm bảo toàn vẹn CSDL nếu triển khai theo kiến trúc cơ sở dữ liệu phân vùng
* Việc triển khai và quản lý microservice nếu làm theo các thủ công so với ứng dụng một khối phức tạp hơn nhiều
* Phải xử lý sự cố khi có kết nối chậm

# Elasticsearch

## Elasticsearch là gì?

Elasticsearch là một công cụ phân tích và tìm kiếm dựa trên engine **Apache Lucene,** được viết bằng ngôn ngữ Java và cũng là kho dữ liệu không theo cấu trúc như SQL.

An index, search engine, analytics database, big data solution, document-oriented database, …

Elasticsearch là cơ sở dữ liệu NoSQL (Not-only SQL).

Mục đích của elasticsearch là để làm giảm thời gian tìm kiếm dữ liệu

## Cấu trúc của elasticsearch

Dưới đây là liệt kê về cấu trúc và cách tổ chức dữ liệu của elasticsearch:

* **Index** = cơ sở dữ liệu (a logical namespace that points to one or more **shards**)
* Mỗi Index có nhiều **Type** (giống table trong SQL dành cho phiên bản trước elasticsearch 6.0)

“**Thực tế 1 index chỉ chứa 1 type**”

Thông tin chi tiết: https://www.elastic.co/blog/removal-of-mapping-types-elasticsearch

* Mỗi 1 Index có 1 hoặc nhiều **Shard**
* Các Shard tồn tại trên các **Node** khác nhau = phần quan trọng giúp thực hiện công việc một cách nhanh chóng
* Shard có: **Primary shard và Replica shard** tồn tại trên các Node khác nhau. Số lượng “Primary shard” cho 1 index không thể thay đổi sau khi index được tạo, chính vì vậy nên cần cân nhắc kĩ trước khi tạo index.
* Elasticsearch hoạt động dựa trên **Lucene của Apache**
* 1 shard thực chất là một **Lucene index**, là nơi lưu trữ dữ liệu và là **search engine** (mỗi một Shard là một search engine riêng)
* Vì là Lucene index => Shardtạo từ nhiều **Segment** và mỗi segment là **Inverted Index** với đầy đủ chức năng
* **Mỗi segment hoạt động độc lập => yêu cầu sử dụng CPU và bộ nhớ**
* Nhiều **Node** hợp lại tạo thành **Cluster**
* Có 3 loại **Node**: **Master, Data** và **Client**.
  + **Master:** Cluster sẽ tự động chọn ra một Node làm Master; có nhiệm vụ phân phối shard, tạo/xóa các index và cập nhật trạng thái cluster
  + **Data:** Được phân bố cho các Shard bởi Master; có nhiệm vụ index và tìm kiếm dữ liệu
  + **Client:** = Load Balancer
* Trong mỗi Node sẽ thường có 1 **Primary shard** riêng và nhiều **Replica shard** sẽ được tạo ra từ bản sao của **Primary shard** giúp tăng tốc độ đọc và đề phòng shard chính dừng hoạt động hoặc gặp trục trặc vấn đề gì đấy.

Những từ khóa cần lưu ý: Index, Type, Shard, Node, Primary shard và Replica shard, Lucence của Apache, Lucene index, Segment và Inverted Index, Node, Cluster

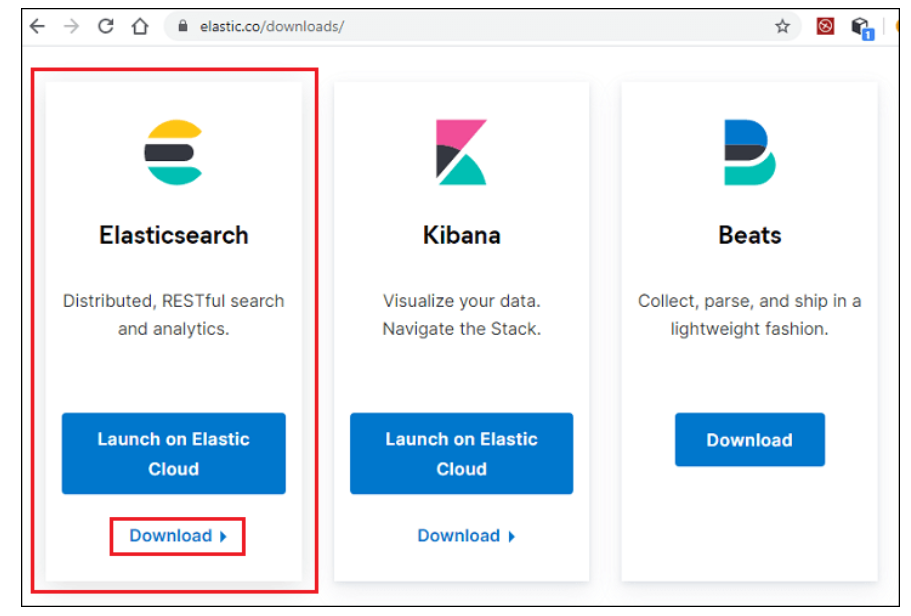
## Cài đặt elastic stack

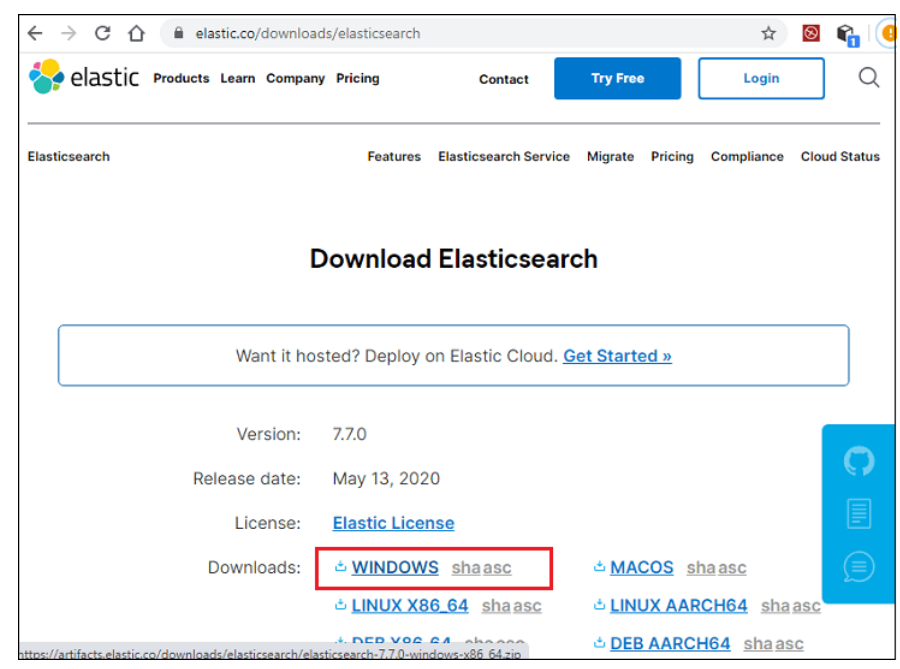
B1: Bạn cần cài java trước khi cài đặt Elastic search

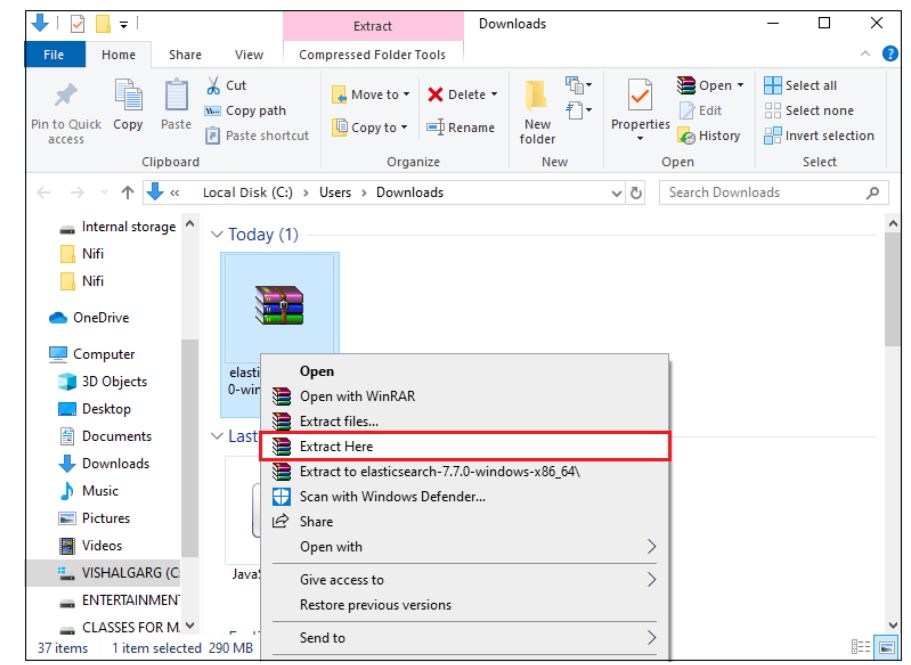
B2: Check java version của bạn phải lớn hơn hoặc bằng 11

B3: Vào trang chủ của elastic search để tải về:

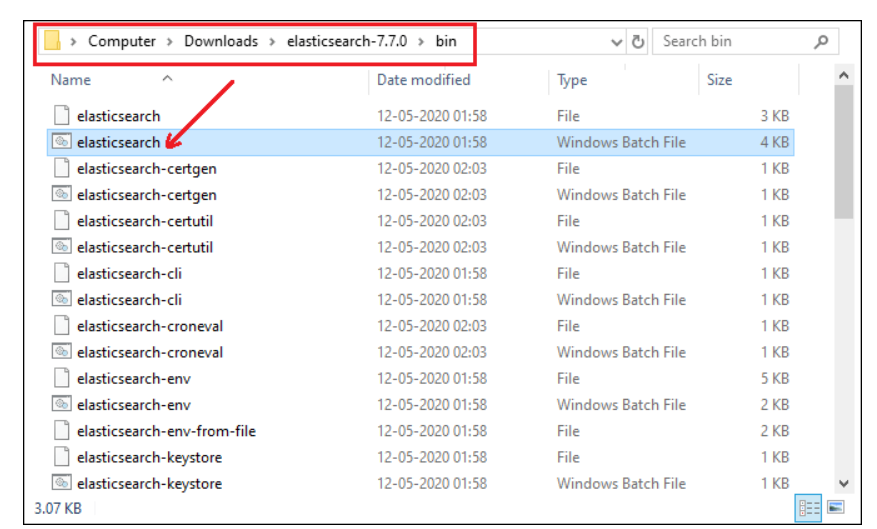
<https://www.elastic.co/downloads/>



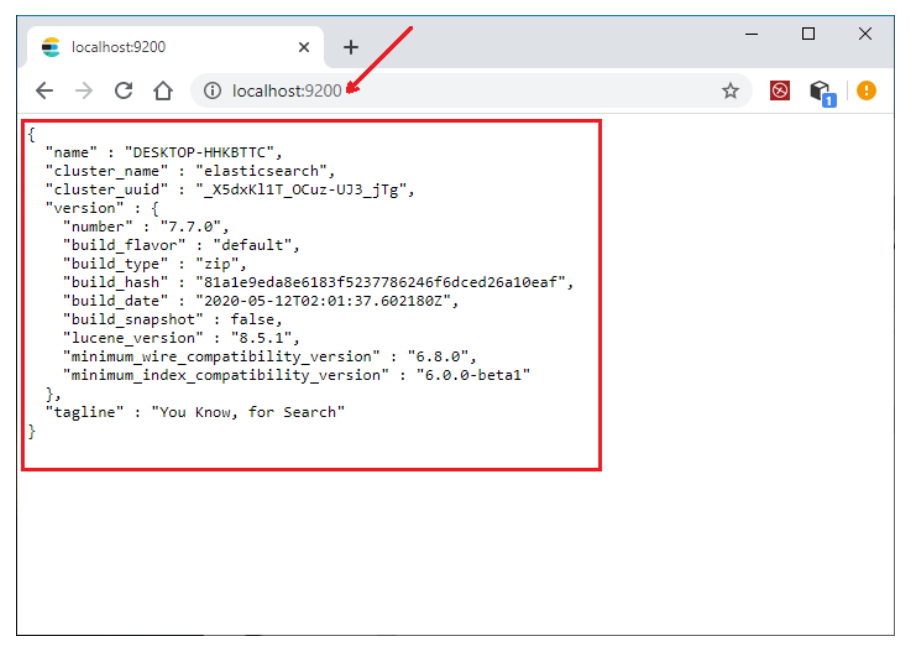




B5: Vào file bin của folder và run **elastic.bat** file



B6: Mở trình duyệt và gõ: <http://localhost:9200>



## Elasticsearch APIs

### Elasticsearch APIs

Elasticsearch APIs có 5 kiểu APIs:

+ Document APIs

+ Search APIs

+ Aggregation

+ Index APIs

+ Cluster APIs

### ES Document APIs

ES Document APIs lưu dữ liệu theo document form

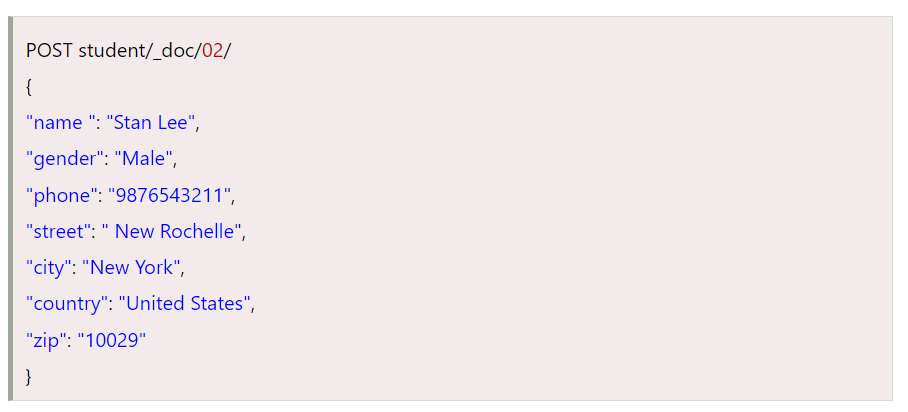
Có 2 loại ES Document APIs

* + Single Document API
  + Multi-Document API

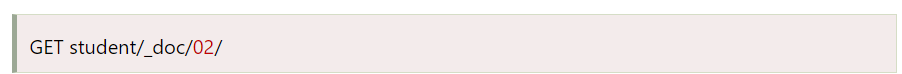
1. Single Documant API

Các API này rất hữu ích nếu bạn muốn thực hiện các hoạt động trên một tài liệu thay vì một nhóm tài liệu. Vì vậy bạn có thể sử dụng Single Document API này khi bạn muốn thực hiện các hoạt động trên một tài liệu . Được chia là 4 cái như sau:

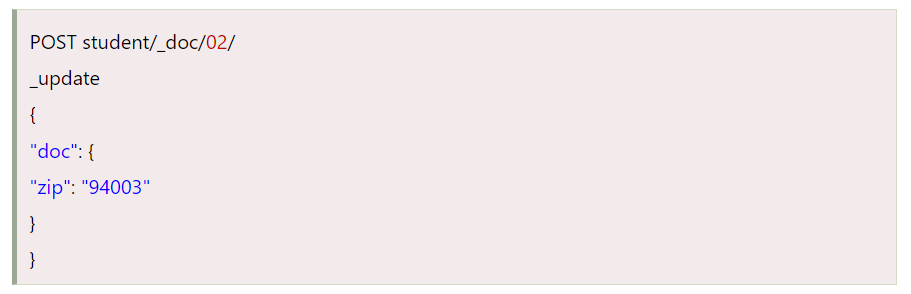
* + Index API
    - This API giúp add một document vào một index cụ thể khi request được tạo
    - Nó cho phép bạn update tài liệu nếu tài liệu đã tồn tại
    - Nó sử dụng thay thế toàn bộ tài liệu đã tồn tại
    - Sử dụng Post method để thêm dữ liệu và chúng ta phải nếu rõ id của document



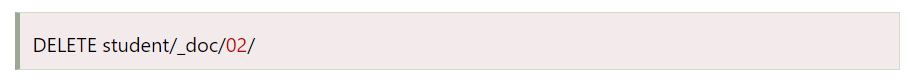
* + Get API
    - Get API giúp để nhận hoặc đọc dữ liệu được lưu trong Index
    - API này có nhiệm vụ nhận tài liệu JSON từ index



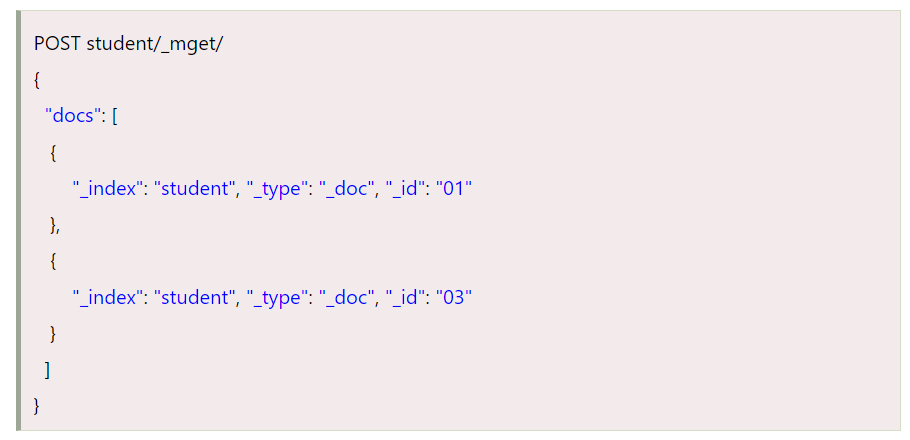
* + Update API



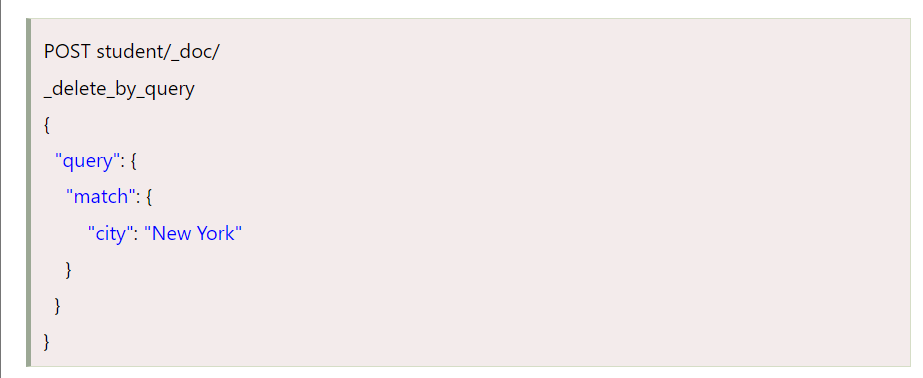
* + DeleeteAPI



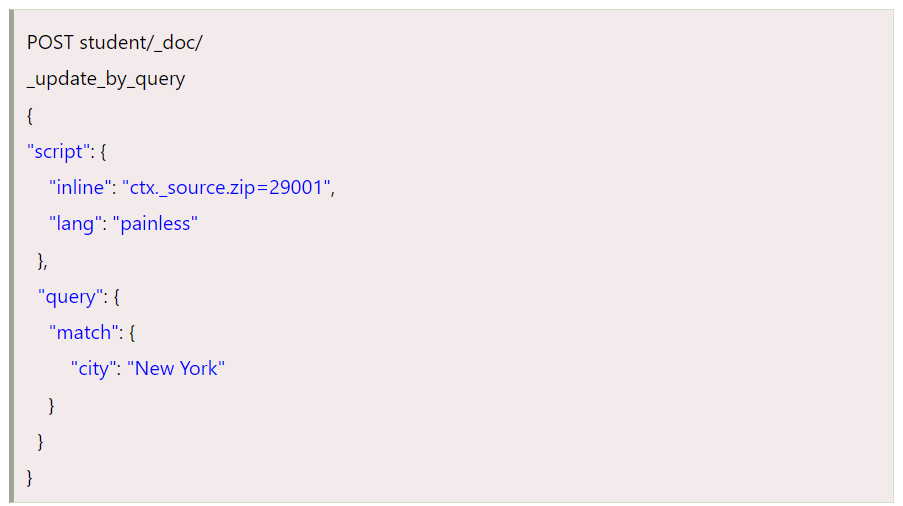
1. Multi-Document API
   * Không giống Single Document API bạn có thể sử dụng Multi-Document cho query giữa nhiều documents
   * Có 5 loại API:
     + Multi Get API



* + - Bulk API
      * Add \_bulk keyword to call API
      * Nó được sử dụng để delete/update/index trong bulk
    - Delete by query API
      * Dùng để delete tất cả các match document



* + - Update By Query API

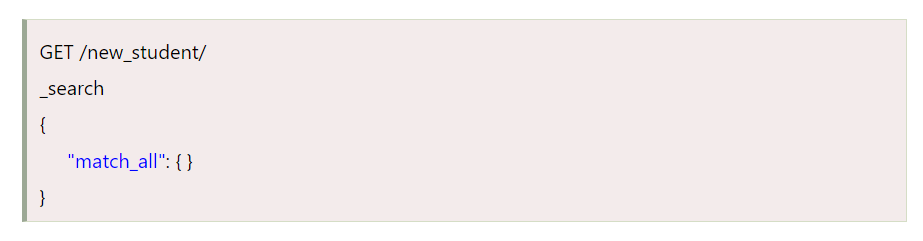


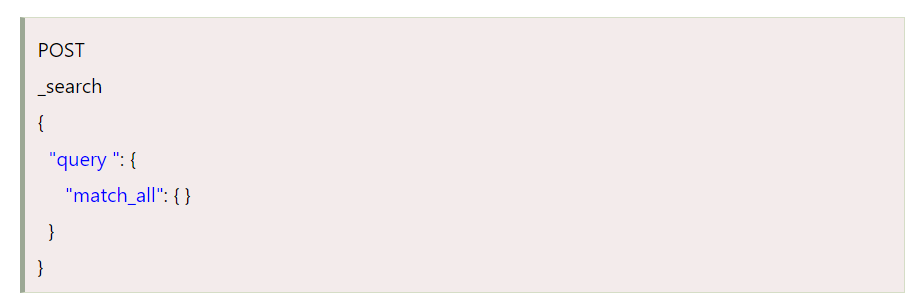
* + - Reindex API



### ES Search APIs

* + Được sử dụng để search dữ liệu trên index và tất cả các types





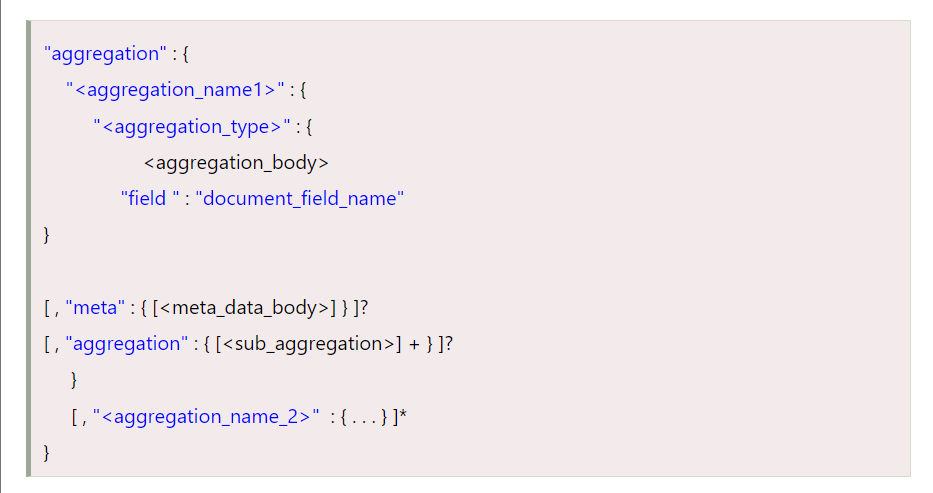
### ES Aggregation APIs

Elasticsearch cung cập API tổng hợp, được sử dụng để tổng hợp dữ liệu. Khung tổng hợp cung cấp dữ liệu tổng hợp dữ trên truy vấn tìm kiếm. Nói một cách đơn giản, Khung tổng hợp thu thập tất cả các dữ liệu được truy vấn tìm kiếm và cung cấp cho người dùng. Nó chứa một số khối xây dựng giúp xây dựng một bản tóm tắt dữ liệu phức tạp. Tập hợp tạo ra thông tin phân tích có sẵn trong Elastic search

Một số điểm quan trọng của Aggregation APIs:

* + - Aggregation APIs có thể được cấu hình thành cùng nhau để xây dựng các bản tóm tắt dữ liệu phức tạp.
    - Nó có thể được coi là một unit-of-work. Tạo ra thông tin phân tích trên một bộ tài liệu có sẵn trong Elasticsearch
    - Về cơ bản nó dựa trên các khối xây dựng

Syntax:

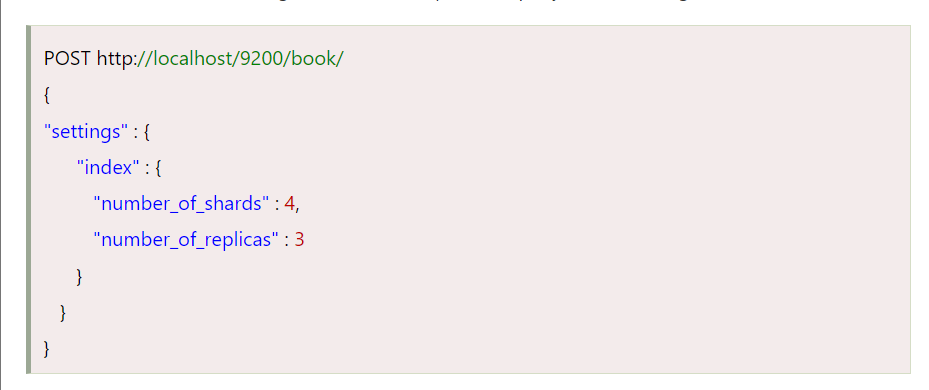


Chi tiết có thể lên trang: h<ttps://www.javatpoint.com/elasticsearch-a>ggregation-apis

### ES Index APIs

ES Index APIs để quản lí tất cả các khía cạnh của một index ví dụ như index template, mapping, aliases, settings…

Một số ví dụ:



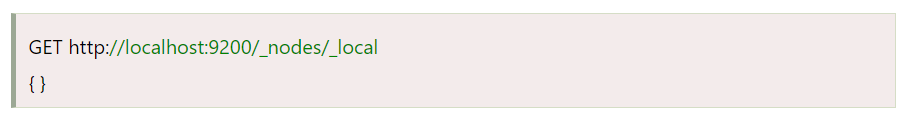


Chi tiết lên trang: https://www.javatpoint.com/elasticsearch-aggregation-apis

### ES Cluster APIs

Trong Elasticsearch, Cluster API lấy thông tin về một cụm và nút của nó. Nó cũng thực hiện những thay đổi hơn nữa trong chúng. Với sự trợ giúp của Cluster API , ta có thể thực hiện 21 hoạt động ở cập Cluster. Chúng ta có thể sử dụng Cluster API để quản lý các cluster của chúng tôi.

Chúng ta cần add thêm \_local keword vào trong GET method



Chi tiết Tham khảo ở: https://www.javatpoint.com/elasticsearch-cluster-apis

# Kafka

## Kafka là gì

## Kafka installation

## Kafka core concept

### Kafka topics

### Topic replication

### Kafka producer

### Consumer & Comsumer Groups

### Kafka Use Cases

### Kafka Application

### Advantage & Disadvantage

### Kafka Multiple Clusters

### Kafka Architecture

## Kafka với Spring boot

### Kafka Template

### Kafka Message Listener Container

### Kafka Listener

### Kafka Transaction Manager

### Spring Kafka Test

# Redis Cache

# Nguồn tham khảo

Tài liệu hướng dẫn elasticsearch ở trang chủ chính:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

Cấu trúc và cách tổ chức dữ liệu của elasticsearch

<https://viblo.asia/p/elasticsearch-cach-to-chuc-du-lieu-OeVKB3oyZkW#_mo-hinh-luu-tru-4>

Giao tiếp với elasticsearch qua power-shell

<https://www.thecodebuzz.com/invoke-webrequest-vs-invoke-restmethod-differences/>

Configuring SAML single-sign-on on the Elastic Stack

<https://www.elastic.co/guide/en/elasticsearch/reference/current/saml-guide-stack.html>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/security-api-saml-logout.html>

too many indexes is not good for ES and that it is better to have as least indexes as possible

Object field type

<https://www.elastic.co/guide/en/elasticsearch/reference/current/object.html>

if the elastic user does not already have a password, it uses a default bootstrap password

<https://discuss.elastic.co/t/detecting-bottleneck-in-elasticsearch-indexing/242060/3>

<https://lukemerrett.com/getting-started-with-elasticsearch/>

**B) STRUCTURE SYSTEM**