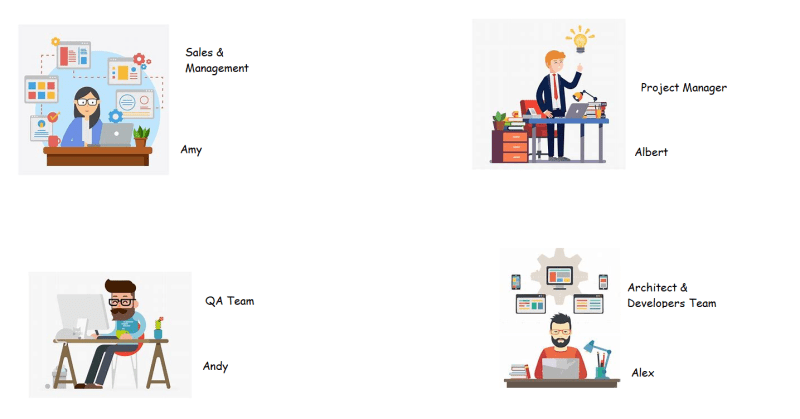
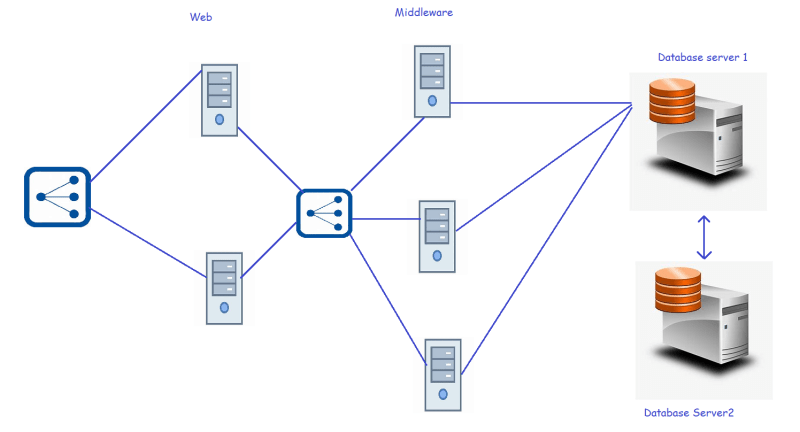
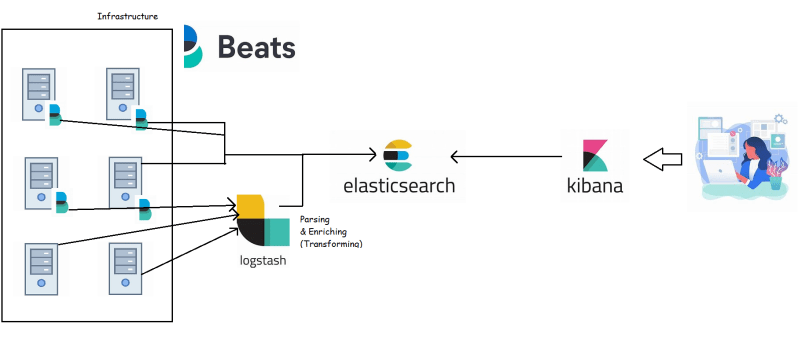
## Class-1: 14/Sept/2021

Story of an Organization

* Learning Thoughts (a fictitious company) is having an application lt-hrms which is a human resource management system and is used by multiple organizations
* For this Learning Thoughts is hosting the application lt-hrms and they have the following team
* 
* Customers of LT are
* 
* The architecture of the application is as follows
* 
* Learning Thoughts updates a new release in every two weeks
* It is often observed that the following issues occur randomly
  + functionality stops working
  + Some server disks get filled up and there will no space left so servers do not respond
  + In Some server CPU utilization is above 95% all the time and users experience slow/unresponsive behaviour
  + And many more…
* During Peak times, some users are facing request timed out errors etc.
* Hardware/Network failures happen randomly.
* Now We need to find a solution
  + to stop as many failures as possible from occurring
  + In the case of failures to resolve as early as possible
* We need to have monitoring in place to
  + monitor systems (Whether they are up or not)
  + monitoring health of your application
  + Monitor system resources
    - CPU
    - Memory
    - Storage
    - Network
* Applications generally create logs which do not have any standard approach. Reading text is tricky and creating meaningful information from text is quite difficult, so in majority of the case, we would use humans to find issues by going through logs
* So learning thoughts have decided that they would use a monitoring system which can not only read metrics but also parse log files and also helps in finding error patterns in logs.
* We would try to understand on resolving/identifying/trouble shooting failures with Elastic Stack (Which can do monitoring, APM, Log parsing, Alerting…)

# Class-2: 15/Sept/2021

Elastic Stack – Introduction

* Elastic Stack is a rich ecosystem of components serving as full search and analytics stack.
* Elastic Stack is **a group of open source products from Elastic** designed to help users take data from any type of source and in any format and search, analyse, and visualize that data in real time.
* Elastic Stack can be deployed on premises or made available as Software as a Service (SaaS).
* The main components of Elastic Stack are
  + Elastic Search
  + Logstash
  + Kibana
  + Beats
  + X-Pack
  + Elastic cloud
* **Elasticsearch** is at the heart of the Elastic Stack providing storage, search and analytics capabilities.
* Elasticsearch is a [RESTful](https://searchapparchitecture.techtarget.com/definition/REST-REpresentational-State-Transfer) distributed [search engine](https://whatis.techtarget.com/definition/search-engine) built on top of [Apache Lucene](https://whatis.techtarget.com/definition/Apache-Lucene) and released under an [Apache](https://whatis.techtarget.com/definition/Apache) license. It is [Java](https://www.theserverside.com/definition/Java)-based and can search and index document files in diverse formats.
* **Kibana** is referred as user interface for Elastic Stack with great visualization capabilities
* Kibana is an open source [data visualization](https://searchbusinessanalytics.techtarget.com/definition/data-visualization) and exploration tool from that is specialized for large volumes of streaming and real-time data. The software makes huge and complex data streams more easily and quickly understandable through graphic representation.
* **Logstash** and **Beats** help the data into Elastic stack
* Logstash is a data collection engine that unifies data from disparate sources, [normalizes](https://searchsqlserver.techtarget.com/definition/normalization) it and distributes it. The product was originally optimized for log data but has expanded the scope to take data from all sources.
* Beats are “data shippers” that are installed on servers as agents used to send different types of operational data to Elasticsearch either directly or through Logstash, where the data might be enhanced or archived.
* **X-Pack** provides features including altering, security, graph & machine learning to make Elastic Stack production ready
* 

Elastic Search:

* Elastic Search intro by official documentation is as follows
  + - "Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stack.
    - Elasticsearch is where the indexing, search, and analysis magic happens.
    - Elasticsearch provides near real-time search and analytics for all types of data. Whether you have structured or unstructured text, numerical data, or geospatial data, Elasticsearch can efficiently store and index it in a way that supports fast searches. You can go far beyond simple data retrieval and aggregate information to discover trends and patterns in your data. And as your data and query volume grows, the distributed nature of Elasticsearch enables your deployment to grow seamlessly right along with it."
* Elastic stack is built on the radically different technology ‘Apache Lucene’
* Key Benefits of Elastic Search
  + Schema less, document-oriented
  + Searching
  + Analytics
  + Rich client library support and the REST API
  + Easy to operate and Easy to Scale
  + Near real time
  + Lightning-fast
  + Fault-tolerant
* Exercise: [Refer Here](https://www.youtube.com/watch?v=ggOmHlnhPaM&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=15) to this video to understand JSON and YAML (<https://www.youtube.com/watch?v=ggOmHlnhPaM&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=16>)

Schema less and document oriented

* Elasticsearch does not impose a strict structure on your data; you can store any json documents.
* These JSON documents are first-class citizents of Elastic search as opposed to rows and columns in a relational database

========================================

Class-3:16/Sept/2021 Schema less and document-orientiated

* Elastic search does not impose a strict structure on your data, you can store any json documents.
* JSON documents are first class citizens of Elastic search (as opposed to rows and columns in the Relational database).
* A document in Elastic Search is roughly equivalent to record in relational database table
* Traditional databases require a schema to defined beforehand
* JSON documents naturally support dynamic data

//first record

{

"name": "John Doe",

"email": "johndoe@gmail.com",

"address": {

"city": "New York"

}

}

//second record

{

"email": "jane\_doe@gmail.com",

"age": 45,

"address": {

"country": "USA",

"zipcode": "10010"

}

}

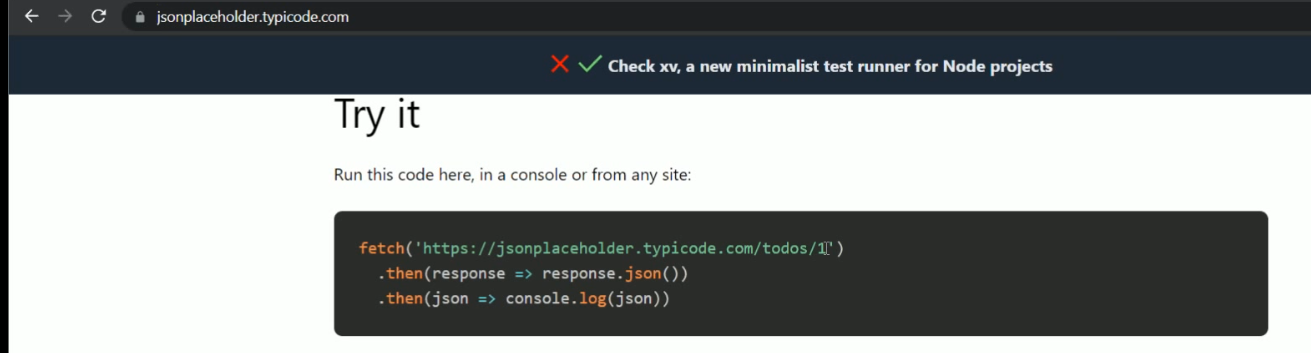
## Searching capabilities

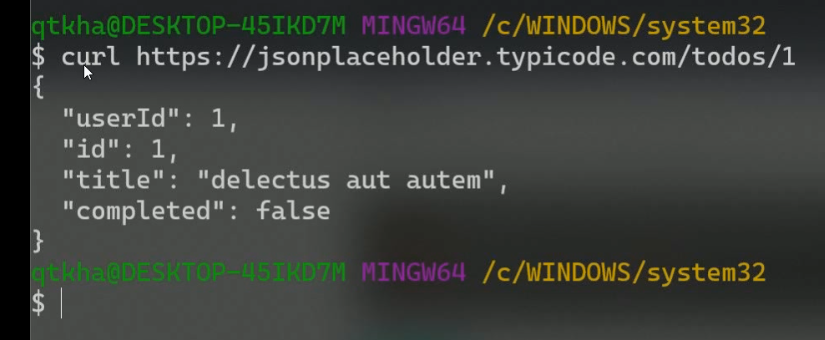
* The core strength of Elastic Search lies in its text-processing capabilities.
* Elastic search is great at searching especially full-text searches
* Full text search means searching through all the terms of the documents available in database, this requires entire content of all documents to be parsed and stored beforehand.
* When you want to perform a search similar to Google search on your own data, Elastic search is your best bet, you can index emails, text documents, pdf files, web pages
* At a high level, Elastic search breaks up text data into terms and makes every term searchable by building lucene indexes

## Analytics

* Elastic search supports wide variety of aggregations for analytics (search in Google like kibana heat map)

## Rich Client Library Support and REST API

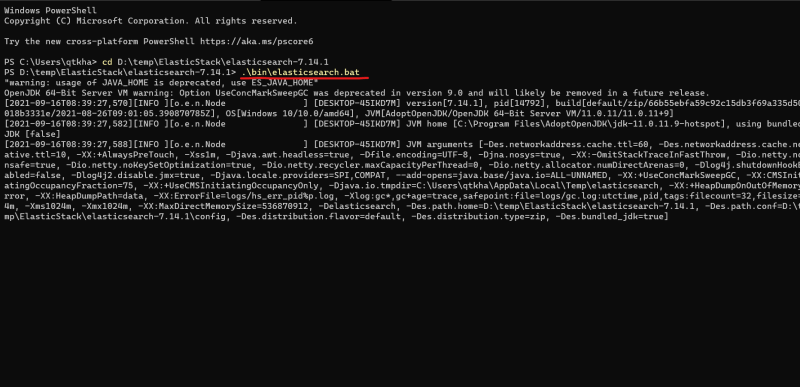
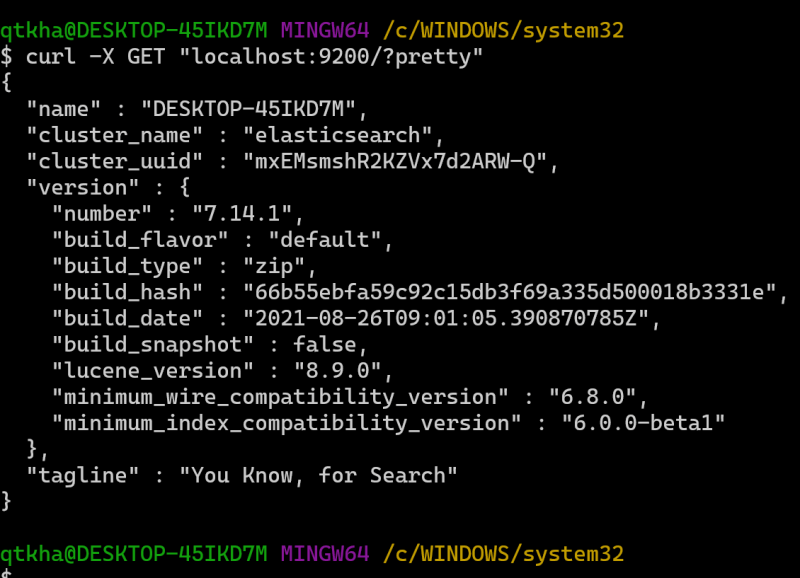
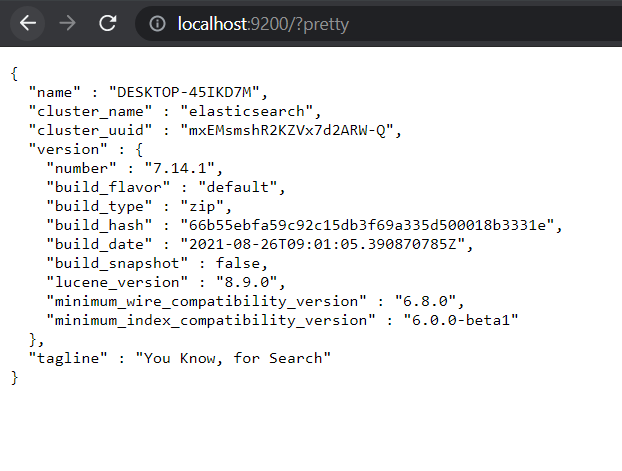
* Elastic search has a very rich client library support to access from languages like JAVA, C#, python, JavaScript, Ruby & more
* Elastic search has a very rich REST API (Representational state transfer) which works on HTTP protocol.
* Search a fake rest api and copy link and past in terminal
* 



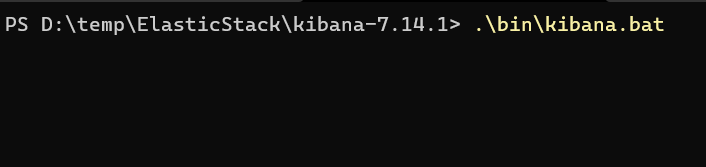
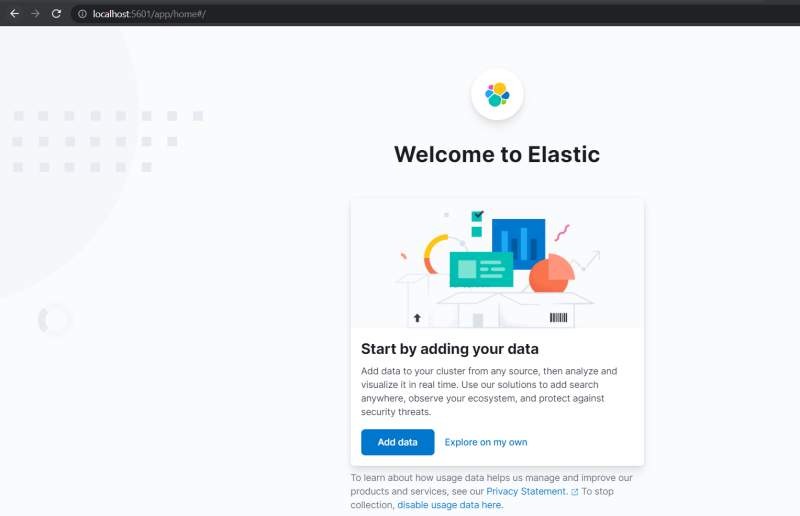
## Easy to Operate and easy to Scale

* Elastic search can run on single node and easily scale out to hundreds of nodes
* Unlike most traditional database that only allow vertical scaling, Elastic search can be scaled horizontally
* The client application doesn’t need to change whether it is running against a single node or a hundred node cluster
* Elastic search clusters can keep running even when there are hardware failures such as node failure, network failure etc. i.e. Elastic search is designed to scale and is fault-tolerant

## Installing Elastic Search

* [Refer Here](https://www.elastic.co/downloads/elasticsearch) to download elastic search (<https://www.elastic.co/downloads/elasticsearch>)
* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html#install-elasticsearch) for the installation of elastic search (<https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html#install-elasticsearch>)
* For understanding features of elastic search let’s try to run elastic search locally
* [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/zip-windows.html) (<https://www.elastic.co/guide/en/elasticsearch/reference/current/zip-windows.html>)
* Unzip the downloaded file and do the following
* 
* Run the command curl [http://localhost:9200?pretty](http://localhost:9200/?pretty) from git bash
* 
* Open browser and navigate to [http://localhost:9200?pretty](http://localhost:9200/?pretty)
* 

## Install Kibana

* Download kibana zip from [Refer Here](https://www.elastic.co/cn/downloads/kibana) (<https://www.elastic.co/cn/downloads/kibana>)
* Extract the file and RUN bin\kibana.bat
* 
* Now open [http://localhost:5601](http://localhost:5601/) in your browser
* 

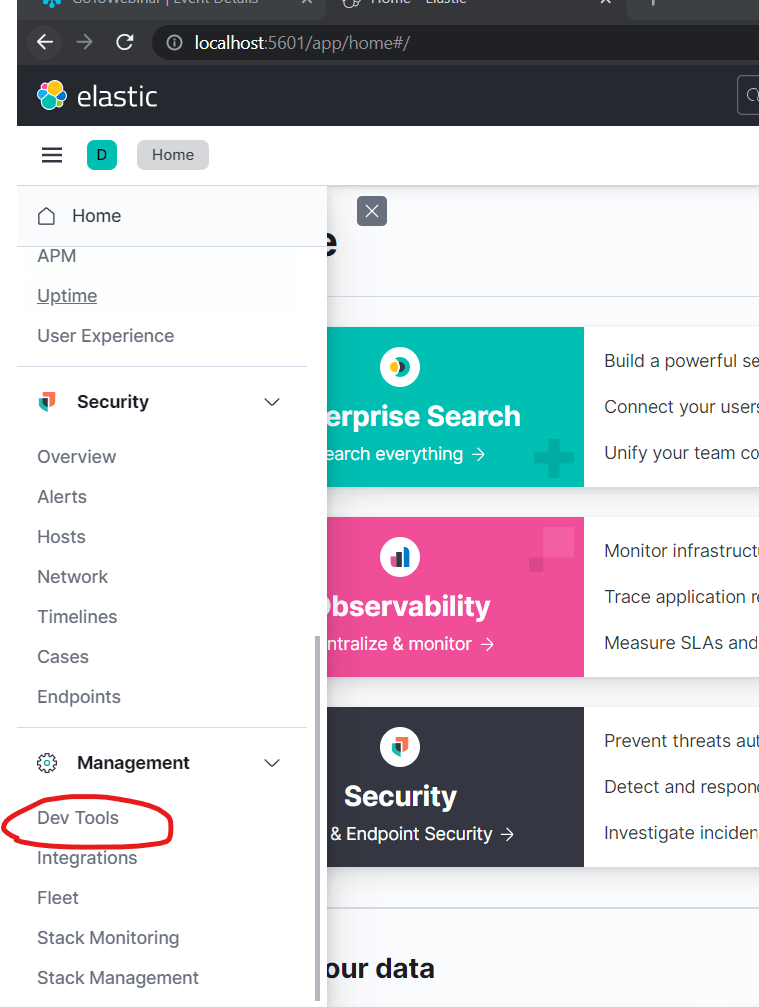
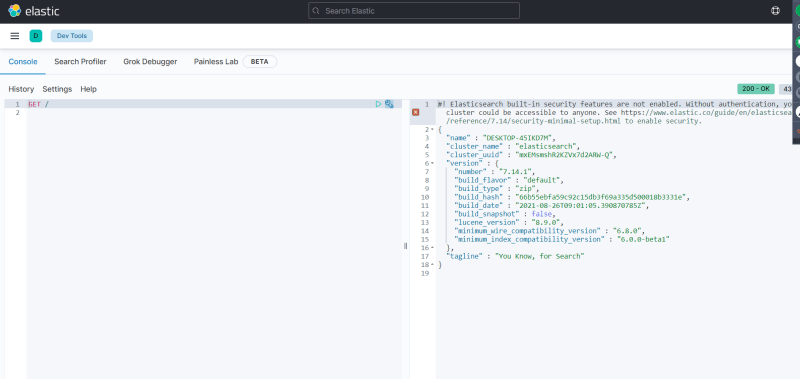
## Lab Setup

* Installing necessary softwares on Windows [Refer Here](https://www.youtube.com/watch?v=mRILfUNbsIo&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=14) (<https://www.youtube.com/watch?v=mRILfUNbsIo&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=15>)
* Install Windows Terminal [Refer Here](https://www.youtube.com/watch?v=qLVn2EvPsYc&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=11) (<https://www.youtube.com/watch?v=qLVn2EvPsYc&list=PLuVH8Jaq3mLud3sVDvJ-gJ__0zd15wGDd&index=12>)

========================================

# Class-4: 17/Sept/2021

## Using Kibana Console

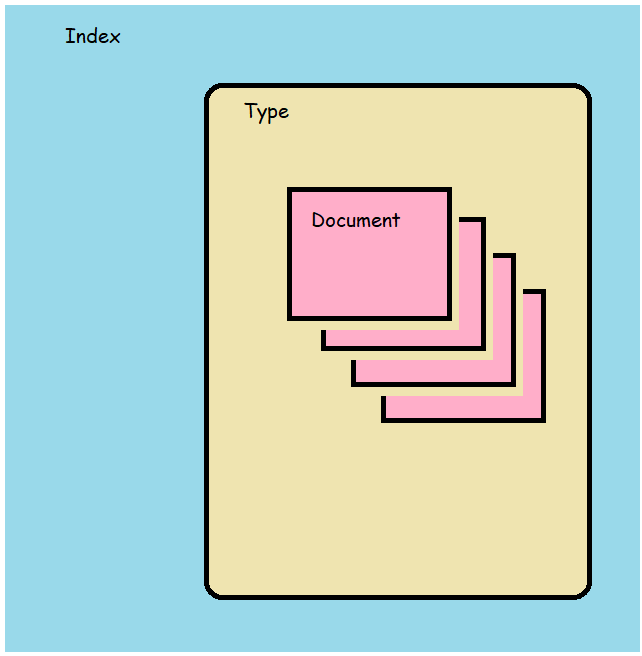
* To communicate with Elastic Serach we need to use REST API
* REST stands for Representational State Transfer. Its an architectural style that is used to make system inter operatate and interact with each other. REST has evolved over HTTP protocol.
* HTTP protocol supports different methods
  + GET
  + POST
  + PUT
  + DELETE
  + HEAD
* Kibana Console gives us a Console UI which is part of developer tools which makes it easier to interact with Elastic Search
* So we will be using Kibana Dev Tools to understand Elastic Search  

## Core Concepts of Elastic Search

* The JSON documents in Elastic Search are organized as Elastic Search is a document-oriented store.
* Following are core concepts of Elastic search
  + Indexes
  + Types
  + Documents
  + Clusters
  + Nodes
  + Shards and replicas
  + Mappings and Types
  + Inverted Indexes

## Index

* An index is a container that stores and manages documents of single type in Elastic Search



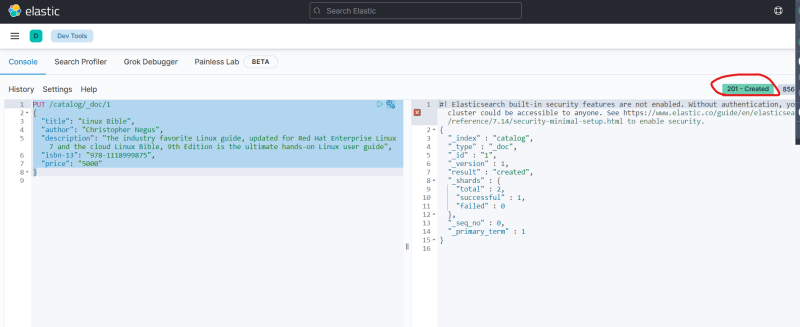
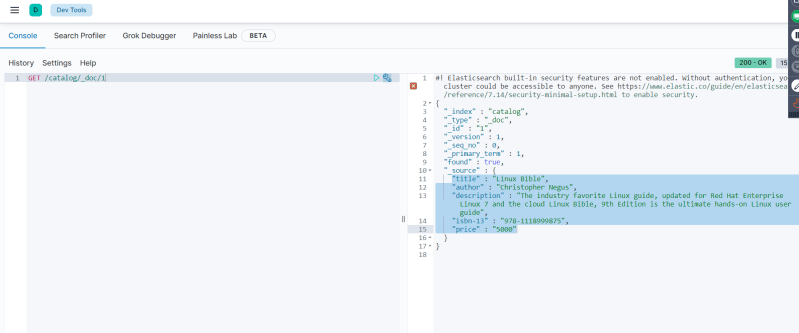
## Type

* In practice, we should avoid mixing different entitites such as customers and products into single type, It makes sense to store them in seperate types with seperate Entities

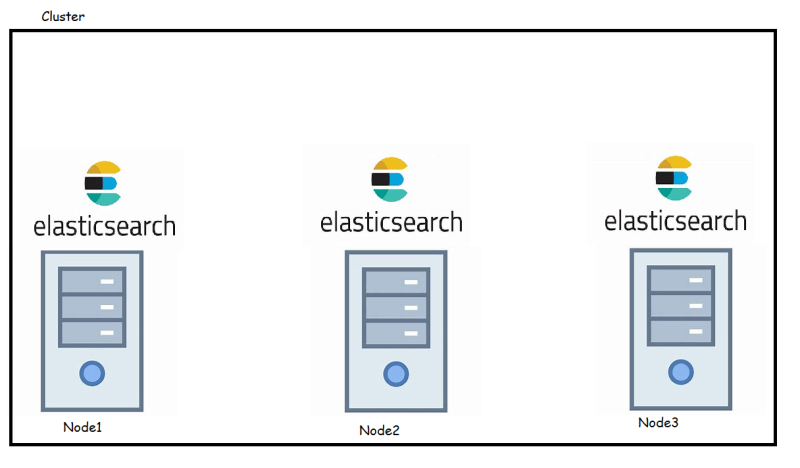
## Documents

* Documents contains multiple fields & each field in JSON document is of particular type
* In addition to the fields sent by user in the document, Elastic search maintains internal meta fields
  + \_id: This is unique identifier of the document within a type
  + \_type: This field contains the type of the document
  + \_index: This field contains the index name of the document

## Scenarios

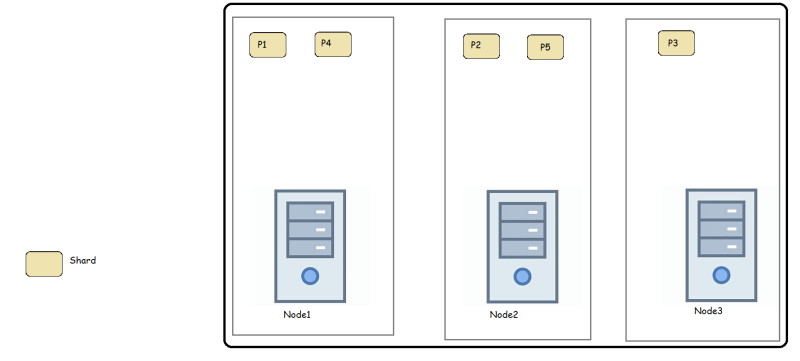
* Create a new index and document
* 
* Get the document created in the index
* 
* Note: HTTP Status codes [Refer Here](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status) (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>)

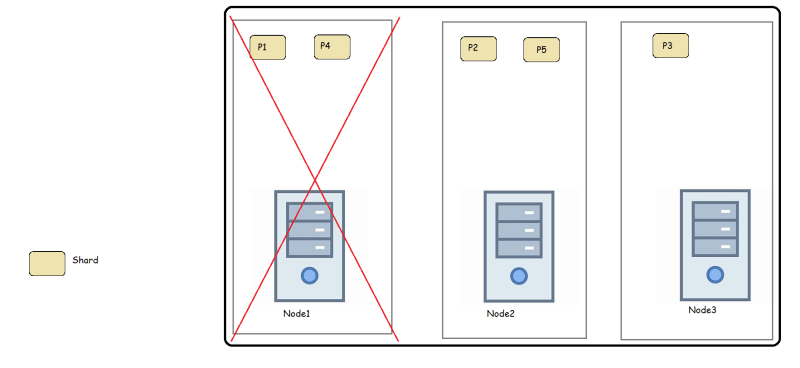
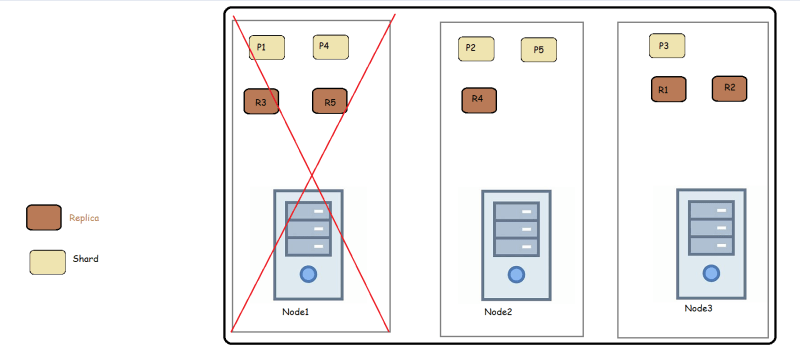
## Nodes and Clusters

* Elastic search is a distributed system and it consits of multiple processes running on different machines that communicate
* Each machine which runs elastic search is a Node
* Every node is associated with unique id and name
* Node details can be configured in elasticsearch.yml
* A cluster is formed by one or more nodes. Every Elastic search node is always part of cluster.
* By default, every Elastic search node tries to join a cluster with the name Elasticsearch
* A cluster consists of multiple nodes, where each node takes the responsibility for storing, managing its share of data.
* One cluster can host one or more indexes.
* 

## Shards and replicas

* Lets try to understand what shard is, An index contains documents & Shards help in distributing an Index over cluster.
* Shards help in dividing the documents of single index over multiple nodes (Shards split data of single index over the cluster), hence allowing the storage, memory and processing capacities of cluster to be utilized
* The process of dividing data among shards is called sharding
* By default every index is configured to have five shards in Elastic Search
* While creating index we can specify number of shards

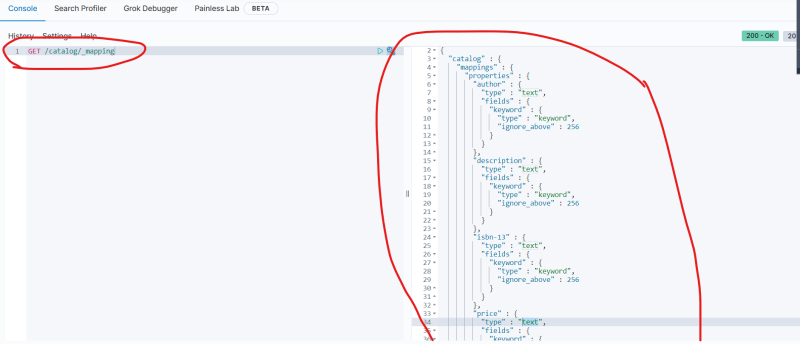
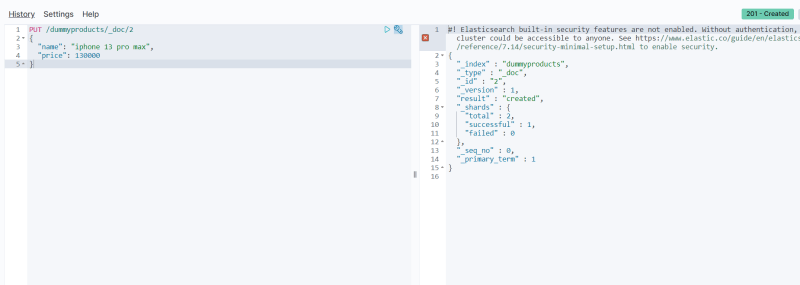
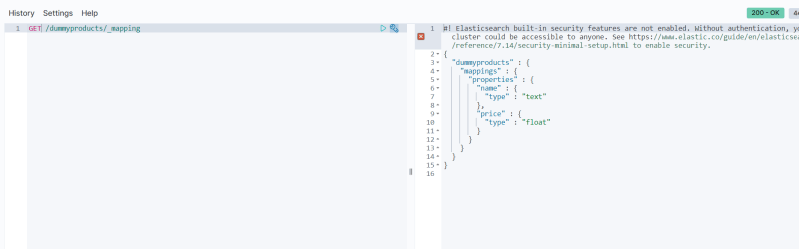
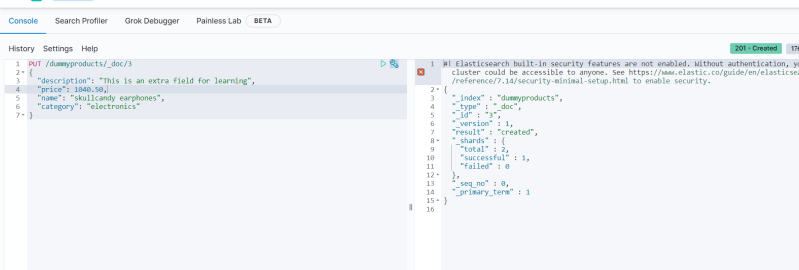
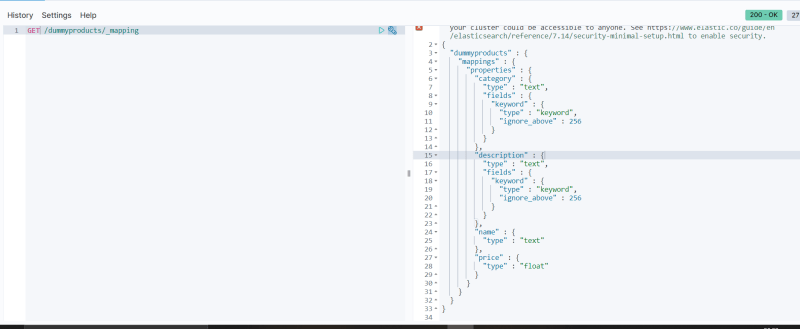
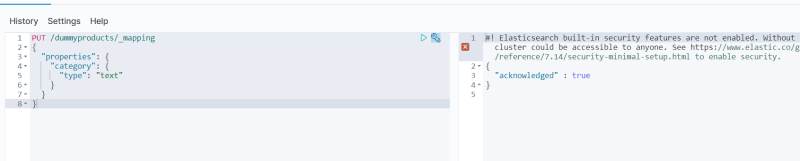


* Now lets assume Node 1 is down
* 
* Distributed systems such as Elastic search are expected to run inspite of hardware failures. This issue is address by replica shards or replicas.
* Each shard in index is configured to have zero or more replica shards. Replica Shards are extra copies of the original shards and they created for high availability of data
* 
* As you can observe in the image even if the Node 1 is down, then the data is getting served from other nodes with replicas ()

**=================================================================**

# Class-5: 20/Sept/2021

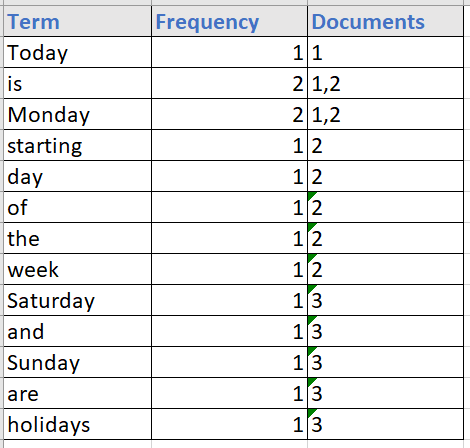
## Mappings and Data types

* Elastic search supports wide variety of data types
  + Core data types:
    - String data types:
      * Text: The text data type is useful for supporting full-text search for fields that contain a description or lengthy text values.
      * keyword: The keyword type enables analytics on string fields to support sorting, filtering and aggregations
    - Numeric data types:
      * byte (8), short (16),integer(32) and long(64)
      * float (32) and double (64)
      * half float (16)
      * scaled float
    - Date Data type
      * date
    - Boolean datatype
      * boolean
    - Binary data type:
      * binary
    - Range datatypes
      * integer\_range, float\_range, long\_range, double\_range and date\_range
  + Complex datatypes
    - Array datatype
    - Object datatype
    - Nested datatype
  + Other datatypes:
    - Geo-point
    - Geo-shape
    - Ip datatype
* Get a mapping for the index
* 
* Let’s create an index dummyproducts with some mapping   
* Now create a document in the dummy products with extra fields not specified in mapping
* 
* Now let’s get mapping for dummy products
* 
* Let’s try to update mappings [Refer Here](https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html) (<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html>)
* 

## Inverted Index

* An inverted index is the core data structure of Elastic search.
* Let’s assume we have following documents created

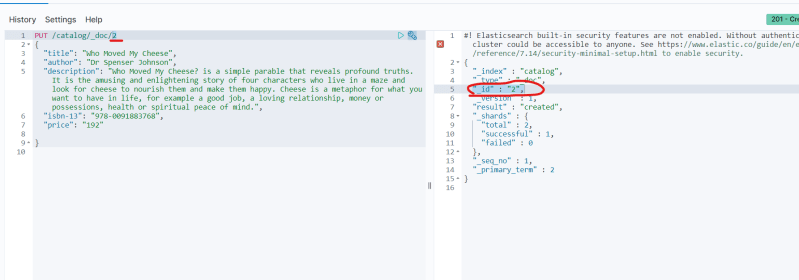
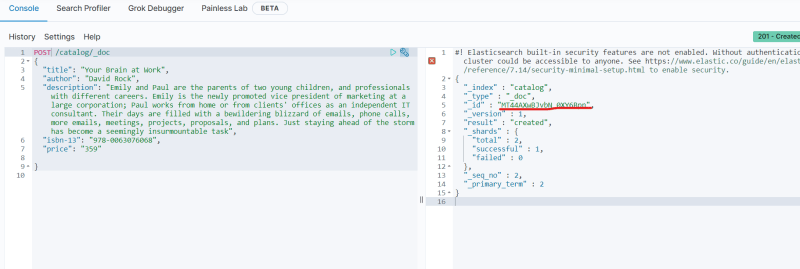
| **Document ID** | **Document** |
| --- | --- |
| 1 | Today is Monday |
| 2 | Monday is starting day of the week |
| 3 | Saturday and Sunday are holidays |

* Elastic search builds a data structure from the above three documents which is called as inverted index
* 
* When searching for terms in documents, it fast to locate the documents in which given term appears and inverted index is created.
* By default Elastic search builds an inverted index on all the fields in the document

## CRUD Operations

* To understand how to perform CRUD operations lets try to understand the following APIs
  + Index API
  + Get API
  + Update API
  + Delete API

## Index API

* In Elastic search terminology, adding a document to a type with in an index is called indexing operation.
* We have already learnt, indexing involves
  + adding the document to the index by parsing all the fields within the document
  + building an inverted index
* There are two ways we can index a document
  + Indexing a document with ID
  + 
  + Indexing a document without provding ID
  + 

**=================================================================**

# Class-6: 21/Sept/2021

**=================================================================**

# Class-7: /Sept/2021

**=================================================================**

# Class-8: /Sept/2021

**=================================================================**

# Class-9: /Sept/2021

**=================================================================**

# Class-10: /Sept/2021

**=================================================================**

# Class-11: /Sept/2021

**=================================================================**