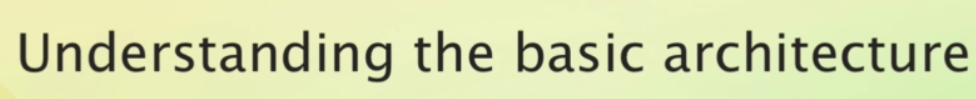
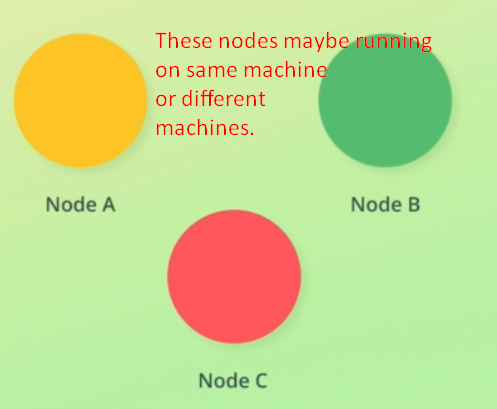
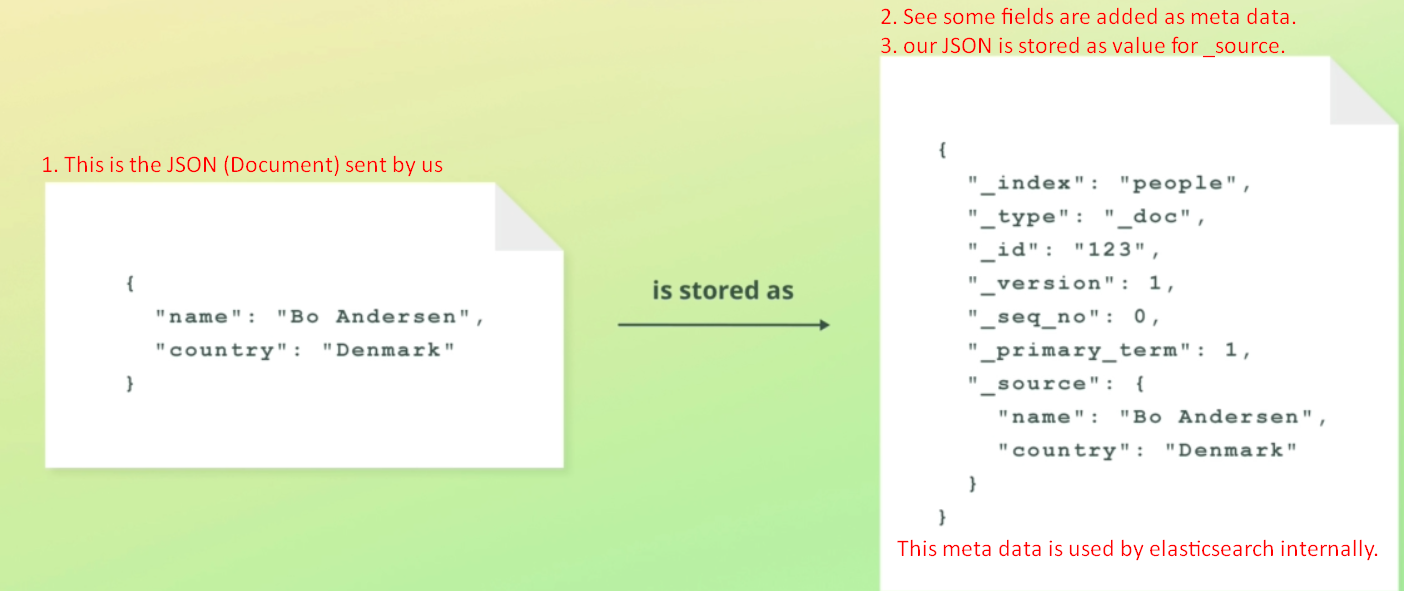
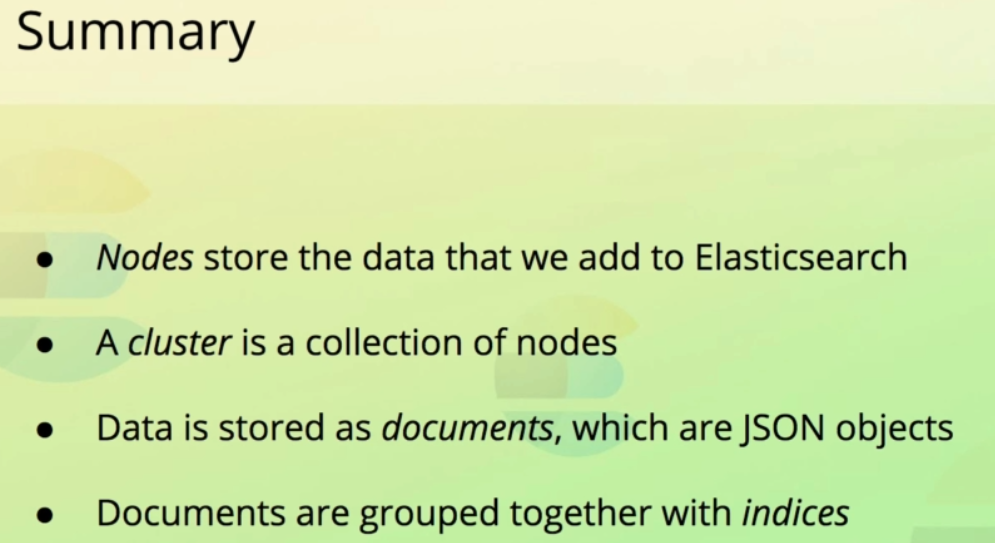
1. 
2. When we start up **Elasticsearch** we start up a node.
3.  **Node**: Node is essentially an **instance of Elasticsearch** that **stores** data.   
   To store many Terabytes, we can run as many nodes as we want.  
   Each node will then store a part of our data as our data may be 1000GB but the node capacity is 200GB but we have many nodes.

Node refers to an instance of Elasticsearch not to a machine. So we can start up five nodes if we want without having to deal with virtual machines or containers.  
**NOTE**: But we should typically separate things in a production environment so that each node run on a dedicated machine (virtual or container)

1. **Cluster**: A collection of related nodes that together contain all of the data.  
   Each node belongs to a cluster.  
   We can have many clusters but usually one is enough.  
   Clusters are independent of each other **by default**. It’s possible to perform **Cross –Cluster Searches** but it’s not very common to do so.  
   You might run multiple clusters that serve different purposes.  
   One for powering the search of an e-commerce application. One for Application Performance Management (APM).  
   The reasons for splitting things into multiple clusters are typically to separate things logically and to be able to configure things differently.  
   NOTE: We will be working with one cluster throughout the course.  
   No matter what happens, each node is part of a cluster.   
   **Scenario**: when elasticsearch is started up, its instance which is called node is created and we know that node always belongs to a cluster. How?  
   **Case 01**: If cluster doesn’t exist.  
    Then first node would create its own cluster consisting of just that node.   
   **Case 02**: If cluster exists already?  
    It will join an existing cluster if configured to do so.
2. There are some problems with only having a single node in terms of **availability and scalability**.  
   But for development purpose, it’s fine.   
   **NOTE**: We will discuss in detail but later.
3. Let’s see how data is organized and stored?  
   
   1. **Document**: Each **unit** of data being stored within cluster is called a **document**.
   2. Documents are JSON objects.
   3. JSON data sent by you to Elasticsearch is stored along with same **meta data** used internally by Elasticsearch.
4. Now the question is how documents are managed?
   1. The answer is within the **indices**.
   2. Every document within the Elasticsearch is stored within the index.
   3. An Index groups document together logically as well as provides configuration options that are related to scalability and availability which we will discuss alter.
   4. So, index is a collection of documents having similar characteristics and those documents are logically related.
   5. Index can store data as you want. There is no hard limit.
   6. Search query is run against index. Jatin🡺 Index plays the role of table.
5. Let’s take a short moment to recap on the key takeaways from this lecture.  
   

**Cluster**: An elasticsearch is a collection of nodes responsible for storing data.  
**Node**: Running instance of Elasticsearch running on physical machine or virtual machine or within a Docker Container.

**Document**: Data is stored as document. Unit of info.  
**Index**: Collection of related documents.