Bahaaldine Azarmi

Learning Kibana 5.0

Exploit the visualization capabilities of Kibana and build powerful interactive dashboards

Packt>

Learning Kibana 5.0

Exploit the visualization capabilities of Kibana and build powerful interactive dashboards

Bahaaldine Azarmi

BIRMINGHAM - MUMBAI

Learning Kibana 5.0

First published: February 2017

Production reference: 1100217

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.
ISBN 978-1-78646-300-5

www.packtpub.com

Credits

Author

Bahaaldine Azarmi

Reviewers

Alan Hardy
Bharvi Dixit

Commissioning Editor

Amey Varangaonkar

Acquisition Editor

Prachi Bisht

Content Development Editor

Manthan Raja

Technical Editor

Dharmendra Yadav

Copy Editor

Safis Editing

Project Coordinator

Nidhi Joshi

Proofreader

Safis Editing

Indexer

Aishwarya Gangawane

Graphics

Tania Dutta

Production Coordinator

Nilesh Mohite

About the Author

Bahaaldine Azarmi, Baha for short, is a Solutions Architect at Elastic. Prior to this position, Baha co-founded reachfive, a marketing data-platform focused on user behavior and social analytics. Baha also worked for different software vendors such as, Talend or Oracle, where he held the positions of Solutions Architect and Architect. Before *Learning Kibana 5.0*, Baha authored books such as *Scalable Big Data Architecture*, by *Apress* and *Talend for Big Data*, by *Packt Publishing*. Baha is based in Paris and has a Master's Degree in computer science from Polytech'Paris.

There are a few people I would like to acknowledge, as this book is not just me writing chapters but the work of many folks from Elastic regrouped into one book:

- Alan Hardy, EMEA Director of Solutions Architecture, for his mentoring, support, and for his excellent review
- Steve Mayzak, VP of Solutions Architecture, for supporting innovation in the SA team and me writing this book :-)
- Christian Dahlqvist, Solution Architect, for providing the Apache web logs demo used in `Chapter 4`, *Logging Analytics with Kibana 5.0*
- Rich Collier, Steve Dodson, and Sophie Chang, respectively Solutions Architect, Machine Learning Tech, and Team Lead, for helping me to understand the concept of anomaly detection and providing the demo used in `Chapter 8`, *Anomaly Detection in Kibana 5.0*
- Court Ewing and Spencer Alger, respectively Tech Lead and Javascript Developer, and actually, the whole Kibana team, for answering all my questions!

About the Reviewers

Alan Hardy has spent over 20 years in the software industry with a breadth of experience in different businesses and environments, from multinationals to software startups and financial organizations, including global trading and exchanges. He started out as a developer on real-time, monitoring, and alerting systems before moving into packet-switching technology and financial data processing in C, C++, and Java-based environments. Alan now works for Elastic, where he gets to fully explore his data-wrangling passion, leading EMEA Solution Architecture.

Bharvi Dixit is an IT professional and an engineer with extensive experience of working on the search servers, NoSQL databases and cloud services. He holds a master's degree in computer science and is currently working with Sentieo, a USA-based financial data and equity research platform where he leads the overall platform and architecture of the company, spanning hundreds of servers. At Sentieo, he also plays a key role in the search and data team.

He is also the organizer of Delhi's Elasticsearch Meetup Group, where he speaks about Elasticsearch and Lucene and continuously building the community around these technologies.

Bharvi also works as a freelance Elasticsearch consultant and has helped more than half a dozen organizations adapt Elasticsearch to solve their complex search problems in different use cases, such as creating search solutions for big data-automated intelligence platforms in the area of counter-terrorism and risk management, as well as in other domains, such as recruitment, e-commerce, finance, social search, and log monitoring.

He has a keen interest in creating scalable backend platforms. His other areas of interest are search engineering, data analytics, and distributed computing. Java and Python are the primary languages in which he loves to write code, and he has also built proprietary software for consultancy firms.

In 2013, he started working on Lucene and Elasticsearch, and has authored two books on Elasticsearch, *Elasticsearch Essentials* and *Mastering Elasticsearch 5.0*, both published by *Packt Publishing*.

You can connect with him on LinkedIn at `https://in.linkedin.com/in/bharvidixit` or can be followed on Twitter `@d_bharvi`.

www.PacktPub.com

For support files and downloads related to your book, please visit `www.PacktPub.com`.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www.packtpub.com/mapt`

Get the most in-demand software skills with Mapt. Mapt gives you full access to all Packt books and video courses, as well as industry-leading tools to help you plan your personal development and advance your career.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Customer Feedback

Thanks for purchasing this Packt book. At Packt, quality is at the heart of our editorial process. To help us improve, please leave us an honest review on this book's Amazon page at `https://www.amazon.com/Learning-Kibana-5-0-Bahaaldine-Azarmi-ebook/dp/B01LZ4ESK0/`.

If you'd like to join our team of regular reviewers, you can email us at `customerreviews@packtpub.com`. We award our regular reviewers with free eBooks and videos in exchange for their valuable feedback. Help us be relentless in improving our products!

For Aurelia, June, and Colin-Harper

Table of Contents

Preface

Today, understanding data, whatever the nature of the data is, keeps getting harder. They are couple of reasons for that such as the volume, the variety of data, the pace at which the data is created and the complexity to correlate data from different sources.

It's hard for anyone to cope with this constant increasing challenge, that's why more and more applications are built to facilitate data management, at every level: ingesting data, processing data, storing data, and ultimately visualizing the data to understand it.

All those levels put together are the fundamental layers to build a data-driven architecture that needs to scale with a growing demand and expectation from users.

There is tons of software and applications out there that could answer those challenges, but rarely will you find a stack that could fulfill all the requirements altogether and across many types of use cases.

The Elastic Stack is one of them: it gives the user a way to access their data in an agile and scalable way. Kibana is part of the Elastic Stack and provide a visualization layer on top of data indexed in Elasticsearch, the storage layer.

In *Learning Kibana 5.0*, we'll go through the holistic visualization experience that Kibana offers to address very different use cases, such as creating dashboards using accidents data, or building statistics on top system data, or even detecting anomalies in data.

Rather than listing and going through Kibana features one by one, this book adopts a pragmatic approach where you will learn based on concrete examples and hands-on.

What this book covers

`Chapter 1`, *Introduction to Data-Driven Architecture*, describes the fundamental layers that compose a data-driven architecture, and how the Elastic Stack can be used to build it.

`Chapter 2`, *Installing and Setting up Kibana 5.0*, covers the installation of Elasticsearch and Kibana, and a walkthrough in Kibana 5.0 anatomy.

`Chapter 3`, *Business Analytics with Kibana 5.0*, tackles the first use case of this book, namely business analytics, with the help of Paris accidentology data.

`Chapter 4`, *Logging Analytics with Kibana 5.0*, covers a technical logging use case on top of Apache logs data.

`Chapter 5`, *Metric Analytics with Metricbeat and Kibana 5.0*, walks the reader through the brand new feature of metrics analytics in Kibana 5.0 with the help of system data from Metricbeat.

`Chapter 6`, *Graph Exploration in Kibana*, explains the concept of graphs in the Elastic Stack and introduces forensic graph analysis on top of Stack Overflow data.

`Chapter 7`, *Customizing Kibana 5.0 Timelion*, shows how to extend the capabilities of Timelion and build an extension to fetch data from Google Analytics.

`Chapter 8`, *Anomaly Detection in Kibana 5.0*, covers the Elastic Stack machine learning features and how to use Kibana to visualize anomalies on top of system data.

`Chapter 9`, *Creating a Custom Plugin for Kibana 5.0*, explains how to create a plugin to visualize the Elasticsearch cluster topology.

What you need for this book

In this book, you will need to download and install the Elastic Stack, specifically, Elasticsearch, Kibana, Metricbeat, Logstash, and the X-Pack. All the software is available from the following page: `http://www.elastic.co/downloads`.

The Elastic Stack can be run on a various environment on commodity machines; here is the support matrix: `https://www.elastic.co/support/matrix`.

Who this book is for

This book is for developers, operation teams, business analytics, and data architects who want to learn how to deploy a data-driven architecture using the Elastic Stack 5.0, and more specifically, how to enable visualization on top of the data indexed in Elasticsearch with Kibana 5.0.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "We can include other contexts through the use of the include directive."

A block of code is set as follows:

```
PUT /_snapshot/basic_logstash_repository
{
  "type": "fs",
  "settings": {
  "location":
    "/Users/bahaaldine/Dropbox/Packt/sources/chapter3/
      basic_logstash_repository",
  "compress": true
  }
}
```

Any command-line input or output is written as follows:

```
GET _cat/indices/basic*
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Clicking the **Next** button moves you to the next screen."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the book's webpage at the Packt Publishing website. This page can be accessed by entering the book's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Learning-Kibana-5`. We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `https://www.packtpub.com/sites/default/files/downloads/LearningKibana5_ColorImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the **Errata** section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support`and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Introduction to Data-Driven Architecture

If you are reading this book, it certainly means that you and I have something in common: we are both looking for a solution to effectively visualize and understand our data.

Data can be anything: business data, infrastructure data, accounting data, numbers, strings, structured, or unstructured. In any case, all organizations reach a point where trying to understand data and extract the value of it begins to be a real challenge, for different reasons:

- **Data brings complexity**: If we take the example of an e-commerce IT operation team where one must find why the orders just dropped, it can be a very tricky process to go to the log to get the issue.
- **Data comes from a variety of sources**: Infrastructure, applications, devices, legacy systems, databases, and so on. Most of the time, you need to correlate them. In the e-commerce example, maybe the drop is due to an issue in my database?
- **Data increases at a very fast pace**: Data growth implies some new questions, such as which data should I keep? Or how do I scale my data management infrastructure?

The good news is that you won't need to learn it the hard way, as I'll try in this book to explain how I've tackled data analytics projects for different use cases and for different types of data based on my experience.

The other good news is that I'm part of the **Solutions Architecture** (**SA**) team at Elastic, and guess what? We'll use the Elastic stack. By being part of the SA team, I'm involved in a variety of use cases, from small to large scale, with different industries; the main goal is always to give to our users better management of and access to their data, and a better way to understand their data.

In this book, we'll dig into the use of Kibana, the data analytics layer of the Elastic stack. Kibana is the data visualization layer used in an overall data-driven architecture.

But what is data-driven architecture? This is the concept I will illustrate in this chapter by going through industry challenges, the usual technology used to answer this need, and then we'll go into the description of the Elastic stack.

Industry challenges

Depending on the industry, the use cases can be very different in term of data usage. Within a given industry, data is used in different ways for different purposes, whether it's for security analytics or order management.

Data comes in various formats and different scales of volumes. In the telecommunications industry, it's very common to see a project about the quality of services where data is grabbed from 100,000 network devices.

In every case, it always comes down to the same canonical issues:

- How to decrease the complexity of handling fast growing data at scale
- How to enable my organization to visualize data in the most effective and real-time fashion

By solving these fundamental issues, organizations would be allowed to simply recognize visual patterns without having to deal with the burden of exploring tons of data

To help you get a better understanding of the actual challenges, we'll start by describing the common use cases met across industries and then see what technologies are used and their limits in addressing these challenges.

Use cases

Every application produces data, whether it be in daily life when you use your favorite map application to geo-locate yourself and the best restaurant around you; or be it in IT organizations, with the different technical layers involved in building recommendations depending on your location and profile.

All computers and the processes and applications running on them are continuously producing data, effectively capturing the state of the system “now”, driven by a CPU tick or user click.

This data normally stays in obscure files, located physically on the computer and hidden deep within data centers. We need a means to extract this data (ship), convert it from obscure data formats (transform), and eventually store it for centralized access.

This flow of data streaming in the system, based on event triggering functional processes, needs a proper architecture to be shipped, transformed, stored, and accessed in a scalable and distributed way.

The way we interact with applications dramatically changed the legacy architecture paradigm that we used to lay out. It's not anymore about building relational databases, it's about spin up on demand distributed data stores based on the throughput; it's not only about having batch processing data overnight, but it's also about pushing data processing to boundaries that weren't met so far in terms of real-time and machine learning aspects; it's not anymore about relying on heavy business intelligence tools to build reporting, but more about an iterative approach to data visualization close to real-time insights.

End users, driven by the need to process increasingly higher volumes of data, while maintaining real-time query responses, have turned away from more traditional, relational database or data warehousing solutions, due to poor scalability or performance. The solution is increasingly found in highly distributed, clustered data stores that can easily be.

Take the example of application monitoring, which is one of the most common use cases we meet across industries. Each application logs data, sometimes in a centralized way, for example by using syslog, and sometimes all the logs are spread out across the infrastructure, which makes it hard to have a single point of access to the data stream.

When an issue happens, or simply when you need to access the data, you might need to get:

- **The location**: where the logs are stored.
- **The permission**: can I access the logs? If not, who should I contact to get them?
- **The understanding of the log structure**: I can take here the example of Tuxedo with multiline logs, which is not a trivial task at all.

The majority of large organizations don't retain logged data for longer than the duration of a log file rotation (a few hours or even minutes). This means that by the time an issue has occurred, the data which could provide the answers is lost.

When you actually have the data, what do you do? Well, there are different ways to extract the gist of logs. A lot of people start by using a simple string pattern search (GREP). Essentially, they try to find matching patterns in logs using a regular expression. That might work for a single log file but that doesn't scale as the log files rotate and you want to get insights over time, plus the fact that you may have more than one application and the need to make correlations.

Without any context regarding an issue (no time range, no application key, no insight), a user is reduced to brute force, assuming you are also looking in the correct file in the first place.

GREP is convenient, but clearly doesn't fit the need to react quickly to failure in order to reduce the **Mean Time To Recovery** (**MTTR**). Think about it: what if we are talking of a major issue on the purchase API of an e-commerce website? What if the users experience a high latency on this page or, worse, can't go to the end of the purchase process? The time you will spend trying to recover your application from gigabytes of logs is money you could potentially lose.

Another potential issue could be around a lack of security analytics and not being able to blacklist the IPs that try to brute force your application. In the same context, I've seen use cases where people didn't know that every night there was a group of IPs attempting to get into their system, and this was just because they were not able to visualize the IPs on a map and trigger alerts based on their value.

A simple, yet very effective, pattern in order to protect a system would have been to limit access to resources or services to the internal system only. The ability to whitelist access to a known set of IP addresses is essential.

The consequence could be dramatic if a proper data-driven architecture with a solid visualization layer is not serving those needs: lack of visibility and control, increasing the MTTR, customer dissatisfaction, financial impact, security leaks, and bad response time and user experience.

Fundamental steps

The objective is then to avoid these consequences, and build an architecture that will serve the different following aspects.

Data shipping

The architecture should be able to transport any kind of data/events, structured or unstructured; in other words, move data from remote machines to a centralized location. This is usually done by a lightweight agent deployed next to the data sources, on the same host, or on a distant host with regards to different aspects:

- Lightweight, because ideally it shouldn't compete for resources with the process that generates the actual data, otherwise it could reduce the expected process performance
- There are a lot of data shipping technologies out there; some of them are tight to a specific technology, others are based on an extensible framework which can adapt relatively to a data source
- Shipping data is not only about sending data over the wire, it's also about security and being sure that the data is sent to the proper destination with an end-to-end secured pipeline.
- Another aspect of data shipping is the management of data load. Shipping data should be done relative to the load that the end destination is able to ingest; this feature is called back pressure management

It's essential for data visualization to rely on reliable data shipping. Take as an example data flowing from financial trade machines and how critical it could be not to be able to detect a security leak just because you are losing data.

Data ingest

The scope of an ingest layer is to receive data, encompassing as wide a range of commonly used transport protocols and data formats as possible, while providing capabilities to extract and transform this data before finally storing it.

Processing data can somehow be seen as **extracting, transforming, and loading** (**ETL**) data, which is often called an ingestion pipeline and essentially receives data from the shipping layer to push it to a storage layer. It comes with the following features:

- Generally, the ingestion layer has a pluggable architecture to ease integration with the various sources of data and destinations, with the help of a set of plugins. Some of the plugins are made for receiving data from shippers, which means that data is not always received from shippers and can directly come from a data source, such as a file, network, or even a database. It can be ambiguous in some cases: should I use a shipper or a pipeline to ingest data from the file? It will obviously depend on the use case and also on the expected SLAs.
- The ingestion layer should be used to prepare the data by, for example, parsing the data, formatting the data, doing the correlation with other data sources, and normalizing and enriching the data before storage. This has many advantages, but the most important is that it can improve the quality of the data, providing better insights for visualization. Another advantage could be to remove processing overhead later on, by precomputing a value or looking up a reference. The drawback of this is that you may need to ingest the data again if the data is not properly formatted or enriched for visualization. Hopefully, there are some ways to process the data after it has been ingested.
- Ingesting and transforming data consumes compute resources. It is essential that we consider this, usually in terms of maximum data throughput per unit, and plan to ingestion by distributing the load over multiple ingestion instances. This is a very important aspect of real-time visualization which is, to be precise, near real-time. If ingestion is spread across multiple instances, it can accelerate the storage of the data, and therefore make it available faster for visualization.

Storing data at scale

Storage is undoubtedly the masterpiece of the data-driven architecture. It provides the essential, long-term retention of your data. It also provides the core functionality to search, analyze, and discover insights in your data. It is the heart of the process. The action will depend on the nature of the technology. Here are some aspects that the storage layer usually brings:

- Scalability is the main aspect, the storage used for various volumes of data which could start from GB, TB, to PB of data. The scalability is horizontal, which means that as the demand and volume grow, you should be able to increase the capacity of the storage seamlessly by adding more machines.

- Most of the time, a non-relational and highly distributed data store, which allows fast data access and analysis at a high volume and on a variety of data types, is used, namely a NoSQL data store. Data is partitioned and spread over a set of machines, in order to balance the load while reading or writing data.
- For data visualization, it's essential that the storage exposes an API to make analysis on top of the data. Letting the visualization layer do the statistical analysis, such as grouping data over a given dimension (aggregation), wouldn't scale.
- The nature of the API depends on the expectation on the visualization layer, but most of the time it's about aggregations. The visualization should only render the result of the heavy lifting done at the storage level.
- A data-driven architecture can serve data to a lot of different applications and users, and for different levels of SLAs. High availability becomes the norm in such architecture and, like scalability, it should be part of the nature of the solution.

Visualizing data

The visualization layer is the window on the data. It provides a set of tools to build live graphs and charts to bring the data to life, allowing you to build rich, insightful dashboards that answer the questions: What is happening now? Is my business healthy? What is the mood of the market?

The visualization layer in a data-driven architecture is one of the potential data consumers and is mostly focused on bringing KPIs on top of stored data. It comes with the following essential features:

- It should be lightweight and only render the result of processing done in the storage layer
- It allows the user to discover the data and get quick out-of-the box insights on the data
- It brings a visual way to ask unexpected questions to the data, rather than having to implement the proper request to do that
- In modern data architectures that must address the needs of accessing KPIs as fast as possible, the visualization layer should render the data in near real-time
- The visualization framework should be extensible and allow users to customize the existing assets or to add new features depending on the needs
- The user should be able to share the dashboards outside of the visualization application

As you can see, it's not only a matter of visualization. You need some foundations to reach the objectives.

This is how we'll address the use of Kibana in this book: we'll focus on use cases and see what is the best way to leverage Kibana features, depending on the use case and context.

The main differentiator with the other visualization tools is that Kibana comes along a full stack, the Elastic stack, with a seamless integration with every layer of the stack, which just eases the deployment of such architecture.

There are a lot of other technologies out there; we'll now see what they are good at and what their limits are.

Technologies limits

In this part, we'll try to analyze why some technologies can have limitations when trying to fulfill the expectations of a data-driven architecture.

Relational databases

I still come across people using relational databases to store their data in the context of a data-driven architecture; for example, in the use case of application monitoring, the logs are stored in MySQL. But when it comes to data visualization, it starts to break all the essential features we mentioned earlier:

- A **Relational Database Management System** (**RDBMS**) only manages fixed schemas and is not designed to deal with dynamic data models and unstructured data. Any structural changes made on the data will need to update the schema/tables, which, as everybody knows, is expensive.
- RDBMS doesn't allow real-time data access at scale. It wouldn't be realistic, for example, to create an index for each column for each table, for each schema in a RDBMS; but essentially that is what would be needed for real-time access.
- Scalability is not the easiest thing for RDBMS; it can be a complex and heavy process to put in place and wouldn't scale against a data explosion.

RDBMS should be used as a source of data that can be used before ingestion time to correlate or enrich ingested data to have a better granularity in the visualized data.

Visualization is about providing users with the flexibility to create multiple views of the data, enabling them to explore and ask their own questions without predefining a schema or constructing a view in the storage layer.

Hadoop

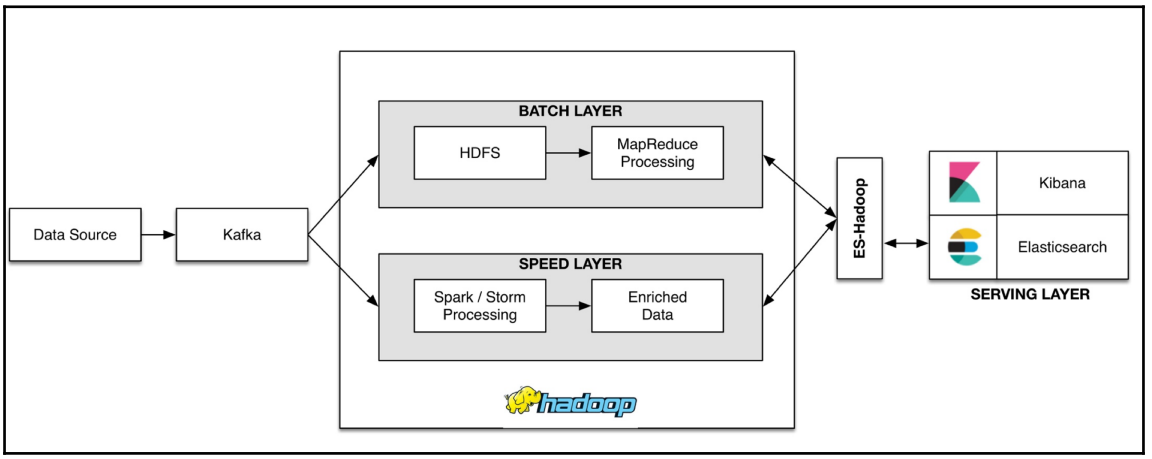
The Hadoop ecosystem is pretty rich in terms of projects. It's often hard to pick or understand which project will fit the ones needed; if we step back, we can consider the following aspects that Hadoop fulfills:

- It fits for massive-scale data architecture and will help to store and process any kind of data, for any level of volume
- It has out of-the-box batch and streaming technologies that will help to process the data as it comes in to create an iterative view on top of the raw data, or longer processing for larger-scale views
- The underlying architecture is made to make the integration of processing engines easy, so you can plug and process your data with a lot of different frameworks
- It's made to implement the data lake paradigms where one will essentially drop its data in order to process it

But what about visualization? Well, there are tons of initiatives out there, but the problem is that none of them can go against the real nature of Hadoop, which doesn't help for real-time data visualization at scale:

- Hadoop Distributed File System (HDFS) is a sequential read and write filesystem, which doesn't help for random access.
- Even the interactive ad hoc query or the existing real-time API doesn't scale in terms of integration with the visualization application. Most of the time, the user has to export its data outside of Hadoop in order to visualize it; some visualizations claim to have a transparent integration with HDFS, whereas under the hood, the data is exported and loaded in the memory in batches, which make the user experience pretty heavy and slow.
- Data visualization is all about APIs and easy access to the data, which Hadoop is not good at, as it always requires implementation from the user.

Hadoop is good for processing data, and is often used conjointly with other real-time technology, such as Elastic, to build Lambda architectures as shown in the following diagram:



Lambda architecture with Elastic as a serving layer

In this architecture, you can see that Hadoop aggregates incoming data either in a long processing zone or a near real-time zone. Finally, the results are indexed in **Elasticsearch** in order to be visualized in **Kibana**. This means essentially that one technology is not meant to replace the other, but that you can leverage the best of both.

NoSQL

There are a lot of different very performant and massively scalable NoSQL technologies out there, such as key value stores, document stores, and columnar stores, but most of them do not serve analytic APIs or don't come with an out-of-the box visualization application.

In most cases, the data that these technologies is using is ingested in an indexation engine such as Elasticsearch to provide analytics capabilities for visualization or search purposes.

With the fundamental layers that a data-driven architecture should have and the limits identified in existing technologies in the market, let's now introduce the Elastic stack, which essentially answers these shortcomings.