

# Reference Architecture: Confluent and Elastic



Elastic and Confluent provide their customers a better overall experience building contextual event driven applications leveraging a modern document-based distributed database. Confluent provides distributed, scalable, and secure event delivery that can scale to handle trillions of events a day with Confluent Platform and Confluent Cloud. Elastic offers secure and flexible data storage, aggregation and search & real-time analytics in Elastic on premise and Elastic Cloud.

This document provides an overview of Confluent and Elastic offerings, some detailed tutorials for getting started with the integration, opinionated recommendations on how to best utilize Confluent and Elastic Search together both on-prem and in cloud, some guidelines for deployment, and unique considerations to keep in mind when working with these two technologies.

## Contents

<b>Use Cases</b>	<b>2</b>	<b>Self-Managed</b>	<b>8</b>	<b>Additional Resources</b>	<b>11</b>
<b>Technical Brief</b>	<b>2</b>	On-Prem	8	Use Cases	11
<b>Partner Product Overview</b>	<b>3</b>	In-Cloud	8	Concepts, Documentation & Training	11
Product Components	3	Kubernetes	9	Tutorials & Quickstarts	12
Product Concepts	4	<b>Self Managed Kafka Connect for</b>		Deployment & Production	12
Technology Overlap	4	<b>Elastic Considerations</b>	<b>9</b>		
Terminology Mapping	5	Schema	9		
<b>Deployment</b>	<b>6</b>	Message Sizes	9		
Hybrid On-Prem/in-Cloud	6	<b>Confluent Cloud and</b>			
<b>Managed-Cloud</b>	<b>7</b>	<b>Elastic Demo</b>	<b>10</b>		
		Demo	10		

# Use Cases

Although the potential use cases for the integration are infinite, the most common use cases are listed below. If your use cases don't match these, you might be subject to additional considerations.

- **Bidirectional ETL 1:1** (Data source / RDBMS <-> Confluent -> Elastic)
- **Bidirectional ETL many:many** (Lots of data sources <-> Confluent -> Elastic)
- **SIEM\Security Analytics\Infra monitoring** (Network device/ logs -> Confluent -> Elastic)
- **IoT** (IoT Device -> Confluent -> Elastic)

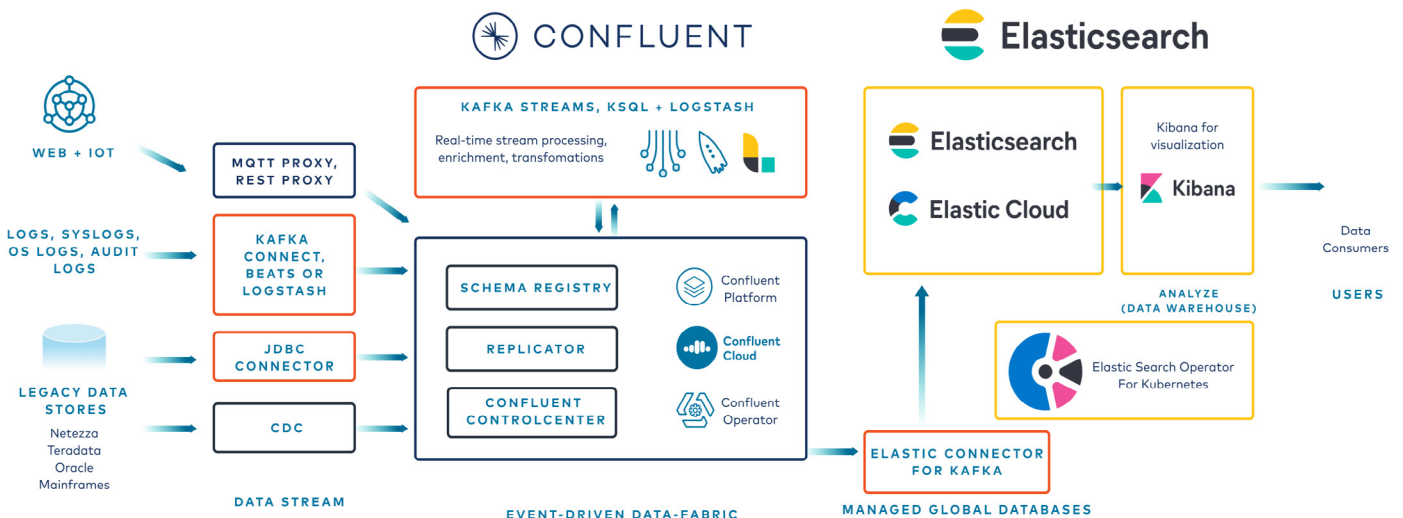
# Technical Brief

The recommended Elastic integration with Confluent is via the [Elastic Connector for Apache Kafka](#) supported by Confluent. The sink connector for Elastic is integrated with Schema Registry and converts the data into native JSON format before ingesting into Elastic.

Other way of Elastic integration with Confluent is using Elastic supported Beats or Logstash. [Beats output to Kafka](#) which is similar to Kafka connect based connectors to read events from multiple sources and ingest into Kafka. Key difference being Kafka Connect integration with the Schema registry which is not available in Beats.

The same holds true for Logstash source as well as sink connector which can consume and produce data to Kafka but lacks integration with schema registry.

<https://www.elastic.co/guide/en/logstash/current/plugins-inputs-kafka.html>  
<https://www.elastic.co/guide/en/logstash/current/plugins-outputs-kafka.html>



COMMON TOOLS ON PREMISE OR ON ANY CLOUD Google Cloud Microsoft Azure AWS

The above diagram contextualizes the connector as both a source and sink to Confluent Platform and Confluent Cloud

# Partner Product Overview

In order to properly use the connector, it's important to understand some broader context, such as the components that might be deployed on the Elastic side as well as the concepts used to represent data in Elastic.

## Product Components

Elastic Offers a data platform consisting of:

- **Elasticsearch** is a distributed JSON based search and analytics engine.
- **Beats** is a free and open platform for single-purpose data shippers, they collect data from a variety of sources like log files, network data etc. and publish it to Elastic search, Logstash or Kafka etc.
- **LogStash** is a free and open server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to multiple targets like Elastic, Kafka etc.
- **Kibana** — enables you to interactively explore, visualize, and share insights into data stored in Elastic and manage and monitor the stack.
- **Elastic cloud** is a hosted service of elasticsearch and Kibana on AWS, GCP and Azure. Beats and Logstash are not fully managed as part of Elasticsearch service.

Beats is a lightweight component for collecting data from various event sources and ingest into targets like Elastic Search or Kafka. Beats has limited transformational capabilities. It is primarily used to deploy remotely for ex; on servers to collect metrics and forward to ES or Kafka for further analysis.

Beats family (data source connectors):

- Audit beat
- File beat
- Function Beat
- Heart Beat
- Metric beat
- Packet beat
- Winlog beat

Logstash is primarily a stream processing tool to process/transform the data especially good with security logs as it has a prebuilt library to build parsers very quickly. Along with that Logstash can also connect with a wide variety of data sources directly like ETL tools. Logstash is deployed across multiple nodes for high availability and fail over.

Beats and Kafka connect are meant for similar use cases i.e.

- deploy remotely
- lightweight infrastructure
- limited data processing

File source is a common connector between Beats and Kafka connect apart from this rest of the connectors don't overlap.

Another consideration would be Beats and Logstash does not support integration with Schema registry, hence a use case where Schema registry is leveraged for data governance Kafka connect and KSQL would be recommended.

Logstash data pipeline configurations have to be manually deployed across nodes to achieve high availability and failover unlike KSQL which runs in a cluster mode. Logstash has a very powerful library to build log parsers quickly.

## Elastic Concepts

Elasticsearch is a distributed, open source search and analytics engine for all types of data, including textual, numerical, geo-spatial, structured, and unstructured. Due to its handling of structured and unstructured data it falls into the category of Non relational and NoSQL database. In order to contextualize the Elastic / Confluent integration it's important to understand the key concepts Elastic works with..

A **document** is the core data element in Elastic, its equivalent to a row in a relational database. An Elasticsearch **Index** is a collection of documents that are related to each other. An Index is similar to Table in relational systems without the strict table definition. Elasticsearch stores data as **JSON** documents. Each document correlates a set of keys (names of fields or properties) with their corresponding values (strings, numbers, Booleans, dates, arrays of values, geolocations, or other types of data).

The documents stored in Elasticsearch are distributed across different containers known as **shards**, which are duplicated to provide redundant copies of the data in case of hardware failure. These copies are known as a **replication group** and must be kept in sync when documents are added or removed.

## Technology Overlap

PURPOSE	CONFLUENT OFFERING	ELASTIC OFFERING
Data onboarding to Kafka from various data sources	pre-built Kafka Connectors	Beats family
Stream processing	Kafka Streams\ksqlDB	LogStash

Confluent connectors and Elastic beats are built for Data collection from multiple source applications into Kafka or Elastic. Beats are designed to be deployed remote or close to source and deployed in standalone mode, whereas Confluent connectors can be deployed close to source (standalone mode) or run close to Kafka (distributed mode) in a cluster mode.

KsqlDB is deployed as a cluster whereas Logstash configurations work in standalone mode which brings in the challenges on managing failover, parallelism, resource management etc.

While ksqlDB is a more generic all purpose stream processing tool, logstash is good at log parsing with a lot of in-built libraries to parse the unstructured logs.

## Terminology Mapping

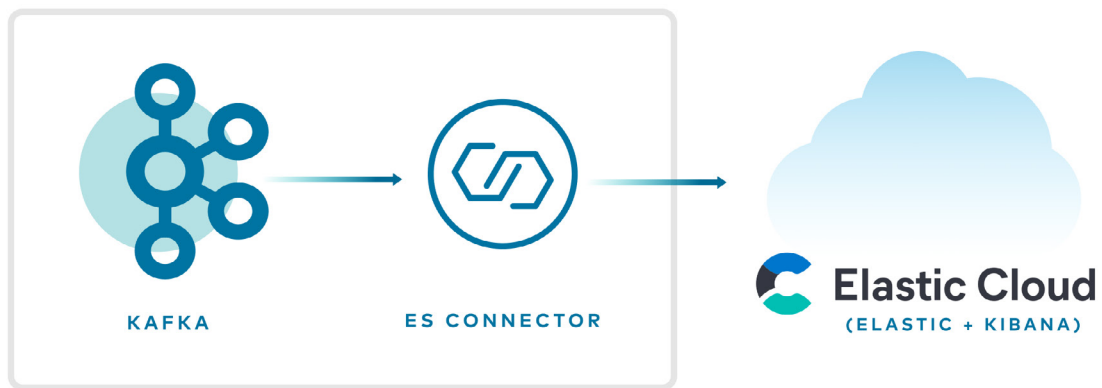
Sometimes it makes sense to compare terminology with something you already know. For example, someone who knows Apache Kafka well might understand Elastic better by trying to map terminology from one system to the other. It helps to understand how these concepts are similar and different. Below is a simple table that attempts to differentiate the core concepts in Elastic from those in Apache Kafka.

THIS	ROUGHLY APPROXIMATES THIS	BECAUSE...	BUT NOT REALLY BECAUSE...
<b>node</b>	<b>broker</b>	Like a kafka broker, node is equivalent in Elastic for scaling out horizontally.	broker serves events and Elastic serves document data. Elastic nodes are also responsible for the indexing as well as storage, whereas in Confluent these functions are broken out.
<b>document</b>	<b>event</b>	A document is the core data element in Elastic, an event is the core data element in Apache Kafka	Events are strongly ordered by offset and limited in size. Documents tend to be a little bigger and are indexed based on fields for faster search. Also, a document in Elastic is in JSON while an event in Kafka is a key + value encoded in bytes..
<b>index</b>	<b>topic</b>	Both are groupings of data in their respective platform.	It's a best practice that all events in a topic match the same schema. While Elastic is schemaless, but its recommended to similar documents together in a Index like Tables in RDBMS
<b>shard</b>	<b>partition</b>	Each constitutes a subset of the data set that is replicated across a cluster	Similarly, topics are partitioned, meaning a topic is spread over a number of "buckets" located on different Kafka brokers.

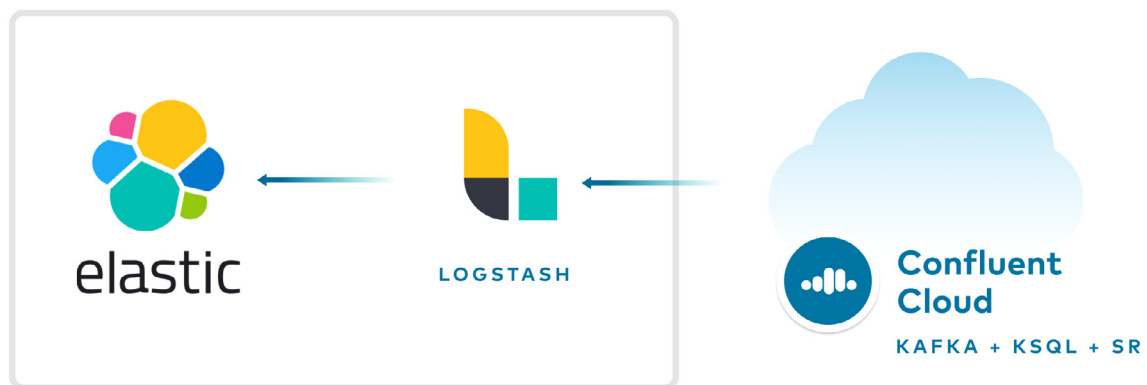
# Deployment

## Hybrid on-prem/in-Cloud

**Elastic cloud** is a fully-managed Elastic Search developed by the same people that build Elastic. Elastic Cloud handles all the complexity of deploying, managing, and operation on the cloud service provider of your choice. Elastic cloud includes fully managed Elastic Search and Kibana.



In the above scenario wherein Kafka is on-premise and Fully managed elastic on Cloud, Elasticsearch sink connector from Confluent would sync the events in real time between the two ecosystems.



In the above scenario wherein Elastic is on premise and fully managed confluent on Cloud there are two options:

- Logstash to sink data between Confluent cloud and Elastic Cloud (above) or
- Fully managed Elastic search sink supported by Confluent (below)



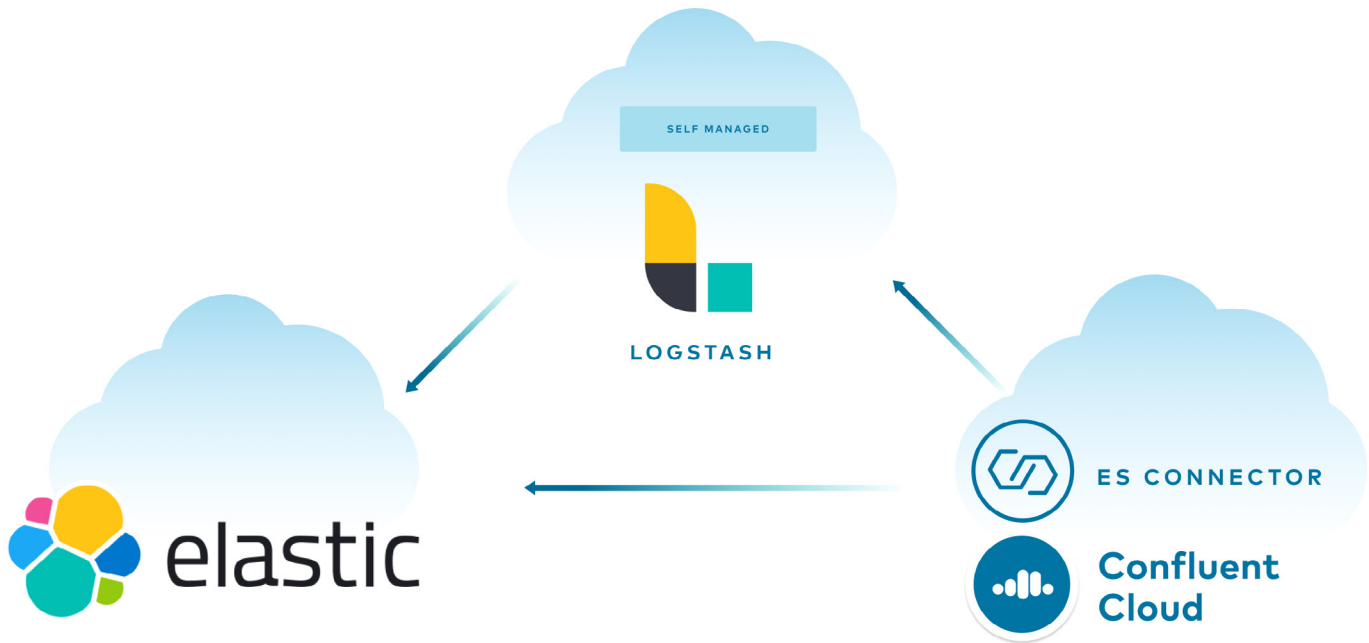
**Managed-Cloud**



In the above scenario where events need to be ingested into Elastic cloud from Confluent cloud there are two options:

1. A fully managed ES connector from Confluent to consume from confluent cloud and index into Elastic (above)
2. A self managed Logstash instance to consume from Confluent cloud and index into Elastic (below)

Elastic cloud has fully managed Elastic search and Kibana while Logstash and Beats are not available on Cloud as fully managed. On the other hand Confluent cloud has fully managed Kafka, KSQL, Connectors, Schema registry to build end to end streaming pipelines.



# Self-Managed

## On-Prem

Both Elastic and Confluent Platform install on-prem in the following formats:

- zip/tar for manual installation
- Linux native packaging (OS packages such as apt-get or yum)
- Docker images
- Ansible playbooks

## In-Cloud

- Confluent Cloud is strongly recommended, but recipes are available for each cloud provider for self-managed deployments.
- Elastic Cloud is strongly recommended, but Elastic Cloud enterprise is available for self-managed deployments of Elastic ecosystem.

## Kubernetes

**Confluent Operator** allows you to deploy and manage Confluent Platform as a cloud-native, stateful container application on Kubernetes and OpenShift. The automation provided by Kubernetes, Operator, and Helm greatly simplifies provisioning and minimizes the burden of operating and managing Confluent Platform clusters. Operator also provides you with the portability to use Apache Kafka® in multiple provider zones and across both your private and public cloud environments.



**Elastic Cloud on Kubernetes** simplifies setup, upgrades, snapshots, scaling, high availability, security, and more for running Elasticsearch and Kibana in Kubernetes. Built on the Kubernetes Operator pattern, extends Kubernetes orchestration capabilities to support the setup and management of Elasticsearch and Kibana on Kubernetes.

Both operators are controlled and configured via standard Kubernetes mechanisms such as helm, yaml files, and kubectl commands. Deploying the two operators together in production is an exercise in **multi-operator** deployment, leveraging node affinity and pod affinity to ensure proper resource isolation. Specific best practices for running these two operators together have yet to be identified.

# Self managed Kafka connect for Elastic considerations

Below are some of the features / considerations while configuring Kafka connect for ES:

- **Exactly once delivery** - the connector relies on Elasticsearch's idempotent write (document id) semantics to ensure exactly once delivery to Elasticsearch. By setting IDs in Elasticsearch documents, the connector can ensure exactly once delivery.
- **Mapping Inference:** The connector can infer mappings from Connect schemas. When enabled, the connector creates mappings based on schemas of Kafka messages.
- **Schema Evolution:** The connector supports schema evolution and can handle backward, forward, and fully compatible schema changes in Connect. It can also handle some incompatible schema changes such as changing a field from an integer to a string.
- **Mapping Management:** Index templates can be helpful when manually defining mappings, and allow you to define templates that are automatically applied when new indices are created.
- **Index Management:** Connector provides support to create a new index on time based intervals such as Daily automatically via configuration in the connector which helps in managing indexes in ElasticSearch.

## Schema

ElasticSearch is schema less, it accepts documents in JSON format. Confluent Platform uses Schema Registry to enforce schema, standardize on Avro for serialization, and facilitate schema evolution in Apache Kafka topics. Schema registry ensures data is governed on Kafka topics in turn ensuring governance on ElasticSearch. Also its highly recommended to define Index templates for additional governance on ElasticSearch.

## Message Sizes

Kafka defaults to a 1MB message size. If the JSON string size of the change stream document is greater than 1MB, you will need to configure Kafka to handle larger sized documents.

# Confluent and Elastic Demo

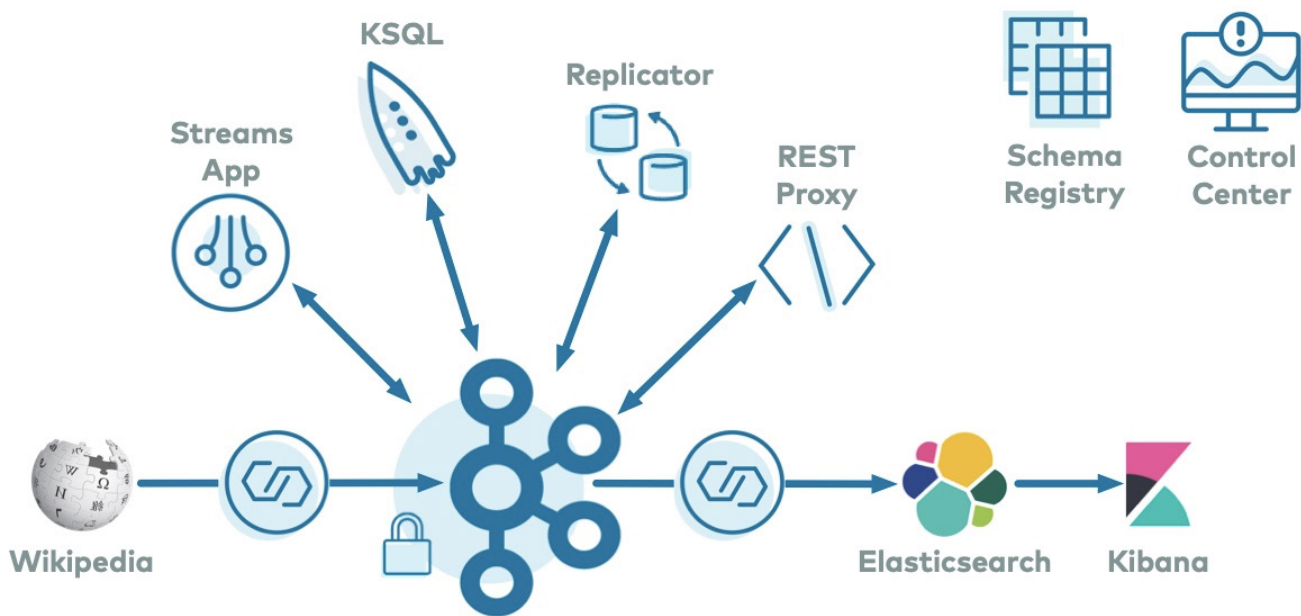
Most of the Confluent demos have Elastic Search & Kibana dashboard as an endpoint to showcase the joint capabilities.

One of the most frequent demos used is the CP demo or Wiki demo available at:

<https://docs.confluent.io/current/tutorials/cp-demo/docs/index.html>

This is a great demo to capture the edits to the wikipedia pages published to IRC channels in real time. Kafka source connector for irc (kafka-connect-irc) streams raw messages from these IRC channels and are written to a kafka cluster. These raw messages are then processed using ksqlDB and Kafka streams applications. Finally Kafka connect for Elasticsearch is set up to stream the processed events to Elasticsearch Index from Kafka. All the processed data can be viewed/analysed on Kibana dashboards in real time.

Below diagram illustrates the flow of events.



# Additional Resources

## Use Cases

- Case studies published by Elastic:
  - Featured customers: <https://www.elastic.co/customers/>
  - <https://www.elastic.co/customers/furuno>
  - <https://www.elastic.co/elasticon/tour/2020/stockholm/volvo-group-connected-solutions-starting-the-observability-journey-with-elastic>
  - Bayer customer of Confluent and Elastic: <https://www.elastic.co/elasticon/tour/2019/munich/large-scale-patent-analytics-at-bayer>
  - <https://www.elastic.co/blog/a-round-of-applause-for-the-2019-emea-elastic-search-awards-honorees>
  - <https://www.elastic.co/elasticon/conf/2016/sf/all-quiet-digital-front-security-analytics-usaa>
- Kafka connect and Elasticsearch
  - <https://rmoff.net/2019/10/07/kafka-connect-and-elasticsearch/>
- Fine tuning and Considerations
  - <https://www.confluent.io/blog/kafka-elasticsearch-connector-tutorial/>

## Concepts, Documentation and Training

- Basic Concepts:
  - <https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html>
- Scalability - Cluster, Nodes, Shards
  - <https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html>
- Data model, data types
  - <https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html/>
  - <https://www.elastic.co/blog/introducing-the-elastic-common-schema>
- Beats (Connectors equivalent of ES)
  - <https://www.elastic.co/guide/en/beats/libbeat/current/index.html>
- Logstash (streams\ksql equivalent of ES)
  - <https://www.elastic.co/logstash>
- Kibana
  - <https://www.elastic.co/kibana>
- Confluent Platform documentation
  - <https://docs.confluent.io/current/platform.html>
- Apache Avro, used by the schema registry
  - <http://avro.apache.org/>

## Tutorials and Quickstarts

- Tutorial: Deploying Elastic Search ecosystem:
  - <https://www.elastic.co/guide/en/elastic-stack-get-started/7.6/get-started-elastic-stack.html>
  - <https://www.elastic.co/guide/en/apm/get-started/7.6/quick-start-overview.html>
- Managed ES on Cloud
  - <https://www.elastic.co/cloud/>
- Kafka connect for ES
  - <https://www.confluent.io/blog/kafka-elasticsearch-connector-tutorial/>
- Blog for Kafka and ES
  - <https://www.confluent.io/blog/elasticsearch-ksqldb-integration-for-data-enrichment-and-analytics>
  - <https://medium.com/@bravnic/kafka-streams-elasticsearch-and-kibana-4f7889a27dcf>

## Deployment and Production

- ES Reference
  - <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>
- ES architecture best practices:
  - <https://www.elastic.co/webinars/elasticsearch-architecture-best-practices>
- The connector on Confluent Hub
  - <https://www.confluent.io/hub/confluentinc/kafka-connect-elasticsearch>
- The connector source code:
  - <https://github.com/confluentinc/kafka-connect-elasticsearch>
- Connector documentation:
  - <https://github.com/confluentinc/kafka-connect-elasticsearch/commit/882c11f8b2d673e576c32c3ee94f8f217c7adf65>
- Kubernetes Resources
  - <https://www.elastic.co/guide/en/cloud-on-k8s/current/k8s-deploy-eck.html>
  - Operator: <https://www.elastic.co/blog/introducing-elastic-cloud-on-kubernetes-the-elasticsearch-operator-and-beyond>
  - Operator setup: <https://medium.com/@izekchen/step-by-step-installation-for-elasticsearch-operator-on-kubernetes-and-metricbeat-filebeat-and-67a6ec4931fb>