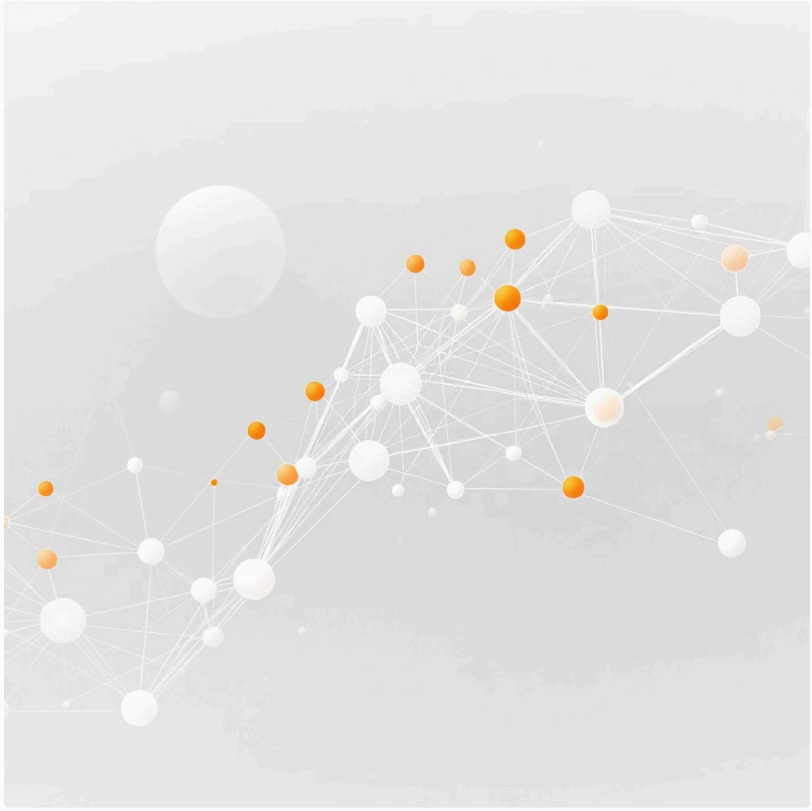
Introduction to the Elastic Stack

Exploring distributed search and analytics for telecom and enterprise environments

Elasticsearch: The Foundation

Elasticsearch is a distributed search and analytics engine built on Apache Lucene, created by Shay Banon in 2010. Since then, it has evolved from a simple search tool to an enterprise-grade platform powering real-time analytics.

- JSON document storage for flexible data models
- Distributed architecture for horizontal scaling
- Real-time indexing and search capabilities
- RESTful API for easy integration



Evolution Through Key Releases

1

2010-2012: Foundation

Full-text search with distributed architecture and clustering capabilities

2

2013-2015: Stack Formation

Integration with Logstash and Kibana creates the ELK Stack ecosystem

3

2016-2018: Enterprise Features

Security, alerting, and machine learning capabilities added

4

2019-Present: APM & Observability

Application performance monitoring and full observability platform

NoSQL vs RDBMS: Fundamental Differences

RDBMS

Structured Data

Fixed schemas with tables and relationships

ACID Compliance

Strong consistency and transaction integrity

SQL Queries

Standardized query language for relational operations

NoSQL

Schema Flexibility

Dynamic schemas for diverse data formats

Horizontal Scaling

Distributed architecture for high volume

Eventual Consistency

Optimized for availability and performance

The CAP Theorem

The CAP theorem, also known as Brewer's theorem, was first proposed by computer scientist Eric Brewer in 2000 at the PODC symposium. It states that in distributed systems, you can only guarantee two of three properties:

Consistency

All nodes reflect the same data simultaneously

Availability

System remains operational and responsive

Partition Tolerance

Continues operating despite network failures

Elasticsearch prioritizes Availability and Partition Tolerance, accepting eventual consistency for telecom's high-volume, real-time requirements.

The Complete Elastic Stack

Beats

Lightweight data shippers for logs, metrics, and network data

Logstash

Data processing pipeline for transformation and enrichment

Elasticsearch

Core search and analytics engine with distributed storage

Kibana

Visualization and management interface with rich dashboards

Application Performance Monitoring

Elastic APM provides distributed tracing and application monitoring with minimal overhead, essential for telecom service reliability.

1 Service Maps

Visualize service dependencies and communication patterns

2 Performance Metrics

Track response times, throughput, and error rates

3 Distributed Tracing

Follow requests across microservices architecture

Lucene vs Solr vs Elasticsearch

Apache Lucene

Core Java search library providing IR algorithms. Requires custom development for applications. Foundation for both Solr and Elasticsearch.

Apache Solr

Enterprise search platform with faceted search and REST APIs. Rich features but requires manual cluster setup and administration.

Elasticsearch

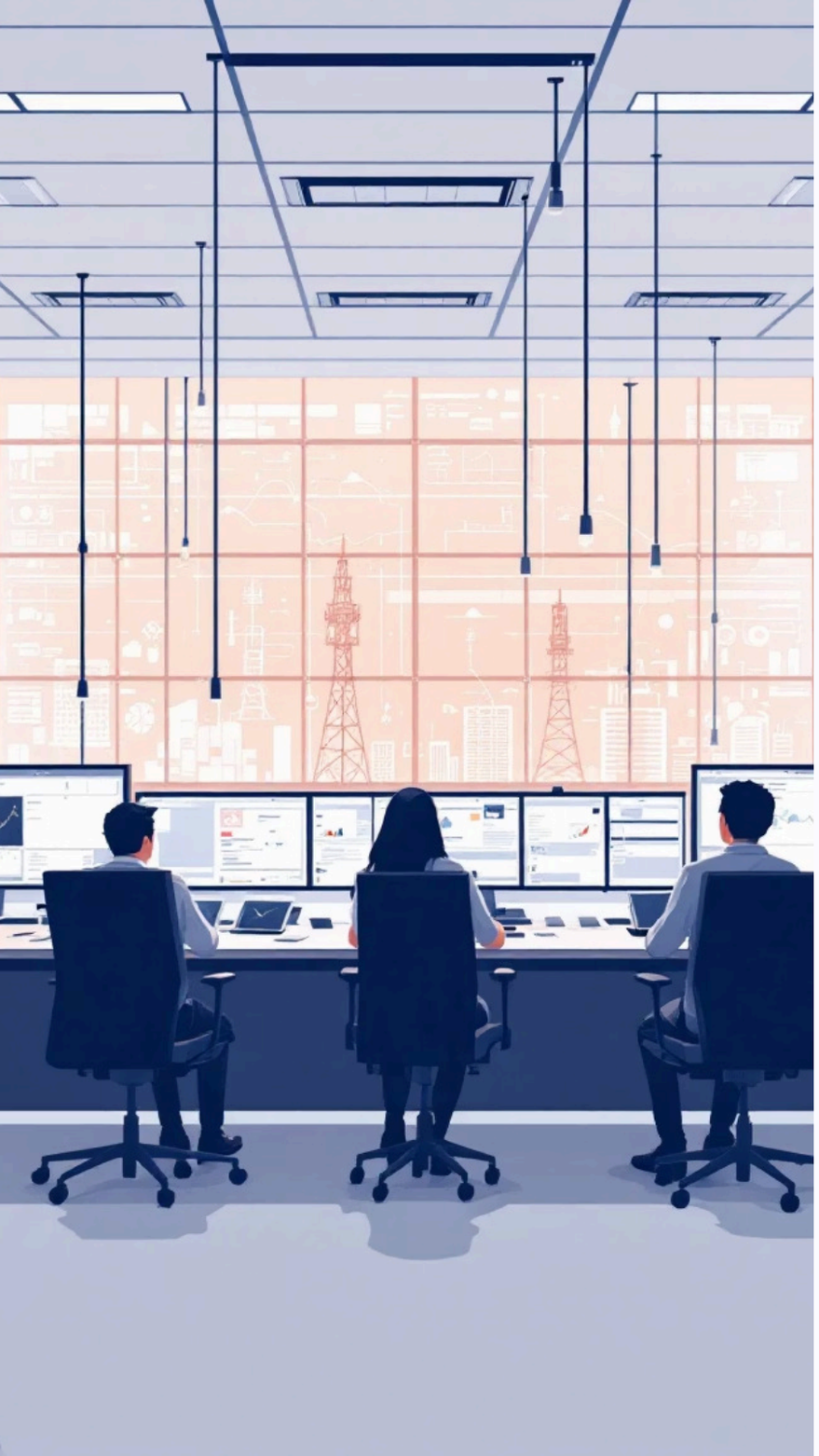
Distributed, RESTful engine with dynamic schemas. Cloud-native design with built-in scaling and ecosystem integration.

Solr vs. Elasticsearch: A Comparison

While both Solr and Elasticsearch are powerful search engines built on Apache Lucene, they have evolved with different philosophies and cater to distinct use cases.

Category	Solr	Elasticsearch
Architecture	Master-slave architecture, often relying on Apache ZooKeeper for cluster management.	Distributed peer-to-peer architecture, self-managing cluster, built-in coordination.
Features	Mature, rich full-text search, advanced faceted search, powerful text analysis, geospatial search.	Real-time search, complex analytics, native integration with Kibana for visualization, APM.
Benefits	Highly configurable, robust, well-established community, strong for traditional search applications.	Easier to set up and scale, cloud-native design, strong ecosystem for logging and observability.
Limitations	More manual setup and administration for distributed environments, less integrated for log analytics.	Can be resource-intensive, eventual consistency by design, less mature for specific text analysis nuances.

Elasticsearch's distributed nature and integrated stack make it ideal for real-time analytics and observability in modern telecom environments.



Why Elasticsearch for Telecom?

Real-time Processing

Handle massive volumes of CDRs, network events, and log data with sub-second query response

Horizontal Scaling

Seamlessly scale across clusters to accommodate growing data volumes and user demands

Schema Flexibility

Adapt to diverse telecom data formats without rigid schema constraints or migrations

Thank you!