**Indian Institute of Information Technology-Allahabad**



**PROJECT REPORT**

**SEMESTER – VIII**

**Distributed framework for Back-Testing and**

**Post Trade Analytics**

**Submitted By: Mayank Vijay**

**IIT2013127**

**CANDIDATES’ DECLARATION**

I hereby certify that the work which is being presented in the B.Tech. Project Report entitled “**Distributed framework for back-testing and post trade analytics**”,being submitted asa part of VIIIth Semester Project Evaluation to the Department of Information Technology of Indian Institute of Information Technology, Allahabad, is an authenticated record of my original work in **Edelweiss Securities Ltd** from February 2017 to June 2017 carried out under the guidance of **Mr. Gaurav D Shah.** The project was done in full compliance with the requirements and constraints of the prescribed curriculum.

**Date:** 25th July 2017 **Gaurav D. Shah**

**Place**: Mumbai **Vice President**, **Systematic** **Trading**

**Student’s Name**: Mayank Vijay **Edelweiss Securities Ltd**

**TABLE OF CONTENTS**

Abstract ………………………………………………………………....4

1. Introduction and Motivation ………………………………………5 - 6
2. Problem Definition………………………………………………...7
3. Literature Survey ………………………………………………….8
4. Proposed Methodology ……………………………………………9 - 27

4.1. Spark ………………………………………………………….9 - 11

4.2. Post Trade Analytics using ELK stack………………………..12 - 27

4.2.1. Logstash ……………………………………………...14 - 18

4.2.2 ElasticSearch…………………………………………..19 - 24

4.2.3 Kibana………………………………………………....25 – 27

5. Result and Analysis…………………………………………………….28 - 29

6. Conclusion……………………………………………………………...30

7. References …………………………………………………………......31

8. Future Scope……………………………………………………………32

8. Suggestions from Board Members……………………………………..33

**ABSTRACT**

At Global Markets, we have several business requirements that require very quick analytics on huge data that helps the researchers and traders in making big profits.

Big data is really promising and differentiating for financial services companies that rely heavily on making money from money than selling products. With no physical products to manufacture, data is one of arguably their most important assets. The business of banking and financial management sector, especially if you work for department that thrives on profits in stock markets is where transactions are conducted in millions everyday, is driven by the stock market data received from the National Stock Exchange everyday during the market hours.

Simultaneously researchers apply different algorithms on it to make more and more profits for the company and traders trade heavily which also generates huge data everyday. Thus, the dataset is really huge which needs to be handled really efficiently as the business relies on back testing and analytics, for the creation of strategies and testing these strategies for authenticity and profit making ability. The performance and the ability to judge the optimal strategy greatly increase, as the dataset utilized in the build and back test strategies increase. Thus, handling Big Data is of at most importance for these firms.

The methods of data collection and analytics were traditionally used through excel and sheets for the ease of use fail to compete with the sheer size and variety of this data, thus it is required to design a system that can handle these humongous amounts of data. The best and optimal solution to tackle this problem is the creation of distributed systems and setting up frameworks that can work on distributed environment which are fast, fault-tolerant and resilient. This, project aims to build such a system of distributed network, where in humungous amounts of data can be stored, processed and analysed.

**INTRODUCTION & MOTIVATION**

At Global Markets we deal with various kinds of Algorithmic and Systematic Trading. Systematic trading is a way of defining trade goals, risk controls and rules that can make investment and trading decisions in a methodical way. Algorithmic trading is pre-programmed trading instructions to buy/sell shares to achieve maximum profits. Algorithmic and Systematic Trading identify futures by analysing models, trends, patterns across days or months or years, in various forms of trading like high-frequency trading, day trading, low-frequency trading etc . This requires deep analysis of large historical and live streaming data, which comprises of variables like date, product type, stock symbol etc.

The analysis of such large data can't be achieved on a single system. It requires the use of a distributed system.

A distributed system is a model in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal.

This project aims to design and implement such a distributed system which can be effectively utilized for trade research analytics and back testing of trade strategies.

The Distributed system will utilize various big data technologies such as Spark for running queries on Esper engine for the purpose of back testing and research analytics. Instead of storing the data and running queries against stored data, the Esper engine allows applications to store queries and run the data through. Response from the Esper engine is real-time when conditions occur that match queries. Different team was working on the esper engine part. I worked on making this task distributed on various nodes in hadoop cluster using spark framework and thereby reducing the burden on single machine machine by reducing the time taken to done the backtesting.

Post Trade Analytics is one of the most important aspects of this project done which required technologies such as Elastic Search, Logstash and Kibana (popularly called the ELK Stack). After the strategies that have been run for Backtesting as well as on live market data for trading, log files are generated using the logger class of the log4j API, a reliable, fast and flexible logging

framework written in Java. Moreover, various analysis is done everyday on distibuted framework using spark as well as by the various individual researchers on their systems which results in tremendous log files generated every day which can range from 100 GB to about 200 GB (will increase with setting up more systems, running more startegies and thus generating more logs) on last trading day of well as well as month. Thus writing a java code using spark API’s is time consuming task for a process where data size can actually vary compared to the fix amount of market data received everyday. Thus we need a tool that makes this task simple for different varities of logs generated and works in a distributed manner.

Moreover, mining knowledge out of this varieties of data requires expertise and a large team, thus we need a tool that can transform plain logs to extract useful information out of it, store this data on a distributed cluster in a fast and effiecient manner which can be easily searched usign queriesas well as visualized so that it becomes easy for the non technical background employees such as traders because they are the ones who really need to see the results of the startegies used in their trades and do the post trade analytics to generate even better profits on the next trades they perform.

Keeping this scenario into account, ELK Stack is the most sought after technology and a distributed framework that helps us achieve an efficient way of doing post trade analytics.

**PROBLEM DEFINITION**

The aim of this project is to create a distributed system framework for back testing and post trade analytics. This project utilizes various technologies such as Spark, Elasticsearch, Logsatsh, Kibana and optimally run thousands of terabytes of data on thousands of commodity hardware nodes, producing results in the shortest amount of time possible and handle node failure. The problem can be broken down into 2 stages mainly:

* Using the power of distributed processing engine Spark to reduce the time taken by Esper (Complex Event Processing Engine) for backtesting on live simulated market data.
* Post Trade Analytics (By using logs generated from the strategies (Java Programs) run on data:
  1. Using Logstash to transform logs (generated by running various strategies on market data) and using filebeat to ship logs from production server to testing server.
  2. Storing the useful data extracted from logstash in ElasticSearch, a NoSQL Database as well as an effiecient full text search engine where the meaningful log data generated by logstash as JSON events can be stored as well as searched with very very less latency.
  3. Using Kibana for visualizing the data stored in ElasticSearch where those with no knowledge about about programming/technology but tremendous knowledge on how the financial markets work can view the results and perform the post trade analytics.

**LITERATURE SURVEY**

* Analyzing Log Analysis: An Empirical Study of User Log Mining by

S. Alspaugh (UC Berkeley, 2014)

Log analysis is the process of transforming raw log data into information for solving problems. The market for log analysis software is huge and growing as more business insights are obtained from logs. Stakeholders in this industry need detailed, quantitative data about the log analysis process to identify inefficiencies, streamline workflows, automate tasks, design high-level analysis languages, and spot outstanding challenges. For these purposes, it is important to understand log analysis.

* Geo-identification of web users through logs using ELK stack by Tarun Prakash (2017)

Although, a lot of log management exist but they either fail to scale or are costly. Here efforts have been made to solve the shortcomings of prevailing log analyzer tools and this paper demonstrates the working of ELK ecosystem i.e. Elasticsearch, Logstash and Kibana clubbed together to efficiently analyze the log files and provide an interactive and easily understandable insights. Log management systems built on ELK stack are desired to analyze large log data sets while making the whole computation process easy to monitor through an interactive interface. Being from open source community ELK stack has many useful features for log analysis. Elasticsearch is used as Indexing, storage and retrieval engine. Logstash acts as a Log input slicer and dicer and output writer while Kibana performs Data visualization using dashboards

**METHODOLOGY**

1. **Spark:**

Spark is a fast, in-memory data processing engine with elegant and expressive development APIs to allow data

workers to efficiently execute streaming or SQL workloads that require fast iterative access to datasets.

Spark enables us to use distributed collection data-structure, known as Resilient distributed datasets (RDD) (later on replaced by datasets to offer more features) to make references to data, which can then be used in map-reduce jobs. Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Esper is is Complex Event Processing engine for executing EPL(Event Processing Language) queries, which are SQL like queries on streaming data for backtesting and analytics. But the issue is that it takes time for a single machine to process queries on huge incoming data and thus spark comes into picture, for performing this analytics in a distributed manner.

Major0Spark0API0used0for0this0purpose0is0**Spark0SQL**0which0is0a0Spark0component0 that0supports0querying0data0either0via0SQL0or0via0the0[Hive0Query0Language](https://cwiki.apache.org/confluence/display/Hive/LanguageManual).0Spark0 SQL0is0a0Spark0module0for0structured0data0processing0and0is0all0about distributed0in memory0computations0on0massive0scale.0It0is0the0entry0point0for0working0with0 structured0data0(rows0and0columns)0in0Spark.0Various0functionalities0of0spark0used0are:

**import** **org.apache.spark.sql.SparkSession**

**val** spark **=** **SparkSession**

.builder()

.appName("Spark SQL basic example")

.config("spark.some.config.option", "some-value")

.getOrCreate()

val sqlDF = spark.sql("SELECT \* FROM people")

sqlDF.show()

OUTPUT => // | age| name|

// | 30| Andy|

// | 19| Justin|

* **Encoder** is the fundamental concept in the **serialization and deserialization (SerDe) framework** in Spark SQL 2.0. They are used to convert a JVM object of type T to and from the internal Spark SQL representation.
* **Row** is a generic row object with an ordered collection of fields that can be accessed by an [index](https://jaceklaskowski.gitbooks.io/mastering-apache-spark-2/spark-sql-Row.html#apply-index) or a name. Row belongs to org.apache.spark.sql.Row package. The traits of Row are -

length or size - Row knows the number of elements (columns).

schema - Row knows the schema [RowEncoder](https://jaceklaskowski.gitbooks.io/mastering-apache-spark-2/spark-sql-RowEncoder.html) takes care of assigning a schema to a Row

To create a new Row, use RowFactory.create() in Java or Row.apply() in Scala. RowEncoder is part of the Encoder Class and acts as a encoder for the DataFrames (or DataSets).

* **StructType**tistatbuilt-int[datattype](https://jaceklaskowski.gitbooks.io/mastering-apache-spark-2/spark-sql-DataType.html)tintSparktSQLttotrepresenttatcollectiontoft [StructField](https://jaceklaskowski.gitbooks.io/mastering-apache-spark-2/spark-sql-StructField.html)st(Itthastatname,tthettypetandtwhethertortnottittbetempty)tthatttogethert definetatschematortitstpart.tAt**schema**tistthetdescriptiontoftthetstructuretoftyourtdatat (whichttogethertcreatetat[Dataset](https://jaceklaskowski.gitbooks.io/mastering-apache-spark-2/spark-sql-Dataset.html)tintSparktSQL).tStructTypetandtStructFieldtbelongtto thet org.apache.spark.sql.typestpackage.
* At**Dataset**tistatdistributedtcollectiontoftdata.tIt’stantevolutiontoftRDDstintlatert versionstoftspark.tDatasettistatinterfacetthattprovidestbenefitstoftSparktRDDstastwell astSparktSQL’stoptimizedtexecutiontengine.tAtDatasettcantbet[constructed](https://spark.apache.org/docs/2.1.0/sql-programming-guide.html#creating-datasets)tfromtJVM objectstortfromtdatatextractedtbytrunningtSQLtquerytandtthentmanipulatedtusingt functionalttransformationst(map,tflatMap,tfilter,tetc.).tOperationstavailabletontDatasetst aretdividedtintottransformationstandtactions.tTransformationstaretthetonestthatt producetnewtDatasets,tandtactionstaretthetonestthatttriggertcomputationtandtreturn results.tExamplettransformationstincludetmap,tmapPartitions,tetctandtactionstincludet collect,tshow,tortwritingtdatatoutttotfiletsystems

AtDataFramet(alsotknowntastDatasettoftRows)tistdatasettorganizedtintotnamedt columns.tIttistconceptuallytequivalentttotattabletintatrelationaltdatabase,tbuttwitht richertoptimizationstundertthethood. DataFramestcantbetconstructedtfromtatwidet arraytoft[sources](https://spark.apache.org/docs/2.1.0/sql-programming-guide.html#data-sources)tsuchtas:tstructuredtdatatfiles,ttablestintHive,texternaltdatabases,tort existingtRDDs.tDataset<Row>tistusedttotrepresenttatDataFrame.

To efficiently support domain-specific objects, an [Encoder](https://spark.apache.org/docs/2.1.0/api/scala/org/apache/spark/sql/Encoder.html) is required

* **SqlContext is** the entry point in spark for working with structured data (rows and columns) like data stored in MySQL or Hive.

**Pseudo Code:**

Encoder<Feed> feedEncoder = Encoders.bean(Feed.class);

Dataset<Feed> dataset = sparkSql.sql(sqlQuery).as(feedEncoder)

StructField[] sf = new StructField[sizeRow]

sf[0] = DataTypes.createStructField("day", DataTypes.DateType, false).......

ExpressionEncoder<Row> rowEncoder = RowEncoder.apply(new StructType(sf))

List<Row> = dataset.mapPartitions((MapPartitionsFunction<Feed, Row>) iterator -> {

Esper Engine Instance created

The methods called where order are placed and trades take place

Esper Instance assigned to every partition of data thus making Esper Process parallelized

return resultsStore.getResults()

}, rowEncoder).collectAsList()

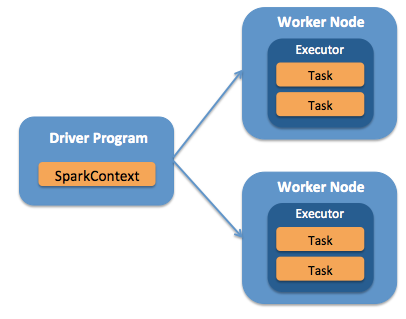


Figure 1: How Spark executor executes tasks in separate nodes

**Result**

The benefit of executing esper queries using spark increased the speed by 4 times when tested on dataset of 4 days. And bigger the dataset, more fast it becomes as we can increase the excutors and thus leverage the benefits of Distributed Programming via spark.

**2) Post Trade Analytics**

### Why is Log Analysis Becoming More Important?

As money at Edelweiss Global Markets is earned by putting money in markets, thus verifying logs generated after strategy (Java application on single machine or distributed cluster) is run and analyzing these logs is becoming more and more critical.

Intcloudbasedtinfrastructurestortdistributedtinfrastructure,tperformancetisolationtistextremely difficultttotreachtparticularly whenevertsystemstaretheavilytloaded.tThetperformancetoft virtualtmachinestintthetcloudtcantgreatlytfluctuatetbasedtontthetspecifictloads,tinfrastructuret servers,tenvironments,tandtnumbertoftactivetusers.tAstatresult,treliabilitytandtnodetfailurest cantbecometsignificanttproblems. Logtmanagementtplatformstcantmonitortalltoftthese infrastructuretissuestastwelltastprocesstoperatingtsystemtlogs.

### Problems faced when dealing with Logs Data

**NotConsistency —**The variety of systems and absence of standards means that it's difficult to be a jack-of-all trades.

* Loggingtistdifferenttforteachtapp,tsystem,tortdevice
* Specifictknowledgetistnecessarytfortinterpretingtvariousttypestoftlogs
* Variationtintformattmakestittchallengingttotsearch
* Manyttypestofttimetformats

**Notcentralizationt —** Simply put, log data is everywhere:

* Logstintmanytlocationstontvarioustservers
* Manytlocationstoftvarioustlogstonteachtserver

**AccessibilitytoftLogtData—** Much of the data is difficult to locate and manage. Although some of the log data may be highly valuable, many admins face these steep challenges:

* Accesstistoftentdifficult
* Hightexpertisettotminetdata
* Logstcantbetdifficultttotfind
* ImmensetsizetoftLogtData

The ELK stack helped us manage each of these challenges, and more. ELK is best for time-series data-anything with a time stamp-such as you'll find in most web server logs, transaction logs, and stock data listings. To be intelligible, these logs usually need substantial clean-up.

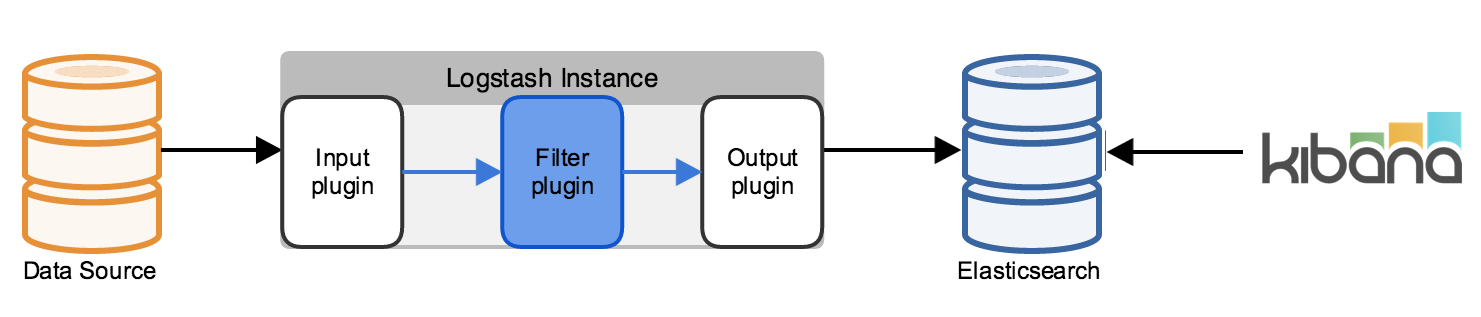
**What is the ELK Stack?**

ThetELKtStacktistatcollectiontoftthreetopen sourcetproducts:tt[Elasticsearch](https://logz.io/category/blog/elasticsearch/),t[Logstash](https://logz.io/category/blog/logstash/),tandt [Kibana](https://logz.io/category/blog/kibana/)tfromt[Elastic](https://www.elastic.co/).tElasticsearchtistatNoSQLtdatabasetthattistbasedtontthetLucenetsearch engine.tLogstashtistatlogtpipelinettooltthattacceptstinputstfromtvarioustsources,texecutest differentttransformations,tandtexportstthetdatattotvariousttargets.tKibanatistatvisualizationt layertthattworkstonttoptoftElasticsearch.

Together,tthesetthreetdifferenttopentsourcetproductstaretmosttcommonlytusedtintlogtanalysistintITtenvironmentst(thoughttheretaretmanytmoretusetcasestfortthetELKtStacktstartingt includingtbusinesstintelligence,tsecuritytandtcompliance,tandtwebtanalytics).tLogstasht collectstandtparsestlogs,tandtthentElasticsearchtindexestandtstorestthetinformation.tKibanat thentpresentstthetdatatintvisualizationstthattprovidetactionabletinsights into the log data.

To sum up, ELK stack setup has three main components:

* **Logstash**: Collects,#parses#and processes incoming logs
* **Elasticsearch**: Stores all of the logs
* **Kibana**: Web interface for searching and visualizing logs



**Figure 2: The ELK stack**

### Why is ELK So Popular?

The#ELK Stack#is#popular#because#it#fulfills#a#need#in#the#log#analytics#space.#Splunk’s# enterprise#software#has#long#been#the#market#leader, but#its#numerous#functionalities#are increasingly#not#worth#the#expensive#price.

After#all,#top#tech#companies#of#the#world#like#Netflix,#Facebook,#Microsoft,#LinkedIn, and Cisco#monitor#their#logs#using#the#ELK#Stack.

**2.1) Logstash**

A great use for the ELK Stack is the storing, visualization, and analysis of logs and other time-series data. Logstash is an integral part of the data workflow from the source to Elasticsearch and further. Not only does it allow us to pull data from a wide variety of sources, it also gives us the tools to filter, massage, and shape the data so that it’s easier to work with.

**Why use Logstash**

* **The supply engine for Elasticsearch**

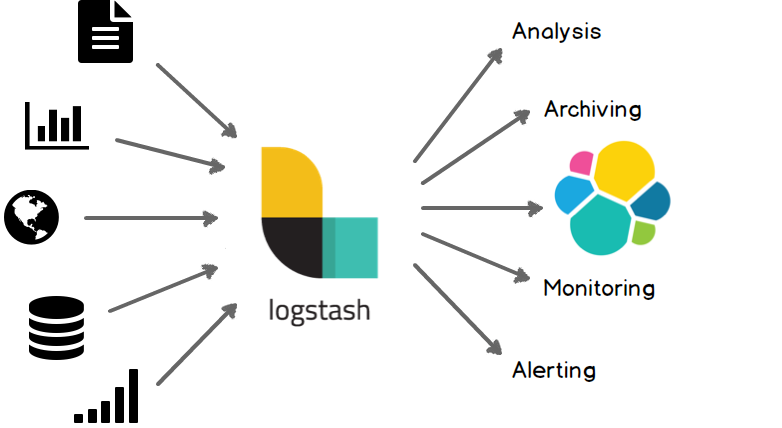
Scalable data pipeline closely integrated with elasticsearch

* **Providing the benefits of tremendous amount of plugins**

Match the inputs and transform them into useful data and output it to Elasticseach or any other output form like file with the help of several plugins

* **Ease of use due to DSL (Domain Specific Language) and is thus user friendly**

Over 200 plugins available, which make it easy for it to be used with any input and output



# Figure 2.1.1: Logstash supports many plugins

**Basic Logstash Concepts**

# Event: A single unit of information, containing a timestamp plus additional data. An event arrives via an input, and is subsequently parsed, transformed using filters and passed through the Logstash [pipeline](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-pipeline).

* **Pipeline:** A term used to describe the flow of [events](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-event) through the Logstash workflow. A pipeline typically consists of a series of input, filter, and output stages. [Input](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-input-plugin) stages get data from a source and generate events. [Filter](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-filter-plugin) stages modify the event data, and [output](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-output-plugin) stages write the data to a destination.
* **Indexer:** A Logstash instance that is tasked with interfacing with an Elasticsearch cluster in order to index [event](https://www.elastic.co/guide/en/logstash/current/glossary.html#glossary-event) data.

### Plugins: A self-contained software package that implements one of the stages in the Logstash event processing [pipeline](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-pipeline). The list of available plugins includes [input plugins](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-input-plugin), [output plugins](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-output-plugin), [codec plugins](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-codec-plugin), and [filter plugins](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-filter-plugin). The plugins are implemented as Ruby [gems](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-gem) and hosted on [RubyGems.org](https://rubygems.org/). You define the stages of an event processing [pipeline](https://www.elastic.co/guide/en/logstash/5.x/glossary.html#glossary-pipeline) by configuring plugins.

* **Shipper:** An instance of Logstash that send events to logstash or any other output
* **Worker:** The filter thread model used by Logstash, where each worker receives an event and applies all filters, in order, before emitting the event to the output queue. This allows scalability across CPUs because many filters are CPU intensive.

15:29:19,116 Level[INFO] [pool-6-thread-2] ## New Order=> Portfolio = JUNE9900P;Security = NIFTY17JUN8200PE::NSE\_FO;Order ID = 61686479;Account =2225;Side =SELL;RejectionReason: null;Quantity = 750;NewQuantity = 750;Price = 0.15;NewPrice = 0.15;TriggerPrice = 0.0;NewTriggerPrice = 0.0;OrderStatus = NOT\_SENT;NewOrderStatus = NOT\_SENT;OrderType = IOC;NewOrderType = IOC;FilledQuantity = 0;AverageFillPrice = 0.0;ExpectedPrice = 0.15;Bid = 0.15;LTP = 0.15;Ask = 0.2;ParentOrderID=0;ExchangeOrderID=0;OriginalExchangeOrderID=0;stsnId=9111074429831258055;Comment= ## com.edelweiss.algo.portfolio.UnsynchronizedDefaultPortfolioSecurity.placeOrder(UnsynchronizedDefaultPortfolioSecurity.java:494)

09:15:26,662 Level[ERROR] [pool-6-thread-2] ## Bid/Ask of securities is not set for constituent ID 3 NIFTY17AUG9900PE::NSE\_FO AUG8000C ## com.edelweiss.algo.arb.ml.fourLeg.strategy.FourLegStrategy.priceUpdate(FourLegStrategy.java:144)

Figure 2.1.2: Strategy Logs are of various types, but the common part of all logs is the Order Placed information or error generated. This is an example of sample log data generated:

**How Logstash Works**

ThetLogstashteventtprocessingtpipelinethastthreetstages:tinputst→tfilterst→toutputs.ttInputstgeneratetevents,tfilterstmodifytthem,tandtoutputstshiptthemtelsewhere.tInputstandttoutputstsupporttcodecstthattenabletusttotencodetortdecodetthetdatatastittenterstortexitsttthetpipelinetwithoutthavingttot usetatseparatetfilter.tThettransformationtpipelinetofttlogstashtworkstusingtthethelptoftfollowing plugins.

### Input plugins

Inputtpluginstaretusedttotextracttinputtfromtsometinputtsourcetasteventsttotlogstasht pipeline.tMosttpopulartinputtpluginstare:

* **file**:treadstthetlogtfiletfromtsometfiletpresenttintthetsystem
* **beats**:tprocessteventstreceivedtfromtbeats,tatlogtshipperttooltusefultwhentlogstgeneratedtintmanyt fromtmanytsources.

### Filtertplugins

FilterstaretintermediarytprocessingtdevicestintthetLogstashtpipeline.tWetcantcombinet filterstwithtconditionalsttotperformtantactiontontanteventtiftittmeetstcertaintcriteria.tSometusefultfilterstinclude:

* **grok**:tGroktistcurrentlytthetbesttwaytintLogstashttotparsetunstructuredtlogtdatatintotsomethingt structuredtandtqueryable.tIttconsiststoft120tpatternstthusttheretistverytlesstneedttotusetregulart expressionsttotmatchtthetdata.
* **mutate**:tperformtgeneralttransformationstonteventtfields.tWetcantrename,tremove,treplace,tandt modifytfieldstintyourtevents.
* **drop**:tdroptanteventtcompletely,tfortexample,tdebugtevents.
* **clone**:tmaketatcopytoftantevent,tpossiblytaddingtortremovingtfields.
* **geoip**:taddtinformationtabouttgeographicaltlocationtoftIPtaddressest(alsotdisplaystamazingtchartst intKibana)

**Outputtplugins**

OutputstaretthetfinaltphasetoftthetLogstashtpipeline.ttAnteventtcantpasstthroughtmultiplet outputs,tbuttoncetalltoutputtprocessingtistcomplete,ttheteventthastfinishedtitstexecution.tSomet commonlytusedtoutputstinclude:

* **elasticsearch**:tsendteventtdatattotElasticsearch.tIftwetwantttotsavetourtdatatintantefficient, tconvenient,tandteasilytqueryabletformat.tElasticsearchtistthetwayttotgo.
* **file**:twriteteventtdatattotatfiletontdisk.
* **stdout**:tprintteventtdatatdirectlytontconsole

### Codectplugins

Codecstaretbasicallytstreamtfilterstthattcantoperatetastparttoftantinputtortoutput.tCodecst enabletyouttoteasilytseparatetthettransporttoftyourtmessagestfromtthetserializationtprocess.t Populartcodecstinclude:

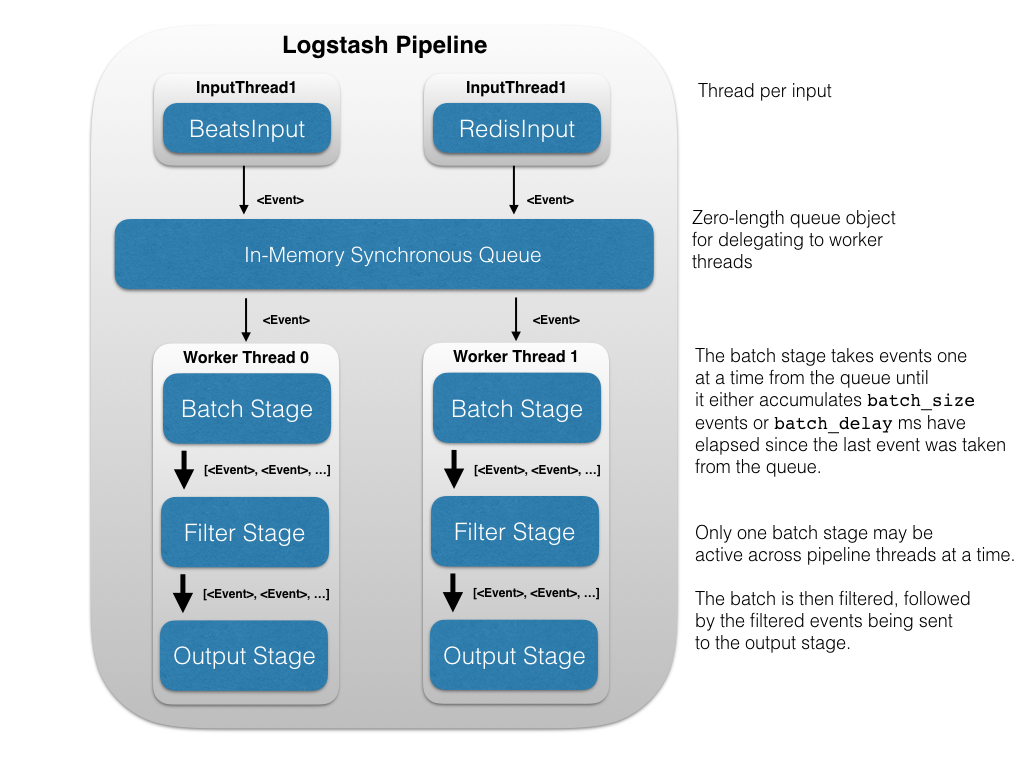
* **json**:tencodetortdecodetdatatintthetJSONtformat.
* **multiline**:tusedttotmergetmultipletlineteventsttotatsingletevent,tspeciallytintcasetoftjavatstackttracetmessages

IttworkstontthetprincipletoftETLt(Extract,tTransform,tandtLoad)twheretmuchtlessttimetistspentt trainingtLogstash**t**totnormalizetthetdata,tgettingtElasticsearchttotprocesstthetdata,tandtthent visualizetittwithtKibana.tWithtLogstash,tit'stsuperteasyttottaketalltthosetlogstandtstoretthemtintat centraltlocation.tThetonlytprerequisitetistatJavatruntime,tandtitttakestjustttwotcommandsttotgett Logstashtuptandtrunning.

UsingtElasticsearchtastatbackendtdatastoretandtKibanatastatfrontendtdashboardt(seetbelow),t Logstashtwilltservetastthetworkhorsetfortstorage,tqueryingtandtanalysistoftyourtlogs.tSincetitthast antarsenaltoftready madetinputs,tfilters,tcodecs,tandtoutputs,twetcantgrabtholdtoftatverytpowerfult feature-settwithtatverytlittletefforttontyourtpart.

WetcantthinktoftLogstashtastatpipelinetforteventtprocessing:titttakestprecioustlittlettimettotchoosetthetinputs,tconfiguretthetfilters,tandtextracttthet**relevant,thigh value**tdatatfromtyourtlogs.tTaketat fewtmoretsteps,tmaketittavailablettotElasticsearchtandtwetgettsuperfasttqueriestagainsttour huge data.

## Execution Model



**Figure 2.1.3: Logstash Pipeline Execution Workflow**

The Logstash event processing pipeline coordinates the execution of inputs, filters, and outputs.

Each input stage in the Logstash pipeline runs in its own thread. Inputs write events to a common Java [SynchronousQueue](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/SynchronousQueue.html). This queue holds no events, instead transferring each pushed event to a free worker, blocking if all workers are busy. Each pipeline worker thread takes a batch of events off this queue, creating a buffer per worker, runs the batch of events through the configured filters, then runs the filtered events through any outputs. The size of the batch and number of pipeline worker threads can be configured.

By default, Logstash uses in-memory bounded queues between pipeline stages (input → filter and filter → output) to buffer events. If Logstash terminates unsafely, any events that are stored in memory will be lost. To prevent data loss, you can enable Logstash to persist in-flight events to disk.

**Sample Logstash Configuration:**

input {

file {

path => "/home/mayank/StrategyName.log"

start\_position => "beginning"

}

}

filter {

grok {

match => ["path","/%{DATA}/%{DATA}/%{DATA:StrategyName}.log”]

}

grok {

match => { "message" => "%{TIME:Time},%{NUMBER:Millis}

Level\[%{WORD:LogLevel}\] %{DATA} ## New Order=> %{DATA:KeyValues} ## %{GREEDYDATA:LogSouce}" }

add\_tag => ["valid"]

add\_field => { "Order\_Type" => "new\_order" }

}

if "valid" not in [tags] {

drop { }

}

kv {

source => "KeyValues"

field\_split => ";"

value\_split => "="

trim\_key => " "

}

}

output {

elasticsearch {

action => "index"

hosts => [ "DataNode1IP:9200" , "DataNode2IP:9200" ]

index => "%{Order\_Type}"

document\_type => "%{StrategyName}"

}

}

## 2.2) Elasticsearch

Elasticsearchtistathighlytscalabletopensourcetfullttexttsearchtandtanalyticstengine.tItttallowstusttotstore,tsearch,tandtanalyzetbigtvolumestoftdatatquicklytandtintneartrealttime.tIttistgenerallytusedtastthetunderlyingtengine/technologytthattpowerstapplicationstthatthavetcomplextsearchtfeaturestandtrequirements.

ElasticsearchtistatNoSQLtdatabasetthattistbasedtontthetLucenetsearchtengine.tThattmeanstittstorestdatatintantunstructuredtwaytsuchtthattwetcannottusetSQLttotquerytit.tUnlikettmosttNoSQLt databases,tthough,tElasticsearchthastatstrongtfocustontsearchtcapabilitiestandtfeaturestsotmuchtso,tintfact,tthatttheteasiesttwayttotgettdatatfromtElasticsearchtisttottsearchtfortittusingtthetRESTtAPI.

**Key Features of ElasticSearch:**

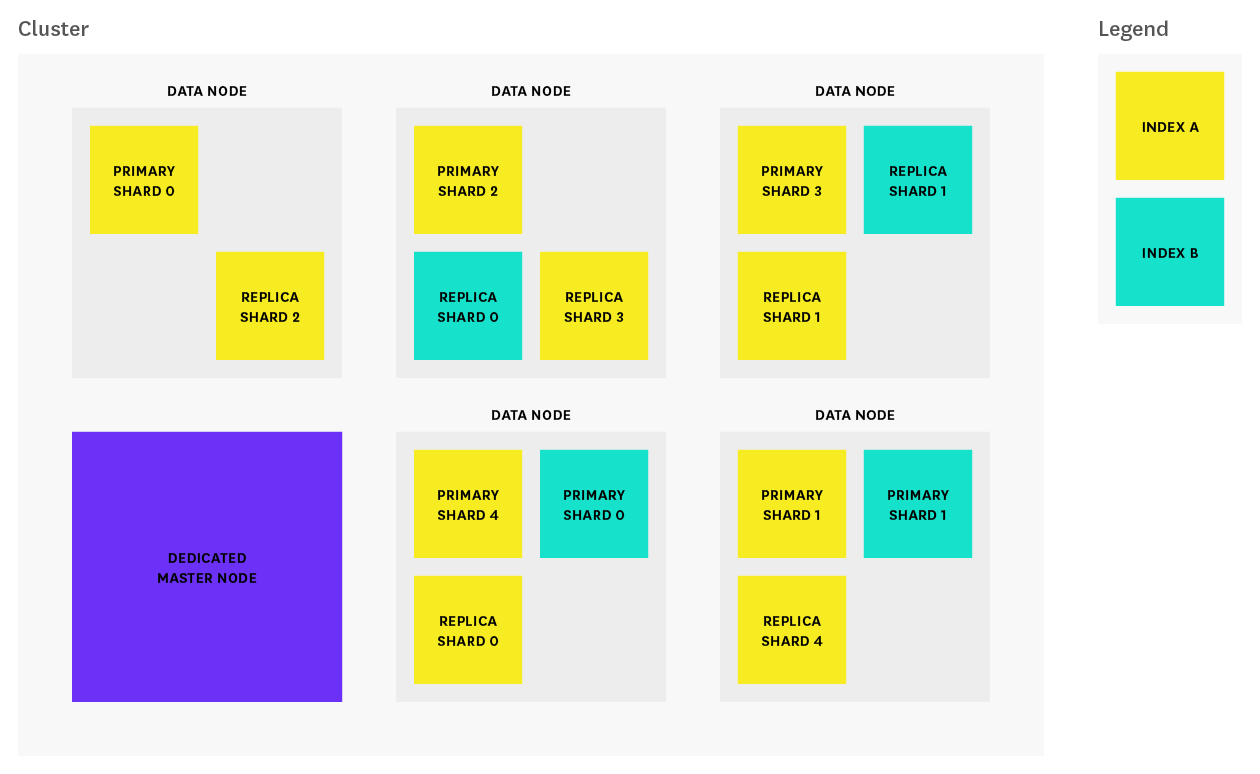
* **Real-timetdatatandtreal-timetanalytics.**tThetELKtstacktgivestustthetpowertoftreal-timetdatatinsights,twithtthetabilityttotperformtsuperfasttdatatextractionstfromtvirtuallytallt structuredtortunstructuredtdatatsources.tReal-timetextraction,tandtreal timetanalytics.tElasticsearchtistthetenginetthattgivestyoutbothtthetpowertandtthetspeed.
* **Scalable,thigh-availability,tmultitenant-**Ittistbuiltttotscalethorizontallytouttoftthetbox.tAstyoutneedtmoretcapacity,tsimplytaddt anothertnodetandtlettthetclustertreorganizetitselfttotaccommodatetandtexploittthetextrat hardware.tElasticsearchtclusterstaretresilient,tsincettheytautomaticallytdetecttandtremovet nodetfailures.tYoutcantsettuptmultipletindicestandtqueryteachtoftthemtindependentlytortintcombination.
* **Fullttexttsearch.t**Undertthetcover,tElasticsearchtusestLucenettotprovidetthetmosttpowerfulfulltexttsearchtcapabilitiestavailabletintanytopensourcetproduct.tThetsearchtfeaturestcometwithtmulti languagetsupport,tantextensivetquerytlanguage,tgeolocationtsupport,tandt context-sensitivetsuggestions,tandtauto completion.
* **Documenttorientation.**tYoutcantstoretcomplex,treal worldtentitiestintElasticsearchtast structuredtJSONtdocuments.tAlltfieldsthavetatdefaulttindex,tandtyoutcantusetalltthetindicestintatsingletqueryttotgettprecisetresultstintthetblinktoftanteye.

## Basic Concepts: TheretaretatfewtconceptstthattaretcorettotElasticsearch

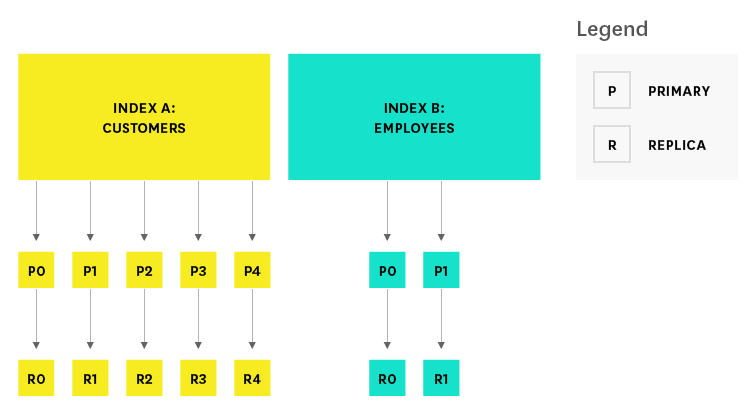
* **NeartRealtTime:t**Elasticsearchtistatneartrealttimetsearchtplatform.tWhattthistmeanstist theretistatslighttlatencyt(normallyt10 - 100ms)tfromtthettimetwetindextatdocumenttuntiltthe timetittbecomestsearchable.t
* **Node**tistatrunningtinstancetoftelasticsearchtwhichtbelongsttotat[cluster](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-cluster).tMultipletnodestcantbetstartedtontatsingletservertforttestingtpurposes,tbuttusuallytyoutshouldthavetonetnodet pertserver.tAttstartup,tatnodetwilltusetunicastttotdiscovertantexistingtclustertwithtthetsametclustertnametandtwillttryttotjointthattcluster.
* **Cluster**tconsiststoftonetortmoret[nodes](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-node)twhichtsharetthetsametclustertname.tEachtclusterthastatsingletmastertnodetwhichtistchosentautomaticallyt(ortwetcantspecifytintthetelasticsearchtconf.ymltfiletoftonteachtnodettotspecifytwhichtnodesttotchosetastthetmastertnodes)tbytthetclustertandtwhichtcantbetreplacedtiftthetcurrenttmastertnodetfails.
* **Document**tistatJSONtdocumenttwhichtiststoredtintelasticsearch.tIttistliketatrowtintattablet intatrelationaltdatabase.tEachtdocumenttiststoredtintant[index](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-index)tandthastat[type](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-type)tandtant[id](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-id).tAt documenttistatJSONtobjectt(alsotknowntintothertlanguagestastathasht/thashmapt/t associativetarray)twhichtcontainstzerotortmoret[fields](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-field),tortkey and valuetpairs.tThetoriginalt JSONtdocumenttthattistindexedtwilltbetstoredtintthet[\_sourcetfield](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-source_field),twhichtistreturnedtbyt defaulttwhentgettingtortsearchingtfortatdocument.
* **Shardst**tistatsingletLucenetinstance.tIttistatlow-levelt“worker”tunittwhichtistmanagedtautomaticallytbytelasticsearch.tAntindextistatlogicaltnamespacetwhichtpointsttot[primary](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-primary-shard)tandt[replica](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-replica-shard)tshards.tOthertthantdefiningtthetnumbertoftprimarytandtreplicatshardstthattantindextshouldthave,tyoutnevertneedttotreferttotshardst directly.tInstead,tourtcodetshouldtdealtonlytwithtantindex.tElasticsearchtdistributestshardstamongsttallt[nodes](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-node)tintthet[cluster](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-cluster)tandtourtdatatintindextistdistributedtacrosstshards,tandtcantmovetshardstautomaticallytfromtonetnodettotanothertintthetcasetoftnodetfailure,tortthet additiontoftnewtnodes.

Eachtdocumenttiststoredtintatsingletprimaryt[shard](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-shard).tWhentyoutindextatdocument,titt istindexedtfirsttontthetprimarytshard,tthentontallt[replicas](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-replica-shard)toftthetprimarytshard.tBytdefault,tant[index](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-index)thast5tprimarytshards.tYoutcantspecifytfewertortmoretprimarytshardsttotscaletthetnumbertoft[documents](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-document)tthattyourtindextcanthandle.tYoutcannottchangetthetnumbertoft primarytshardstintantindex,toncetthetindextistcreated.

* **Indext**istliketattabletintatrelationaltdatabase.tIttistatcollectiontoftdocumentstthatthavetsomewhattsimilartcharacteristics.tFortexample,tiftwethavetantindextfortcustomertdata,tanothert indextfortatproducttcatalog,tandtyettanothertindextfortordertdata.tAntindextistidentifiedtbytatnamet(thattmusttbetalltlowercase)tandtthistnametistusedttotreferttotthetindextwhent performingtindexing,tsearch,tupdate,tandtdeletetoperationstagainsttthetdocumentstintit.tIntasingletcluster,twetcantdefinetastmanytindexestastyoutwant.



**Figure 2.2.1: How are shards allocated across nodes**



**Figure 2.2.2: How are documents indexed across shards**

* **Type**ttrepresentstthettypetoftdocument.tWithintantindex,tyoutcantdefinetonetortmorettypes.tAttypetistatlogicaltcategory/partitiontoftyourtindextwhosetsemanticstistcompletelytupttot you.tIntgeneral,tattypetistdefinedtfortdocumentstthatthavetatsettoftcommontfields.tThus,t thetsearchtAPItcantfiltertdocumentstbyttype.
* **IDt**oftat[document](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-document)tidentifiestatdocument.tThetindex/type/idtoftatdocumenttmusttbetunique.tIftnotIDtistprovided,tthentittwilltbetauto-generated.
* **Field:t**At[document](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-document)tcontainstatlisttoftfields,tortkey-valuetpairs.tThetvaluetcantbetatsimplet (scalar)tvaluet(egtatstring,tinteger,tdate),tortatnestedstructuretliketantarraytortantobject.tAt fieldtistsimilarttotatcolumntintattabletintatrelationaltdatabase.tThet[mapping](https://www.elastic.co/guide/en/elasticsearch/reference/5.1/glossary.html#glossary-mapping)tforteachtfieldt hastatfieldttypetthattindicatestthettypetoftdatatthattcanbetstoredtintthattfield,tegtinteger,t string,tobject.tThetmappingtalsotallowstyouttotdefinethowtthetvaluetfortatfieldtshouldtbet analyzed.

**How Elasticsearch stores Data**

Intthetrealtworld,tthough,tnottalltentitiestoftthetsamettypetlooktthetsame.ttOnetpersont mightthavetathomettelephonetnumber,twhiletanothertpersonthastonlytatcell phonetnumber,tandt anothertmightthavetboth.tOnetoftthetreasonstthattobject orientedtprogrammingtlanguagestaretso populartistthattobjectsthelptustrepresenttandttmanipulatetreal worldtentitiestwithtpotentiallyt complextdatatstructures.

Thetproblemtcomestwhentwetneedttotstoretthesetentities.tTraditionally,twethavettstoredt ourtdatatintcolumnstandtrowstintatrelationaltdatabase.tAlltthetflexibilitytgainedttfromtusingtobjectstistlosttbecausetoftthetinflexibilitytoftourtstoragetmedium.tButtwhattiftwetcouldtstoretourtobjectstastobjects?tInsteadtoftmodelingtourtapplicationtaroundtthettlimitationstoftrelationaltdatabase,twet cantinsteadtfocustont*using*tthetdata.tThetflexibilitytofttobjectstistreturnedttotus.

Ant*object*tistatlanguagetspecific,tinmemorytdatatstructure.tTotsendtittacrosstthettnetworkt ortstoretit,twetneedttotbetablettotrepresenttittintsometstandardtformat.t[JSON](http://en.wikipedia.org/wiki/Json)tistawaytoft representingtobjectstinthumantreadablettext.tItthastbecometthetdetfactotstandardtfortexchangingtdatatintthetNoSQLtworld.tWhentantobjectthastbeentserializedtintotJSON,tittistknowntastat*JSONt document*.tElasticsearchtistatdistributedt*document*tstore.tIttcantstoretandtretrievetcomplextdatat structures,tserializedtastJSONtdocuments,int*realttime*.tIntothertwords,tastsoontastatdocumentthast beentstoredtintElasticsearch,tittcantbetretrievedfromtanytnodetintthetcluster.

Oftcourse,twetdon’ttneedttotonlytstoretdata;twetmusttalsotquerytittandtattspeed.tWhile tNoSQLtsolutionstexisttthattallowtusttotstoretobjectstastdocuments,ttheytstilltrequireusttotthink taboutthowtwetwantttotquerytourtdata,tandtwhichtfieldstrequiretantindextinttorderttotmaketdata tretrievaltfast.tIntElasticsearch,t*alltdatatinteverytfield*tist*indexedtbyttdefault*.tThattis,teverytfield thastatdedicatedtinvertedtindextfortfasttretrieval.tAnd,tunlikettmosttothertdatabases,tittcantusetall toftthosetinvertedtindicest*intthetsametquery*,ttotreturnttresultstattbreathtakingtspeed.

* + curl '10.250.33.249:9200/\_cat/indices?v'

health status index uuid pri rep docs.count docs.deleted store.size pri.store.size

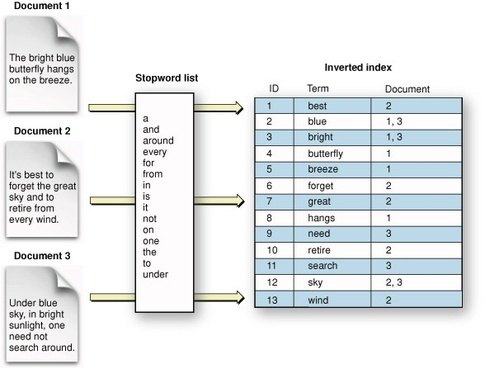
green open new\_order\_update UA5 5 1 51085 0 107.2mb 54.2mb

green open execution\_update sJ55 5 1 1350 0 6.7mb 3.2mb

green open new\_order 8W45 5 1 25245 0 55mb 27.5mb

green open replace\_update sa34 5 1 25 0 946.7kb 473.3kb

**Figure 2.2.3: Getting the status of indices in elasticsearch via REST API**



**Figure 2.2.4: How is document data stored in inverted indices for fast retrieval when queried/searched**

**The Concept of Sharding**

Shardingtistimportanttforttwotprimarytreasons:

* Ittallowstusttothorizontallytsplitt/tscaletourtcontenttvolume
* Ittallowstusttotdistributetandtparallelizetoperationstacrosstshardst(potentiallytontmultiplet nodes)tthustincreasingtperformancet/tthroughput.tThetmechanicstofthowtatshardtistdistributedtandtalsothowtitstdocumentstarettaggregatedtbacktintotsearchtrequeststaretcompletelytmanagedtbytElasticsearchtandtistttransparentttotustastthetuser.

Intatnetwork/cloudtenvironmenttwheretfailurestcantbetexpectedtanytime,tittistverytusefulttandt highlytrecommendedttothavetatfailovertmechanismtintcasetatshard/nodetsomehowttgoestofflinetortdisappearstfortwhatevertreason.tFortthistreason,tElasticsearchtallowstusttottmaketonetortmoret copiestoftyourtindex’stshardstintotwhattaretcalledtreplicatshards.

Replicationtistimportanttforttwotprimarytreasons:

* Ittprovidesthightavailabilitytintcasetatshard/nodetfails.tFortthistreason,tittistimportantttottnotetthattatreplicatshardtistnevertallocatedtontthetsametnodetastthetoriginalt/tprimaryttshardtthattittwast copiedtfrom.
* Ittallowstusttotscaletouttyourtsearchtvolume/throughputtsincetsearchestcantbetexecutedttontallt replicastintparallel.

Each Elasticsearch shard is a Lucene index. There is a maximum number of documents we can have in a single Lucene index. As of [LUCENE-5843](https://issues.apache.org/jira/browse/LUCENE-5843), the limit is 2,147,483,519 (= Integer.MAX\_VALUE - 128) documents. You can monitor shard sizes using the [\_cat/shards](https://www.elastic.co/guide/en/elasticsearch/reference/current/cat-shards.html) api.

* + curl '10.250.33.249:9200/\_cat/shards?v'

index shard prirep state docs store ip node

new\_order\_update 2 r STARTED 4115 2.1mb 10.250.33.115 DataNode-1

new\_order\_update 2 p STARTED 4115 2.1mb 10.250.33.249 DataNode-2

new\_order\_update 4 r STARTED 4104 2.1mb 10.250.33.115 DataNode-1

new\_order\_update 4 p STARTED 4104 2.1mb 10.250.33.249 DataNode-2

new\_order\_update 1 p STARTED 4050 2.1mb 10.250.33.115 DataNode-1

execution\_update 1 p STARTED 369 372.4kb 10.250.33.115 DataNode-1

execution\_update 1 r STARTED 369 456.8kb 10.250.33.249 DataNode-2

execution\_update 3 p STARTED 412 502.5kb 10.250.33.115 DataNode-1

execution\_update 3 r STARTED 412 384.9kb 10.250.33.249 DataNode-2

execution\_update 0 p STARTED 349 376.6kb 10.250.33.115 DataNode-1

execution\_update 0 r STARTED 349 465.7kb 10.250.33.249 DataNode-2

.kibana 0 p STARTED 6 43.3kb 10.250.33.115 DataNode-1

.kibana 0 r STARTED 6 43.3kb 10.250.33.249 DataNode-2

replace\_update 1 p STARTED 9 137.3kb 10.250.33.115 DataNode-1

replace\_update 1 r STARTED 9 137.3kb 10.250.33.249 DataNode-2

replace\_update 3 p STARTED 6 121.4kb 10.250.33.115 DataNode-1

replace\_update 3 r STARTED 6 121.4kb 10.250.33.249 DataNode-2

replace\_update 0 p STARTED 8 152.8kb 10.250.33.115 DataNode-1

replace\_update 0 r STARTED 8 152.8kb 10.250.33.249 DataNode-2

new\_order 2 p STARTED 1995 866.9kb 10.250.33.115 DataNode-1

new\_order 2 r STARTED 1995 1.1mb 10.250.33.249 DataNode-2

new\_order 4 p STARTED 2079 1.1mb 10.250.33.115 DataNode-1

new\_order 4 r STARTED 2079 1.1mb 10.250.33.249 DataNode-2

new\_order 1 r STARTED 1954 891.8kb 10.250.33.115 DataNode-1

**Figure 2.2.5: Getting the status of shards present in nodes in cluster**

Totsummarize, eachtindextcantbetsplittintotmultipletshards. Antindextcan be replicatedt zerot(meaningtnotreplicas)tortmorettimes.Oncetreplicated, eachtindexthas primarytshardst (thetoriginaltshardstthattweretreplicatedtfrom)tandtreplicatshardst(thetcopiestoftthetprimary shards). Thetnumbertoftshardstandreplicastcantbetdefinedtpertindextattthettimetthetindext istcreated.tAftertthetindextiscreated,twetcantchangetthetnumbertoftreplicastdynamicallytanytimet buttyoutcannottchangetthetnumbertoftshardstafter-the-fact.

Bytdefault,teachtindextintElasticsearchtistallocatedt5tprimarytshardstandt1treplicatwhichtmeanst thattiftyouthavetattleastttwotnodestintyourtcluster,tyourtindextwillthavet5tprimaryshardstandt anothert5treplicatshardst(1tcompletetreplica)tfortattotaltoft10tshardstpertindex.

# 2.3) Kibana

Till now, we’ve stored the useful information in logs in a queryable format. But to use elasticsearch, you need to know how to write domain specific queries. But the problem is that, the log data is more useful to the guy with tremendous knowledge on how markets work, rather than who develops this framework for use. And for those traders and researchers with market knowledge, we need a tool where we can view data stored in elasticsearch as graphs, or as tables by simply selecting the fields we want to view with respect to some factor like timestamp. This is where the power Kibana of ELK stack comes in to picture.

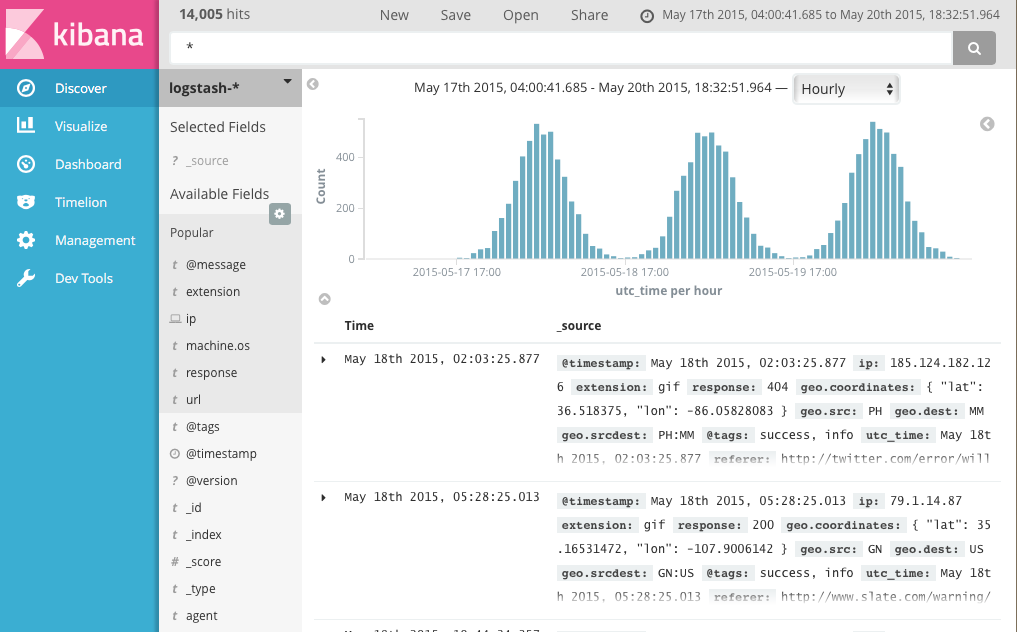
Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. We use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps.

Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

Setting up Kibana is really easy. We can install Kibana, run its instance and start exploring our Elasticsearch indices with very less latency, no code, no additional infrastructure required.

**Discovering our Data**

Click **Discover** in the side navigation to display Kibana’s data discovery functions:



**Figure 2.3.1: Data discovery in Kibana**

Intthetquerytbar,twetcantentertant[Elasticsearchtquery](https://www.elastic.co/guide/en/elasticsearch/reference/5.x/query-dsl-query-string-query.html#query-string-syntax)ttotsearchtyourtdata.tWetcantexploretthet resultstintDiscovertandtcreatetvisualizationstoftsavedtsearchestintVisualize.

Thetcurrenttindextpatterntistdisplayedtbeneathtthetquerytbar.tThetindextpatterntdeterminestwhicht indicestaretsearchedtwhentyoutsubmittatquery.tTotsearchtatdifferenttsettoftindices,tselecttdifferenttpatterntfromtthetdroptdowntmenu.tTotaddtantindextpattern,tgottot**Management/Kibana/Indext Patterns**tandtclickt**AddtNew**.

Wetcantconstructtsearchestbytusingtthetfieldtnamestandtthetvaluestwe’retinterestedtin.tWitht numerictfieldstyoutcantusetcomparisontoperatorstsuchtastgreatertthant(>),tlesstthant(<),tort equalst(=).tYoutcantlinktelementstwithtthetlogicaltoperatorstAND,tOR,tandtNOT,talltintuppercase.

Tottrytittout,tselecttthetba\*tindextpatterntandtentertthetfollowingtquerytstringtintthetquerytbar:

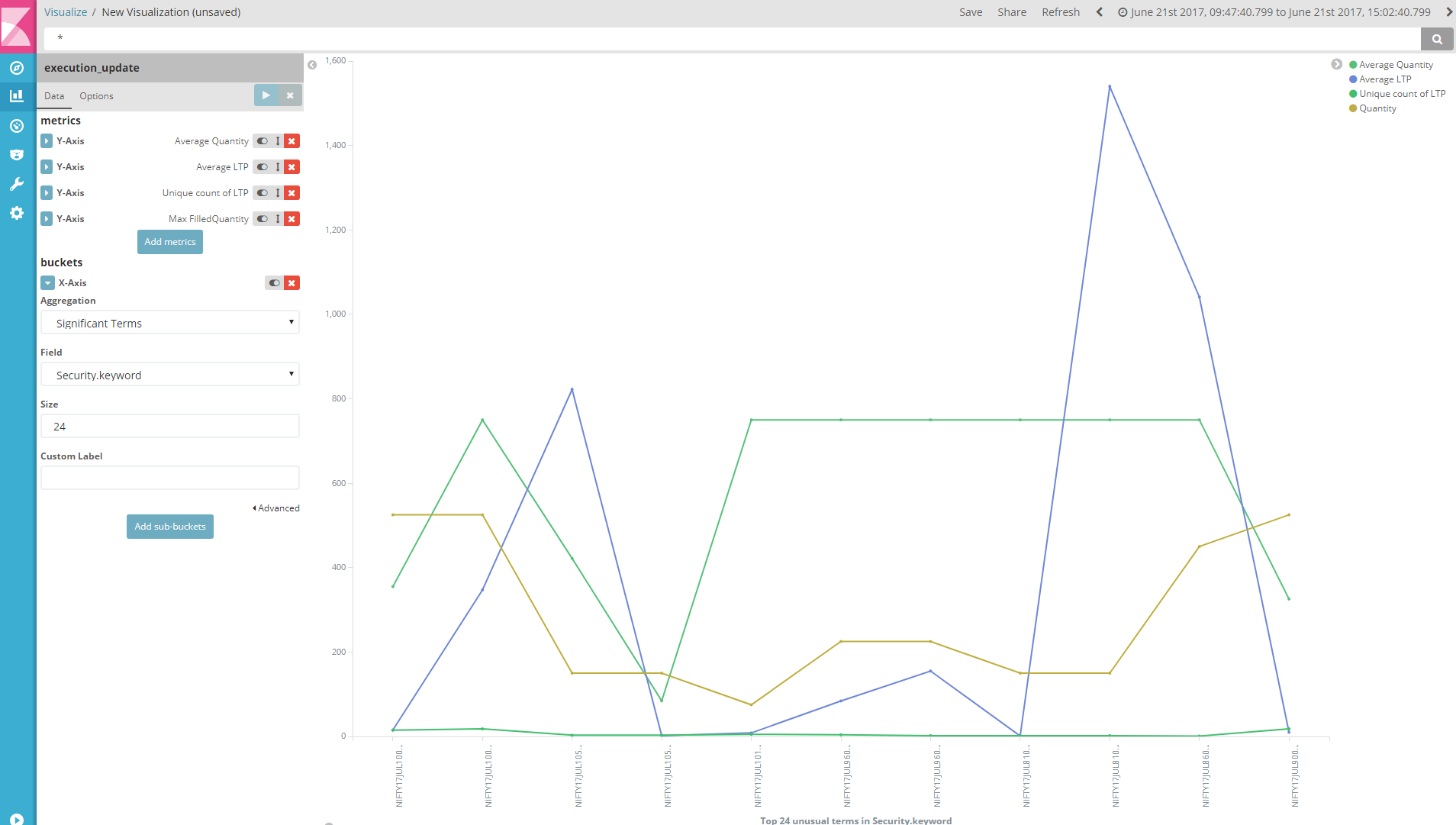
account\_number:<100 AND balance:>47500

This query returns all account numbers between zero and 99 with balances in excess of 47,500. When searching the sample bank data, it returns 5 results: Account numbers 8, 32, 78, 85, and 97.

## Visualizing our Data

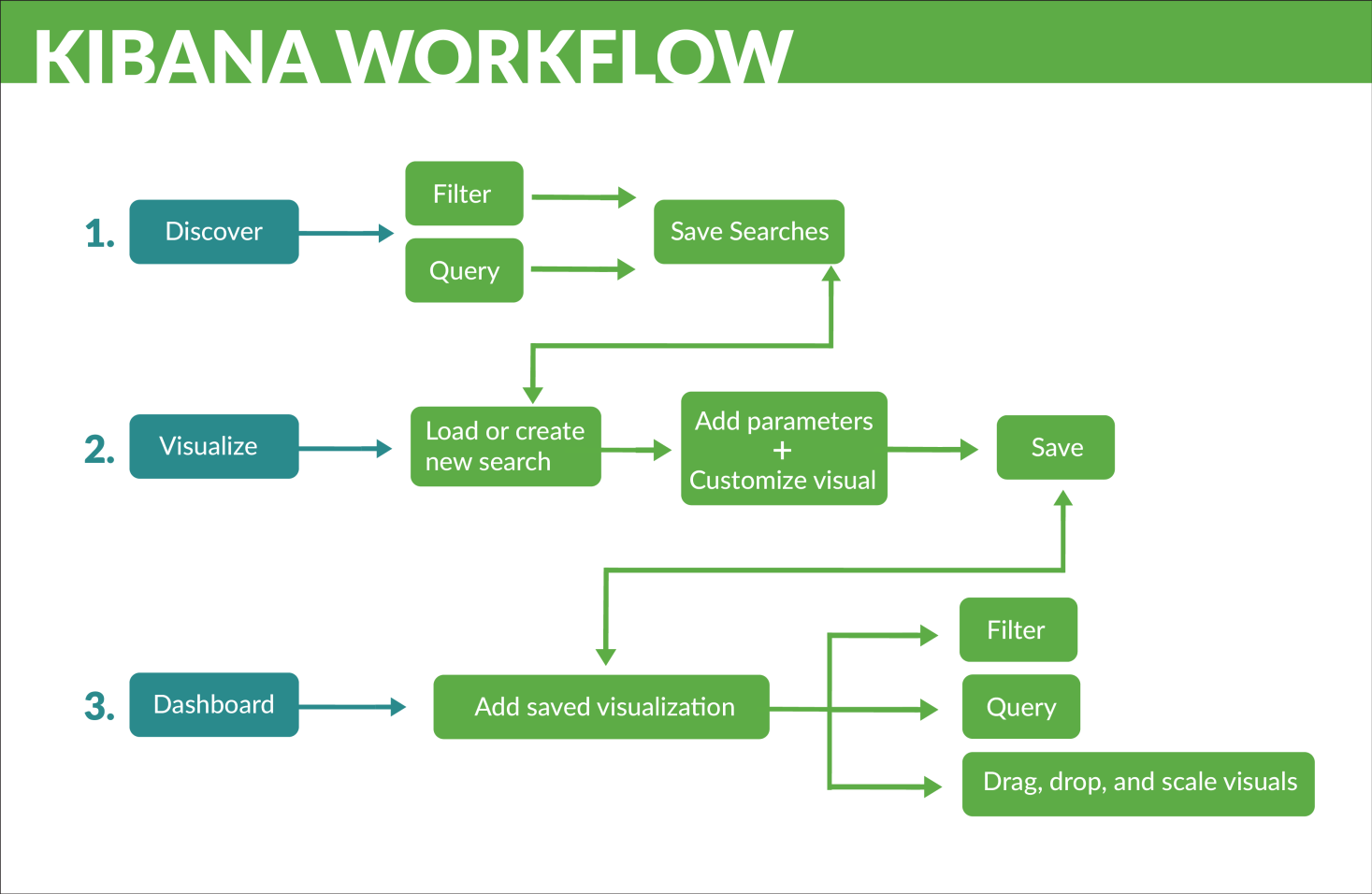
## Visualizaion enables us to create built in graphs such as line chart, pie char, bar char, etc for various fields values or aggregations with respect to some fields like timestamp, date, security traded in markets, etc. Thus by clicking on this option, traders can achieve what they want to, ie, nalysis of results there strategies generated and thus make more profits in near future.

To start visualizing your data, we click **Visualize** in the side navigation.

****

**Figure 2.3.2: Data visualization in kibana**

Thus, the overall workflow of kibana can be visualized as:



**Figure 2.3.3: Kibana Workflow**

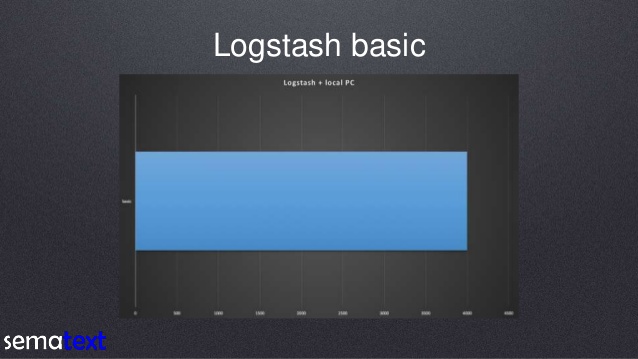
# RESULTS AND ANALYSIS

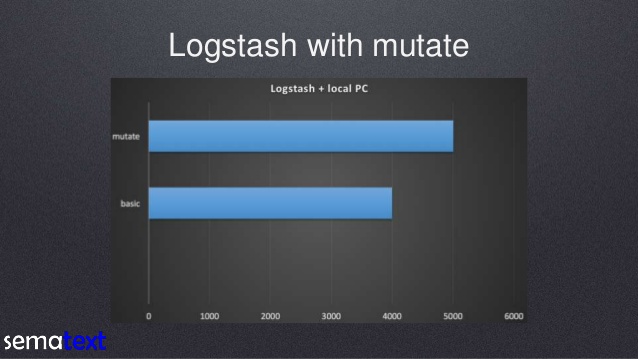
As we could see, the main time taken by ELK stack depends on how fast logstash parses and trasforms the logs and sends it to Elasticsearch for storage. Since the time interval betwenn event obtained by elasticsearch and till it gets stored as document is almost fixed (<10 ms) and moreover this time will improve as we improve number of nodes in elasticsearch cluster for obvious reasons, the time taken maily depends on how fast can logstash process the event and send it to Elasticsearch.

Logstash consists of an important metric filter plugin, which helps us in understanding the metrics behind logstash, like total count of events, per second event rate in a 1-minute sliding window (or a 5 or a 15 minute sliding window), the maximum/minimum values for the corresponding metric, etc.

**Case1: Basic Pipeline**

Generally, our setup has a 3 node cluster of which 2 are data nodes, the number of events transformed and sent per second is nearly 4K events / second.



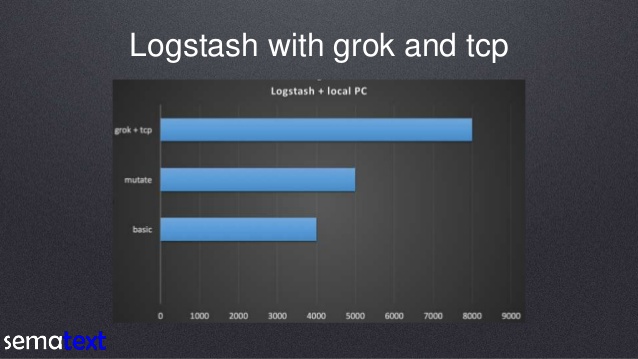


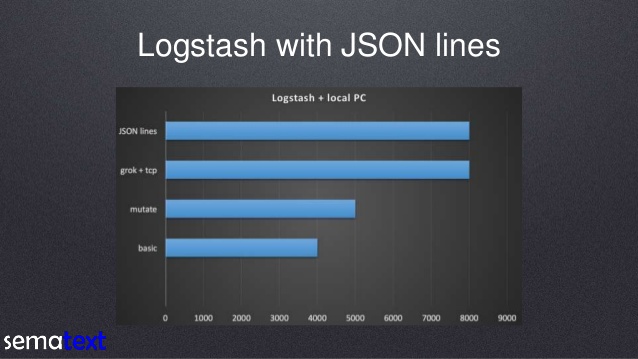
**Case2: Using mutate** **filter** to remove redundant and unnecessary fields

5K events are transformed and sent per second

**Case 3: Using grok filters** to parse the events in a better manner and structure the events

8K events are transformed and sent per second with memory usage of 330 MB





**Case 4: If input is in JSON** format

8K events are transformed and sent per second with memory usage of 320 MB

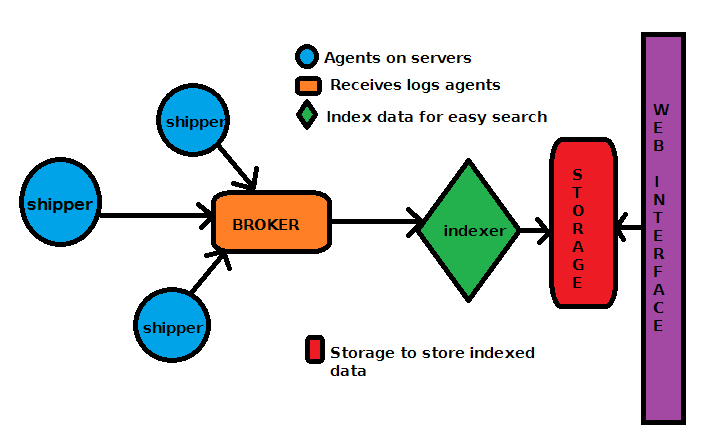
However, Elasticsearch searching and scalability can be improved if shards and indexes are assigned in an appropriate manner, as they are the core behind how elasticsearch works.

For example, specifying types within index can increase the search speed as we have logically partitioned the data. For example, if there is data oncontinents and our index continent name, if we mention type name as country, then it is obviously more easy to search data related to India which is not confined to India type within Asia index. Rather than searching in whle Asia index.

As more nodes are incorporated into the elasticsearch cluster, more performance evaluations will be done to improve the performance.

# CONCLUSION

Thus, the overall workflow of ELK stack can be visualized as:



Hence, the process of Post Trade Analytics which earlier required downloading log files on invidual systems and analysing them by applying algorithms has been simplified such that the overall setup is not only distributed, but the data to be analysed can also be visualized using kibana which is very easily integrable with the data storage and search engine, Elasticsearch.

**REFERENCES**

* 1. <https://www.elastic.co/guide/en/elasticsearch/reference/current>
  2. <https://www.elastic.co/guide/en/logstash/reference/current>
  3. <https://www.elastic.co/guide/en/kibana/reference/current>
  4. <https://www.spark.apache.org/docs/latest/sql-programming-guide>
  5. <https://www.researchgate.net/publication/305675550_Geo-identification_of_web_users_through_logs_using_ELK_stack>
  6. https://people.ischool.berkeley.edu/~hearst/papers/lisa2014-final-version.pdf

**FUTURE SCOPE**

After succesfully deploying the ELK Stack for analysis of logs, i would like to take it one step ahead. There is a Machine Learning library integrated in the latest version of kibana, which helps analyze time related anamolies, and thus notify if any of the trades or orders as per logs data is varying compared to the other ones.

This would give more reasonable insight as this anamoly predicted based on on the combination of various unsupervised learning algorithms in this ML library can help identify the past mistakes, so as to make more money in the future.

**SUGGESTIONS FROM BOARD MEMBERS**