Selvan Arumugam

Developing Cloud Based Log Analytic Services

Helsinki Metropolia University of Applied Sciences

Master Of Engineering

Information Technology

Master’s Thesis

26 May 2017

Preface

This graduate study has been part of Master of Engineering Degree Programme in Information Technology. The motivation for this thesis was the reliability challenges posed by the old dedicated physical server log analytic system in analysing growing volumes of events data.

The study was undertaken as part of log analytic system cloudification work in my company. As such, I would like to thank my colleagues especially Leevi Nieminen for facilitating the thesis work and supporting my studies.

I would also like to thank my supervisor Juha Kopu, for his time, patience and constant feedback throughout the work on my thesis.

Finally, I would like to thank my family who has inspired and supported me during my studies.

Selvan Arumugam
Espoo, 26.05.2017

Author Title Number of Pages Date	Selvan Arumugam Developing Cloud Based Log Analytic Services 50 pages 26 May 2017
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Ville Jääskeläinen, Principal Lecturer Juha Kopu, Project Supervisor

Over the years, cloud computing has been attracting considerable interest from developers and researchers. Virtual machines, the key to cloud computing environment, provide software computers that, like physical computers, run an operating system and applications. These virtual machines provide high availability and scalability in the cloud compared to dedicated physical servers.

This thesis considers virtual machines in cloud for log analytic purposes. The thesis focuses on developing cloud based log analytic services using virtual machines as hosts in the case company private cloud, eliminating the need for dedicated physical servers.

The benefits of virtual servers over physical servers are discussed together with a commonly used log analytic service, the ELK Stack.

As a part of testing and evaluation of the solution for the thesis, cloud based log analytic services were configured in the case company's private cloud. The acquired test results clearly show that the chosen cloud based solution improves the stability and reliability of the company log analytic system, with high availability and scalability as compared to a physical server.

Keywords	Logstash, Kibana, Elasticsearch, Virtual Machines, ELK

Helsinki Metropolia University of Applied Sciences

Contents

Abbreviations and Terms

AWS	Amazon Web Services
CRUD	Create Read Update Delete
ELK	Elasticsearch Logstash Kibana
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
HTTP	Hypertext Transfer Protocol
JVM	Java Virtual Machine
NE	Network Elements
NIC	Network Interface Controller
OS	Operating System
VM	Virtual Machine

List of Figures

List of Tables

List of Listings

1 Introduction

Many large organizations and businesses with big data analytics are still using log analytic services on dedicated physical servers due to earlier infrastructure investments.

With ever growing data demands in big data analytics, the volumes of data analysed for statistical or troubleshooting purposes increase in log analytic systems. Data constantly flows into these physical server log analytic systems and, as the data sets grow larger, the analytic ability of the log analytic systems slows down, resulting in sluggish performance. Analysing these huge volumes of data demands computing power which strains the computing resources of dedicated physical servers.

Strained computing resources provide challenges to the stability of this type of log analysis setup as the volumes of data to be analysed grow. When these outdated hardware servers are subjected to much larger workloads, performance-related problems occur frequently. Often these problems cause the services to be unavailable for log analytic purposes.

Outdated hardware and growing volumes of data increases demand for cloud based virtual machines as one of the emerging technologies to solve the physical servers' hardware limitations. Virtual machines release hardware limitation away from locally hosted infrastructure into cloud based solutions. The scalability of the cloud based virtual machines offers unprecedented advantages against physical servers by providing the ability to add or remove virtual resource within minutes. By distributing load across these virtual machines, high availability of computing power is ensured all the time.

Many of the existing tools and techniques can also be well adapted to virtual machines running in the cloud. The analytic tools to collect, process and analyse these huge volumes of data also evolve in such a way that they are optimized for less resource demanding systems. One of the commonly used log analytic tool suites is the ELK Stack. ELK Stack is a combination of three open source projects, Elasticsearch, Kibana and Logstash, which search, analyse, and visualize the data.

1.1 Objective

The objective of this thesis was to develop cloud based log analytic services and determine if the cloud based system was a better solution compared to locally hosted dedicated physical servers.

The thesis project was done at one of leading global telecommunication companies. The study uses the case company’s large data generated by the company’s automation system. The data used is events’ troubleshooting data from syslog produced by the test systems in the case company’s automation system.

The study starts with the investigation of the existing physical server based log analytic system used in the case company. The study continues by identifying the potential log analytic services which can be performed in the cloud based solution. Based on the results, the ELK services in the case company’s private cloud were configured. All the relevant data to evaluate the performance of the cloud based log analytic services was collected. After the testing and evaluation period the study aimed to conclude whether the cloud based log analytic system is the best solution for the company's log analytic system problem.

1.2 Outline

Chapter 1 is the Introduction which discusses the current trend in log analytic systems and the objective of the thesis.

In Chapter 2, the typical log analytic systems and their architecture is discussed. Next the basics and the benefits of a commonly used log analytic tool, the ELK Stack, is discussed in Chapter 3, followed by the benefits of Virtual Machines in Chapter 4.

Chapter 5 introduces the case company’s log analytic system and its shortfalls.

Chapter 6 outlines how the cloud based ELK stacks discussed in Chapter 3 and the benefits of Virtual Machines in Chapter 4 can solve the case company’s log analytic problem.

Chapter 7 discusses the performance of the cloud based ELK service configured for the case company, and Chapter 8 summarizes the thesis.

2 Log Management

This chapter discusses the basics of data logging, why data logging is done and typical log analytic system.

2.1 Logging

In computer log management, log analysis attempts to make sense of the events data that comes out of computer generated records. The process of creating such records is known as data logging. Some of the reasons why log analysis is performed are:

i. Fulfilling security policies
ii. Compliance with audit or regulation
iii. System troubleshooting
iv. Security incident response

Network devices, operating systems, applications and all kinds of intelligent or programmable devices emit logs. A stream of messages in time sequence often include a log. Devices may direct these streams of messages to files and store them on disk, or they can be directed as a network stream to a log collector. [1] Most of these devices or systems generate syslog which follows the logging standard defined in Internet Engineering Task Force (IETF RFC 5424). Syslog mainly consists of the following information:

Facility - numerical indicator to identify the sender component or application.
E.g. 0 is for kernel messages, 2 for mail subsystem, etc.

Severity - integer value to indicate the importance or severity of the message or event that took place. Lower number indicates higher importance of the message. E.g. 0 is for emergency and 7 for debugging.

Host – The name or IP address of the sender component or application.

Timestamp - The local time when the event takes place or the component or application message is generated.

Message – Textual part of the syslog message with additional information about the component or application that generated the message. Often this message is unstructured and human readable.

An example of a syslog from Cisco IOS device is as shown in Listing 1. [2]

```
*Mar  6 22:48:34.452 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback0,
   changed state to up
```

Listing 1 Syslog example from Cisco IOS device

Not all these fields are present in the syslog, depending on the implementation or configuration of the systems or devices. With this standardization and best practices, syslogs are commonly used for log analytic systems.

2.2 Log Analytic System

Typically, a log analytic system consists of four main elements, as shown in Figure 1: **Generation**, **Transport, Storage** and **Analysis**.

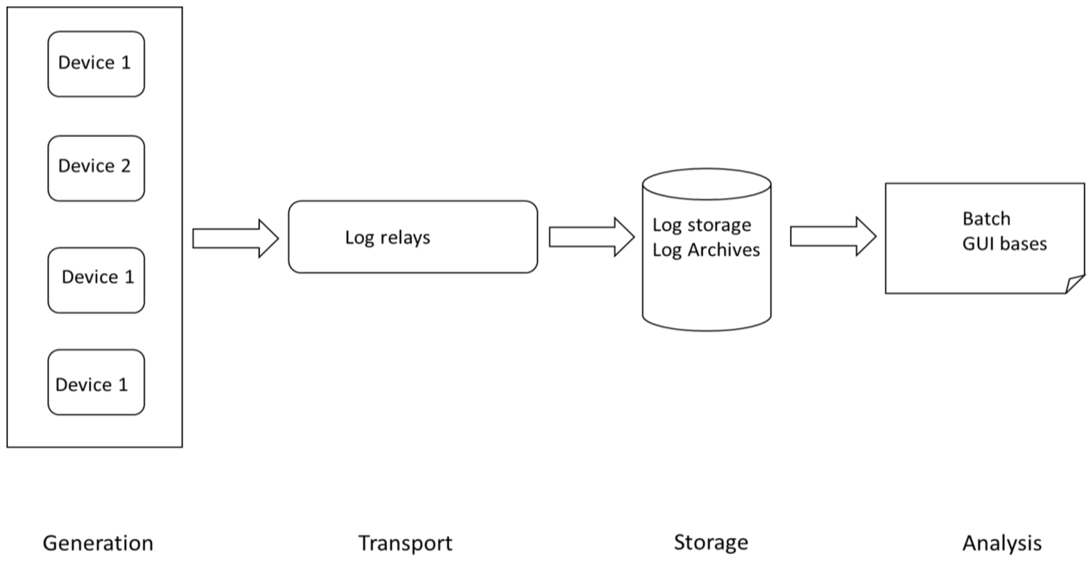


Figure 1 Typical log analytic system architecture

Each of these elements runs as services in a host. In earlier generations of log analytic systems, a single personal computer or workstation can host the entire solution.
The logs are generated locally or fed into the host computer and then transported via local sockets to storage on local hard disks. A simple analysis can be done with grep.

Generation in log analytic system refers to the components where devices, network devices, operating systems, applications, firewall etc. generate logs. Applications generate logs in many ways. Some applications log the event in the application's own directory and some through syslog. If the application is running in a Linux based host environment, there are more log files generated in /var/log as well as in the application's own directory. The collection frequency of the generated logs depends on the nature and intended purpose of the log analytic system.

If the log analytic system is intended for troubleshooting purposes, then real-time solutions which can monitor the changes to the log files are the best choice. In contrast, if the intended purpose of the log analytic system is for analysing log data offline, for example calculating metrics, then a file replication strategy would be the best method.

In a file replication based approach, the logs are analysed later by replicating the logs to a local or centralized server on a fixed schedule. A one minute rsync cron job does this reliably.

Transport in log analytic system consists of log relays. When multiple devices and applications are running in a host, the generated and accumulated logs can grow quite quickly. The system must transport these logs and save them in a local or centralized location reliably and effectively to ensure that no data is lost. There are various log relays available for transporting these log files effectively and quickly such as Logstash, Flume, Scribe, fluentd and others. All these transport relays are suitable for transporting large log files even though the approach taken by these tools differ slightly from each other.

Some of these tools require clients to log the data through their own APIs. Application code is written directly to the logging application with the API so that events log directly to the destination such as a local or centralized location. The applications have more control this way on what type of logging is needed and necessary depending on the intended purpose for an efficient analysis later. A typical example of log relays that employs this approach is Scribe. There is another type of log relays which are independent of application APIs and can transport log files reliably. These log relays have more than one source as input and can transport the log files by tailing them. Typical examples of such log relays are Logstash, fluentd and Flume

Storage in a log analytic system is a local or centralized location where the system stores the transported log files. There are a few considerations that need to be considered when choosing the type of centralized location for log analytic system. The duration of the analysed logs is one of them. If the logs are intended for long term archiving purpose, then there are a few service providers who can provide the backup services with low cost and large volume data. Amazon’s S3 and AWS glacier are some of the possibilities. If the logs are stored only for short term duration, then a distributed storage system would be the best alternative. HDFS and MongoDB would fit well for this purpose.

Data volume needs consideration as well. Depending on the devices and applications in the log analytic system, there is variation on the volume of log files. The storage system should allow the possibility of scaling out when there is a need for storing large volumes of data. Finally, how the log analytic system accesses the data is considered. Some storage nodes are not suitable for real-time or even batch analysis. Tape backup or AWS Glacier usually takes hours to load data. If there is a need to perform real-time and interactive analysis, then storing the data in Elasticsearch would be a better choice.

Analysis in a log analytic system consists of components where applications or tools provide better understanding on the performance of the system. Typically, there are two approaches how this data is analysed. Batch oriented process that runs periodically is the most common approach in some log analytic systems. Data stored in HDFS or Apache HIVE facilitates these batch oriented processes, easing and helping in analysing the data. The second most common method is based on the user interface (UI) for analysis. Kibana and Graylog2 are GUI based applications that query and inspect the parsed log data stored in Elasticsearch, and represent it graphically. The log parsing, stored in Elasticsearch, can be done with Logstash or Heka. This approach allows more real-time data access as all the services involved provide the platform for real-time analysis. [3]

3 ELK Stack

Presently the ELK Stack has become the common solution for open-source log analytic systems. ELK consists of Elasticsearch, Logstash and Kibana (ELK), as shown in Figure 2.

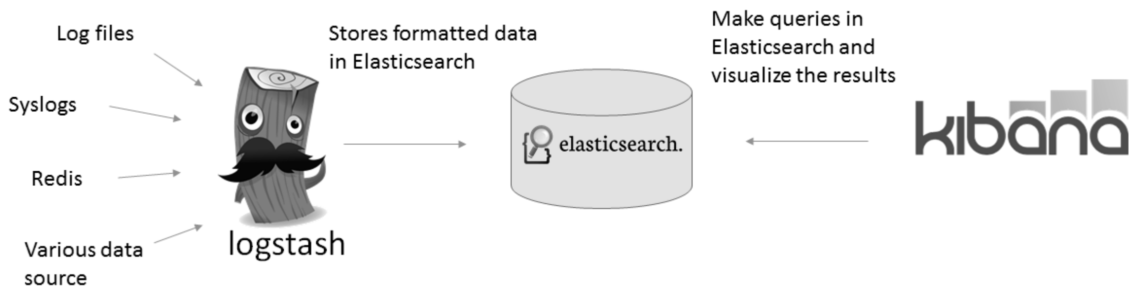

Log files
Syslogs
Redis
Various data source
logstash
Stores formatted data in Elasticsearch
elasticsearch.
Make queries in Elasticsearch and visualize the results
kibana

Figure 2 ELK Stack

These are open source projects that help take data from any source in any format and search, analyse, and visualize it in real time. [4] ELK provides a very powerful real time log analytic solution that is easy to scale.

3.1 Logstash

Logstash is a gateway for the log analytic system. It is an open source data collection tool that takes data as an input, processes it and outputs it. It has an arsenal of ready-made inputs, filters, codecs, and outputs. With this tool, one has access to a very powerful feature to extract the relevant, high-value data and store it in a central location. Figure 3 illustrates a Logstash instance.

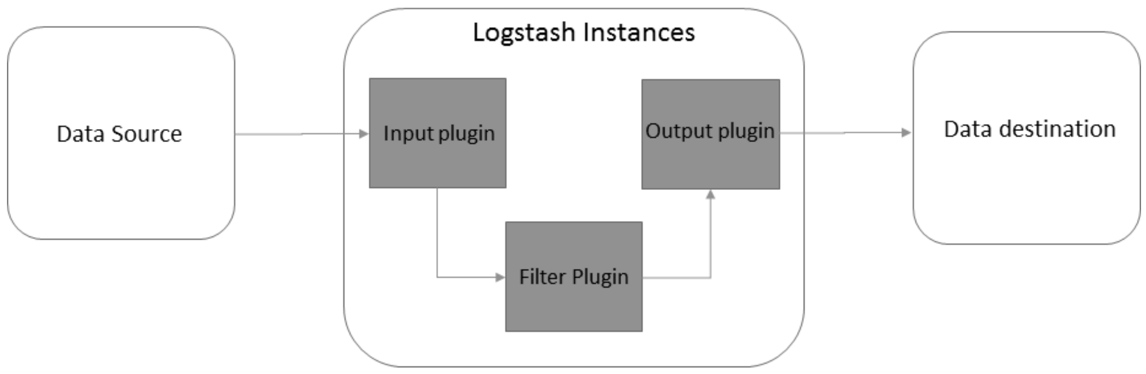


Figure 3 Logstash instance

Logstash extracts the relevant data with event processing, which consists of three stages. Stage one is the input, followed by the filter and, finally, the output. Inputs generate events while the filter modifies the events and, finally, the output sends them to the intended destination, for example Elasticsearch.

Input - Inputs get the data into Logstash system. Some of the most common sources of inputs are text based log files from the applications. Inputs can also be from S3 buckets, syslog files, Redis or process events sent by Filebeat.

Filter - Filter stage is also known as intermediate stage. Upon receiving events from an input, Logstash performs an action based on the conditional filter. The data received from input is transformed from unstructured to structured data that suits the log analytic system by means of available plugins.

Output - In the final output stage, the structured data from intermediate processing is sent to multiple outputs. Some of the commonly used outputs are Elasticsearch, files or other open source for storing and analysis beside Elasticsearch such as Graphite.

3.2 Elasticsearch

Elasticsearch is a real-time log search and analytic engine, used to find high value data underneath huge piles of data. It provides real-time data and analytic services, with the ability to extract data from structured or unstructured files very rapidly. With Elasticsearch all the features are available to make real-time decisions all the time. The core components of Elasticsearch are Cluster, Nodes, Shards and Replicas, as shown in Figure 4.

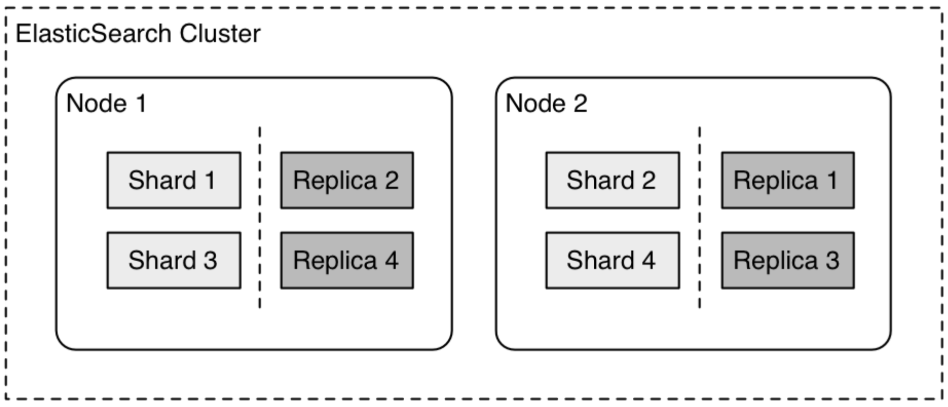


Figure 4 Elasticsearch Cluster

Cluster - A cluster consists of several collections of one or more nodes. By holding the nodes together, the cluster can provide search capabilities across all the nodes by means of indexing.

Node - A node is a single server that is part of the cluster, stores the data, and takes part in the cluster's indexing and search capabilities.

Shards - An index is a collection of documents in a node that can easily store large volumes of data which potentially could limit the resource capabilities of the node, in terms of storage space and performance. This slows the search capabilities in a cluster from a node. To avoid this problem, Elasticsearch provides sharding features. With sharding, the indexes split into small pieces of shard and are stored independently in the nodes. Sharding offers two distinct advantages:

i. It allows horizontal scaling of data volume.
ii. Shard in multiple nodes allows parallelized operation which increases the search performance.

Replicas - In networked or virtual cloud environments, there are possibilities for failures which causes a shard or node to go to offline mode. From the user or application point of view these nodes or shards then disappear from the log analytic system. To ensure availability, Elasticsearch allows to make copies of these shards called replicas, distributed across the nodes. Replicas also provide two distinct advantages:

i. High availability if there are node or shard failures

ii. Replicas in multiple nodes allows for parallelized operation since search operation is executed parallel in all replicas. [5]

3.3 Kibana

Kibana provides the front-end functionality of the ELK stack and is the component that end-users will mostly be interacting with. Trends that are very profoundly tedious to read and interpret can be visualized with Kibana. Large data can be visualized with pie charts, bar graphs, trend lines and scatter plots.

4 Virtual Machines

Virtual machine (VM) is an operating system or application environment that is installed on software which imitates dedicated hardware. The end users have the same experience on a virtual machine as they would have on dedicated hardware. [6]

This chapter discusses the virtual machine’s architecture, comparison between virtual server and dedicated physical server and the benefits of virtual machines.

4.1 Virtual Machine Architecture

A key element in the virtual machines is the virtualization of the hardware. Hypervisor is a software component which can virtualize the system hardware. Figure 5 shows the architecture of a physical server as opposed to a virtual server.

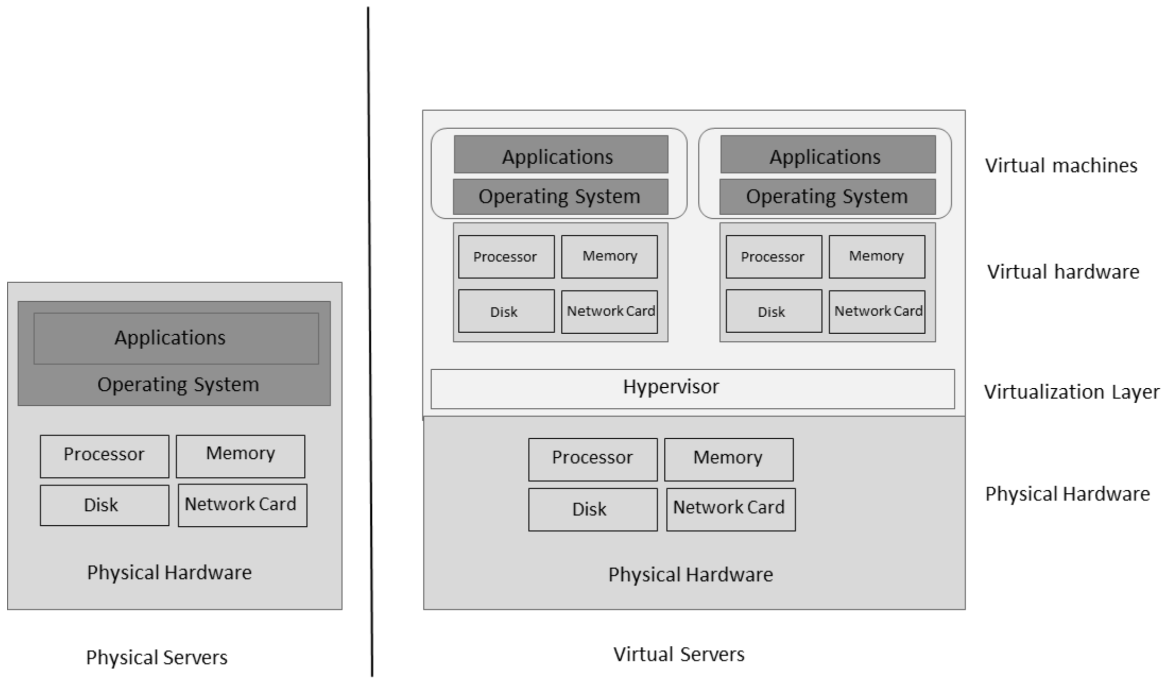


Figure 5 Architecture: physical server vs virtual server

Each of these dedicated physical servers has its own CPU, memory, disk storage etc. OS and applications will be installed in this physical workstation. In virtual machines, all the same resources are available as in the physical server but these virtual machines loaded from hypervisors emulate the hardware, i.e. virtual hardware.

The created virtual machines' hardware is emulated in this hypervisor layer. As in the dedicated physical servers, these virtual machines have its own virtual CPU, memory, disk storage etc. Guest OS and needed applications installed in these virtual machines run as in the traditional physical servers. At the bottom end of the tier, the CPU cycles, memory, disk capacity and the I/O bandwidths are shared among the virtual machines. It is known that application peaks occur at different times. Virtual machine allows a better utilization of the resource pools by spreading each of these applications to its own virtual machine.

4.2 Virtual Machine Flavors

Flavors are hardware resources needed when creating virtual machines. Flavor types depends on the available hardware resources. Some flavors provide options for the highest performing processors while others provide large RAM capacity in memory applications. Table 1 shows examples of flavors.

Flavor	VCPU	RAM (GB)	Storage Disk (GB)
t1.micro	1	1536	25
m1.small	2	8192	40
m1.medium	2	15360	60
c1.medium	4	15360	100
m1.large	4	30720	200
m1.xlarge	6	30720	200
c1.xlarge	6	46080	300
m2.xlarge	8	46080	400
hs1.8xlarge	30	204800	2800

Table 1 Virtual machine flavors

VCPU shows the number of CPU cores created when the flavor is used to spin up virtual machines. The choice of flavors depends on the task performed in the virtual machine. For lightweight task, small size flavor and for demanding task a bigger size flavor is used.

4.3 Benefits of Virtual Machines

There are a number of advantages of virtual machines over the dedicated physical servers.

Cost - As discussed in Chapter 4.1, instead of running one OS per dedicated physical server, virtual machine allows the possibilities of running several OSs and applications with one shared physical hardware. This virtualization of hardware brings significant cost savings to enterprises.

High Availability - By having several virtual machines in a system, load of the system can be distributed to several virtual machines. This allows the virtual host to have high availability of application and data to the system via this virtual machine. When one of the virtual machines fails, another virtual machine can be created with minimal downtime and without any loss of data.

Scalability - Virtual machines allow scalability on demand without the need to add any physical resources. When a virtual machine needs more resources, for e.g. RAM, it can be easily added within minutes if compared to physical servers.

Hardware independence and portability - Each of these virtual machines consists of virtual hardware and guest OS. The guest OS is only aware of virtual hardware configuration and not the physical server hardware. This means the virtual machines are hardware independent and not tied to certain hardware. It allows portability where virtual machines can me moved from one datacentre to another datacentre in different locations.

5 Log Analytic System in Case Company

Work at the case company requires analysing the test results and data that are generated by the case company's automation system. The automation system consists of several test systems with each one of them generates troubleshooting events data in the form of syslog. The automation system feeds these syslogs into the case company's physical dedicated log analytic system.

Figure 6 shows the log analysis system of the case company.

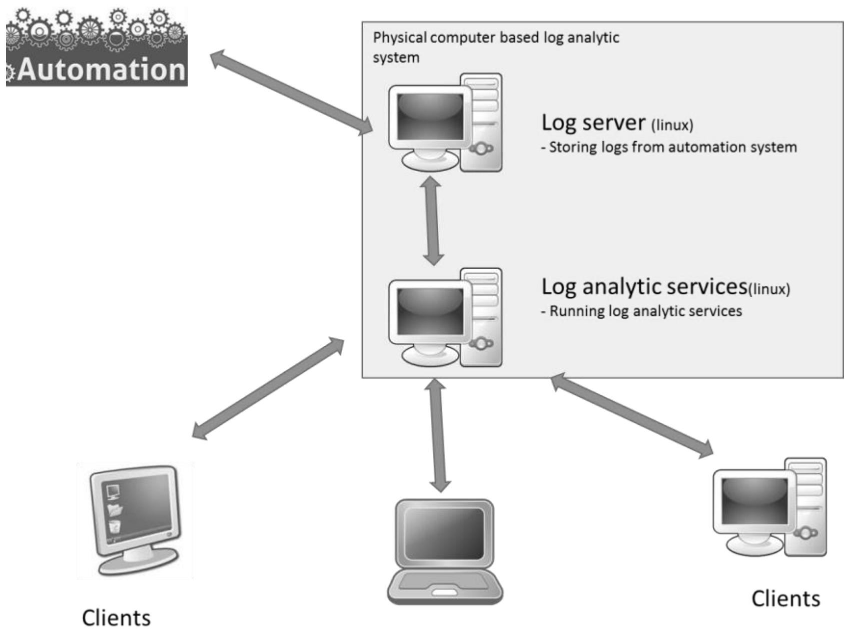

Automation
Physical computer based log analytic system
Log server (linux)
- Storing logs from automation system
Log analytic services(linux)
- Running log analytic services
Clients
Clients

Figure 6 Case company log analytic system

As shown in Figure 6, the log analysis system of the case company consists of two dedicated physical servers running Linux operating system. Linux based log analytic services are running in these two work station computers providing the possibility to collect substantial amounts of data for analytic purposes. Analysing these huge volumes of data is now done in local network.

As shown in Figure 7, the log analytic system in the case company follows the earlier adaptation approach where a single workstation, running Linux based OS hosts the entire log analytic system. All the three log analytic services (Elasticsearch, Kibana, Logstash) are configured and run in the single host physical server.

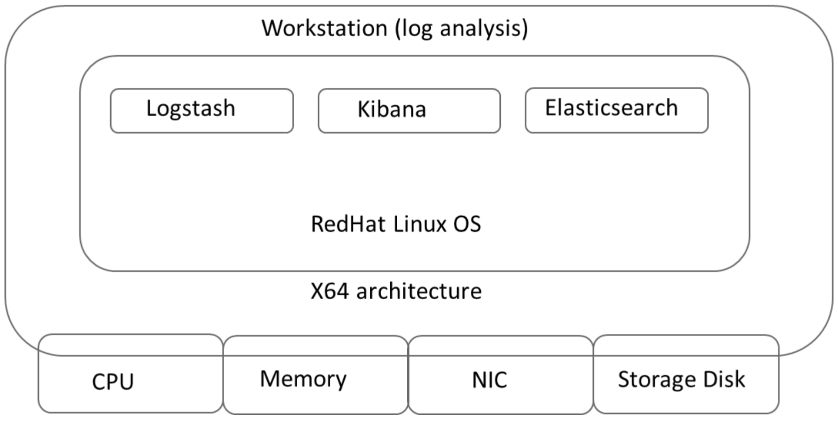


Figure 7 Case company log analytic architecture

Table 2 shows the hardware used to develop the single host dedicated physical server log analytic system in case company.

CPU	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz
Cores	4
RAM (GB)	12
Storage (GB)	500
OS	RHEL 5.9 2013-01-07 2013-01-07 RHEA-2013-0021 2.6.18-348

Table 2 Environment used for case company's log analytic system

The hardware used to develop the single host log analytic system consist of four cores Xeon CPU with 12 GB of RAM. Red Hat Linux version RHEL 5.9 was installed as an operating system for this log analytic system.

5.1 Configured Log Analytic Services

The case company has already developed a basic log analytic system which runs in a single host dedicated physical server. The commonly used ELK log analytic suite is already in use in the case company's log analytic system. The configured single host physical server log analytic system in the case company includes all the four log analytic system components discussed in Chapter 2.2.

Automation test results as log **Generation** - Logs from the case company's automation system are fed into the log analytic system via a log server, also based on the physical server running Linux operating system.

Logstash shipper as **Transporte**r - After test case execution, the company's automation system copies the logs to a centralized server. From the centralized log server, the logs are fed into the log analytic system with Logstash shipper.

Workstation and Elasticsearch as **Storage** - The case company's automation system stores the test results in two different storage hosts. One of them is the company's storage host which also serves as a backup and log server for the log analytic system. In this host the logs are stored in their original format. The same set of logs is fed into the log analytic system and stored in Elasticsearch, but in a formatted form for easy analysis.

Kibana for **Analysis** - GUI based log analysis is done with Kibana from several client terminals connected to the log analytic system. Each of these elements runs as a service in the host workstation. The logs are fed into the host computer via NIC from the case company's automation system and then transported via local sockets for storage on a local hard disk.

5.2 Obstacles to Solve

With entire log analytic solutions hosted in a single workstation, the case company has managed to build a log analytic system which is up and running for short periods of time. The single host log analytic system serves its purpose well, but over the time more test systems have been added into the case company's automation system. The addition of test systems in the case company's automation system increase the amount of data that needs to be analysed.

The way the case company analyses and shares the data also evolves from local network drive to a wider network over intranet and internet. This information and results are shared not only locally but also worldwide in the case company's global location. The addition of test systems and the increase in volumes of data that need to be analysed provide challenges to the stability of the current log analysis setup. The workload for the workstation, which hosts the entire log analytic services, has multiplied and the outdated log analysis setup cannot keep up processing the huge volumes of data.

All these changes lead to performance related problems when the single host log analytic system breaks down often. Log services are not available all the time for analytic purposes and hinders the troubleshooting activities in case company. Investigation to the case company's log analytic system reveals several shortcomings regarding the design and set up. Some of the observations were:

i. Minimal design, Oversight in future expansion

   The case company had configured the log analytic services in a single host dedicated physical server with the intention of analysing data from one test system only. When more test systems are taken into use, the system fails to process the huge volumes of data fed into the log analytic system.

ii. Design oversight

   The choice of putting the entire log analytic services in a single host workstation is not optimal for log analytic system. Configuring all services in one dedicated physical server overloads the workstation.

iii. Hardware limitations

   The hardware configuration chosen when the log analytic system was setup could have been more powerful. Configuring all the services in one host system requires a powerful and high performance hardware. From the case company's chosen hardware, it is obvious that the hardware used could lead to a big disadvantage against the various Java based log analytic services. Logstash runs on JVM and consumes a considerable amount of resources for filtering and indexing. [7] This creates CPU processing power limitation for the inefficient Elasticsearch cluster design (described in detail in chapter v below). As the volumes of data multiply the situation gets worse, leading to frequent freezes or crashes of the log analytic services.

iv. Non-optimal data filtering and parsing

The case company log analytic system uses the Grok Filter plugin to parse and structure the data for analysis. The Grok filter is known to be resource intensive especially in regular expression computation. [8] Running this filter in already resource limited machines only adds more instability to the log analytic system.

v. Inefficient cluster design for Elasticsearch.

With single host physical server, the log analytic system has only one Logstash and Elasticsearch service running. The Logstash connected directly to the Elasticsearch also performs dedicated cluster management. This approach of Elasticsearch configuration is not recommended, as it brings instability to the Elasticsearch cluster if Elasticsearch needs to handle large volumes of data. This poses problems to an already resource limited system.

To keep the log analytic service up and running, the end user or system administrator from the case company needs to reboot the log analytic workstation frequently.

5.3 Requirements

There is a need to ensure that all these log analytic services are running in an orderly manner, providing continued log analytic service without any interruption and downtime. This is to ensure that the log analytic services are available all the time to the end user. For the log analytic system to reach the level of stability and performance with the current trend of big data, the case company must embrace a new technology and method to perform the log analysis.

One of the identified options to achieve this is by moving the log analytic system from the dedicated physical server based solution to a cloud based solution. The research objective related to this is to determine if cloud technology could achieve this requirement.

6 Cloud Based ELK Log Analytic System for Case Company

In the case company, with a single host dedicated physical server, there were critical stability and performance related problems in their log analytic system with the current trend of big data analysis. As described in Chapter 5.2, investigation already reveals some of the shortcomings of the current log analytic system in case company. Design and future expansion oversight, hardware limitation and non-optimal filtering and parsing of data to be transported are some of the identified problems causing instability to the log analytic system. Chapter 3 discussed how the ELK Stack is emerging as the current widely used common tool for big data log analytic systems. In Chapter 4 the benefits of virtual machines compared to the physical hardware based workstation were discussed.

6.1 Common Solutions

A scalable cloud based log analytic system was proposed for the case company to provide the required stability and performance, using the combination of ELK Stack and virtual machines. A good stable scalable log analytic system typically consists of tiers, as shown in Figure 8.

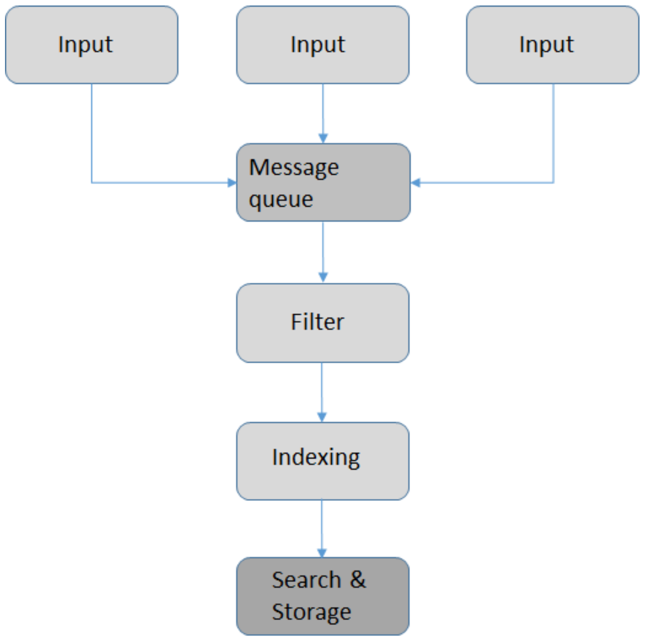


Figure 8 Log analytic tiers

Input tier – consists of data sources which feed the data to the log analytic system.

Messaging queue tier – provides a temporary buffer to protect against surges in traffic.

Filter tier – takes data from the messaging queue, and parses the data from unstructured to required structured format.

Indexing tier – moves the processed data in filter tier to Elasticsearch.

Search & storage tier – consists of log search and storage engine.

Figure 9 shows the cloud based log analytic system design for the case company.

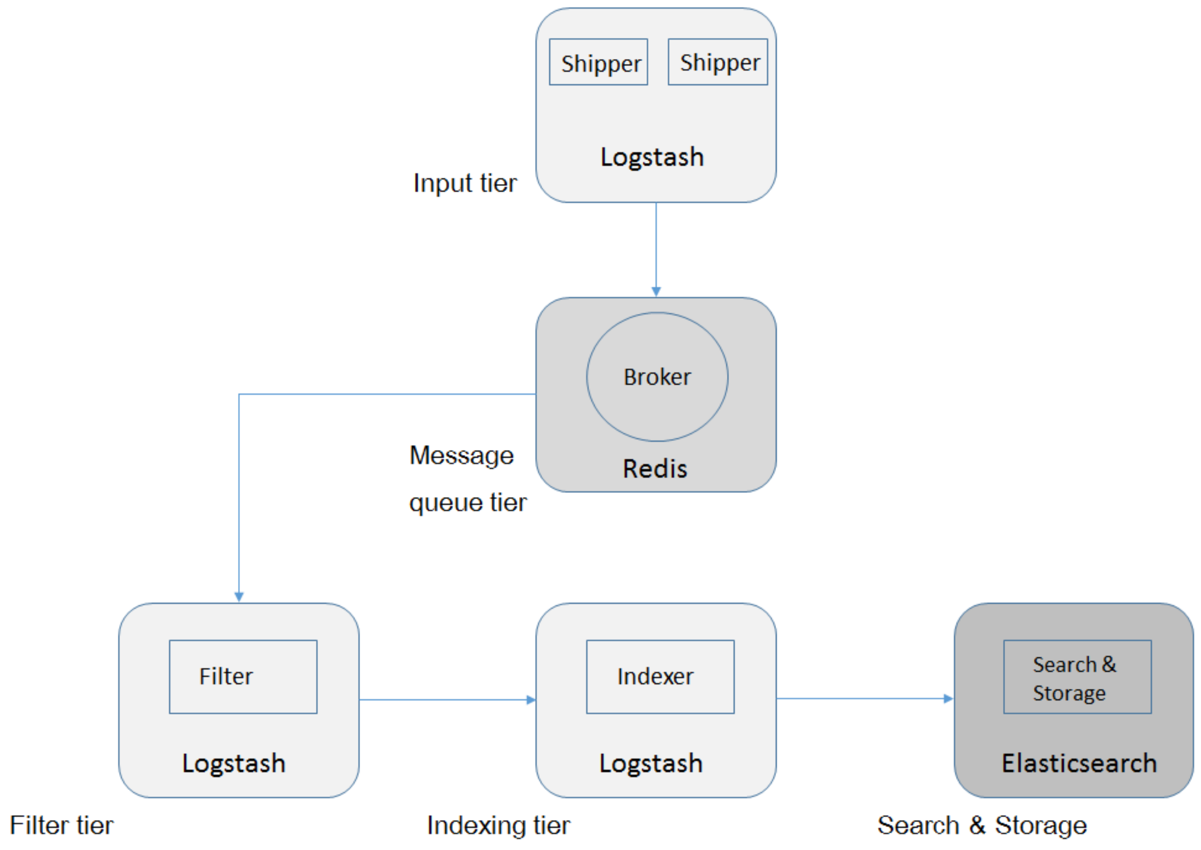


Figure 9 Cloud based log analytic system for case company

For each of these tiers, the services needed to perform the log analysis were identified. Logstash was chosen for the input, filter and indexing tiers while Elasticsearch was chosen for the Search & Storage tier. For the message queue tier, Redis was chosen as a broker.

6.2 Architecture

To achieve a scalable log analytic system for the case company, the cloud based log analytic system was configured with ELK service layers deployed to their own cloud virtual machines as shown in Figure 10.

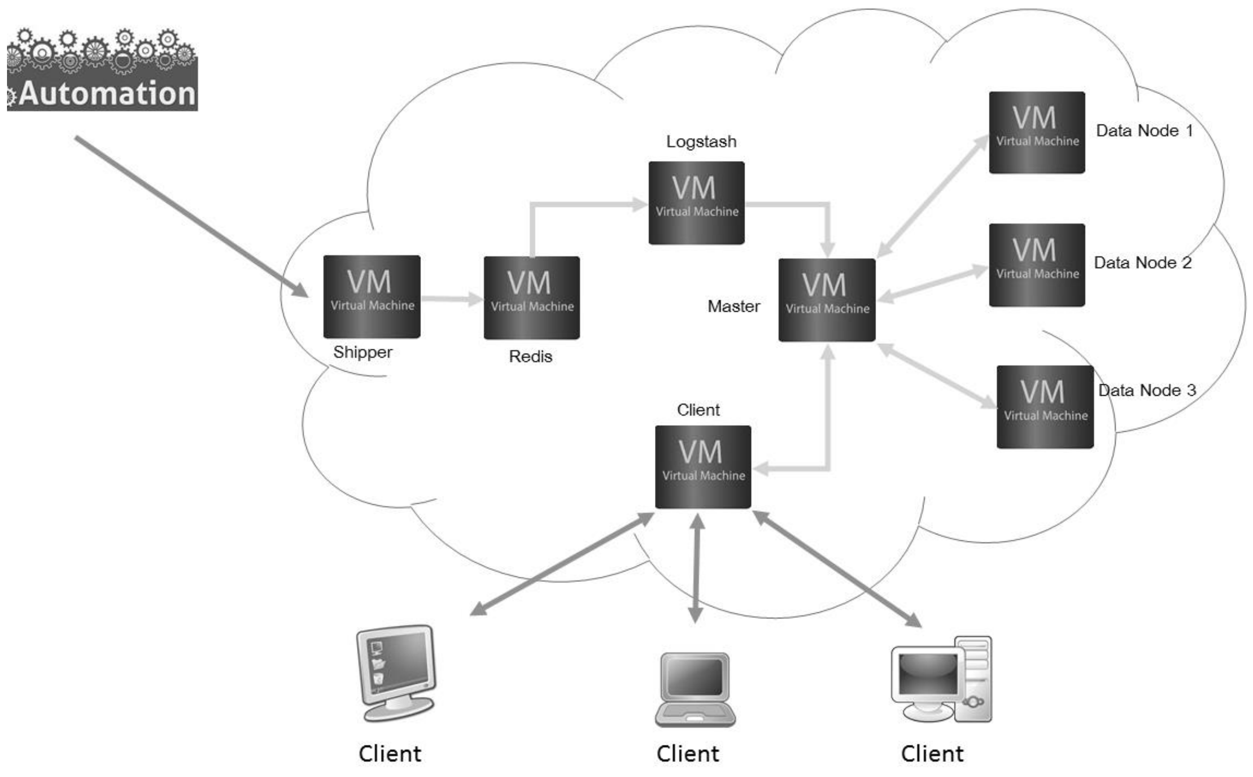


Figure 10 Virtual machines overview in case company's private cloud

By having the services running in their own virtual machines, the resource utilization within the virtual machine is distributed and confined. This provides stability to each of the running services with the added benefit of increased uptime, reliability and better manageability. Additionally, tools and services such as Redis, were introduced to give a robust and overall stability to the log analytic system.

6.3 Logstash Shipper

As more log files are fed into the log analytic system from various test system sources, the log shipper instance was introduced. The log shipper collects the event logs from the log server and forwards these to the destination instance. In this cloud based log analytic setup, the logstash shipper collects the event logs from the automation system and sends them to the Redis instances.

The Logstash shipper uses the Logstash agent itself for forwarding the data and filter plugins to parse and tag the log data. In this case, the shipper tags the log data as per test system source, as shown in the configuration files in Listing 2 (page 34). In this cloud based log analytic system, the log shipper is isolated from the other virtual machine instances. This approach gives advantage to the log analytic system since the log shipper will be using the computing resources of the machine hosting the source data, rather than from other virtual machine instances in the log analytic system.

6.4 Redis as Messaging Queue in Virtual Machine

With the default, basic configuration, Logstash throttles incoming events when the Logstash indexer consumption rates fall below incoming data rates. This leads to buffered events at the data source and backpressure build ups.

To tackle this problem in the case company setup, Redis was introduced. Often Redis is used as a broker in a centralized Logstash installation, which queues Logstash events from remote Logstash shippers. Redis provides a temporary buffer to protect against surges in traffic whenever there is a throughput or message spikes going into Logstash pipelines. In the case company deployment, with huge volumes of data coming from various test systems, adding Redis as a message queue is important in managing the log analytic system.

The Redis message broker is quite memory intensive. Configuring this message queue tier to its own virtual machine and isolating it from other virtual machine instances, will provide the required resources and stability to this broker task.

6.5 Logstash as Filter and Indexing in Virtual Machine

As discussed in Chapter 3.1, Logstash consists of three stages in event processing pipeline: input, filters and outputs. The roles and function of these three stages were also discussed in that chapter. As for the case company setup, with the introduction of Redis as message broker, the functionality of Logstash exists in two places or tiers. The first functionality exists in the Logstash shipper (input tier) which handles the incoming data and places the data in the message queue, in this case Redis. The second functionality exists in this filter and indexing tier, which retrieves the data from the message queue, applies the configured filter and passes it over to Elasticsearch index.

In this tier, the Logstash modifies the unstructured data from various test systems in the case company to structured data. One of the most commonly used filters in Logstash is the Grok filter. As mentioned in Chapter 5.2., the Grok filter uses regular expression computation and it is resource intensive. In case company, the log analytic system handles quite a large number of event logs. Logstash collects unstructured event logs from various test systems. Adding Grok filter plugin to modify huge volumes of data logs to structured form significantly affects the performance of the host. To overcome the increased demand in resources, the Logstash service is configured with these filter plugins to its own virtual machine. By running this service in its own virtual machine, it was possible to provide enough resources to the Logstash services when needed.

6.6 Elasticsearch in Scaled Out Virtual Machines

In this cloud based log analytic system for the case company, the Elasticsearch is scaled out with a few virtual machines to add more nodes to the cluster and to spread the load and reliability between them. Each of the Elasticsearch managements and tasks were separated by configuring them to their own virtual machines.

i. Elasticsearch as Master node in virtual machine

   The master node handles lightweight cluster-wide actions such as creating or deleting an index, tracking which nodes are part of the cluster, and deciding which shards to allocate to which nodes. It is important for cluster health to have a stable master node. Indexing and searching data is CPU, memory and I/O intensive work which can put pressure on a node’s resources. To ensure that the master node is stable and not under pressure, it is a good practise in a bigger cluster to split the roles between dedicated master-eligible nodes and dedicated data nodes.

ii. Elasticsearch as Data node virtual machine

   Data nodes hold the shards that contain the indexed documents. Data nodes handle data related operations like CRUD, search, and aggregations. These operations are I/O, memory, and CPU intensive. To avoid overload, the data was distributed by adding more data nodes, in this case three data nodes. Three data nodes, allow the possibility of taking advantage of shard and replica features of the Elasticsearch. The main benefit of having dedicated data nodes is the separation of the master and data roles.

iii. Client node virtual machine

Taking away the ability to handle master duties and to hold data, the client node can only route requests, handle the search reduce phase and distribute bulk indexing. Essentially, client nodes behave as smart load balancers. Standalone client nodes can benefit large clusters by offloading the coordinating node role from data and master-eligible nodes. Client nodes join the cluster and receive the full cluster state, like every other node, and they use the cluster state to route requests directly to the right places.

Elasticsearch supports HTTP, transport and node protocol to move around the data among the nodes. HTTP protocol for data transport was used in the case company. Using the HTTP protocol provides the possibility to use the Logstash Elasticsearch output plugin to automatically load-balance whenever there is an indexing request.

To speed up the access and load distribution, shards and replicas were used. The data was distributed into multiple data nodes in a cluster for sharding. These shards are then replicated into several data nodes in a cluster. These arrangements guarantee the availability of the data for the log analytic system if there is a failure such as a lost data node.

7 Testing and Evaluation of Solution

The objective of the evaluation is to ensure that all the configured ELK services in the cloud virtual machines are running in an orderly manner, providing continued log analytic service without any interruption and downtime. This is to ensure that the log analytic services are available all the time to the end user of the case company. The uptime of the log analytic services and the virtual machines hosting the log analytic services is a key factor in determining the conclusion.

7.1 Virtual Machine Hardware

As discussed in Chapter 4.2, the choice of the virtual machine flavors is important as it determines the overall performance of the cloud based log analytic system. Configuring the log analytic services to the correct hardware resources should be done effectively. Typically, virtual machines come with different hardware flavor configurations. The case company categorized the flavors based on usage purpose as described below:

General purpose (m1 series): a baseline level of configuration which provides basic level of CPU performance all the time consistently and can burst above this baseline when the workload increases.

Compute optimized (c1 series): optimized for highest performing processors, providing computing resources. These flavors consist of high performance CPUs.

Memory optimized (m2 series): optimized for e.g. in-memory applications and comes with quite large RAM capacity.

Storage optimized (hs1 series): comes with high I/O performance with large amount of disk capacity.

For the case company's cloud based log analytic system, the virtual machine hardware setup for the log analytic systems was configured as shown in Table 3.

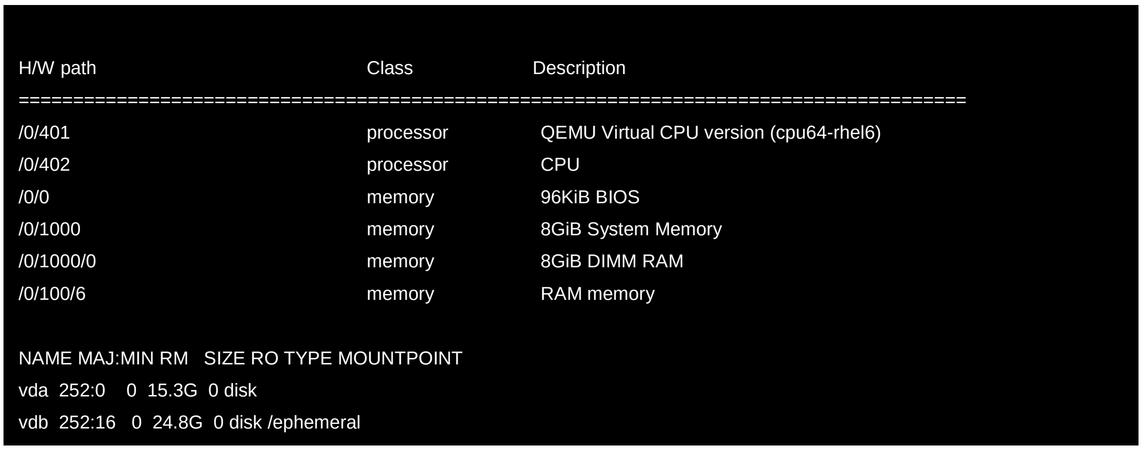
Services	VM Type	CPU	RAM (GB)	Disk Size(GB)
Redis	m1.small	2	8	40
Logstash	c1.medium	4	16	100
Master	m1.medium	2	16	60
Data	m2.xlarge	8	48	400
Client	m1.medium	2	16	60
OS	RHEL 6.6 2014-10-14 2014-10-13 RHEA-2014:1608 2.6.32-504			

Table 3 Virtual machine flavors from case company's private cloud

For the cloud based analytic system, the flavor for the virtual machine was chosen based on the type of the log analytic service it was assigned to. Many of these log analytic services are running on JVM, which consumes a hefty amount of resources for filtering and indexing. Therefore, consideration was given to the choice of RAM and CPU selected. As the virtual machines allow scalability, the design of the cloud based log analytic system for the case company was started with moderate virtual machine capacity.

Redis – Ideally, for standalone Redis virtual machine, a memory optimized virtual machine is the preferred choice, as the entire Redis dataset always resides in RAM.

However, due to limited availability of this flavor in case company's private cloud, only 2 vCPUs with 8GB RAM memory from the general purpose virtual machine was configured as shown in Figure 11.

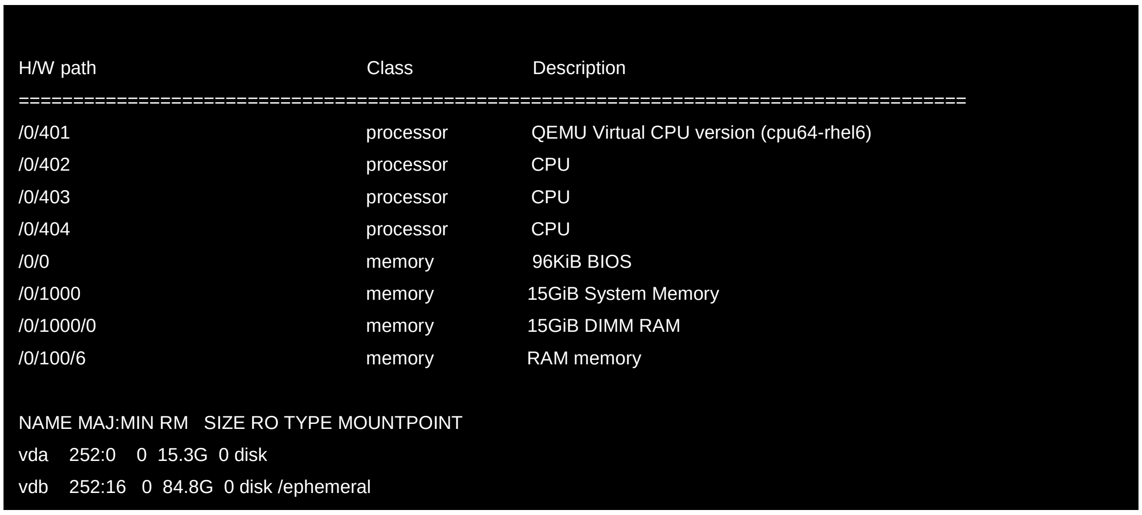

```
H/W path                                Class               Description
=====================================================================================
/0/401                                  processor            QEMU Virtual CPU version (cpu64-rhel6)
/0/402                                  processor            CPU
/0/0                                    memory               96KiB BIOS
/0/1000                                 memory               8GiB System Memory
/0/1000/0                               memory               8GiB DIMM RAM
/0/100/6                                memory               RAM memory

NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda  252:0    0  15.3G  0 disk
vdb  252:16   0  24.8G  0 disk /ephemeral
```

Figure 11 Redis virtual machine flavor

For the case company log analytic system, the general purpose virtual machine is sufficient for the task since Redis was configured to its own virtual machine and it is the only service running in the virtual machine.

Logstash – Logstash runs on JVM and consumes a hefty amount of resources for filtering and indexing. In addition to this, the usage of Grok filter makes the Logstash virtual machine one of the resource intensive virtual machines on both ends, from CPU utilization to RAM limitation. For this reason, a virtual machine from the compute optimized virtual machines with 4 vCPUs and 16 GB of RAM memory was configured as shown in Figure 12.

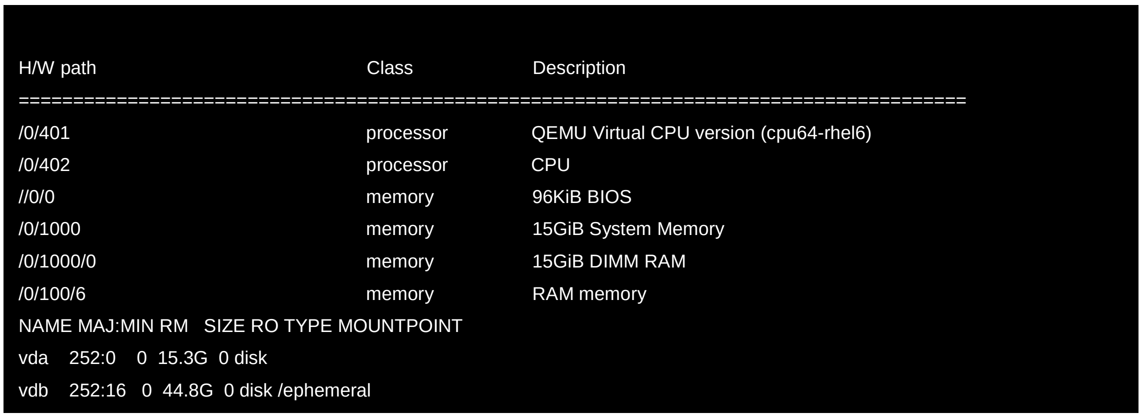


```
H/W path                                Class                  Description
=========================================================================================
/0/401                                  processor              QEMU Virtual CPU version (cpu64-rhel6)
/0/402                                  processor              CPU
/0/403                                  processor              CPU
/0/404                                  processor              CPU
/0/0                                    memory                 96KiB BIOS
/0/1000                                 memory                15GiB System Memory
/0/1000/0                               memory                15GiB DIMM RAM
/0/100/6                                memory                RAM memory

NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda    252:0    0  15.3G  0 disk
vdb    252:16   0  84.8G  0 disk /ephemeral
```

Figure 12 Logstash virtual machine flavor

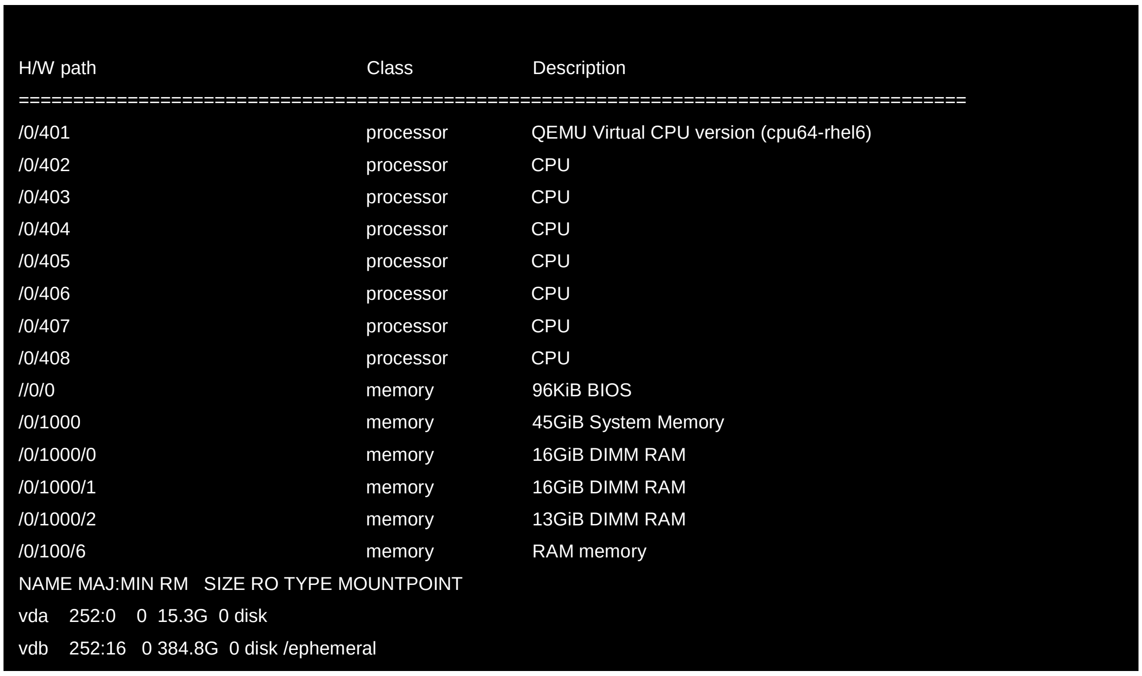
Master – Master performs lightweight cluster operations. From the CPU performance point of view, the processing power of the Master node should be light after separating the Master node from the Data node role. As the Master node runs its services in its own virtual machine, a virtual machine from the general purpose pool with 2 vCPUs and 16GB of RAM memory was configured as shown in Figure 13. This should give the Master node enough resources consistently and added resource when there is a demand (burst) for it.



```
H/W path                                Class                  Description
=========================================================================================
/0/401                                  processor              QEMU Virtual CPU version (cpu64-rhel6)
/0/402                                  processor              CPU
//0/0                                   memory                 96KiB BIOS
/0/1000                                 memory                 15GiB System Memory
/0/1000/0                               memory                 15GiB DIMM RAM
/0/100/6                                memory                 RAM memory
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda    252:0    0  15.3G  0 disk
vdb    252:16   0  44.8G  0 disk /ephemeral
```

Figure 13 Master virtual machine flavor

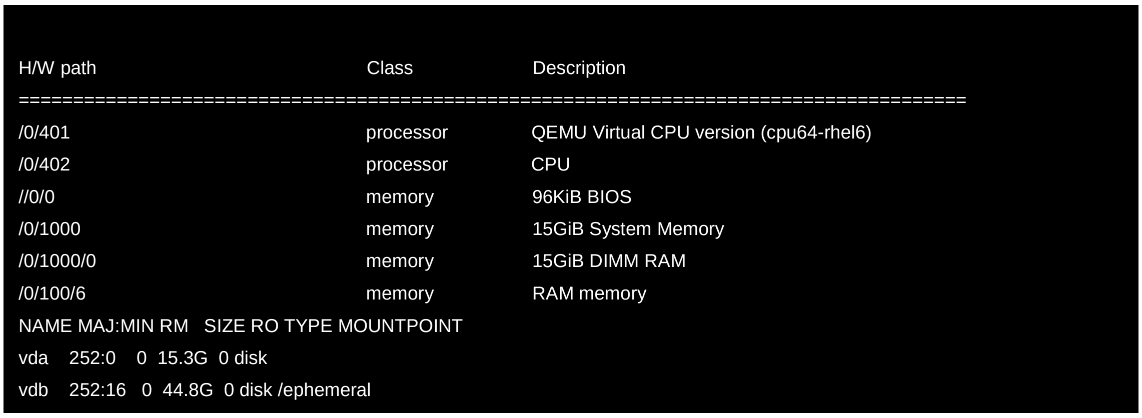
Data – The Data node consists of indexed files. In this node, operations such as data search and aggregations which are I/O memory and CPU intensive were performed. Ideally, a virtual machine from hs1 flavor pool which has the required fast I/O performance and bigger disk capacity can be a good choice. However, virtual machine with this flavor could not be spun due to the limited availability of the virtual machines in the case company’s private cloud. A virtual machine with only 8 vCPUs and 64 GB of RAM from memory optimized pool with large storage capacity was configured, as shown in Figure 14.



```
H/W path                              Class              Description
=========================================================================================
/0/401                                processor          QEMU Virtual CPU version (cpu64-rhel6)
/0/402                                processor          CPU
/0/403                                processor          CPU
/0/404                                processor          CPU
/0/405                                processor          CPU
/0/406                                processor          CPU
/0/407                                processor          CPU
/0/408                                processor          CPU
//0/0                                 memory             96KiB BIOS
/0/1000                               memory             45GiB System Memory
/0/1000/0                             memory             16GiB DIMM RAM
/0/1000/1                             memory             16GiB DIMM RAM
/0/1000/2                             memory             13GiB DIMM RAM
/0/100/6                              memory             RAM memory
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda    252:0    0  15.3G  0 disk
vdb    252:16   0 384.8G  0 disk /ephemeral
```

Figure 14 Data virtual machine flavor

Client – The Client node acts as a good load balancer. It routes requests directly to the corresponding nodes. When it performs searches, it occasionally handles bulk indexing. A virtual machine from the general purpose pool is an ideal choice for these tasks. 2 vCPUs with slightly higher amount of 16GB of RAM capacity for the Client node was configured as shown in Figure 15.



```
H/W path                              Class              Description
=========================================================================================
/0/401                                processor          QEMU Virtual CPU version (cpu64-rhel6)
/0/402                                processor          CPU
//0/0                                 memory             96KiB BIOS
/0/1000                               memory             15GiB System Memory
/0/1000/0                             memory             15GiB DIMM RAM
/0/100/6                              memory             RAM memory
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
vda    252:0    0  15.3G  0 disk
vdb    252:16   0  44.8G  0 disk /ephemeral
```

Figure 15 Client virtual machine flavor

7.2 Log Analytic Service Configurations

Elasticsearch and Logstash require Java in order to run their services. Java 8 was installed in Logstash and in all Data Node virtual machines. For the Logstash node, Logstash version 1.4.2 was installed while for Master, Client and Data Nodes, Elasticsearch version 1.7.2 was installed. Red Hat Enterprise Linux Server release 6.6 was installed as the operating system to all the virtual machines.

7.2.1 Logstash Shipper Configuration

Logstash version 1.4.2 was installed for the shipper virtual machine with default configurations. The task for the shipper instance depends on the content of its configuration file which defines the source of the log file. Table 4 shows the required configuration for the shipper.

Logstash	**logstash 1.4.2**	**Needed Configuration**
	Input:	type
		path
	Event log files	
	Output:-	
		host
	Redis VM	data_type
		key

Table 4 Logstash shipper configuration

The content of the logstash.cnf file at /etc/logstash/conf.d was configured as in Listing 2. The key "TestSystem" in the configuration file is the name of the Logstash that is queued by Redis.

```
input {

# Test system 1 log files
file {
type => "TestSystem1"
path => ["/var/syslog/TS1/Syslog*"]  # watch syslog files
}

# Test system 2 log files
file {
type => "TestSystem2"
path => ["/var/syslog/TS2/Syslog*"] # watch syslog files
}

# Test system 1 alarm files
file {
type => "TestSystem1Alarm"
path => ["/var/syslog/TS1/Alarm*"] # watch alarm files
}
..
..
..
output {
redis {
host => "177.176.136.92" # IP address of Redis VM
data_type => "list"
key => "TestSystem"
}
}
```

Listing 2 Shipper - logstash.cnf configuration

With this configuration, the shipper sends the log messages from the source path defined at each of the test systems to Redis virtual machine. This happens whenever the test system updates or generates new files at source paths, in this case the Syslog and Alarm files.

7.2.2 Redis Configuration

The shipper instances deliver the event logs from each of the test systems to Redis. Logstash instance running logstash service then retrieves these messages from Redis. Redis version 2.8.19 with default parameters was installed for Redis virtual machine. Table 5 shows the needed Redis configuration.

Services	Version	Needed Configuration
Redis	2.8.19	
	Input:	KEYs

Table 5 Redis configuration

It is important to configure the host where Redis connects to so that the Logstash shipper can deliver its messages to Redis. The configuration was defined with bind parameter in the /etc/redis/redis.conf as shown in Listing 3.

```
…….
…….
# Shipper IP
bind 177.176.136.10
…….
…….
```

Listing 3 Redis - redis.cnf configuration

As shown in Listing 3, the Logstash shipper IP address was configured as the bind IP. With this configuration, Redis listen for connection from Logstash shipper virtual machine instead of multiple network interfaces from the server.

7.2.3 Logstash Indexer Configuration

In the case company log analytic system, the Logstash service in the Logstash virtual machine retrieves the log messages from Redis virtual machine. Therefore, it is important to define the Redis host IP as the input to Logstash indexer. The output of the Logstash indexer was directed to the Elasticsearch virtual machine, so that search tasks can be performed and the data can be stored. Logstash version 1.4.2 with default configuration was installed for the Logstash indexer virtual machine. Table 6 shows the configuration for the Logstash indexer.

Services	Version	Needed Configuration
Logstash	2.8.19	
	Input: - Redis	Host data_type key filter
	Output: - Elasticsearch	host cluster

Table 6 Logstash indexer configuration

The task for the Logstash indexer was defined in the configuration file found at /etc/logstash/conf.d. The configuration of the logstash.cnf should be as in Listing 4.

```
input {
  redis {
   host => "177.176.136.92"
   data_type => "list"
   key => "TestSystem"
  }
filter {
   # Parsing for test system log files
   if [type] =~ /^Testsystem..$/ {
    grok {
      match => { "message" => "^%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:sys-
log_program}(?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}$" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
  # Parsing for test system alarm files
   if [type] =~ /^Alarm..$/ {
     csv {
      columns => ["Alarm_ID","Problem","Alarm_Text""]
      separator => "|"
     }
}
output {
  if "_grokparsefailure" in [tags] {
    # write events that didn't match to a file
    file { "path" => "/status/grok_failures.txt" }
  } else {
    elasticsearch {
    host => "177.176.136.94"
    cluster => "Cluster_Pheonix"
    template => "/opt/ls_template.json"
    template_overwrite => true
    }
  }
```

Listing 4 Logstash indexer - logstash.cnf configuration

With the above configuration, Logstash will receive the events from Redis virtual machines, structures the data in the filters section and then sends it to Elasticsearch virtual machine as in the output section. Cluster must be defined for Logstash to send the log events to the intended cluster in Elasticsearch and the IP address of the Master node that performs the cluster management tasks.

The performance of the Grok filter is largely dependent on how successfully it can match the regular expressions. The successful matches can perform differently if compared to unsuccessful matches. Grok filter performance in Logstash service has a direct impact to the performance of the virtual machine. If the regular expression cannot match a line, it tries to find the pattern within the substrings of the first string. This will degrade the performance.

With the introduction of _grokparsefailure in the configuration, there will be indication when ever unsuccessful matches take place in the regular expression. With this configuration, those unsuccessful matches could be directed to a different location for analysis and debugging so that the regular expressions can be corrected.

In addition to _grokparsefailure, anchors (^ and $) were also introduced in the regular expression. The anchors restrict the regular expression to certain positions, for example the start and end of the pattern match. A pattern that does not match the regular expression is not checked or matched. This in turn boosts the performance of the Grok filter. [8]

7.2.4 Master Node Configuration

Elasticsearch version 1.7.2 with default setting was installed in the Master virtual machine. The setting for the master node was defined in the configuration file found at /etc/elasticsearch/elasticsearch.yml. Table 7 shows the needed Elasticsearch configuration to create a Master node.

Services	Version	Needed Configuration
Elasticsearch	1.7.2	
	Role: Master	node.master node.data node.name cluster.name network.host
	Output: Data Nodes	discovery.zen.ping.unicast.hosts

Table 7 Master node Elasticsearch configuration

The configuration files mainly contain node specific settings needed for the node to be discovered and join a cluster. To create a master node, the *node.master* parameter is set to true, while *node.data* is set to false. This takes away the ability of the node to hold any data. Cluster name is mandatory for all the nodes to joint and share the cluster. The default name *elasticsearch* was changed to the cluster name as defined in Logstash indexer configuration.

In a single development node server, the Elasticsearch by default sets the network host to loopback addresses. For the nodes to communicate in the multimode setup such as in this cloud based log analytic system, the non-loopback address must be defined. In this case the IP of the Master node virtual machine is set to the *network.host* parameter. The needed configuration of elasticsearch.yml should be as in Listing 5, with the rest of the parameters set to default.

```
# Cluster name
cluster.name: Cluster_Pheonix

# Node name
node.name: Master

#Node role
node.master: true
node.data: false

# Node host IP
network.publish_host: 177.176.136.94

# Data Nodes
discovery.zen.ping.unicast.hosts: 177.176.136.95,177.176.136.96,177.176.136.97
```

Listing 5 Master Node - elasticseacrh.yaml configuration

Elasticsearch searches for a cluster in a local server by default. In the case company log analytic setup, three data nodes were configured. For the master node to be able to connect to the data nodes, the node IP list of the data nodes is given in the *discovery.zen.ping.unicast.hosts* parameters.

7.2.5 Client Node Configuration

As in the Master Node, cluster name is mandatory for the client node to join the cluster. Cluster name as defined in Logstash shipper was configured for the client node. Table 8 shows the needed Elasticsearch configuration to create a Client node.

Services	Version	Configuration
Elasticsearch	1.7.2	
	Client	node.master node.data node.name: cluster.name:
	Output: - Master node	discovery.zen.ping.unicast.hosts
Kibana	4.1.1	

Table 8 Client node Elasticsearch configuration

Elasticsearch version 1.7.2 was installed to the Client virtual machine with default settings. The setting for the master node was defined in the configuration file found at /etc/elasticsearch/elasticsearch.yml. The needed configuration of elasticsearch.yml for client node should be as shown in Listing 6 with the rest of the parameters set to default.

```
# Cluster name
cluster.name: Cluster_Pheonix

# Node name
node.name: Client

#Node role
node.master: false
node.data: false

# Node host IP
network.publish_host: 177.176.136.98

# Data Nodes
discovery.zen.ping.unicast.hosts: 177.176.136.94
```

Listing 6 Client Node - elasticsearch.yaml configuration

To create a client node, the *node.master* and *node.data* parameters are set to false. This takes away the ability of the node to hold any data or to become a master node. For the client to be able to perform the search, the Master node should be discoverable from the client node.

7.2.6 Data Node Configuration

In this case company log analytic setup, three data nodes were configured as shown in Figure 10, to take advantage of shard and replica features of Elasticsearch. Since Elasticsearch service is also running in the data nodes, the needed setting is the same as in the Master Node. The difference is in the role of the nodes and the IP addresses of the server nodes that join the cluster.

Table 9 shows the needed Elasticsearch configuration for all the three data nodes.

Services	**Version**	**Needed Configuration**
Elasticsearch	1.7.2	
	Role: - Data	node.master node.data node.name cluster.name network.host
	Output: - Data Nodes Master node	discovery.zen.ping.unicast.hosts:

Table 9 Data Nodes Elasticsearch configuration

Elasticsearch version 1.7.2 was installed to all the data node virtual machines with default settings. To create a data node, the *node.master* parameter is set to false, while *node.data* is set to enable. This ensures that the data nodes hold the shards that contain the indexed documents to handle data related operations like search and aggregations. The needed configuration of elasticsearch.yml for each data nodes should be as in Listing 7 with the rest of the parameters set to default.

```
Data Node 1 - elasticsearch.yaml configuration
# Cluster name
cluster.name: Cluster_Pheonix

# Node name
node.name: Data_Node_1

#Node role
node.master: false
node.data: true

# Node host IP
network.publish_host: 177.176.136.95

# Data Nodes
discovery.zen.ping.unicast.hosts: 177.176.136.94,177.176.136.96,177.176.136.97

Data Node 2 - elasticsearch.yaml configuration
# Cluster name
cluster.name: Cluster_Pheonix

# Node name
node.name: Data_Node_2

#Node role
node.master: false
node.data: true

# Node host IP
network.publish_host: 177.176.136.96

# Data Nodes
discovery.zen.ping.unicast.hosts: 177.176.136.94,177.176.136.95,177.176.136.97

Data Node 3 - elasticsearch.yaml configuration
# Cluster name
cluster.name: Cluster_Pheonix

# Node name
node.name: Data_Node_3

#Node role
node.master: false
node.data: true

# Node host IP
network.publish_host: 177.176.136.97

# Data Nodes
discovery.zen.ping.unicast.hosts: 177.176.136.94,177.176.136.95,177.176.136.96
```

Listing 7 Data Nodes - elasticsearch.yaml configuration

As shown in the Listing 7, for the data nodes to be live and contactable with the Master node, the master node IP is listed in the *discovery.zen.ping.unicast.hosts* parameters.

7.3 Test Method and Results

For the case company, the proposed and implemented cloud based log analytic solution should be able to handle the events that are coming from their automation system. Their old dedicated physical server based log analytic system cannot handle large numbers of events and crumbles under load. To evaluate the case company log analytic setup, each of the log analytic services was configured to its own virtual machine, as shown in Table 3 (page 29), with settings as described in Chapter 7.2, and the case company automation system was allowed to feed in the data from various test systems. At the time of writing this thesis, the configured log analytic system has already been in use by the end user at the case company for a few months.

In Elasticsearch deployment and monitoring, the node stats APIs can be used to access various node statistics. By using the API, the performance and uptime of each individual log analytic service and virtual machine were monitored and logged.

i. Log analytic services uptime

Table 10 shows the measured uptime of the running Elasticsearch and Logstash services. In Table 10, it can be seen that all the virtual machines and the configured log analytic services have been running with average measured uptime of 3 months.

Service	Measured uptime
"name": "Logstash", "ip": "177.176.136.93", "version": "1.5.2",	49 days, 19 hours, 42 minutes and 44 seconds (1 month, 19 days)
‘ "name": "Master", "ip": "177.176.136.94", "version": "1.5.2",	136 days, 22 hours, 2 minutes and 6 seconds (4 months, 14 days)
"name": "Node1", "ip": "177.176.136.95", "version": "1.7.2",	136 days, 22 hours, 11 minutes and 1 second (4 months, 14 days)
name": "Node2", "ip": "177.176.136.96", "version": "1.7.2",	136 days, 22 hours, 15 minutes and 23 seconds (4 months, 14 days)
name": "Node3", "ip": "177.176.136.97", "version": "1.7.2",	136 days, 23 hours, 59 minutes and 8 seconds (4 months, 14 days)
"name": "Client", "ip": "177.176.136.98", "version": "1.7.2",	107 days, 21 hours, 6 minutes and 26 seconds (3 months, 16 days)

Table 10 Elasticsearch and Logstash measured uptime

When configuring the log analytic services, continuous debugging and improvement of the scripts lead to the restart of the log analytic services at different times. Hence the test results show different uptime for the log analytic services if compared to each other, for e.g. Logstash service running in Logstash virtual machine.

ii. Logs handled during evaluation period

Additional information such as indices and information about the data (documents) that the log analytic system Data node handled during the evaluation period were also noted, as shown in Table 11.

Data node	Documents counts	Storage size	Indexes	Search performed by client
DataNode1	"indices": { "docs": { "count": 197736748, "deleted": 35293 }	store": { "size_in_bytes": 146449087548, … }	"indexing": { "index_total": 145809089, … }	"search": "query_total": 31470, … }
DataNode2	"indices": { "docs": { "count": 196803691, "deleted": 38430 },	"store": { "size_in_bytes": 146128189846,	"indexing": { "index_total": 141607843,	"search": { "query_total": 31305, … }
DataNode3	"indices": { "docs": { "count": 182901281, "deleted": 49226 },	"store": { "size_in_bytes": 135402081009,	"indexing": { "index_total": 781360429,	"search": { "query_total": 214637, … }

Table 11 Data handled by Data node virtual machines

Indices section shows the number of documents stored in the indices and also the number of documents that will be deleted from the segment. In general indices indicates the number of documents handled by the log analytic system. For the case company, during the evaluation period, the log analytic system handled around 197 million documents which includes the replica shard. The amount of physical storage consumed by the indices are 146 GB of disk space as shown in Store section.

Indexing shows the number of documents that have been indexed by the Logstash service in the log analytic system. During the evaluation period, 145 million of documents were indexed by the Logstash service. The number of queries handled by the client are around 31 thousand as shown in the search section of the node stats. To summarize the indices information, it can be seen from the Table 11 that the cloud based log analytic system for the case company managed to handle 197 million of documents, indexing 145 million of them and performed 31 thousand queries without any breakdown in log analytic services.

7.4 Summary

From the test results, it can be clearly seen that the objective of the thesis was met, i.e. providing uninterrupted log analytic service to end user with cloud based log analytic system. All the virtual machines and the configured log analytic services have been running for an average of 3 months, exceeding the moderate 1 month benchmark setting that was set when the cloud based log analytic system was set up.

By provisioning the log analytic services with their own virtual machines, the log analytic service was able to provide the required resources and stability.

Redis services with their own virtual machine manage to handle any back-pressure build ups with their available resources. On the other hand, Logstash services with their own virtual machine managed to withstand the required processing power when indexing and converting the huge amount of unstructured data to structured data with its Grok filters.

Scaling out the Elasticsearch nodes to their own virtual machine also brings considerable improvement to the log analytic system. Better load balancing and reliability were achieved by configuring each of the Elasticsearch nodes as master, data and client nodes with its own virtual machine. Master node could perform better cluster management with this configuration while the client node, with its main task of performing route request and search, exhibited good performance.

For providing continued log analytic service without interruption and downtime, the cloud based log analytic system, as discussed in Chapters 7.1 and 7.2, is sufficient for the case company organisation. From the test results, it can be observed that there were no uptime issues with the log analytic system, and resources remain available for future needs.

For future optimization, there are a few areas identified which could improve the performance of the log analytic system. As discussed above, one of the main benefits of this cloud based log analytic system is scalability. The log analytic system can be scaled out in future, whenever there is an increase in number of event logs to handle or whenever there are additional test systems needed in the case company automation system.

As the scope of the evaluation in this thesis is only to measure the uptime of the running log analytic services, the hardware used for the test is moderate in capacity and fixed throughout the test. The capacity performance of these virtual machines were not evaluated. The test could be repeated with various virtual machine flavors with a mixed combination of log analytic services to find the optimum performance configuration. This information will be useful when scaling is needed for the log analytic system and for full utilization of the available computing resources.

The current configured cloud based ELK log analytic system for the case company represents the single architecture log analytic design, as shown in Figure 16.

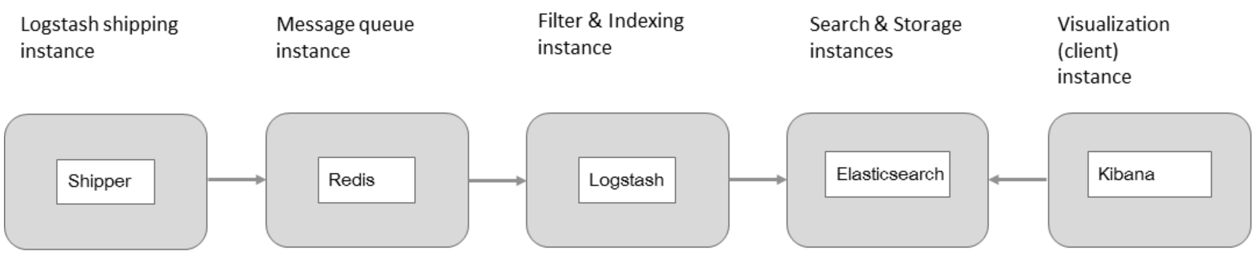

Logstash shipping instance
Message queue instance
Filter & Indexing instance
Search & Storage instances
Visualization (client) instance
Shipper
Redis
Logstash
Elasticsearch
Kibana

Figure 16 Logstash - Single input

With the log analytic system as shown in Figure 16 above, the data from the automation system is fed to the log analytic system with a single input shipper. If that single shipper input service is not available, configured Logstash instances for that specific input will not have any data to process. Hence, an unavailability situation occurs for the whole log analytic system. For high Logstash availability, the log analytic system can be further improved to have a parallel architecture. In this approach, multiple inputs can be configured to Logstash instances as shown in Figure 17.

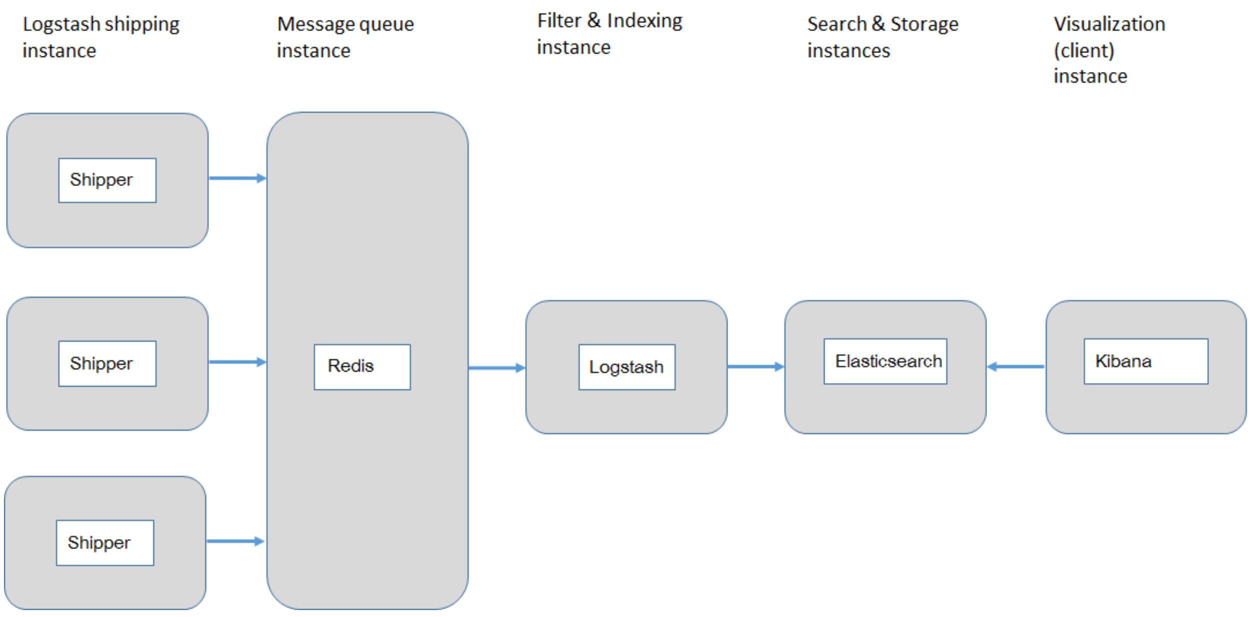

Logstash shipping instance
Message queue instance
Filter & Indexing instance
Search & Storage instances
Visualization (client) instance
Shipper
Shipper
Shipper
Redis
Logstash
Elasticsearch
Kibana

Figure 17 Logstash - Multiple inputs

Multiple inputs with high Logstash availability provide the opportunity to further improve the reliability of the log analytic system, as shown in Figure 18.

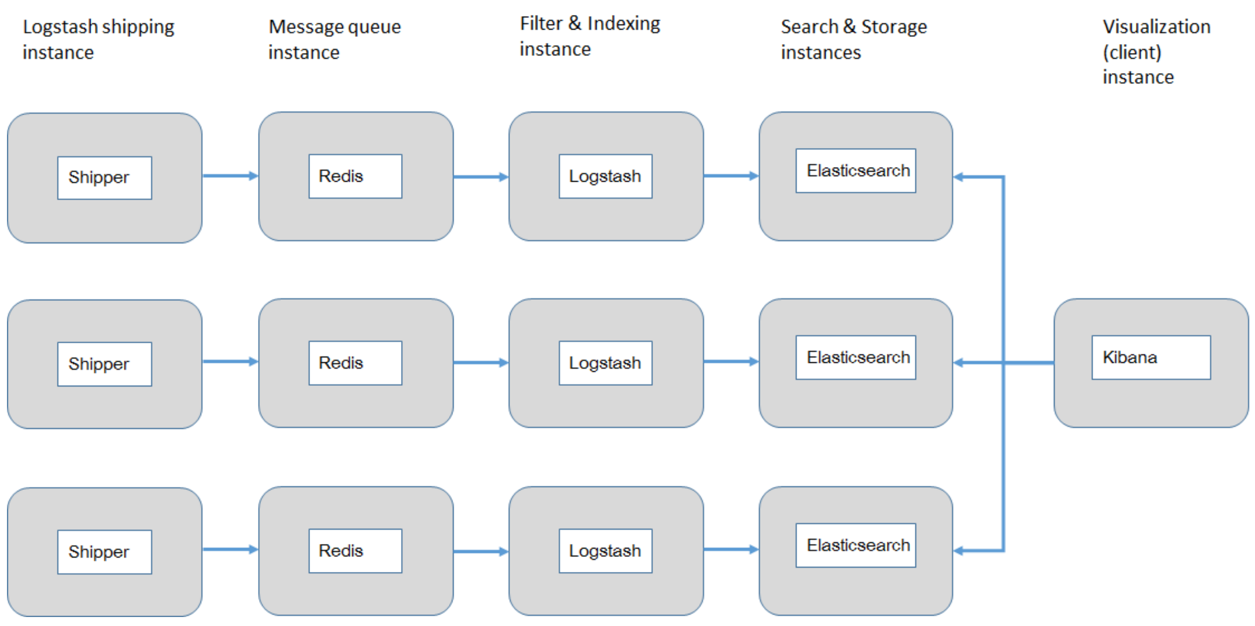


Figure 18 Logstash - Parallel architecture

As shown in Figure 18, each of the inputs can be configured with its own Logstash instance providing horizontal scalability. Also, by parallelizing the data pipeline, high reliability can be provided with no single point of failure.

8 Discussion and Conclusions

The main objective of the thesis was to develop a cloud based log analytic system, where all the log analytic services are running in the virtual machines, providing continuous log analytic service without interruption or downtime. The motivation for this was the reliability challenges posed by the old dedicated physical server log analytic system in analysing the case company's growing volumes of events data.

It quickly became apparent that a log analytic system hosted in single workstation, as configured in case company, is not the optimal solution for big data computing, posing reliability issues. Though the case company already developed an ELK-based log analytic system with a dedicated physical server, there were several shortcomings in the design and implementation of the log analytic system.

The case company chose a single host dedicated physical server without any consideration for future expansion. Underperforming hardware and configuring all the services in a single host workstation introduced workload problem and reliability issues. Filtering and indexing could be optimized to minimize resource clogging.

Keeping in mind the shortcomings (Chapter 5.2) related to old dedicated physical workstation, a scalable cloud based log analytic system was proposed and developed for the case company. Virtual machines with their advantages over dedicated physical servers (Chapter 4.3) were used to host the log analytic services. Each of the log analytic elements and services (Chapter 6.1) was identified and configured to their own virtual machines, eliminating any CPU processing resource issues due to hardware limitations.

Also, new elements such as Redis were introduced for message queueing to prevent backpressure and throttling by Logstash.

Instead of all the Elasticsearch nodes configured in a single host, Elasticsearch was scaled out with virtual machines to add more nodes to the cluster and to spread load between them. Each of the Elasticsearch managements and tasks were separated and configured to its own virtual machine. Filtering and indexing were also improved and simplified to ensure these tasks are lightweight in terms of resource utilization.

During testing and evaluation, it was observed that running the log analytic services in their own virtual machines improves the overall stability and reliability of the log analytic system. Elasticsearch running in its own virtual machine and the scaling out of the Elasticsearch Master, Client and Data nodes to their own virtual machines brings considerable improvement in cluster management.

With this new cloud based log analytic system in the case company, an average virtual machine uptime of 3 months was achieved, exceeding the expectation. At the time of writing of this thesis, the log analytic services were still running in orderly manner without any interruptions or downtime.

Overall, it can be concluded that this thesis has achieved its primary goal, which was to provide continuous log analytic services without downtime. This was achieved during the evaluation period, and also beyond that. This is the first step of cloudification for the log analytic system in the case company and, with these promising results, plans are already in the pipeline for moving all other systems to cloud based solutions.

9 References

[1] "Log analysis," [Online]. Available: https://www.dice.com/skills/Log+analysis.html. [Accessed Sep 2016].

[2] "An Overview of the syslog Protocol," [Online]. Available: http://www.ciscopress.com/articles/article.asp?p=426638.

[3] "Centralized Logging Architecture," [Online]. Available: http://jasonwilder.com/blog/2013/07/16/centralized-logging-architecture/. [Accessed Sep 2016].

[4] "Elastisearch," [Online]. Available: https://www.elastic.co/products. [Accessed Sep 2016].

[5] "Basic Concepts Elasticsearch," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html. [Accessed Sep 2016].

[6] "What is virtual machine?," [Online]. Available: http://searchservervirtualization.techtarget.com/definition/virtual-machine. [Accessed Sep 2016].

[7] T. Levy, "5-logstash-pitfalls-and-how-to-avoid-them," [Online]. Available: http://logz.io/blog/5-logstash-pitfalls-and-how-to-avoid-them/.

[8] J. Duarte, "do-you-grok-grok," [Online]. Available: https://www.elastic.co/blog/do-you-grok-grok.

[9] "Life Inside a Cluster," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/distributed-cluster.html. [Accessed Sep 2016].

[10] "Node," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html. [Accessed Sep 2016].

[11] "Adding Logstash Filters To Improve Centralized Logging," [Online]. Available: https://www.digitalocean.com/community/tutorials/adding-logstash-filters-to-improve-centralized-logging. [Accessed Sep 2016].

[12] "Grok debugger," [Online]. Available: http://grokdebug.herokuapp.com/. [Accessed Sep 2016].

[13] “Welcome to the ELK Stack: Elasticsearch, Logstash, and Kibana,” [Online]. Available: https://qbox.io/blog/welcome-to-the-elk-stack-elasticsearch-logstash-kibana. [Accessed Sep 2016].

[14] “Understanding the Difference Between Physical and Virtual Networking,” [Online]. Available: http://www.windowsnetworking.com/articles-tutorials/netgeneral/understanding-difference-between-physical-virtual-networking.html. [Accessed Sep 2016].

[15] “Virtualization: Physical vs. Virtual Clusters,” [Online]. Available: https://technet.microsoft.com/en-us/library/hh965746.aspx. [Accessed Sep 2016].

[16] “Elasticsearch Reference Basic Concepts,” [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html. [Accessed Sep 2016].

[17] “Ways to Use Log Data to Analyze System Performance,” [Online]. Available: https://blog.logentries.com/2014/06/5-ways-to-use-log-data-to-analyze-system-performance/. [Accessed Sep 2016].

[18] “Ways to index relational data in Elasticsearch,” [Online]. Available: http://voormedia.com/blog/2014/06/four-ways-to-index-relational-data-in-elasticsearch. [Accessed Sep 2016].

[19] “Tips on ElasticSearch Configuration for High Performance,” [Online]. Available: https://www.loggly.com/blog/nine-tips-configuring-elasticsearch-for-high-performance/. [Accessed Sep 2016].

[20] “Redis: What You Need To Know,” [Online]. Available: http://www.interworx.com/community/redis-what-you-need-to-know/. [Accessed Sep 2016].

[21] “Introduction to Redis - In Memory Key Value Datastore,” [Online]. Available: Introduction to Redis - In Memory Key Value Datastore. [Accessed Sep 2016].

[22] “Grok constructor,” [Online]. Available: http://grokconstructor.appspot.com/. [Accessed Sep 2016].