Cloudsearch, Dell Cloud service application

Ajitpal Singh

Submitted as part of the requirements for the degree
of MSc in Cloud Computing
at the School of Computing,
National College of Ireland
Dublin, Ireland.

September 2013

Supervisor Dr. Horacio Gonzalez-Velez

Abstract

Cloud platform is becoming emerging trend in IT infrastructure, service delivering and multi-layered resource sharing architecture which includes SaaS (Software as a service), PaaS (Platform as a service) and Iaas (Infrastructure as a service). With the increasing popularity of cloud computing, user store large amount of data in form of documents, text files, database etc. In Cloud, the large number of applications are running and storing logs on multiple servers and different log files. As today services are getting more decomposed, each layer in cloud stack generates different logs for network, applications, database etc. on different machines. The log provides the important piece of analytical data. At any point in time, cloud provider, owner or application developer need to understand the status of different components, monitor business process and find user documents in real time. Cloudsearch implements a Proof-of-Concept (POC) system to analyse the user documents, logs and folders in real time from different machine using Elasticsearch server which is open source distributed real time search and analytics engine. Furthermore, we have developed rivers (i.e. extension) for Elasticsearch server and extended its data searching capabilities for different operating system and software platform. Cloudsearch extends it capability by searching different OS in private cloud environment using SSH and SAMBA. Cloudsearch also allows users to search documents in their SharePoint server also. It provides simple user interface to add connectors or machine to index the data and allow user to search the data in real time. In this paper we describe the design and implementation of the Cloudsearch. Furthermore, we also describe how to extend Cloudsearch for different software and services.

Contents

Declaration

I confirm that the work contained in this MSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed ..... Date .....

Ajitpal Singh

Contents

List of Tables

List of Figures

Disclaimer

THIS TECHICAL PAPER IS FOR INFORMATIONAL PURPOSES ONLY, AND MAY CONTAIN TYPOGRAPHICAL ERRORS AND TECHNICAL INACCURACIES. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND.

September 15, 2013

Chapter 1

Introduction

Cloud computing has brought major advancements and innovations in IT services Industry. [4] In very short period of time, the term cloud computing has defined a new style of computing in which resources are easily virtualized and provides scalable services that can be accessed over the network. Cloud computing is reshaping our information structure and today is delivered as a traditional utility manner and consumers are paying service provider on the basis of utility usage [1]. Running any application in cloud requires insight into current on-going process and workflow. As cloud computing and big data are evolving, a search engine is required that search documents in cloud. However, today typically most search engine is web based like Google and Yahoo. It is today necessary to search the documents, log files, SQL and NoSQL database, such cloud data is very important for both user and service provider perspective [6]. Logs and document provides important insight into data. Logs are provided to developer, operation team and security analyst. To find specific content in documents can be significant problem when trying to very large subset of them.

We have developed Cloudsearch application which is a proof of concept for Dell cloud customer to search their private data on different platform inside Dell cloud environment. Cloudsearch application is based on Elasticsearch server, which is distributed real time search engine, for effectively searching and indexing document in cloud environment. We have extended the search capabilities of Elasticsearch by extending its plugin to search data on Linux, Windows, SharePoint and Database platform. Elasticsearch engine uses Lucene framework to provide full text based searching using powerful query language, auto search and conflict management for document processing.

In cloud environment, application store logs in multiple folders and machine. Each layer in cloud application generates the log. As customer point of view, to manage and check such log on regular interval could be a big problem [8]. Using Dell cloud search template user can create indexes for all log folders and schedule them to read it real time. We have also provided search capabilities for MySQL database and SharePoint document library. Cloudsearch extends Elasticsearch to read user document in SharePoint library and index it. Cloudsearch provide single place to manage and control all data indexes and search them. Using Cloudsearch template in Dell cloud services user easily increase Elasticsearch node instance as data in their environment grow proportionally. Furthermore, we have developed web based frontend which is integrated with Elasticsearch node.

In next section we will discuss the Cloudsearch application in general and provide details about framework and technology implemented. After that we will see how Cloudsearch application work and its internal design specifications. Again in next section, we will show the implementation of the Cloudsearch application and its working guideline. In last we do some evaluation and testing on the Cloudsearch and finally we conclude.

Chapter 2

Background

Cloudsearch application search data inside DELL cloud platform. Once cloud users are logged inside the cloud, users can install Cloudsearch as a template. A templates are services that user can install inside their cloud platform offered by DELL. Once the service are installed user can control the services as well its configuration. Cloudsearch templates install the Elasticsearch cluster with single VM node and integrate web frontend in web console. User can create search index for virtual machine and schedule them. Search can be deployed for both Windows and Linux platform. Cloudsearch is a POC application which will focus on data search using SSH for Linux, SMB for Window, http for SharePoint service and database from MySQL.

2.1 Technology overview

- **Interfaces**
    1. Web based frontend in servlets
    2. Extended FS-River plugins
    3. Jetty Server
- **Platforms**
    1. Linux/Ubuntu
    2. Servlets web framework
- **Development Platform**

1. Java (Eclipse Kepler project)
2. Maven
3. MySQL
4. Jenkins / GIT Workflow
5. JSch API implementation for SSH2 (V 0.1.50)
6. JCIFS API implementation for SMB (V 1.3.17)

2.2 Elasticsearch

Elasticsearch is open source server which provides distributed and real time search capabilities, it also known as document database. It implements Lucene as its backend for document parsing and structuring [9]. Elasticsearch has following features:

- **Distributed:** Elasticsearch server can start with small data and can be scaled horizontally depending upon the application needs and requirements. If more capacity is needed, we can just add more nodes to store the data.
- **High Availability:** Elasticsearch cluster is error resilient, if any error is detected, it will automatically remove the failed nodes and re-organize itself to make sure data is safe and accessible.
- **Full Text Search:** It provides full query based search capabilities using Lucene.
- **Document Oriented:** Elasticsearch store data or document in JSON format. All documents are indexed by default and provide result at very fast speed.
- **Schema-free:** Documents can be easily stored in JSON format and Elasticsearch will automatically detect the data structure, its data types and index the data accordingly. User can also define its own mapping and can change if required. Documents are versioned for any changes and provide the conflict management automatically.

2.2.1 Elasticsearch Term

We define some basic terminology for the Elasticsearch server.

- **Index:** It is like database, where Elasticsearch server stores the data. In Relational DB terms the index is like a Table. It has mapping which defines multiple types.
- **Document:** Every document is a Elasticsearch is a JSON document. It represents row in table or index. Every document has a type and id field.
- **Type:** Each document has list of fields that can be specified for that type. It can viewed as column as in DB.
- **Shard:** Data in Elasticsearch is divided into shards as smaller parts and each shard is a separate Lucene index. Shards can reside on same server or can be distributed among different server. Shards give Elasticsearch horizontal scaling capabilities.
- **Replica:** Each shard can have one replica or mirror copy to store the information and data.

Elasticsearch cluster architecture is distributed in nature, fault tolerant, can have single node cluster and Elasticsearch can also select automatic leader node from group of nodes. It provides multicast and unicast discovery methods. The server configuration can be changed on runtime without shutting down the node. In cluster node, each Elasticsearch node must restart after river plugins are configured [9].

Elasticsearch use REST based API, user can manage the index, check server health, updates node, search data and can manage the cluster from its API. REST like architecture defines that every request is directed to resource or a concrete entity which is indicated by the path of that object. REST is built upon HTTP protocol, hence it support all the methods of HTTP like get, put, post, delete etc.

2.2.2 Storing Data in Elastic Search

We can implement simple JSON to store the data in the server using REST API. Following sample uses CURL tool to add index to the server. We created (Listing 2.1) a simple JSON example of book information.

```
1 {
2   "title": "Handbook of Cloud Computing",
3   "pages": "656",
4   "edition": 2010,
5   "isbn" : "1441965238",
```

```
  "author" : "Springer",
  "publish_date" : "September 26, 2010",
  "tags" : ["Cloud", "PaaS", "IaaS"]
}
```

Listing 2.1: book json

This example will create index with name as library and type as cloudcompuitng with id 1, when we execute following command data is saved in Elasticsearch node(Listing 2.2)

```
curl -XPUT http://localhost:9200/library/cloudcomputing/1 -d {
  "title": "Handbook of Cloud Computing",
    "pages": "656",
  "edition": 2010,
    "isbn" : "1441965238",
  "author" : "Springer",
  "publish_date" : "September 26, 2010",
  "tags": ["Cloud", "PaaS", "IaaS"]
}
```

Listing 2.2: Create Index

When the index is stored properly in the server, it will create 5 shards by default and with 1 replica on single server. If the data is stored properly on the server, it returns following JSON response (Listing 2.3)

```
{
  "ok":true,
  "_index":"library",
  "_type":"cloudcomputing",
  "_id":"1",
  "_version":1
}
```

Listing 2.3: JSON reponse for index

2.2.3 Retrieving data in Elasticsearch

We can retrieve the documents stored in the server using following url `curl -XGET http://localhost:9200/library/cloudcomputing/1`. The server will return the JSON in response (Listing 2.4)

```
{
  "_index" : "library",
  "_type" : "cloudcomputing",
  "_id" : "1",
```

```
  "_version" : 1,
  "exists" : true,
  "_source" : {
 "title": "Handbook of Cloud Computing",
  "pages": "656",
  "edition": 2010,
  "isbn" : "1441965238",
  "author" : "Springer",
  "publish_date" : "September 26, 2010",
  "tags": ["Cloud", "PaaS", "IaaS"]
}
```

Listing 2.4: JSON retrieved from index

2.2.4 Searching data in Elasticsearch

Elasticsearch provides simple as well as advance search query with REST based API. It provides JSON data as response from query DSL. For example to search the title cloud computing in node, we form the following query.`curl -XGET 'localhost:9200/library/cloudcomputing/search?q=title:cloudcomputing'`. Whereas in response, we will get the following results.

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.39178301,
    "hits" : [ {
      "_index" : "library",
      "_type" : "cloudcomputing",
      "_id" : "1",
      "_score" : 0.39178301, "_source" : {
          "title": "Handbook of Cloud Computing",
    "pages": "656",
    "edition": 2010,
    "isbn" : "1441965238",
    "author" : "Springer",
    "publish_date" : "September 26, 2010",
    "tags": ["Cloud", "PaaS", "IaaS"] }
```

```
    } ]
  }
}
```

Listing 2.5: JSON search result

Score represent the relevancy of the search result. If the results are more than one, the total object in JSON will change. Result also display total time took in milliseconds and shards used to get the result dataset with their success and failure results.

2.2.5 Cluster Administration

Monitoring is important aspect of Elasticsearch cluster because it generates data that can be used to judge the success and to detect possible problem and prevent them before they occur. The REST based API of Elasticsearch provides very detail information that allow to monitor the entire cluster or single node. It includes statistics, information about the server and node parameters. The cluster health API returns the current cluster or node information about its status, number of nodes, primary shards, replica etc. Following command is `curl -XGET localhost:9200/_cluster/health`. It returns as following information.

```
{
   "cluster_name" : "FirstNode",
   "status" : "green",
   "timed_out" : false,
   "number_of_nodes" : 1,
   "number_of_data_nodes" : 1,
   "active_primary_shards" : 5,
   "active_shards" : 5,
   "relocating_shards" : 0,
   "initializing_shards" : 0,
   "unassigned_shards" : 0
}
```

Listing 2.6: Cluster JSON response

Elasticsearch provides query to execute on specific index. The state can also be determined on several levels such as shards, index and cluster. Elasticsearch also provides very complex monitoring API for example

`curl localhost:9200/_cluster/health?level=indices`

`curl localhost:9200/_cluster/health?level=shards`

2.3 SECURE SHELL (SSH)

SSH is a cryptographic protocol for network which employs public and private keys for secure data communication between different user accounts or between two network computer that connect to each other using secure channel via running client and server. The protocol has two versions SSH-1 and SSH-2. In SSH client means the workstation or PC we are logged in and server means the workstation or PC where we visit to login and do some task [2].

Installing open SSH server and Client, type the following command in the terminal `sudo apt-get install openssh-server openssh-client`

Once the SSH server and client are installed, to test it simply run following command `ssh localhost`

To connect to the machine using SSH, we need server user name and IP and password. `ssh server_username@server_ip_address`

For reading the file or documents from different Linux machine, we will use SFTP (SSH file transfer protocol) which basically provide secure data stream using SSH for file access and management. The SFTP work with SSH version 2. SFTP provide security benefit as entire data is encrypted over the network, so other VM (virtual machine) cant parse the data over the network.

2.4 Samba

Samba is suite or free software implementation for CIFS or SMB server and client on Unix like platform. SMB (server message block) is network protocol for file sharing as implemented in the Microsoft windows. A dialect of SMB means that set of message packets define the particular version of the protocol implemented. CIFS (common internet file system) protocol is SMB dialect which is available for the different version of Unix or other OS [5].The packets can be classified as:

- General status packets
- Session control packets
- File access packets

The running mode of SMB is request-response mode. In this mode, Client sends the SMB request packet to the server. Server process and analysed the SMB message packet and send the response back to the client. When we need to share resources form windows platform, we need to install Samba software and its packages. After the packages are installed we need to configure the Smb.conf file. All SMB (citation 3 SMB) clients operate more like as FTP, for transferring file between UNIX and windows machine. We can mount the SMB share on local file system using mount˙smbfs.

Installing Samba in Linux machines run the following command in the terminal `sudo apt-get install samba smbfs`

Once the samba gets installed, we need to configure it, to make it accessible. We need to edit following file.
`sudo gedit /etc/samba/smb.conf`

2.5 Microsoft SharePoint

SharePoint is a Microsoft platform to create websites. It generally brings together all the technology required by business to interact, share and collaborate with content and data using internet or intranet. SharePoint provide enterprise wide information portal that provide centralized space for document sharing. It streamlines the organization data and access management. SharePoint provides the complete stack for application based API and web technology. Cloud-search will only focus on the document part of the SharePoint from where user save and share the document. Cloud-search uses SharePoint Rest based API for the content search and document parsing. Every user in SharePoint has its own document library to store its content [11].

Collection of files that users can share on web pages based on Microsoft SharePoint services is known as document library. Document library is controlled by users and files sharing can be private or public. Document can edit or checked or uploaded in the library by other user with permission. Document libraries can be extended to provide combination of features such as version history, metadata, custom columns, keywords etc. The standard document library is known as Shared document which provided by default. Document library can also create the folder simple as well as nested. SharePoint has another important feature known as Lists which provide collection of information that can be shared. Example of sample list are contact list, calendar list or list meeting or announcements. SharePoint administrator controls the access to the sites, list content to shared, user access rights etc. Cloudsearch application will only

parse the document library of user with shared access right only.

2.6 River in Elasticsearch

River is a service that be plugged in and out of Elasticsearch cluster for pushing or pulling the data that can searched as well as indexed into single or different cluster. Every river is has unique name and type. The name identifies the river in the Elasticsearch cluster and type defines the type of river, which can be anything, that can push meaning full data to the cluster.

Every river is defined by ˙river index. Each river index accepts a new document called as ˙meta and the name of the river. Within its curly braces it will have the river definition and data to be index from a location or service. Example of dummy river syntax:

```
curl -XPUT http://localhost:9200/_river/my_sshriver/_meta -d '{
    "type" : "ssh"
     "ssh":{
        "url": "/temp/sample",
        "username": "demouser",
        "password": "password",
        "includes": [".docx" , ".pdf"]
     }
}'
```

Listing 2.7: JSON for ssh index

To delete the river from the server, use following command
`curl -XDELETE 'localhost:9200/_river/my_sshriver/'`

To get the status of the river and its working, use following command
`curl -XGET 'localhost:9200/_river/my_sshriver/status'`

A diagram overview of river in Elasticsearch

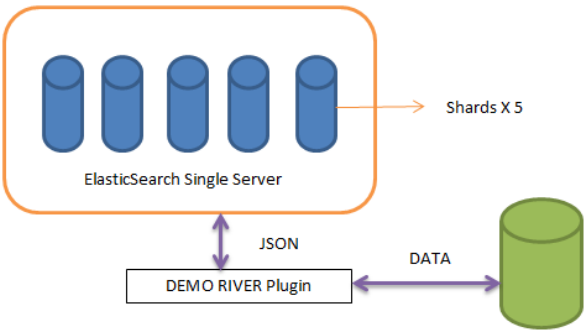

Figure 2.1: River plugin in ES

2.7 MySQL

MySQL is relational database management system and it is open source. It supports SQL as query language for DDL and DML operation on the database. It is built to handle large data volume with very reliable security and provide fast access to data based on relational model. MySQL is open source as any user can modify the source code as per their requirement. MySQL support windows, Linux, Mac OS and etc. It based on client server model and its basic installation includes the MySQL client and server [3]. MySQL support all the functionality of traditional DBMS with ACID properties support. MySQL features are:

- Reliable and very fast access to database
- Light weight database model
- Support general SQL queries
- Indexing and binary objects are supported
- Table structure can be altered while server is running
- Compatible to all the operating system
- Built in replication and partitioning
- Session control packets
- Event scheduler
- Plugin based architecture
- MySQL Cluster

Installing MySQL: `sudo apt-get install mysql-server`

2.8 Maven

Maven is an Apache project for comprehension and project management tool for software projects. Maven concept is much like ANT, but it is quite different in many aspects. Maven concept is based around POM (Project object model) which is an xml file representing the entire configuration for the project. With central piece of information from the maven can manage the documentation, reporting and project build with different phase [12]. Maven requires Java installed on the system with both JDK and JRE. Main advantage of maven is:

- Easy project management which is usable and maintainable.
- Plugins and tools are available to integrate with declarative language.
- Automatic download transitive dependency.

POM is the fundamental unit of work in maven and contains all configuration details required to build the project. POM can contain following configuration:

- Project version
- Goals
- Dependency
- SCM
- Integration Server
- Project developer and mailing list etc

2.8.1 Sample POM configuration file

Maven follows three standard lifecycle phases clean, default and site. A Goal represents a task which helps in managing and building the project and it can have zero or several phases. The XML represent simple maven pom config xml file:

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org ↩
      /2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 ↩
      http://maven.apache.org/maven-v4_0_0.xsd">
2     <modelVersion>4.0.0</modelVersion>
3     <groupId>com.dell.app</groupId>
```

```
4       <artifactId>Cloudsearch</artifactId>
5       <packaging>war</packaging>
6       <version>0.0.1-SNAPSHOT</version>
7       <name>Cloudsearch Maven Webapp</name>
8       <url>http://maven.apache.org</url>
9   </project>
```

Listing 2.8: pom.xml file

Maven uses archetype to create the project, which a plugin to create the project structure as per its described template. For quickstart archetypes, create a sample application in java using following command in terminal
`mvn archetype:generate -DgroupId=com.dell.search.cloud -DartifactId=Cloudsearch -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`

2.9 GIT

GIT is a distributed VCS (version control system). Git is like a mini file system, which has all the file of different and current version. Git works mostly on local file system and dont require network for every operation [7]. Everything in GIT is checked sum; it uses check summing called SHA-1 hash. GIT works on three main stage working directory, staging area and git depository (repository). Basic GIT workflow:

- Create or modify the files in the working directory.
- Stage the changes in the file with snapshots in staging area.
- In commit stage, we take all the files from staging area and store those snapshots in git repository which can be local or remote.

Installing the GIT
`sudo apt-get install git  works for Debian distribution and Ubuntu systems.`

2.10 Jenkins

Jenkins is java based continuous build system which is branched from Hudson. It runs in servlets containers such as Tomcat and glassfish. Jenkins is very highly configuration system which provides support for many plugins required by developer community [10].

It combines very easily with Maven, Ant, Gradle and other build as well as SCM tools. Jenkins features and advantage are:

- It generates the test reports and metrics required by QA and developer teams.
- Very large community support and good documentation.
- Plugins can be created very easily and most of them are available for download.
- Once the project is build, rest all future test and build are done automatically.
- Jenkins master-slave concept, slave nodes can be added to Jenkins.
- Provide continuous deployment, monitor problem and can be roll-back to previous state.
- Most important Jenkins support agile development for robust continuous integration system.

Installing Jenkins: `sudo apt-get install jenkins`

2.11 Jetty

Jetty is free open source project from the Eclipse foundation which is pure java servlets container and simple java based http engine. Jetty also supports SPDY, WebSockets, LMS, JNDI, JAAS and other integration framework. Jetty can easily embedded with in applications, clusters, devices, tools as well as framework. Advantages of Jetty are:

- It licensed under both Eclipse and Apache foundation.
- High level of scalability.
- Can be embedded with application and standalone deployment.
- Very small footprint.
- Easily integrate with Maven and development platform.
- Provides advance features as asynchronous access.

Jetty can be downloaded from `http://jetty.codehaus.org/jetty/`. Current stable version is 9. Extract the content of the Jetty.tar file. Once the content is extracted to the directory, open the terminal and enter the following command. `Java jar start.jar`. Open the browser and go to localhost to check if Jetty is running properly.

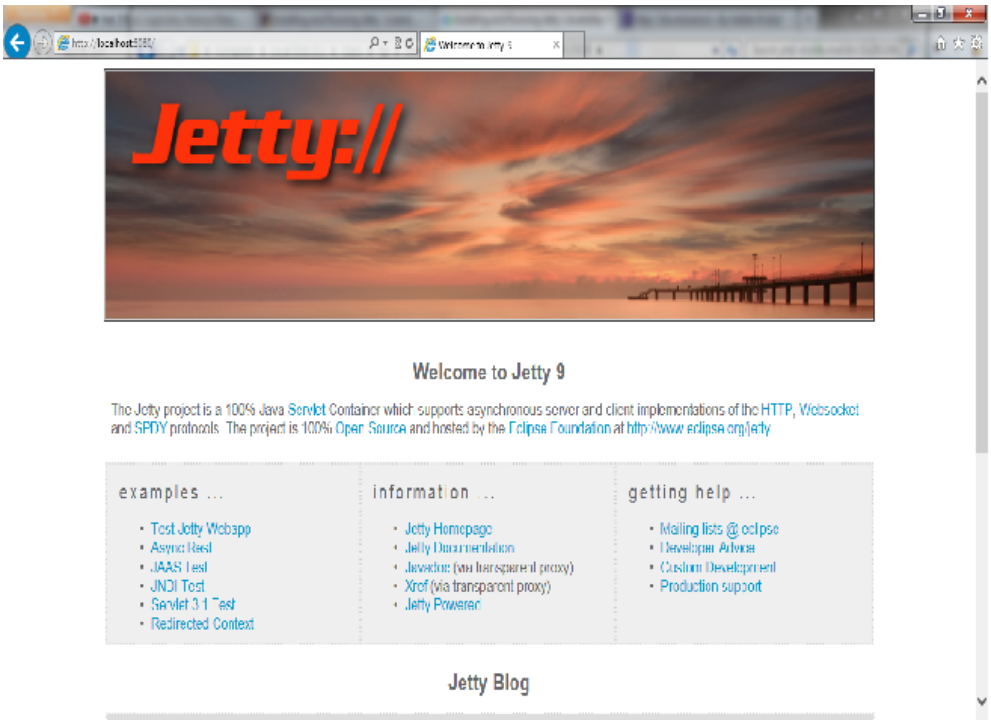

Figure 2.2: Jetty frontend

Chapter 3

Design

Cloudsearch POC is created to provide central search management across all virtual machine and set of services defined by the user to search the data related information for all Dell cloud services. It will provide a simple user interface to create/delete/update the search index for different services and platforms.

3.1 Basic overview

Dell cloud users can install and use Cloudsearch template by first login into Dell cloud services such as vCloud. User will first get authenticated from DCIAM (Dell cloud identity and access management interface) which holds the accounts, users and policies for all cloud system and components within Dell Cloud Services, such as vCloud, CAAS, STAAS and the Dell Cloud Console itself. Dell Cloud Services will interact with DCIAM to authorize and authenticate users and API calls, e.g. allows a Dell cloud user to login, get access and use Dell cloud services, such as vCloud.

The Cloud Console will use the DCIAM to authenticate and authorize users with regard to web activity. On login, user can sign up for Cloudsearch application which will install on the user cloud platform.

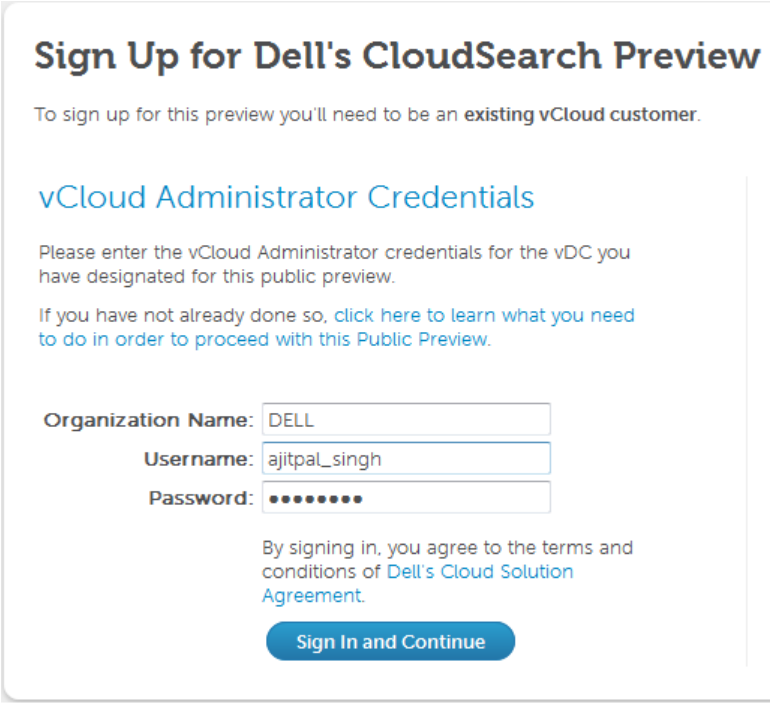

Figure 3.1: cloudsearch signup

Cloudsearch will create and install 2 virtual machines on user cloud platform. First virtual machine will have Elasticsearch package installed on it and running as service. VM will have 5 shards with replication factor of 1. User can increase the Elasticsearch node as required since its horizontally scalable. Second VM install the web frontend package which integrates with user interface at Dell web console. Both VMs are installed as required for Cloudsearch template. Configuration and VM settings are manageable from web interface. On successful installation of template, user will receive notification on the email.

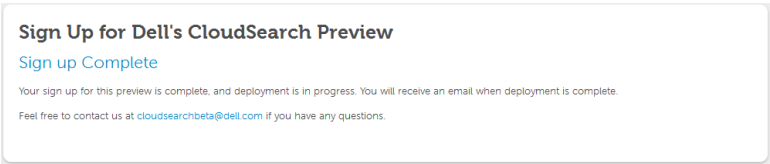

Figure 3.2: cloudsearch notification

User can manage different instances depending upon the cloud setup. In Web console, user can add Cloudsearch template or can edit exiting one. User can add only 4 Elasticsearch node for testing purpose.

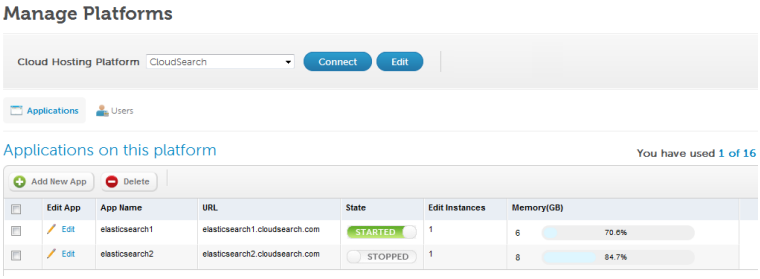

Figure 3.3: manage services and nodes

Elasticsearch dont provide any authentication or authorization method to its REST based interface API. To limit the access to the search interface, user creation/deletion and password changes is implemented using Dell console which is reflected on Cloudsearch template. Only authorized users are able to view the search template and manage the index.

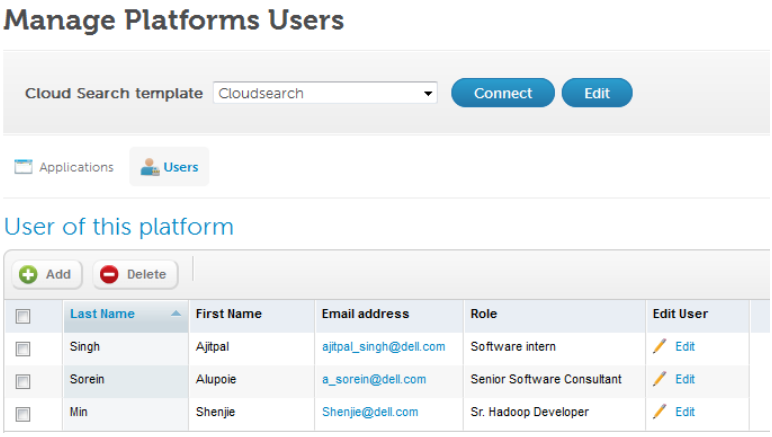

Figure 3.4: user management

3.2 High Level Design

This design represents the overview of installed Cloudsearch application on Dell vCloud as template. The Elasticsearch node is responsible for synchronizing data between the entire nodes using river plugins and sending JSON data back to front end on request. User can manage index of required virtual machine or platforms. Cloud administrators can login from vCloud director to manage the cloud environment and check the metrics of data usage for billing and analytics purpose. Administrator can only collect the data logs for templates installed, audit messages, traces of the cloud health. Administrators have no access to the user cloud environment and its VM

data. Once the free quote for searched has finished, user will charged on the base of search transactions requests. All the data for administrator are stored in vCloud DB.

There is no SLA on this, but in case of failure new Elasticsearch node will be started again. Dell cloud service will handle this where feasible. Elasticsearch node can connect to different VM using SSH, SMB protocols. Each node can have 10 indexes only and each index get a new connection from Elasticsearch node. Index limitation is simply kept for POC purpose only. Each Linux VM must install SSH and SMB utilites to access data.

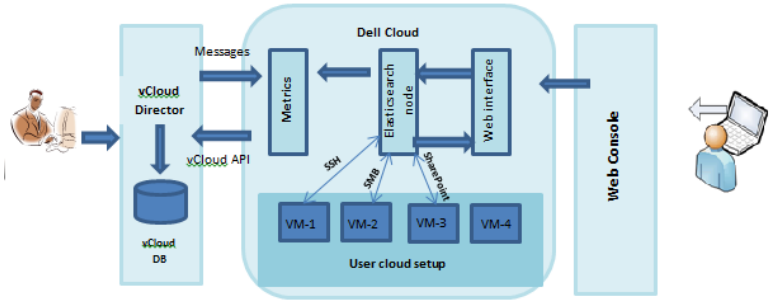

Figure 3.5: Cloudsearch overview

The diagram 3.5 shows the users handling indexes and performing the following

- User login to Dell web console and send authorization token to application for specific user request.
- User token is checked with authorized user and after verification get access to the search web console.
- Adding the index on the virtual machine and template are logged at regular interval to cloud logs.
- On Index start, Elasticsearch node will parse the data on given VM and store its JSON.
- Elasticsearch node will send the request matrix to vCloud and its health is monitored by internal services.
- During issue escalation, vCloud API gather the error message, warning, information and send to vCloud director for verification and DB sync purpose.

The diagram 3.5 does not show an additional interface to the Console, where the Server

forwards audit messages for inspection to the Console and Dell cloud internal running services to manage the cloud environment.

3.3 Elasticsearch Plugins

Elasticsearch provides a way to enhance its core capabilities by adding custom function in form of plugins. The plugins can vary from river to analyser, mapping types, native scripts etc. Plugins can be installed both manually and automatically under Plugins directory. In manual mode plugins must be copied to `/Elasticsearch/plugins/plugin name/plugins` folder. In automatic mode, user can install the plugins using `plugin --install <org>/<username/componentname>/<version>`.

FS-river plugin helps users to send the file system docs and logs in JSON format to Elasticsearch cluster. It indexes the local file system, Linux file system by SSH and Windows file system using SMB protocol libraries. It read your given files at specified directory at particular time intervals, means user can schedule the indexes to read data in real time at specified time intervals. Simple workflow to start the file system river:

- Create the plugin and copy the dependencies to plugin folder in Elasticsearch.
- Start the Elasticsearch cluster again.
- Create your index and mapping file if required.
- Send the mapping to the Elasticsearch cluster.
- Machines should be accessible using to each other via network using SSH or SMB.

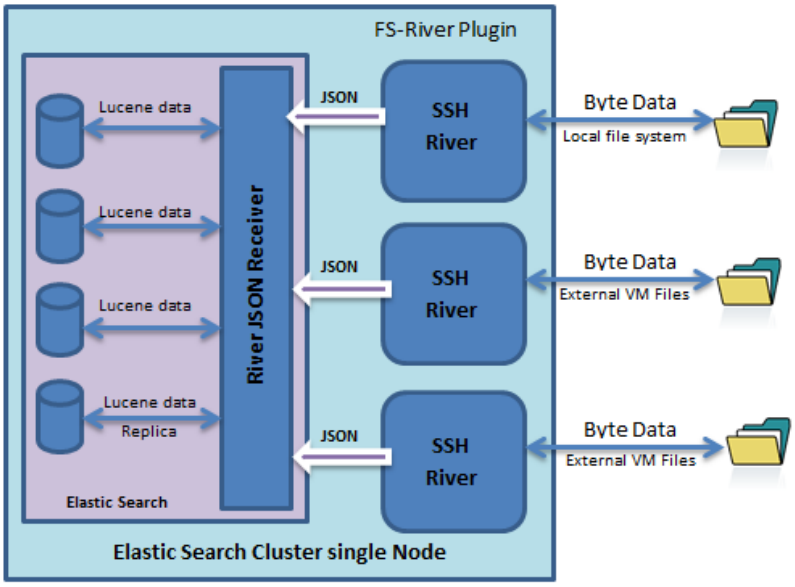

Figure 3.6: River plugin internal components

The diagram 3.6 represents the file system plugin installed in a single Elasticsearch node. Once the index is started, it will check which protocol to use for reading file SSH or SMB. Each river plugin establish one to one relationship with file directory. River read the data in byte stream and then byte stream is passed to Elasticsearch node in JSON format. Every index has its own Lucene data store also knows as shards or replica in Elasticsearch terms. It allows multiple documents in single index and different type of documents can query and indexed. JSON from river plugins are index inside Lucene or shards. Shards can contain all documents or can distribute data in-between to give better performance and routing capabilities. Index is built on Lucene index which can be divided into variable segment at any given point. Each segment can be separate index or index in itself. So each shard can contain single document or can share multiple document to index. Multiple shards provide very high fault tolerance and distributed search capabilities.

3.3.1 Sample index

we define a SSH index for Elasticsearch node required to parse Linux based machine directory file system.

- Create sample index with name sshdemo. Use curl command line tool to pass JSON data to Elasticsearch node.

```
curl -XPUT localhost:9200/sshdemo/
```

- Add properties required for index. Index properties contain:
  - **url:** path of the directory to parse
  - **server:** Server name or IP address of the machine
  - **username:** username for the system.
  - **password:** machine password.
  - **update-rate:** time interval in milliseconds.
  - **includes:** specify the type of document to read eg. Txt, pdf, doc, xml etc.
  - **Domainname:** domain name required for the windows only.
  - **protocol:** it defines which protocol to use
    - **SSH:** use secure shell for Unix/Linux like platform.
    - **SMB:** Server message block for Windows file system.
    - **SP:** SharePoint directory for the user.

3.4 Workflow for application development

Workflow of application is based on continues integration with Jenkins which provides the testing, deploying and integration with other software which automatically build and test the application on regular intervals. It notifies developers, testing team, quality assurance team with report and result of different integration testing. The application deployment is done in testing environment which includes different integration cycles.

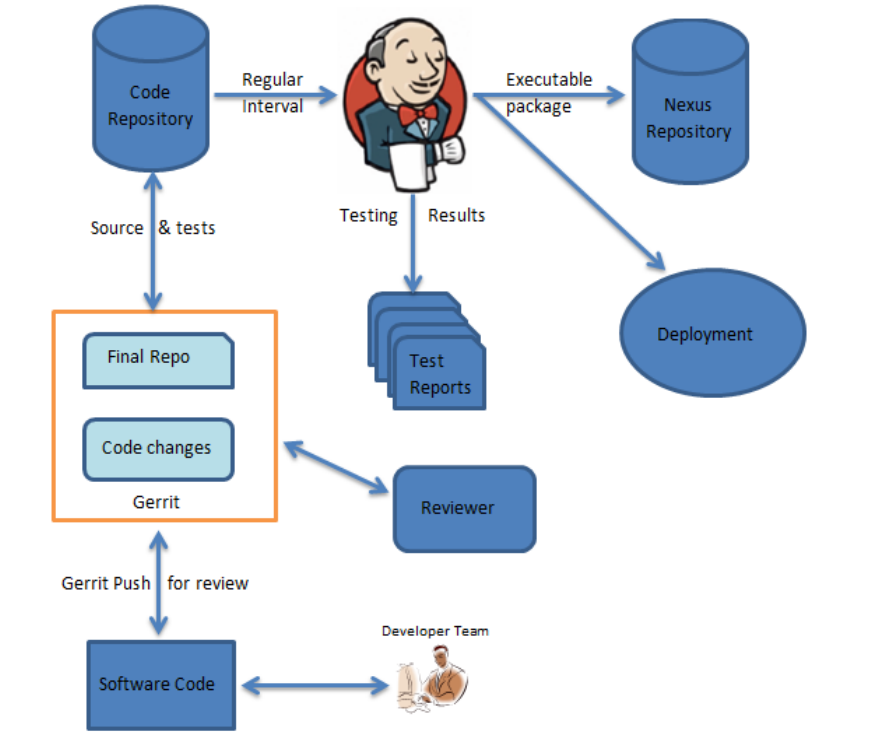

Figure 3.7: Development workflow

The complete workflow shown in figure 3.7 follows as:

- Development team write the software code, testing and software configuration. Developers manage the code using Git and maven workflow among different teams. Once the code lifecycle is completed, updated code is pushed at the Gerrit code repository.
- Team leader review the code using Gerrit, which built on the top of version control system and provide web based code review tool. Any changes required are added to pending list and developer team updates the code.
- After the code is reviewed and pending changes are completed, the project is checked with code repository and Jenkins fetch the project code from code repository. Jenkins automatically starts building and executing the code and test lifecycle process.
- Jenkins download all the required dependency and transitive dependency of code from central repository i.e. nexus. Once the all the dependences are downloaded and updated with repository, Jenkins start the code testing and its integration process.

- Jenkins sends the test result daily about the code and bug to developer mailing team. Jenkins also distributes the code and build test load to multiple computer.
- Deployment is done every time, when new changes are done in code.

Chapter 4

Implementation

In this section, we discuss the implementation of plugin and web based frontend on the Dell cloud services. Cloudsearch extends the Elasticsearch river plugins capabilities to read and parse data over the network using SSH and SMB protocol libraries. Cloudsearch application contain simple web based front to create and manage the search indexes with search page to search the indexed data from Elasticsearch node. Cloudsearch application is divided in two parts:

- FS-River plugin for Elasticsearch node. (backend interface)
- Cloudsearch website (frontend interface)

4.1 FS-River plugin Development

FS-River plugin reads the data over the network using SSH library for Linux machine, SMB library for windows machine and httpasynclient for SharePoint REST interface parsing. The plugin integration with Elasticsearch presents a number of challenges, as Elasticsearch does not support automatic data parsing over the network. We defined the plugin to read the files over the network then convert them to JSON format and send the data to Elasticsearch node using its API. The Elasticsearch node will automatically index the data. We defined a fixed mapping style in which data is formatted and send to Elasticsearch node.

FS-river plugin implements following dependencies in its project

1. Elasticsearch.jar (0.90.3)

2. JSch.jar (0.1.50) support SSH protocol.

3. jCIFS.jar (1.3.17) support SMB protocol.

4. httpasynclient (4.0-beta3) support REST/ HTTP protocol.

Plugin provides a schema mapping which define our index structure to Elasticsearch node. Since each index can have multiple type, but we create single type for each document we parse. File mapping for plugin defined as following:

```
{
  "mappings": {
    "doc": {
      "properties": {
        "file": {"type":"attachment", "path":"full" },
        "name": {"type":"string", "analyzer":"keyword"},
        "pathEncoded": {"type":" string ", "analyzer":"keyword"},
        "postDate": {"type":"date", "format":"dateOptionalTime"},
        "rootPath": {"type":"string", "analyzer":"keyword"},
        "virtualPath": {"type":"string", "analyzer":"keyword"},
        "fileSize": {"type":"long"}
      }
    }
  }
}
```

Listing 4.1: Mapping for river

Every document parsed using Elasticsearch will have following (Listing 4.1) JSON format returned during the search result. Every index created for different files will follows the same mapping.

4.1.1 FS-River Plugin design

Plugin describes four packages in java project which includes **plugin** packages which define main class that interface with Elasticsearch and register the plugin with server, **rest** package provides the mechanism to control the start and stop plugin by providing REST API interface, **river** package introduce main class required to parse the data over the network using multi-threading provided by Elasticsearch library and define other class required to implement those function, **util** package consists primarily of classes that provide static utilities function across all packages as shown in figure 4.1.

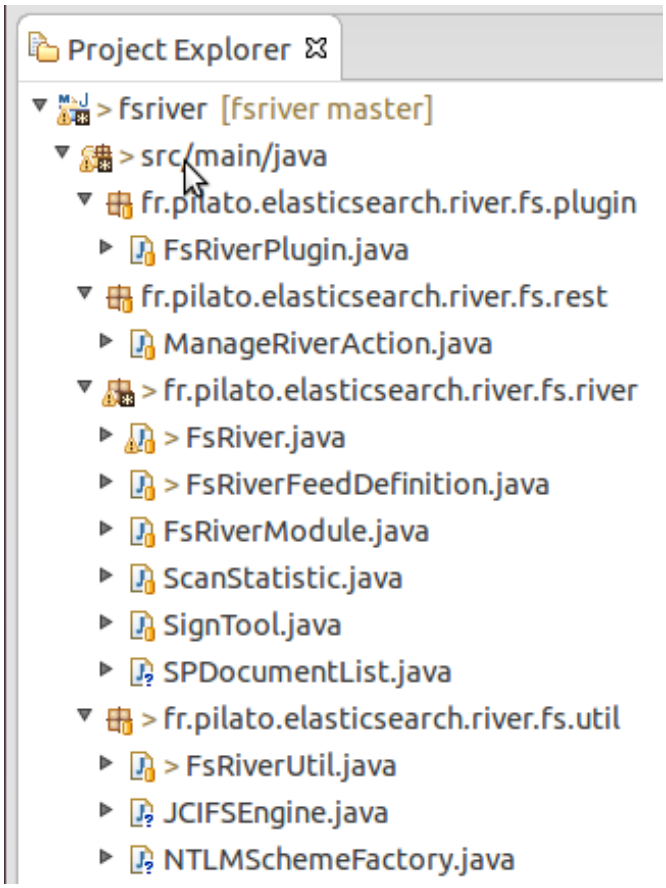

Figure 4.1: Fsriver plugin project

4.1.2 SSH plugin implementation

Plugin implements Jsch library to provide secure remote file reading capabilities, it automatically encrypt and authenticate data over the network. User will provide the details of machine to be parsed over SSH. Plugin will authenticate the server by using credentials and go the directory to parse the data. Plugin will recursively read all the data from the folders and files present in given directory and sends it data in JSON format to Elasticsearch node. User can stop the plugin if required. Sample JSON script to send the SSH request to the plugin:

```
curl -XPUT 'localhost:9200/_river/sshlogs/_meta' -d '{
        "type": "fs",
         "fs": {
           "url": "/usr/data/logs/",
           "server": "193.125.125.125",
           "username": "node001",
           "password": "Pa$$word",
           "protocol": "ssh",
```

```
9            "update_rate": 3600000,
10           "includes": [ "*.doc" , "*.txt", "*.pdf" ]
11          }
12      }'
```

Listing 4.2: SSH mapping

FS-River plugin identify the protocol from JSON request **protocol** field. Each SSH request must provide system credentials with it IP address as well as complete directory path to parse. Every request to plugin will include the **update-rate** which define the time in milliseconds to parse the data again and different type of files to be parsed is defined in **includes** field for example docx, pdf, txt etc.

Code overview for SSH implementation

When plugin receive the request for SSH protocol, it call function **IndexSSHFileSystem** (Listing 4.3). It first creates the connection with machine using the credentials and given details. Once the connection is established, its open a secure FTP channel to the folder. Plugin will recursively parse entire folder for given file types and close the connection once completed. Plugin will again follow the same process when the updated rate event is fired. User can start/stop the plugin for web interface. Method **addSSHFilesRecursively** (Listing 4.4) parse the data inside the folder in recursive manner.

```
1  private void IndexSSHFileSystem(){
2    try{
3     //Create a new jsch session
4      JSch jsch = new JSch();
5     //Set the credentials
6      Session session = jsch.getSession(fsdef.getUsername(),fsdef.getServer());
7      // connect to ssh session
8      session.connect();
9     // open a sftp channel to read the file.
10     Channel channel = session.openChannel("sftp");
11     channel.connect(); //connect the channel for sftp.
12
13    // if channel is connected.
14     if(channel.isConnected()){
15        // goto the desired folder.
16        sftpChannel.cd(fsdef.getUrl());
17        //read all the files recursively and push to server.
18        addSSHFilesRecursively(fsdef.getUrl() , sftpChannel, scanDate);
19     }
20     //disconnect the channel.
```

```
21      //close the session.
22      //Handle the exceptions.
23  }
```

Listing 4.3: IndexSSHFileSystem function

```
1  private void addSSHFilesRecursively(String Filepath , ChannelSftp channelSFtp , Date ↩
       lastScanDate) {
2         //Change the directory path.
3    channelSFtp.cd(Filepath);
4    //check for files and directory.
5    if(!lEntry.getAttrs().isDir()){ // only file to be parsed.
6       indexSSHFile(stats, Filename, Filepath , channelSFtp,lEntry.getAttrs().getMTime ↩
            ());
7    }else{
8           //Directory again parse it.
9       addSSHFilesRecursively(Filepath + Filename + "/" , channelSFtp, lastScanDate);
10   }
11     //reset the path of sftp.
12     channelSFtp.cd("..");
13     //Handle the exceptions.
14 }
```

Listing 4.4: addSSHFilesRecursively function

4.1.3 SMB plugin implementation

Plugin can browse or read data from SMB shares; this can be done whether machine is a samba or Windows server. SMB client installed on Linux machine will provide ftp like interface to Windows machine to parse its data. User will provide detail of window machine credentials and directory to parse. Plugin will recursively read all the data from the folders and files present in given directory and sends it data in JSON format to Elasticsearch node. User can start/stop the plugin if required. Sample JSON script to send the SMB request to the plugin:

```
1  curl -XPUT 'localhost:9200/_river/winlogs/_meta' -d '{
2          "type": "fs",
3           "fs": {
4             "url": "smb://192.168.1.100/demosmb/",
5             "username": "node001",
6             "password": "Pa$$word",
7        "domain": "sample.domain.name",
8             "protocol": "smb",
9             "update_rate": 3600000,
10            "includes": [ "*.doc" , "*.txt", "*.pdf" ]
```

```
11          }
12      }'
```

Listing 4.5: SMB Index

FS-River plugin can identify the protocol implementation using protocol field. Request for windows share must provide SMB URL, windows machine credentials and its domain name. Plugin read data as SmbFile which can be directory or file, SmbFile provides same behaviour as File class in java. SMB URL scheme must be correct and should specify the target directory or file to be parsed.

Code overview for SMB implementation

On SMB protocol usage the plugin calls **IndexSMBFileSystem**(Listing 4.6) method authenticates the Windows using NTLM challenge-response authentication. Once the process of authentication establish between two parties, it creates a SmbFile which represent directory path given in JSON. It internally calls **addSMBFilesRecursively**(Listing 4.7) method, which read all the SmbFile recursively and read the given file extension. Data is converted to JSON and pushed to Elasticsearch search server. REST API can start or stop the recursive data parsing.

```
1  private void IndexSMBFileSystem () throws MalformedURLException{
2     try{
3         //# Login to SMB file system
4    NtlmPasswordAuthentication auth = new NtlmPasswordAuthentication(
5      fsdef.getDomainName(), fsdef.getUsername(), fsdef.getPassword());
6    SmbFile smbDirectory = new SmbFile(fsdef.getUrl() , auth);
7    // index the file and directory recursively
8    addSMBFilesRecursively(smbDirectory, scanDate);
9         }
10 }
```

Listing 4.6: IndexSMBFileSystem function for Windows

```
1  private void addSMBFilesRecursively(SmbFile smbDirectory, Date lastScanDate) throws ↩
       Exception {
2      //List all the files and directory.
3      final SmbFile[] children = smbDirectory.listFiles();
4      if (children != null) {
5    for (SmbFile child : children) {
6                 if (child.isFile()) { //check if it file
7        //index the file and send to es node.
8        indexFile(stats, child);
9             }else{
```

```
10        //if directory, call method again.
11        indexDirectory(stats, child);
12        addSMBFilesRecursively(child, lastScanDate);
13           }
14         }
15      }
16  }
```

Listing 4.7: addSMBFilesRecursively function for Windows

Code doesnt represent the complete functionality, but only the important methods called.

4.2 JDBC plugin for MySQL

Java database connection allows fetching data from JDBC resources and we fetch the data from MySQL server and indexing data into the Elasticsearch node. The relational data will be internally converted to JSON and pushed to Elasticsearch node. We need to implement following steps to parse the MySQL database tables.

1. Install the JDBC river plugin for Elasticsearch using following command line `/bin/plugin -url http://bit.ly/Yp2Drj -install river-jdbc`
2. Download latest version if MySQL JDBC driver.
3. Copy the MySQL JDBC driver jar file to JDBC river plugin installation directory. `cp mysql-connector-java-5.1.26-bin.jar /usr/Elasticsearch/plugins/river-jdbc/`
4. Start the Elasticsearch node with logging enabled in terminal window.
5. Login display installed plugin [fs-river, jdbc-river] in terminal.

4.2.1 JDBC Implementation

To start indexing SQL table data into Elasticsearch node, we define JSON script (Listing 4.8) and send this data to search node.

```
1  curl -XPUT 'localhost:9200/_river/jdbctest/_meta' -d '{
2      "type" : "jdbc",
3      "jdbc" : {
4          "driver" : "com.mysql.jdbc.Driver",
5          "url" : "jdbc:mysql://192.168.145.12:3306/HRM",
6          "user" : "localuser",
```

```
7         "password" : "Pa$$w0rd",
8         "sql" : "select * from user_info"
9         "poll" : "1h",
10        "max_retries" : "10"
11    },
12    "index" : {
13        "index" : "hrmdb",
14        "type" : "hrmdb"
15    }
16 }'
```

Listing 4.8: JDBC index

JSON defines river as type JDBC and specify the JDBC properties for fetching the data from table. Index defines the name of index as hrmdb and its type defined as hrmdb. The table describes the properties of the JDBC river.

Table 4.1: JDBC JSON properties

JSON Properties	Description
type	Define type of river jdbc, fs etc
driver	Jdbc driver class
url	url for jdbc driver
username	Database username
password	Database password
sql	Sql statements or .sql files for batch processing
poll	Time interval in hour h (1h) and seconds s (30s)
max˙retries	Total retries to connect with DB

To query Elasticsearch node for data indexed using JDBC river with following command.

```
curl -XGET 'localhost:9200/hrmdb/hrmdb/_search?pretty&q=*'
```

We have covered the backend implementation of river plugins and their workflow implementation with Elasticsearch node. To interact with Elasticsearch, we created web based front end using servlets and integrated the web pages with Dell cloud web console. User can perform the CURD (create / update / delete / edit) operation on the indexes. Web front end also provides search tab to query the Elasticsearch node in real time to get results of indexing done in backend.

4.3 Cloudsearch Frontend development

Cloudsearch interface is implemented using java servlets which act as an intermediate layer between an http client and request coming from browser. Servlets increase the performance significantly and provide platform independent deployment as well as integration. In general, servlets are objects which follow request and response lifecycle. Every request from user is processes using servlets and in response servlets generate the html pages with user response. For testing purpose, the web application runs with jetty inside the application. During deployment the application will create .war file and store in Jetty web server folder in Dell cloud console.

To interact with REST based interface of Elasticsearch, application integrate the jersey client to send and receive the request from server. Jersey helps in development of the Restful web services using Java API. It provides its own API that extends the JAX-RS toolkit.

4.3.1 Workflow of Cloudsearch

- Dell cloud user login to web console using its credentials and in backend DCIAM (Dell cloud identity and authorization management) manages the entire process.
- After user login to web console, user switches to installed cloud search templates.
- In web console, it provides two tab views containing Search and indexing tab.
- In indexing tab, user can create and manage the indexes for SSH, SMB, local file system and MySQL database.
- User indexing request for different machine are saved in XML files, which are later saved in database.
- Every index can be edited or deleted at any time. Deleting index will remove all the data saved in Elasticsearch node and cant be recovered back.
- To provide better performance over the network, user can stop the index after it runs successfully on entire directory. Once the index is stopped, any changes to data are not updated in Elasticsearch cluster until it is started again.
- User can search the data in real time from search tab.

Frontend implementation

The frontend uses following technology and framework for application to work properly

- Framework
  - Jersey Restful API.
  - Jetty Http web server.
  - Eclipse web development interface.
  - Java servlets.
- Dependency libraries
  - Jetty-server (9.0.0) for embedding http web server
  - javax-servlets API (3.1) for servlets implementation
  - slf4j-api (1.7.2) for logging purpose
  - gson (2.2.2) Google client library for JSON parsing
- Supporting language and library
  - Java
  - Ajax
  - JavaScript
  - Html & CSS

4.3.2 Code Elements

This section is for describing code elements and class implemented n Cloudsearch and their functionality overview. The diagram 4.2 represents the Cloudsearch project packages and their classes. The controller package defines the main servlets classes that handle users request for indexing, searching and controlling the index. The model package implements the simple POJO class (plain old java object) to directly map them with JSON. The utils package provides the utilities class required in different packages. ElastisSearchParser class implements jersey-client which interacts with Elasticsearch node and send results back to FileConsoleServlets. FlexiGridXML manage the index xml generated by user.

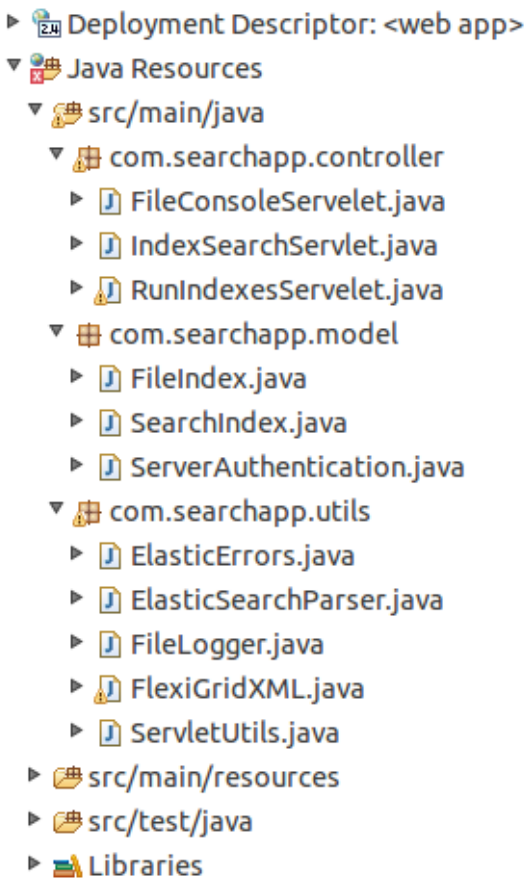

Figure 4.2: cloudsearch website project

4.4 Overview of Cloudsearch website

This section provides step by step guide working of Cloudsearch in Dell web console. User will first login into Dell vCloud console and authenticate to use the Dell cloud services. If the Cloudsearch template is installed, then cloud console will provide pages to manage the index. The frontend show two tabs Search and Indexing.

- Select the indexing tab, to add a new index to Elasticsearch server. Click on Add New Index button.

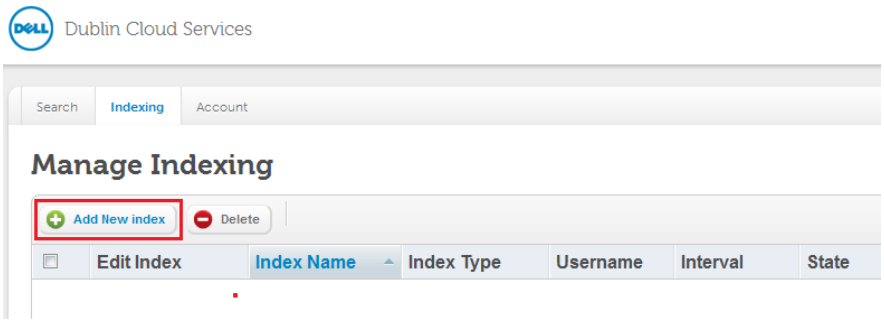

Figure 4.3: Manage index tab

- It opens a pop up window to add new index type. First we will create a SSH index. Select the SSH index from index type dropdown.

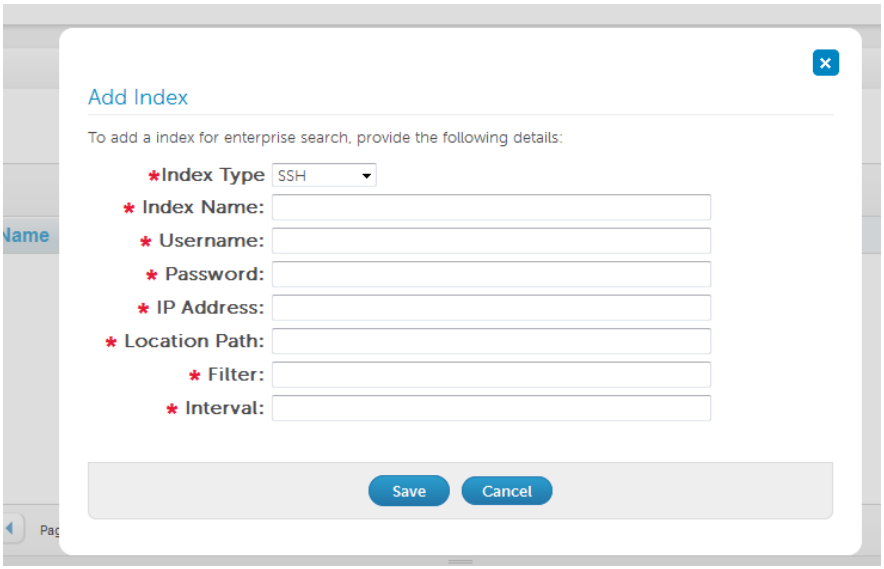

Figure 4.4: Index popup display

- Set index name as sshindex, enter the username and password for Linux machine. Enter the IP address of the machine and set the directory to parse in Location path. Add the filters for file type such as pdf, txt, and doc. Finally set the interval for scheduling.

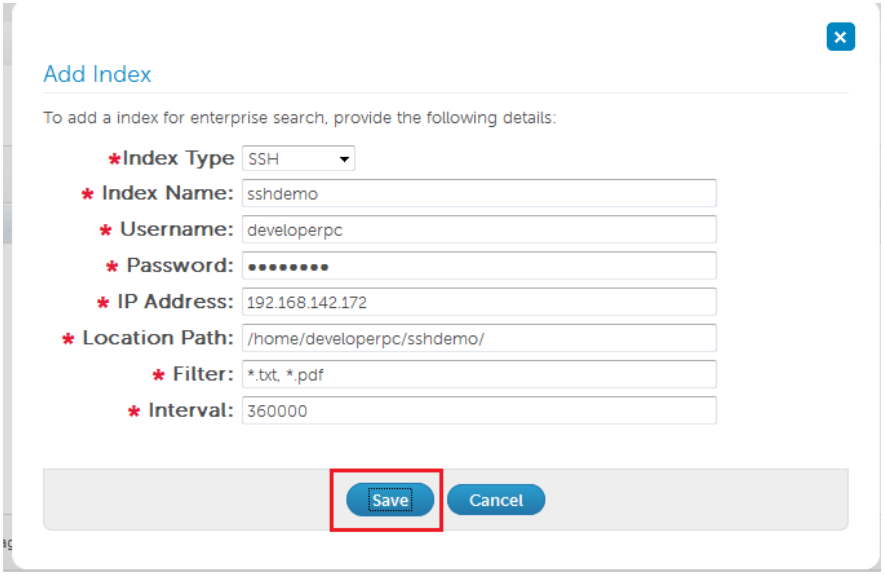

Figure 4.5: Save SSH index

- Click on save button to add the index to grid and Elasticsearch node. User request will be process by servlets, it first checks the authentication of the server and if credentials are correct the index request is sent to Elasticsearch server. Once index is verified, its entry will be shown in grid.

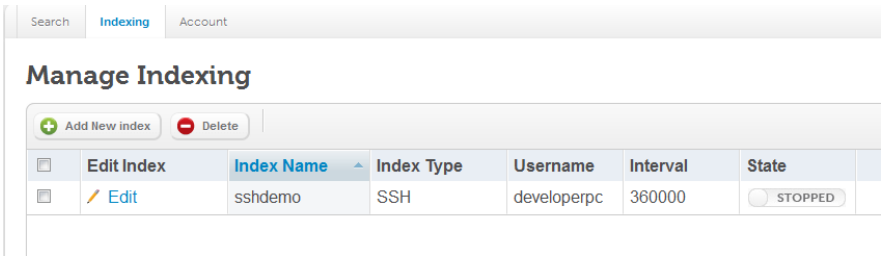

Figure 4.6: Index saved in Flexi-Grid

- To verify that index is created at Elasticsearch node, browse the URL `http://esnode:9200/_plugin/head/` it will show all the indexes in node and its respective shards. Indexes always stopped by default. The docs = 0 at specific index represent that indexing has not started but only defined.

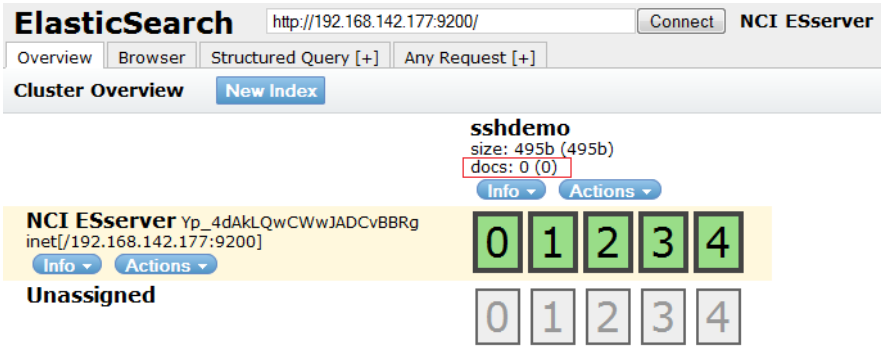

Figure 4.7: Elasticsearch head

- To start the indexing of sshdemo, select the index in grid and click on stopped push button in State column. It will start the index and get the response back from Elasticsearch node. It show index is running properly on the server.

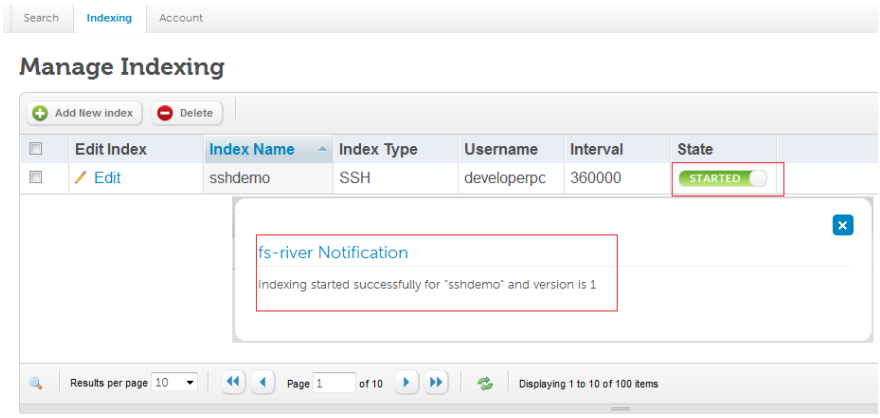

Figure 4.8: start indexing

- To verify the indexing on the Elasticsearch has started, browse the plugin head page. When index is getting started it will create the ˙river index by default. sshdemo index docs count have changed to 7 , it represent that plugin has indexed 7 document which includes files and folder.

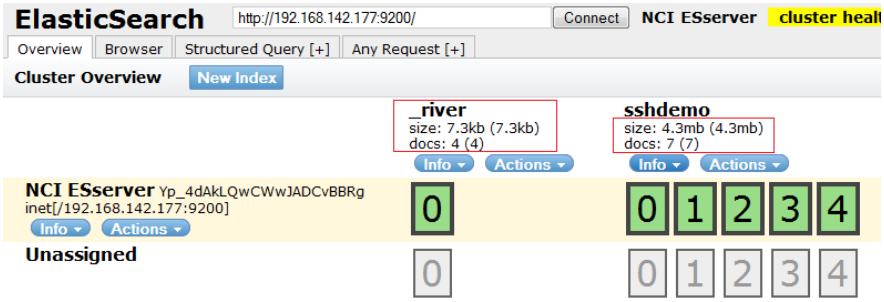

Figure 4.9: check indexing

- All other index for SMB and MySQL are created in same manner. User can delete the index by selecting single or multiple indexes in the grid. Once index is deleted, all the data from Elasticsearch node also get removed.

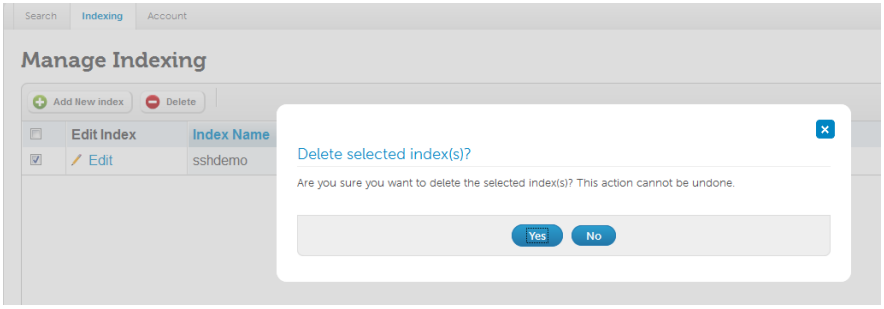

Figure 4.10: Delete index

- To search the data, select the Search tab and enter the text in search text box and click on Search button. We are searching NCI keyword in Elasticsearch node. Search page give details about total results found and time took by server to find the data. Every result show the relevant scoring based on Lucene search algorithm.

Figure 4.11: Search data from index

We can search the entire document indexed by Elasticsearch cluster which includes document from Linux machine, windows platform, SharePoint server and MySQL database. Cloudsearch POC application extends the plugins to index the data over the network system in reliable manner. To test the capabilities of the plugin, we did some evaluation test by reading very large text based file over the network and checking the system processing and network speed. In next section we are going to present the following results and its evaluations.

Chapter 5

Evaluation

In this paper, we evaluate our implementation of Cloudsearch application in Dell cloud services. All components of the Cloudsearch application i.e Elasticsearch server with fs-river plugin, Web based front-end integrated with Dell web console and 2 virtual machine one for Linux platform & another for Windows platform are deployed as well as configured on VMware workstation for testing. However, Cloudsearch front-end is installed on another for standalone machine for testing purpose. Furthermore, we tested the elasticsearch plugin with large log files (1 GB and 2 GB) respectively. We have presented our test result and evaluation of Elasticsearch server in coming section.

5.1 Program Description

We deployed all the component on VMware workstation 9. The virtual machine configuration is shown in Table 5.1.

Elasticsearch node have 2 processor and 6 GB Ram, rest all virtual machine have 1 processor and 2 GB Ram. All VM are connected to each other by VMware virtual network. Client machine with sharing capabilities have common folder name `demoshare` with files and directory listed in it.

Component	Virtual machine name
Elasticsearch version 0.90.3	esserver-01
Linux SSH Client	ubuntu-machine
Windows SMB Client	windows-machine
Webserver	Cloudsearch-webserver

Table 5.1: Test machines configuration

To create testing file of various size, we have written simple script that copies one file into another and rename the file. For demonstration, we have copied Linux machine log file of size 10 MB and created two files of 1 GB and 2GB each. Following script will create file size of 1.28 GB for testing and running script again with 1 loop only create another log file of 2.56 GB.

```
for i in 1..7; do cat testlog.txt testlog.txt > testlog2.txt && mv
testlog2.txt testlog.txt; done
```

For testing purpose, we will try to read following file using fs-river SSH protocol and check CPU and network usage over the time. Once the setup is completed , we created a testing ssh index for fs-river and send the request to Elasticsearch server. Parsing large file in server required to increase the default heap size for JVM to load data in memory. We started server with following configuration to proceed with our test.

```
elasticsearch -f -Xmx1g -Xms3g -Des.index.storage.type=memory
```

In next section, we present our findings and evaluations.

5.2 Findings

First, we evaluate ssh index creation by parsing 1GB file with fs-river. In this evaluation, fs-river parse the 1 GB text file in 61 seconds using ssh protocol. The Graph given below 5.1 shows the CPU usage over the time for 1 minute. As we can see from the graph, the average CPU utilization for reading 1 GB is approximately 58.04%. However, cpu utilization was keep on fluctuating with its mean value (58.04%) with the standard deviation of 19.94. In short we can say that cpu utilization varies between the interval [77.98 - 38.04].

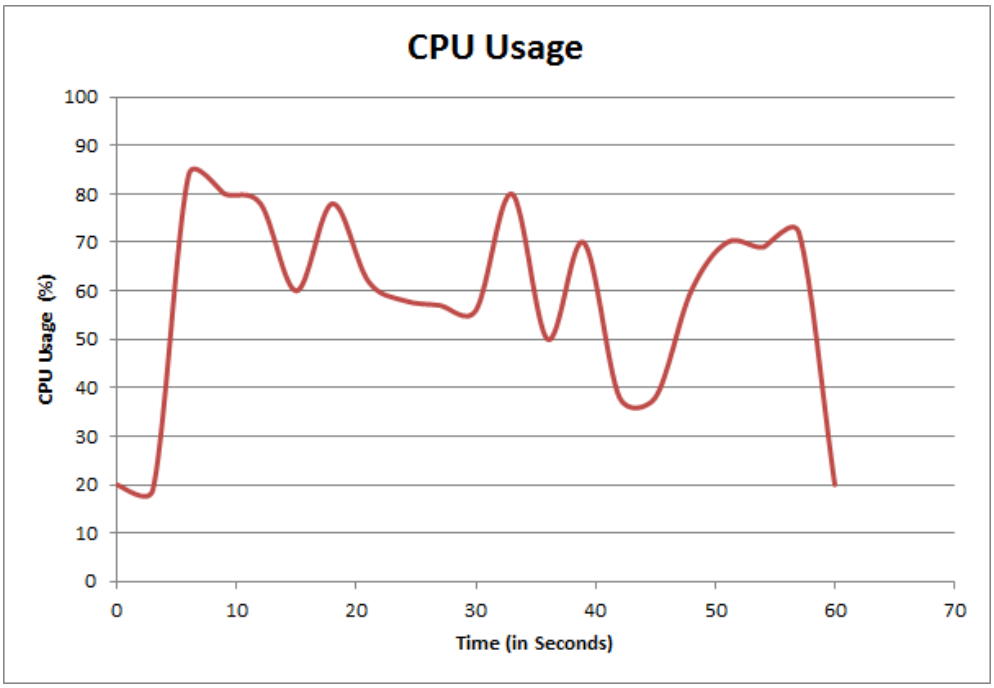

Figure 5.1: CPU usage for 1 GB file.

We also evaluated the network usage result for reading 1 GB file over the network. Figure 5.2 shows average network utilzation for reading 1 GB file is 13.81 MB/s. The network utilization varies between [8.95 - 19.05]MB/s with standard deviation of 5.22.

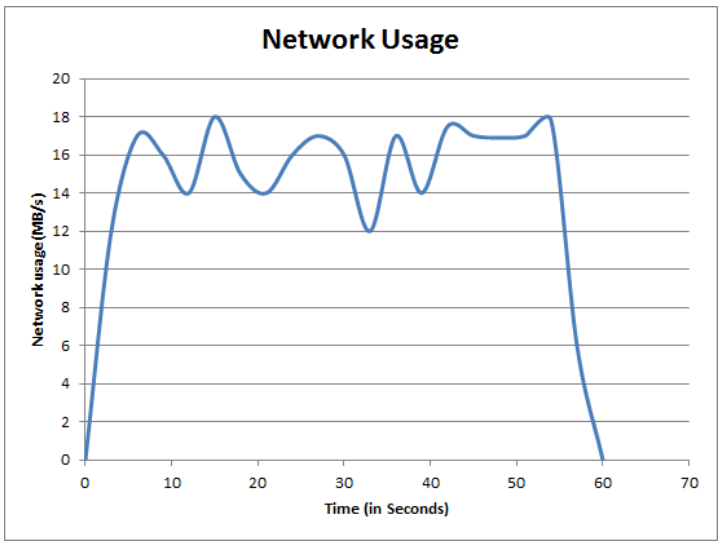

Figure 5.2: Network utilization for 1 GB file.

We also evaluated the file parsing of 2.57 GB file using SSH protocol, but fsriver throws error not enough heap memory, so we increased the memory filter of Elasticsearch

cluster as `-Xmx1g -Xms5g`, as result we were able to read 2.57 GB file with following results. Fs-River took 3.1 minute to parse the file. Figure 5.3 shows cpu utilization for 2.58 GB file. The cpu utilization was keep on fluctuating with its mean value (58%) with the standard deviation of 13.51. In short we can say that cpu utilization varies between the interval [71.52% - 44.5%]. As file size has grown the cpu utilization remain almost stable over the time.

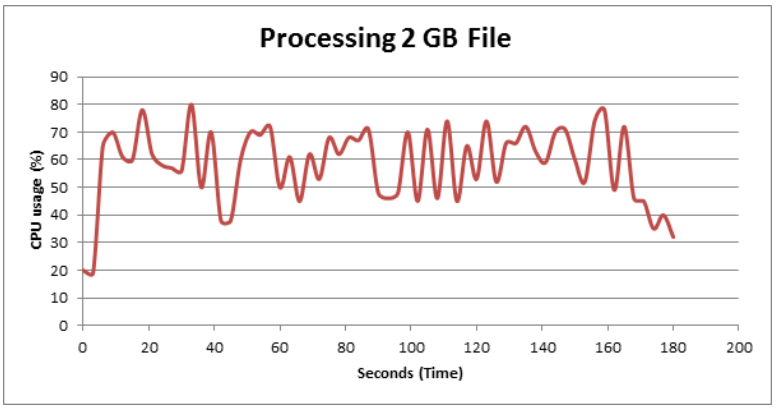

Figure 5.3: CPU utilization for 2.57 GB file.

Whereas the average network utilization as shown 5.3 in 15.54 MB/s with standard deviation of 3.63.

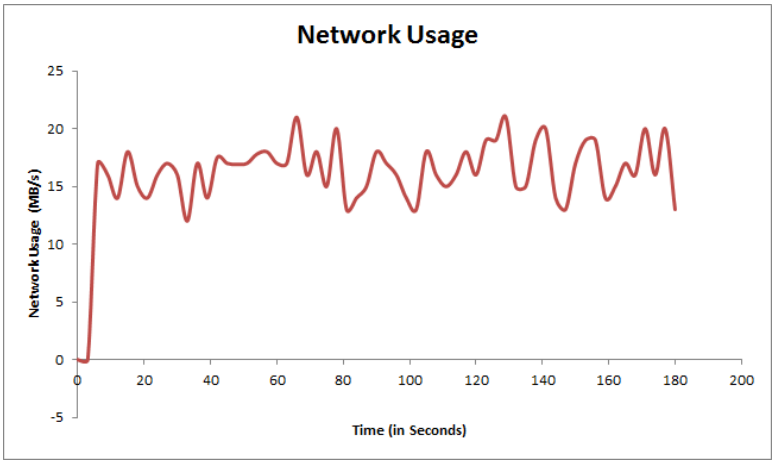

Figure 5.4: Network utilization for 2.57 GB file.

We described evaluation result for different file size but reading file size larger than 3gb will give memory error as Elasticsearch can have memory equal to system RAM. We founded that Elasticsearch require more RAM to parse larger file, which is not possible in every case. But still file parsing is very fast in fs-river plugin. Elasticsearch server

don't have multi part indexing for same file, so we have load the file completely in one time only, which could to lead to memory problems.

Chapter 6

Conclusions

Elasticsearch is very powerful distributed real time search engine which provides quick insight into the data by using different plugins, as shown in previous sections. We presented a Cloudsearch POC (proof of concept) application for Dell cloud environment, which is constructed by developing and integrating multiple plugins for different type of data source. This paper presents how we can extend the core capabilities of search engine which provide great insight into big data and increase customers efficiency to evaluate different business process. Using Elasticsearch engine in cloud environment give Dell customers real time full text based search capabilities over their environment. To increase reliability, Elasticsearch provides redundant sharding. Therefore, if one search node terminate, data can still be searched. Furthermore, the performance can scaled out horizontally.

However, ETL (Extract, Transform and Load) process for Elasticsearch can be complex to implement for every data source which qualifies as big data in cloud environment. As presented in evaluation section, indexing very large dataset into Elasticsearch node requires very carefully designed ETL process else it can lead to exception. Additionally, based on concern that elasticsearch is still under heavy development some features may or may not be supported in future.

Cloudsearch application focuses more on making data available for user insights using different framework and technology. Future work is required to improve the application searching capabilities using complex analysers, facets and term query. Furthermore, we intended to develop plugins for different data source including NoSQL and implementing custom dashboard to get better data insights.

Bibliography

[1] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599 – 616, 2009.

[2] Robert G. Byrnes" "Daniel J. Barrett, Richard E. Silverman. *"SSH, The Secure Shell: The Definitive Guide"*. "O'Reilly Media", "USA", 2 edition, 5 2005.

[3] "Paul DuBois". *"MySQL Developer's Library"*. "Addison-Wesley", "USA", 5 edition, 4 2013.

[4] Borko Furht and Armando Escalante. *Handbook of Cloud Computing*. Springer, Cambridge, Massachusetts, 2010.

[5] Robert Eckstein" "Gerald Carter, Jay Ts. *"Using Samba: A File and Print Server for Linux, Unix and Mac OS X"*. "O'Reilly Media", "USA", 3 edition, 1 2007.

[6] Y. Ichikawa and M. Uehara. Distributed search engine for an iaas based cloud. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 International Conference on*, pages 34–39, 2011.

[7] Matthew McCullough" "Jon Loeliger. *"Version Control with Git: Powerful tools and techniques for collaborative software development"*. "O'Reilly Media", "USA", 2 edition, 8 2012.

[8] T. Miyano and M. Uehara. Proposal for cloud search engine as a service. In *Network-Based Information Systems (NBiS), 2012 15th International Conference on*, pages 627–632, 2012.

[9] Marek Rogozinski" "RafaL Kuc. *"ElasticSearch Server"*. "packtpub", "Birmingham B3 2PB, UK", 2 2013.

[10] "John Ferguson Smart". *"Jenkins the defineitive gudie"*. "O'Reilly Media", "Gravenstein Highway North, Sebastopol, CA", 7 2011.

[11] "Ken Withee". *"SharePoint 2013 For Dummies"*. "Addison Wiley and Sons", "Canada", 1 edition, 1 2013.

[12] "Jason Van Zyl's". *"Maven the Definetive guide"*. "O'Reilly Media", "Cambridge, Massachusetts", 1 edition, 2 2008.