Master of Science in Computer Science
September 2019

Kylo Data Lakes Configuration deployed in Public Cloud environments in Single Node Mode

Rong Peng

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to

Contact Information:
Author(s):
Rong Peng
E-mail: rope17@student.bth.se

University advisor:
Dr. Emiliano Casalicchio
Department of Computer Science

Industrial advisors:
Kim Hindart
Daniel Gustafsson
City Network Hosting AB

ABSTRACT

Data Lake is a platform for centralized storage of massive, multiple sources and multiple types of data, and can quickly process and analyze data. It is essentially an advanced enterprise data architecture. The high demand for big data storage and analytics leads data lake emerging nowadays. Organizations hope to generate business profit from their data by building a data lake.

However, it is worth noting that the promising concept of data lake is still evolving today. People are unfamiliar with the new terminology and technology. The existing paper regarding data lake tends to make it sophisticated and sometimes hard to get the point. I have investigated some technical people around me, about their understanding of data lake, most of them answered me that they have never heard about it, not mention building a data lake by hand. Human are the tools promoter, and they are willing to promote the one they understand and handle well. Thus that, this paper's main goal is to provide the different perspective for readers to understand data lake better. This research tries to figure out the basic question of 'what is data lake'. In addition, the questions like 'why the data lake should be used', and 'what are the differences between the data lakes and the previous data warehouses or databases' will be answered in this paper. This paper aims at landing the concept to the ground and make it easy to understand.

Kylo data lake, is an open-source data lake platform, which is suitable for academic study. Kylo is a relatively new platform, and that's fewer papers regarding the configuration of Kylo data lake currently. And Kylo data lake relies on several underlying technologies, the architectures are worth exploring and studying. Configuration of Kylo data lake can provides a very meaningful perspective for the future data lakes development.

In this paper, Literature Review and Experiment are my main research methods. The Literature Review helps develop knowledge about data lakes. Through searching and reviewing scientific related work and technical documentation to get a comprehensive understanding. Experiment designs to test the single node Kylo data lake's ingestion capability, to see how the data volume and data format influence it. Through these two methods, Kylo data lake will be studied and interpreted in a good way.

Keywords: Big Data, Data Lake, Kylo, NiFi, ActiveMQ, Elasticsearch, Hadoop, Hive, Spark, Data Ingestion

ACKNOWLEDGMENT

I would like to thank City Network Hosting AB for providing me the City Cloud as the public cloud environment in this experiment. The collaboration of BTH and City Network provides me such a cherished opportunity, let me have a real practice of deploying the Kylo data lake in the Cloud.
I also want to thank my academic supervisor Dr. Emiliano Casalicchio and my industrial supervisors Kim Hindart, Daniel Gustafsson. Their constructive advice and kindly support help me move forward.
At last, I am grateful for my parents and my friends, they encouraged me all the time and offered me confidence, because of their warm care I can achieve today.

CONTENTS

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Problem Background

With big data growing rapidly over the past few years, the emergence of the concept of data lakes is getting even more attention from the research and industrial community. We cannot help asking what is the data lake? Laskowski, Nicole[1] gave his explanation, a data lake refers to a massively scalable storage repository that holds a vast amount of raw data in its native format («as is») until it is needed plus processing systems (engine) that can ingest data without compromising the data structure. The terminology explanation from him, the data lake should be a system can save vast amount of raw data. But he did not clearly answered why should we build a data lake? What are the differences between the data lakes and the previous data warehouses or databases?

Miloslavskaya, Natalia, and Alexander Tolstoy [2] argued that in contrast to the data warehouse, data lakes do not require structured data and pre-build static analytical applications, it can integrate seamlessly with various data sources. Data lakes enable organizations to store and analyze a vast amount of data, the data lakes are capable to handle large and quickly arriving volumes of unstructured data and use dynamic analytical applications to derive further insights [2].

According to an Aberdeen survey[3], besides the internal process efficiency, a well-conceived data lake helps set the stage for an elevated level of analytical activity, and those analytical activities can translate into business growth and profit boost. It also claimed that organizations who implemented a Data Lake outperforming similar companies by 9% in organic revenue growth [3]. A data lake can bring business profit from analyzing their data, but how to explore the "golden data"?
Data lake sounds a promising technology, the market is optimistic about this technology. But how to implement such architecture? That's rare to see an article providing the data lake implementation details.

The motivation for choosing Kylo Data Lake [4] has two main reasons. The first one is that Kylo is an open-source and feature-rich data lake platform, which is suitable for academic study. The second reason is from the City Network [5] requirement. The company is interested in the Kylo data lake technology. Kylo data lake built on Apache Hadoop and Spark, provides a turn-key, self-service data ingest, data preparation and data discovery. And it is compatible with Cloudera, Hortonworks, Map, R, EMR, and Vanilla Hadoop distributions [4]. Existing data lakes implementation leverage Cloud technologies for storage, processing, and access to the data. Therefore, the configuration and experiment will deploy on the City Cloud [5]. The reason for choosing the Single Node Mode dues to the function of City Cloud[5], it is time consuming to fully duplicate a same server, the clone function limit that if the original server is `root` role, then the copy one cannot be the same `root` role. The role might influence the installation procedure. The City Cloud servers' disaster recover services are easy to trigger, which means the distributed mode will make the real condition become hard to maintenance. The distributed mode can be explored in the future study. So I chose the Single Node

mode which help control variables easily. In addition, the materials about the configuration of Kylo are quiet rare to find.

1.2 Aim & Objectives

The aim of this research is to get a deep understanding of the data lake concept and know the underlying architecture and data ingestion capability of Kylo data lake. The objectives of the desired aim are:

-Develop knowledge about data lakes through searching and reviewing scientific related work and technical documentation.
-Get practice configuration of Kylo data lake in single-node mode to understand the underlying architecture.
-Ingest with different sizes and formats of datasets to see the data ingestion capability.

1.3 Research Questions

RQ1: What is the data lake? Why should we build data lakes? What are the differences between the data lakes and the previous data warehouses or databases?
RQ2: How to configure a Kylo data lake in the single node mode? What are the underlying technologies of Kylo?
RQ3: How the data size and formats influence the Kylo data lake?

The reason for raising the above questions stems from my study aim. The data lake emerges popular in the big data field nowadays, but what exactly the data lake is? The motivation for this study is to get deep knowledge of data lake. Kylo is an open-source data lake which is suitable to apply in academic research. And through literature review method, I want to land the data lake concept to the ground, explain the concept in an accepted way.

I found there are fewer configuration details of a data lake in current papers. Most of the articles would like to discuss about 'how to apply data lakes', or several big data concepts comparison, or function walk-through of data lake and so on. I hope to provide a configuration perspective for the data lake developers or the technology enthusiast, this perspective can promote the future study. That's why I hold the RQ2.

The RQ3 aims at studying how the data volume and format influence ingestion. The RQ3 plans to conduct several test cases and see the data ingestion results.

Chapter 2

Background

In this experiment involves the public cloud platform (City Cloud [5]) and several relative technologies knowledge like Kylo, NiFi, ActiveMQ, Elasticsearch and Hadoop ecosystem.

2.1 Public Cloud

Cloud computing can be dived into three modes: public cloud, private cloud and hybrid cloud. The public cloud provides services for the public users, and the private cloud provides service for private businesses or organizations, and the hybrid cloud is a combination of public cloud and private cloud.
From the perspective of this academic experiment, the public cloud mode would be the best choice. The deployment of the data lake conducted on the City Cloud. The City Cloud is a City Network Hosting AB Infrastructure as a Service (IAAS) service based on OpenStack [5]. The City Cloud offers network service, hardware service, and provides servers monitor dashboard and can be compatible with the most of OpenStack APIs. It supports the underlying utilities in this experiment.

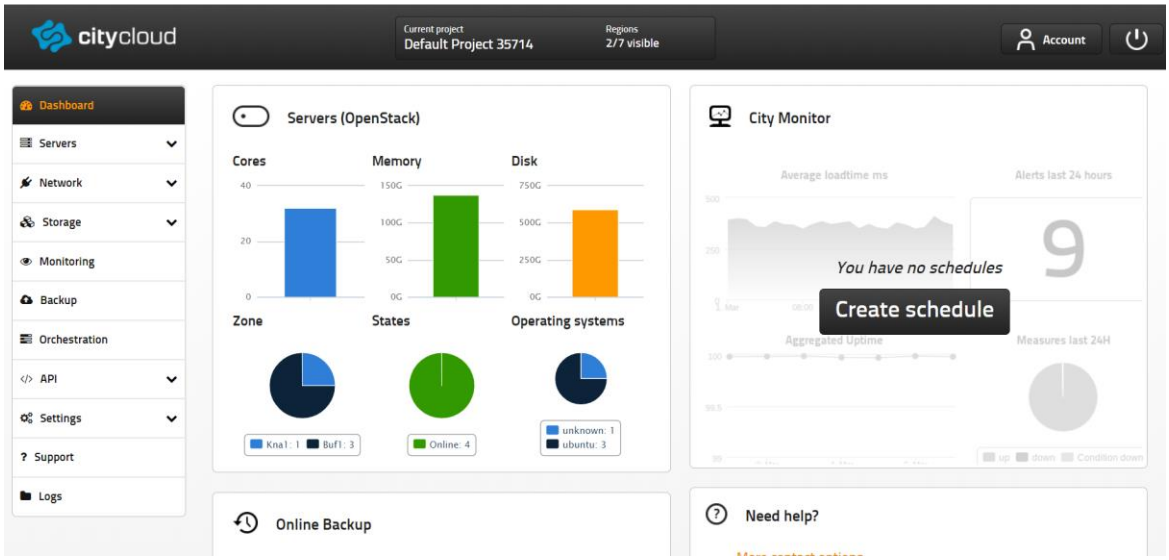


Figure 1.City Cloud Control Panel[5]

2.2 Kylo

The motivation of Kylo has two reasons, one is Kylo acts as an open-source data lake platform, which is suitable for academic research. Another reason is the City Network is optimistic about this technology.
Kylo is a play on a Greek word meaning “flow” and is a full-featured Data Lake platform built on Apache Hadoop and Spark [4]. This open-source data lake solution can provide data ingestion, data preparation, operation dashboard, and global search. Kylo was developed by Teradata company and its web application layer offers

features oriented to business users, including data analysts, data stewards, data scientists, and IT operations personnel[4]. Kylo is compatible with Cloudera, Hortonworks, Map R, EMR, and vanilla Hadoop distribution, and supports Apache NiFi, Hortonworks DataFlow (HDF)[4].
Regarding the components involved in a Kylo deployment, I would like to summary in the following tabulation.

Component	Description
Kylo UI	AngularJS browser app with Google Material Design running in a Tomcat container.
Kylo Services	Services, REST APIs, and plug-ins perform the backbone of Kylo. All features and integrations with other technologies are managed through the services layer.
Kylo Metadata Server	Combination of JBoss ModeShape and MySQL (or Postgres) store all metadata generated by Kylo.
ActiveMQ	JMS queue for inter-process communication.
ElasticSearch	Provides the index for search features in Kylo such as free-form data and metadata.
Apache NiFi	Pipeline orchestration engine and scheduler.
Apache Spark	Executes Kylo jobs for data profiling, data validation, and data cleansing. Also supports data wrangling and schema detection.
Kylo Spark Shell	Manages Spark sessions for data wrangling.
Apache Hadoop	All Hadoop technologies are available but most notably YARN, HDFS, Hive

Figure 2.Kylo Deployment Components [4]

Before configuring Kylo Data Lake into the public cloud, there are several supported dependencies should pay attention. The environment preparation presents as the followings.

Operating System	Version
RHEL, CentOs	6.x, 7.x
SUSE	V11, v12
Ubuntu	16.x, 17.x

Figure 3. Supported Operating System [4]

Hardware	Requirements
CPU	Minimum: 4 cores; Recommend: 8 cores
RAM	Minimum: 16 GB; Recommend: 32 GB

Figure 4. Hardware Requirements [4]

Category	Item	Description
Persistence	MySQL/Postgres/ MS SQL Server	Used to store both the Modeshape (JCR 2.0) and the Operational Relational(Kylo Ops Manager) metadata
JMS	ActiveMQ	Used to send message between different modules and to send Provenance from NiFi to Kylo
NiFi	NiFi	Either HDF or open source NiFi work
Spark	Spark Client	NiFi and Kylo have routines that leverage Spark
Hive	Hive	Required if using Hive and standard ingest template
Hadoop	HDFS	Required if using Hive and standard ingest template
Java	Java	The Kylo install will setup its own Java Home so it doesn 't affect any other Java versions running on the machine
Search	Elasticsearch / Solr	For index and search for Hive metadata and indexing feed data when selected as part of creating a feed

Figure 5.Kylo Stack Dependencies [4]

Kylo closely integrated with the components, the following table is for the checklist of network ports (Default port number).
The specific configuration steps and results would be presented in the Experiment part of this thesis.

2.3 Apache NiFi

NiFi was built to automate the flow of data between systems [6]. It is an easy way to handle and distribute the arriving data, with the graph to view your data routing, transformation, and system mediation. Apache NiFi has the web-based user interface, can design and develop the dataflow efficiently. And it works as Kylo scheduler and orchestration. The following figure is NiFi architecture schema diagram.

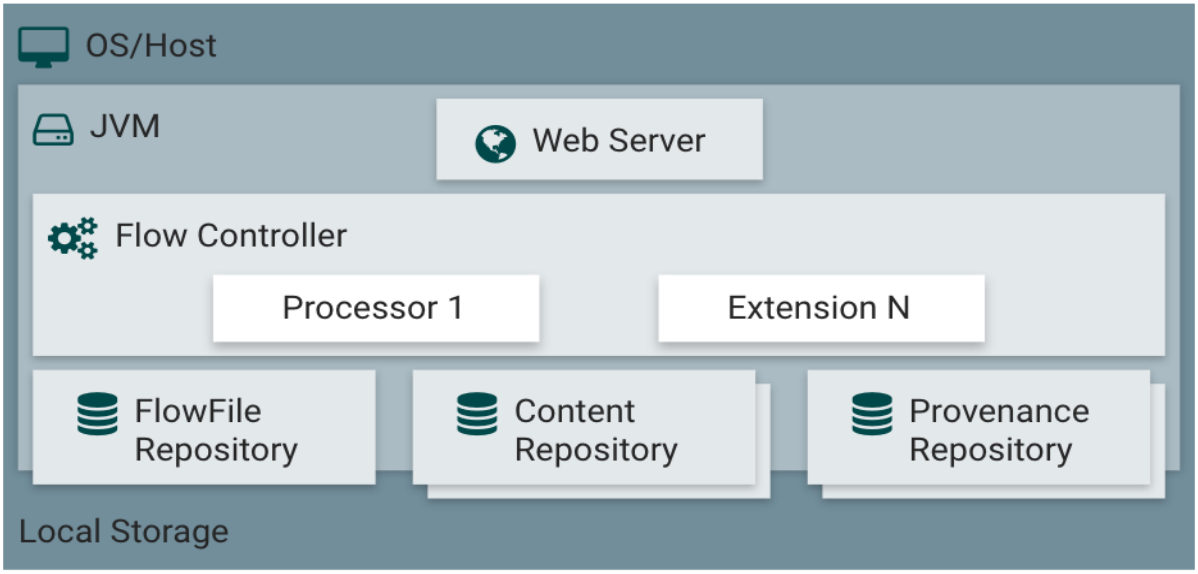


Figure 6. NiFi Architecture [3]

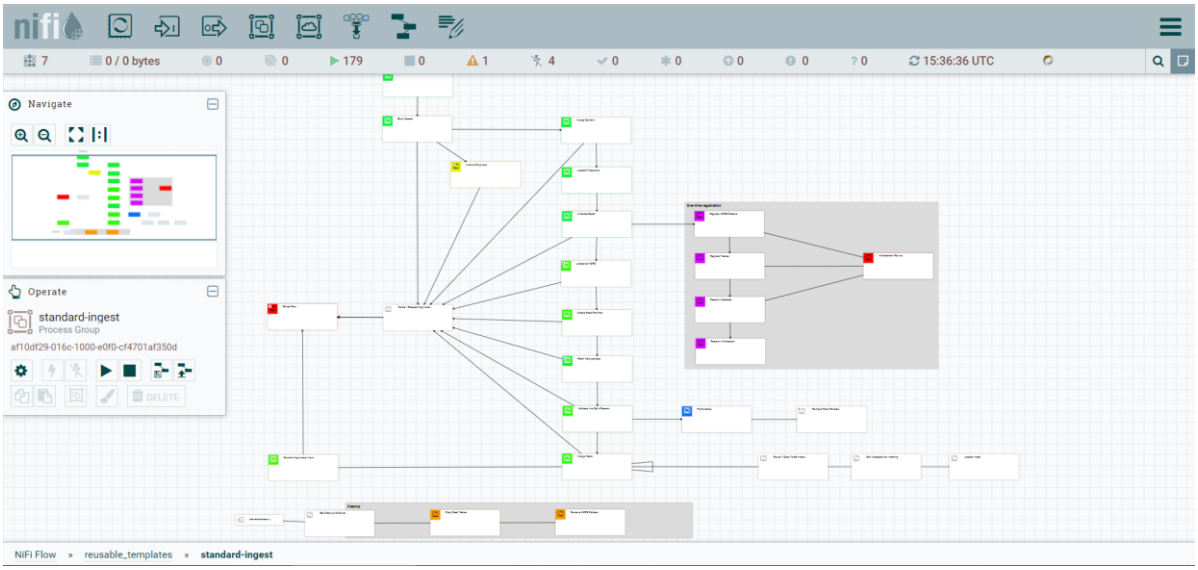


Figure 7.NiFi Page

NiFi is an open-source system, based on the concept of Flow-Based Programming. NiFi executes within a JVM on a host operating system [6]. The web server can handle the HTTP-based command and control API, and Flow Controller is the core part, which focuses on managing the schedule and receiving resources to execute. The FlowFile Repository is where NiFi keeps track of the state of what it knows about a given FlowFile that is presently active in the flow [6]. The Content Repository is for storing the actual data content, and the Provenance Repository is for storing all provenance event data.

2.4 Elasticsearch

Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases [7]. Elasticsearch can search for data with all types, for example, structured or unstructured text, numerical data, or geospatial data.
And it can do distributed storage, search and analyze in real-time. Elasticsearch can efficiently store and index it in a way that supports fast searches, can analyze logs, metrics, and security event data, as the data and query volume grows, the distributed nature of Elasticsearch enables the deployment to grow seamlessly right along with it[7].
Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stack. Logstash and Beats facilitate collecting, aggregating, and enriching your data and storing it in Elasticsearch[7].
Elasticsearch provides real-time search and analytics for all types of data. Whether you have structured or unstructured text, numerical data, or geospatial data, Elasticsearch can efficiently store and index it in a way that supports fast searches.

2.5 Apache ActiveMQ

Apache ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client. It provides "Enterprise Features" which in this case means fostering the communication from more than one client or server [8].
"Message Queue" is a container that holds messages during the transmission of a message. The Message Queue Manager acts as a middleman when relaying a message from its source to its destination. The main purpose of the queue is to provide routing

and guarantee the delivery of the message; if the recipient is not available when the message is sent, the message queue will retain the message until it can be successfully delivered. ActiveMQ acts as this role, it was built for a neutral web-based communicator. ActiveMQ Supports a variety of Cross Language Clients and Protocols from Java, C, C++, C#, Ruby, Perl, Python, PHP, full support for the Enterprise Integration Patterns both in the JMS client and the Message Broker[9].

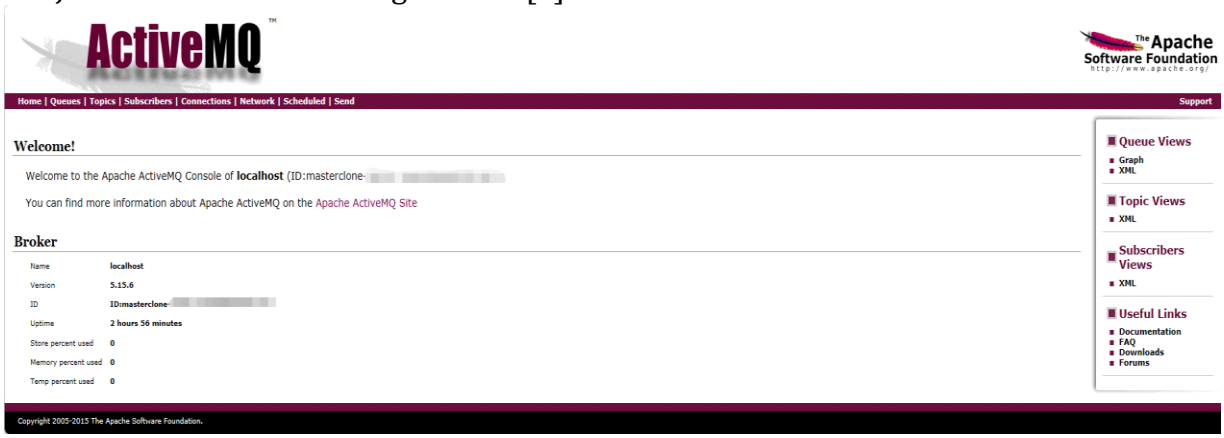


Figure 8. ActiveMQ Page

2.6 Hadoop Ecosystem

Hadoop is a software platform for developing and running large-scale data. It is an open source software framework implemented by Java language in Apache. It realizes distributed computing of massive data in a cluster composed of a large number of computers. Its framework core design are HDFS and MapReduce. HDFS provides storage of massive data, and MapReduce provides calculations for data.
The Figure2 presents the Hadoop Ecosystem architecture, it has grown into a huge system. In this paper, HDFS, Hive and Spark are the necessity parts during the data lake configuration.

HDFS with the feature of high fault-tolerant and high throughput, can be deployed on low-cost hardware and be used to access the large data set. HDFS is a master-slave architecture. A cluster consists of a NameNode and one or more DataNodes. The NameNode is the master server and used to manage the file's namespace and adjust client access files. The DataNode is the slave server and used to manage the storage. Most programs require HDFS file operations on a write-once, read many times.

Thrift Because the interface of the Hadoop file system is served through the Java API, it is cumbersome for other non-Java applications to access the Hadoop file system. The Thrift API can wrap the Hadoop file system with a Thrift service to make up for the deficiencies, allowing any language with Thrift bindings to easily interact with HDFS.

Hive is a data warehouse infrastructure built on Hadoop that performs data extraction and transformation loading (ETL). Hive is also a mechanism for storing, querying, and analyzing large-scale data stored in Hadoop. It also defines the HQL language, which is similar to SQL language, making it easy for users query data.

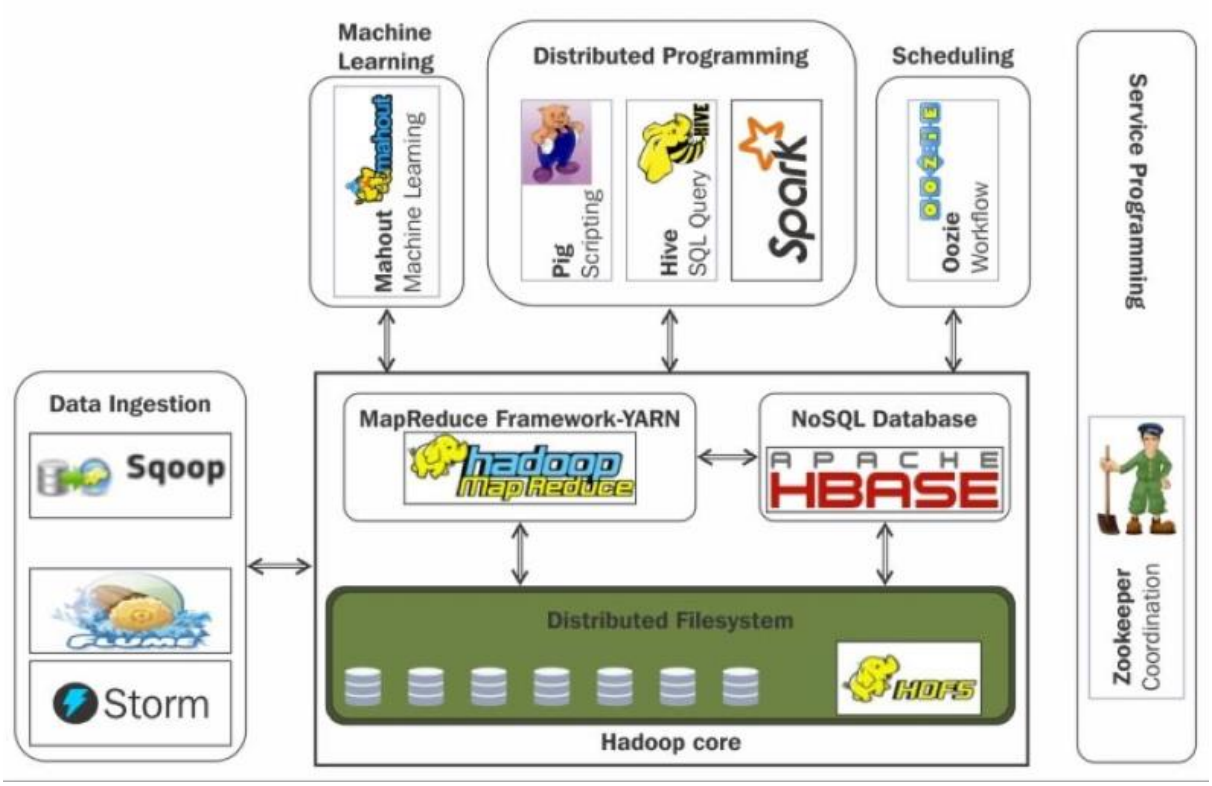


Figure 9.Hadoop Ecosystem [10]

MapReduce is a programming model for parallel computing of large data sets. In this process, you need to program the mapper and reducer. As the well-known word frequency statistics application example, the mapper is used for each word appearing in the map file set, and the reducer is responsible for processing the list of values of different words, and finally, the number of occurrences of different words in the file set can be obtained. This is a divide-and-conquer method.

The core process of MapReduce is divided into a shuffle and sort phases. Shuffle refers to the process starting from the map, including system execution sorting and transferring Map output to Reduce as input. The sort phase is to sort the output of the key by the map.

Spark is a fast and general-purpose cluster computing system[11]. Spark has the advantages of Hadoop MapReduce, but unlike MapReduce, Job intermediate output can be stored in memory, eliminating the need to read and write HDFS, so Spark can be better suited for data mining and machine learning.

Spark is implemented in the Scala language. Scala language expressive ability, more concise than Java program specification, fast development. Scala combines object-oriented and functional programming in one concise, high-level language [12].

Spark is based on memory for data processing. The spark has a DAG directed acyclic graph, and the DAG directed acyclic graph reduces the number of shuffles and landing disks in the process. Spark is a coarse-grained resource application, that is, when submitting a spark application, the application will apply for all resources. If the application does not have resources, it will wait. If the application is applied to the resource, the task does not need to be executed. To apply for a resource, the task executes quickly, and the task will be released when the last task is executed.

2.7 Ambari

The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters [13]. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs [13]. In a word, Ambari is a tool to make Hadoop and related big data software easier to use. And Kylo supports the combination of Ambari.

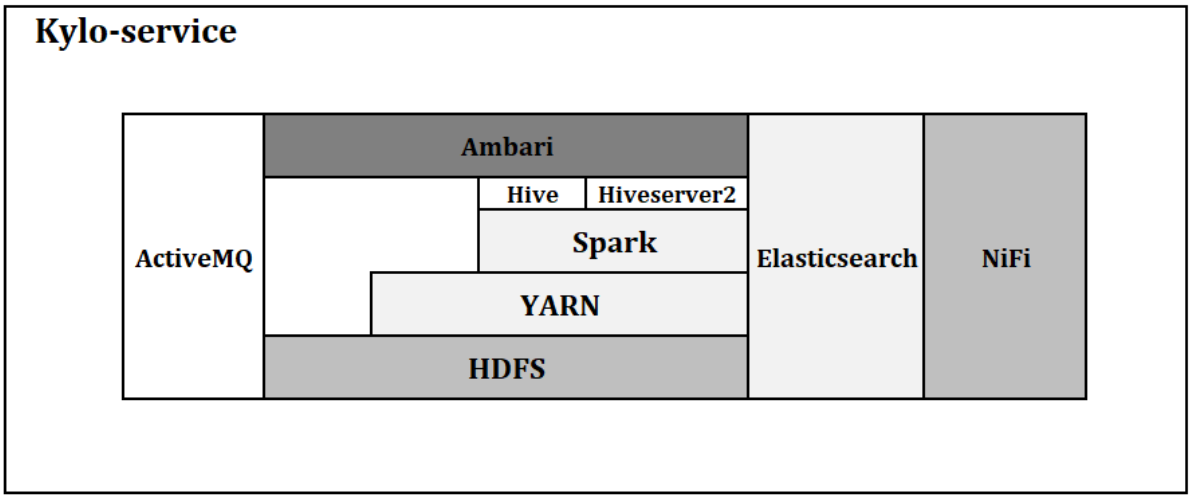


Figure 10. Basic Underlying Technologies Architectures

Chapter 3

Related work

A data lake still without an accepted definition now, what components or functionality a data lake should have or how an architecture looks like without clear regulation[14], Christian Mathis gave his view about Data Lake, a data lake should be able to storage, data ingestion, data profiling, and processing and encoding database systems, but data lake are still evolving concept, how to find new information from data lake requires techniques, tools, and processes which help data scientists, developers, and other stakeholders inside an organization to efficiently work with the data[14]. I agreed with this view, the data lake is a relative new concept today with the growth of big data, and data is still an evolving concept. And data lake replied on several kinds of technologies, such as Hadoop, Hive, Spark. Therefore, the concepts between kinds of big data technologies are hard to distinguish. Natalia Miloslavskaya and Alexander Tolstoy[15] compared Big Data, Fast Data, and Data Lake Concepts, they found the data lakes are well-managed and protected, have scale-out architectures with high availability, centralized cataloging, and indexing, shared-access model from any permitted modern device, use agile analytics and advanced data lineage (tracking), and data lake is on a higher turn of the spiral [15]. In their comparison, a data lake should be able to index data, centralized cataloging. Marilex Rea Llave*[16] pointed out the challenges of data lakes pointed out the data governance, data quality, and data retrieval are the most three challenges. And regarding data quality, for example, CLAM algorithm help improve enterprise data lake. CLAMS uses a new integrity constraint formalism to capture both relational model-like constraints and more expressive quality rules based on graph patterns[17].

Rihan Hai, Sandra Geisler and Christoph Quix[18] walk through the Constance data lake, named after Lake Constance, is the largest data lake in Germany. Constance is divided into three function layers: ingestion, maintenance, and querying. They introduced the demo with embedded query rewriting engines supporting structured data and semi-structured data and provides users a unified interface for query processing and data exploration. It's good to provide an overview of the functional perspective, but still not clear about the underlying architecture.

Articles regarding Kylo data lake are rare to see, not to mention the introduction of the underlying technologies. Data lake deployed in the Public Cloud is a nice perspective to study. In this study, I hope to make the concept easy to understand, study Kylo data lake underlying technologies and share the configuration details with the readers.

Chapter 4

Research method

The research methods chosen to answer the research questions posed in this thesis are the Literature Review and Experiment.

4.1 Literature Review Summary

A few years ago (in 2010) a new concept of «data lakes» or «data hubs» has been appears. The term itself was introduced by James Dixon (Dixon, 2010), but sometimes it is disparaged as being simply a marketing label for a product that supports Hadoop. Or we know also another vision: yesterday's unified storage is today's enterprise data lake (McClure, 2016) [19]. The data lake is an evolving concept as we can see, but it should content with the following features at least:

1. Allow to ingest and save raw data from multiple sources , and low-cost;
2. Data profiling and processing in quick time, allow to do data analysis and combine with machine learning;
3. Open and self-service.

The reason we build data lakes because of machine learning and data science becoming popular nowadays. Data analysis and forecasting in machine learning usually choose the raw data to process and analyze, while data warehouse dimensional models are often used for aggregation. On the other hand, machine learning will not just satisfied with using structured data. User's comments, images, and other unstructured data can also be applied into machine learning. Data lakes fit with the specified demand, then more and more businesses are willing to establish it.

Amazon compare the data lake and data warehouse, as following table:

Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

Figure 11. Compare data warehouse and data lake [20]

The differences between the data lake and data warehouse are obvious. The traditional way of working in a warehouse is centralized: the business staffs should present the requirements to the data team, and the data team processes and develops the dimensional table according to the requirements of the business.The traditional data warehouse is schema-on-write, the ingested data should be wrangled and processed in advance.But data lake is schema-on-read, can save the native format data. The traditional database storage capacity is smaller than data warehouse. Data warehouse

focused on OLAP, database focused on OLTP. Databases and data warehouses are more security than data lake.

Data Lake is an open, self-service system. The data lakes often include a semantic DB, a conceptual model that leverages the same standards and technologies used to create Internet hyperlinks, and add a layer of context over the data that defines the meaning of the data and its interrelationships with other data[19]. The data lake strategies can combine SQL and NoSQL database approaches and online analytics processing (OLAP) and online transaction processing (OLTP) capabilities[19].

Giving a scenario as an example, if a company contains a lot of data regarding the user's behavior, sales reports, employee information, operating conditions and so on. The decision-maker wants to buy a BI system for their company, then the BI company teams should understand their business, document business and confirm requirements, design BI demo, and then begin to do the ETL development. But if build a data lake, decision-maker can upload the company raw data into the lake by themselves, and it would be easier and quicker to get the data analytics result, then grap the market first. If the data lake deployed in the Cloud, the global company can share the common data lake platform, which saves power and reduces costs, make business coordination become simple.

Marilex Rea Llave*[16] provides empirical studies on the use of the data lake approach in enterprises. She identified three important purposes of implementing data lakes in an enterprise: (1) as staging areas or sources for data warehouses, (2) as a platform for experimentation for data scientists and analysts, and (3) as a direct source for self-service business intelligence[16].

Kylo data lake underlying architecture built on Hadoop, Spark, Hive, ActiveMQ, NiFi and Elasticsearch, and NiFi works as the pipeline orchestration engine. Kylo is a modern web application installed on a Linux and contains a number of special purposed routines for data lake operations leveraging Spark and Apache Hive [4].

Kylo utilizes Apache NiFi as its scheduler and orchestration engine, providing an integrated framework for designing new types of pipelines with 200 processors (data connectors and transforms). It has an integrated metadata server currently compatible with databases such as MySQL and Postgres [4].

4.2 Experiment

My experiment deployed in City Cloud [5] as the Public Cloud environment. And setup the Kylo data lake as the research object. Unexpectedly, Kylo requires complexed configuration steps. The following will introduce the whole configuration and experiment steps.

4.2.1 Hardware Parameters & Software Version

The following is the hardware parameters and software version. The minimum of hardware requires 4 cores CPU, 16 GB RAM. But the recommendation is the 8 cores CPU, 32 GB RAM. The following figure display my case of the experiment:

Name	Parameter
java	Jdk1.8.0
Kylo	V0.10.0
RAM	32GB
CPU	8 Cores
Disk	100GB
Metadata Storage	MySQL
Hadoop	2.7.6
Hive	2.3.4
Spark	2.3.3
NiFi	1.6.0
ActiveMQ	5.15.6
Elasticsearch	5.5.0
Operating System	Ubuntu 16.04

Figure 12. Hardware Parameters & Software Version

4.2.2 Network Ports Setting

The following ports list set by myself, which helps to check the background running jobs, to see and judge whether the services have been actived. You are also supported to configure and adjust the ports for your trial case.

Port	Service
8079	NiFi
8400	Kylo-ui
8451	Spark Jobs
10000	HiveServer2
8088	Hadoop Yarn
8420	Kylo-service
9200	Elasticsearch
8161	ActiveMQ Web
9300	Elasticsearch 2.x
50075	Hadoop DataNode
50070	Hadoop NameNode
8042	Hadoop NodeManager
8099	Kylo Configuration Inspector

Figure 13. Network Ports List

4.2.3 Configuration Steps

STEP1. Get prepared your Cloud server, using SSH remote terminal to connect your server, make sure the network works and update the version of the local command to the latest. For my case, I chose City Cloud and XShell. The server should contain the needed commands, such as `curl`, `netstat` etc.

STEP2. Install mysql-server, java and all the necessary tools refer to the following table left columns. Using `useradd`&`groupadd` commands to add roles, and change ownership for the corresponding files. Setting the *bin* path into environment path in ~/.bashrc file, and source it, example like the following.

```
export JAVA_HOME=/opt/java/jdk1.8.0_201
export JRE=$JAVA_HOME/jre
export JAVA=$JAVA_HOME/bin/java
export MVN_HOME=/opt/apache-maven-3.6.1
export MAVEN_OPTS="-Xms256m -Xmx512m"
export HADOOP_HOME=/opt/hadoop
export HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
export HIVE_HOME=/opt/hive
export HBASE_HOME=/opt/hbase
export SPARK_HOME=/opt/spark
export SPARK_SUBMIT=/opt/spark/bin/spark-submit
export SPARK_CONF=/opt/spark/conf
export PATH=$PATH:$JAVA_HOME/bin:$MVN_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$SPARK_HOME/bin:$HIVE_HOME/bin:$HBASE_HOME/bin:$SPARK_HOME/bin:$HADOOP_HOME/lib/native
export JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib:$HADOOP_HOME/lib/native"
```

Figure 14. Environment Path Configuration

Please note that you should download the version up to your real occasion, consider your server operating system and Kylo version, for my case Kylo is v0.10.0. Unzip the packages, ensure tomcat is in your environment, in my case, tomcat is under Hadoop's directory.

Installation List	Role/Group
Mysql	mysql
NiFi	nifi
Elasticsearch	elasticsearch
ActiveMQ	activemq
Kylo	kylo
Java JDK	java
Tomcat	tomcat
Hadoop	hadoop/hdfs
Hive	hive
Spark	spark
Maven	
mysql-java-connector.jar	

Figure 15. Installation List

STEP3. Configuration of technologies details introduced as following.
Under the Kylo setup package, you can find NiFi, ActiveMQ and Elasticsearch install scripts, input all the required paths as parameters to execute and install, the order of input parameters should pay attention. As the following tables, $1 to $5 parameters should locate in correct way.

Script Name	$1	$2	$3	$4	$5
install-nifi.sh	NiFi_Version	NiFi_Install_Home	NiFi_USER	NiFi_Group	Setup_Folder
install-activemq.sh	ActiveMQ_Install_home	ActiveMQ_User	ActiveMQ_Group	ActiveMQ_Java_Home	
install-elasticsearch.sh	Setup_Folder	ES_Java_Home			
install-component.sh	NiFi_Install_Home	Kylo_Install_Home	NiFi_User		
post-intsall.sh	Install_Home	Install_User	Install_Group		

Figure 16. Kylo installation scripts list and input parameters order

Above install scripts you should execute during configuration. In my case, the path of Kylo folder is /opt/kylo. The path for Setup_Folder is different from the path of Kylo_Install _Home. For example, Setup_Folder is /opt/kylo/setup in my case, and Kylo_Install _Home is /opt/kylo. The other Home Path you should specify where you want to locate your directories. The post-install.sh script is used to generate the kylo-services and kylo-ui under the /etc/init.d.

3.1 Configure NiFi

Export Java_Home path to the NiFi_Path/current/bin/nifi.sh and `sudo systemctl daemon-reload ` then start nifi service. You can check the 8079 port or surf the websit http://server_IP:8079 to check whether NiFi can run.

3.2 Configure ActiveMQ

Open the ActiveMQ_Path/bin, find the *activemq* file, export Java path to the file.
Copy *activemq* to the /etc/init.d and use command `sudo systemctl daemon-reload ` to refresh, then `sudo service activemq start` to start your activemq as service. Then copy *env* to the /etc/default directory.

3.3 Configure Elasticsearch

Change the ownership for elasticsearch setup file, it cannot start by root.
Modified Elasticsearch/config/elasticsearch.yml, uncomment Java_HOME and add your java home path. And uncomment ` SysV init.d`. When executing the init script, this user will be used to run the elasticsearch service. The default value is 'elasticsearch' and is declared in the init.d file. You can specify your port for elasticsearch in the .yml file, specify "http.port:9200". The *elasticsearch.yml* file should put in /etc/default directory, but *elasticsearch* should copy under the /etc/init.d directory and run as a service.
Try `curl –XGET http://localhost:9200` can see the JSON format result as the following, it means your configuration has done.

```
ubuntu@masterclone:/etc/init.d$ curl localhost:9200
{
  "name" : "Y796RJd",
  "cluster_name" : "demo-cluster",
  "cluster_uuid" : "P91BipLASl6AWmFF2H0IbQ",
  "version" : {
    "number" : "5.5.0",
    "build_hash" : "260387d",
    "build_date" : "2017-06-30T23:16:05.735Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 17. Elasticsearch Test

After that, to leverage an existing Elasticsearch instance, you must update all feed templates that you created with the correct Elasticsearch URL. Input the following command to shell:
'Kylo_Home/bin/create-kylo-indexes-es.sh localhost 9200 1 1'

3.4 Configure Hadoop

This case I used Apache Hadoop, actually Apache Ambari project provided easy-to-use Hadoop management, whatever which one you chose, you have to configure as your case. Go to Hadoop_Dir/etc/hadoop, you can find *hadoop-env.sh, core-site.xml*, *hdfs-site.xml*, *mapred-site.xml*, *yarn-site.xml* and *slaves,* they are the files should be configured. Consider my case is configuring in the Single Node, thus the parameter will set Hadoop in standalone mode. Parameters showing like the following table:

File	name	value
core-site.xml	fs.defaultFS	hdfs://localhost:9000
	hadoop.proxyuser.kylo.groups	*
	hadoop.proxyuser.kylo.hosts	*
	hadoop.proxyuser.root.groups	*
	hadoop.proxyuser.root.hosts	*
	hadoop.proxyuser.root.groups	*
	hadoop.proxyuser.root.hosts	*
	hadoop.native.lib	true
hdfs-site.xml	dfs.replication	1
mapred-site.xml	mapred.framework.name	yarn
yarn-site.xml	yarn.resourcemanager.hostname	localhost
	yarn.resourcemanager.webapp.address	0.0.0.0:8088
	yarn.timeline-service.webapp.address	0.0.0.0:8188
	yarn.log.server.url	http://0.0.0.0:19888/jobhistory/logs
	yarn.acl.enable	false
	yarn.resourcemanager.resource-tracker.address	0.0.0.0:8025
	yarn.resourcemanager.scheduler.address	0.0.0.0:8030
	yarn.resourcemanager.scheduler.class	org.apache.hadoop.yarn.server.resourceman ager.scheduler.capacity.CapacityScheduler
hadoop-env.sh	export JAVA_HOME=/opt/java/jdk1.8.0_201	
slaves	localhost	

Figure 18. Hadoop Configuration Parameter

3.5 Configure Hive

Hive should configure the hive-site.xml, the parameter as the table:

File	name	value
hive-site.xml	javax.jdo.option.ConnectionURL	jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=tr ue &useSSL=false
	javax.jdo.option.ConnectionDriverName	com.mysql.jdbc.Driver
	javax.jdo.option.ConnectionUserName	root
	javax.jdo.option.ConnectionPassword	hadoop
	hive.server2.allow.user.substitution	true
	hive.server2.authentication	NONE
	hive.server2.thrift.port	10000
	hive.server2.use.SSL	false
	hive.server2.transport.mode	binary
	hive.server2.thrift.http.path	cliservice
	hive.security.authorization.manager	org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd. SQLStdConfOnlyAuthorizerFactory
	hive.metastore.warehouse.dir	/app/warehouse
	hive.metastore.schema.verification	false

Figure 19.Hive Configuration Parameter

Copy the mysql-connector-java.jar to Hive_path/lib, and then initialize hive schema in mysql, use `schematool -dbType mysql -initSchema` command, let you initialize.
Hive has three interaction ways. The first one is running the Hive_path/lib/hive, entering into Hive shell to input Hive commands. The second one is using command directly, such as Hive_path/bin/hive –e 'select * from <table_name>' , it can show the result in the current shell window. The third one is using beeline, run the Hive_path/bin/hiveserver2, 'beeline>' will display, input command like '!connect jdbc:hive2://localhost:10000' can remote access the hive. Test in these three ways can help check whether Hive configuration has done.
By the way, if you use derby as metadata storing, then it should be `schematool -dbType derby -initSchema`.

3.6 Configure Spark

Copy the hive-site.xml to Spark_Dir/conf, and then export Java, Hadoop and Spark path to spark-env.sh. In order to provide an interface Spark SQL queries, we should open the thrift interface. Run the Spark_Dir/sbin/start-thriftserver.sh , it will show "starting org.apache.spark.sql.hive.thriftserver.HiveThriftServer2…" which means started.

Ambari supports most Hadoop components, including HDFS, MapReduce, Hive, Pig, Hbase, Zookeeper, Sqoop, and Hcatalog. The packages as the following, Ambari provides an option to graphically install and manage Hadoop clusters. My personal view to study on Kylo underlying technologies, configure on Apache Hadoop can help understand.

```
[root@sandbox current]# ls
hadoop-client                 hadoop-hdfs-zkfc              hbase-client        kafka-broker    shc                  spark-thriftserver
hadoop-hdfs-client            hadoop-httpfs                 hbase-master        livy2-client    slider-client        sqoop-client
hadoop-hdfs-datanode          hadoop-mapreduce-client       hbase-regionserver  livy2-server    spark2-client        sqoop-server
hadoop-hdfs-journalnode       hadoop-mapreduce-historyserver hive-client        livy-client     spark2-historyserver storm-slider-client
hadoop-hdfs-namenode          hadoop-yarn-client            hive-metastore      livy-server     spark2-thriftserver  tez-client
hadoop-hdfs-nfs3              hadoop-yarn-nodemanager       hive-server2        phoenix-client  spark-client         zookeeper-client
hadoop-hdfs-portmap           hadoop-yarn-resourcemanager   hive-server2-hive2  phoenix-server  spark-historyserver  zookeeper-server
hadoop-hdfs-secondarynamenode hadoop-yarn-timelineserver    hive-webhcat        pig-client      spark_llap
```

Figure 20. Ambari Package

STEP4. Configure Kylo-service and start the setup-wizard.sh.
Open the Kylo_Home/conf/application.properties, modify the following attributes, the following is my case as an example:

Attribute	Value
spring.datasource.username	root
spring.datasource.password	hadoop
hive.datasource.username	root
hive.datasource.password	hadoop
hive.metastore.datasource.username	root
hive.metastore.datasource.password	hadoop
nifi.service.mysql.database_user	root
nifi.service.mysql.password	hadoop
nifi.service.kylo_mysql.database_user	root
nifi.service.kylo_mysql.password	hadoop
nifi.service.kylo_metadata_service.rest_client_password	thinkbig

Figure 21. Kylo application.properties Configuration Parameter

After finishing this step, go to Kylo_Setup_Dir, and start setup-wizard.sh:

```
ubuntu@master:/opt/kylo/setup$ sudo ./setup-wizard.sh
The working directory is /opt/kylo/setup
Welcome to the Kylo setup wizard. Lets get started !!!

Please enter Y/y or N/n to the following questions:

Enter the kylo home folder location, hit Enter for '/opt/kylo':
Enter the kylo linux user, hit Enter for 'kylo':
Enter the kylo linux group, hit Enter for 'users': kylo

Would you like to install the database scripts in a database instance? Please enter y/n: y
Would you like Kylo to manage installing and upgrading the database automatically? Please enter y/n: y
Which database (Enter the number)?
1) MySQL
2) PostgresSQL
3) SQL Server
> 1

Please enter the database hostname or IP, hit Enter for 'localhost'
>
Please enter the database ADMIN username
> root
Please enter the database ADMIN password
> Creating MySQL database 'kylo'
mysql: [Warning] Using a password on the command line interface can be insecure.

Please enter the password for the dladmin user
>
Please re-enter the password for the dladmin user
>
```

Figure 22. Kylo setup-wizard.sh

When ask to install ActiveMQ ,NiFi and elasticsearch, you should input ' N' if you have done the installation in the previous steps. Then run the Kylo_Setup_Dir/nifi/create-symbolic-links.sh to link the correspond nar files tou your NiFi working library.

STEP5. Start all services in turn. After starting Hadoop, using `hadoop fs –mkdir <file_name>` to create the following directories in HDFS.

HDFS Urls	Description
/etl	Root HDFS locations for new raw files
/archive	Root Archive location for new raw files
/model.db	Root HDFS location for Hive ingest processing tables (raw,valid,invalid)
/app/warehouse	Root HDFS location for Hive master table

Figure 23. Locations in HDFS

STEP6. Before starting Kylo services, check the other dependent network ports has started, to make sure servies are actived successfully. 'netstat -tpln' can help check the ports.

Kylo has its kylo-install-inspector service, start kylo-install-inspector service, and input the url Your_Server_IP:8099 in Chrome, you can see the inspector web page. Input your Kylo_Home path, it will help to check whether you configuration right or not. As following screenshot showing:

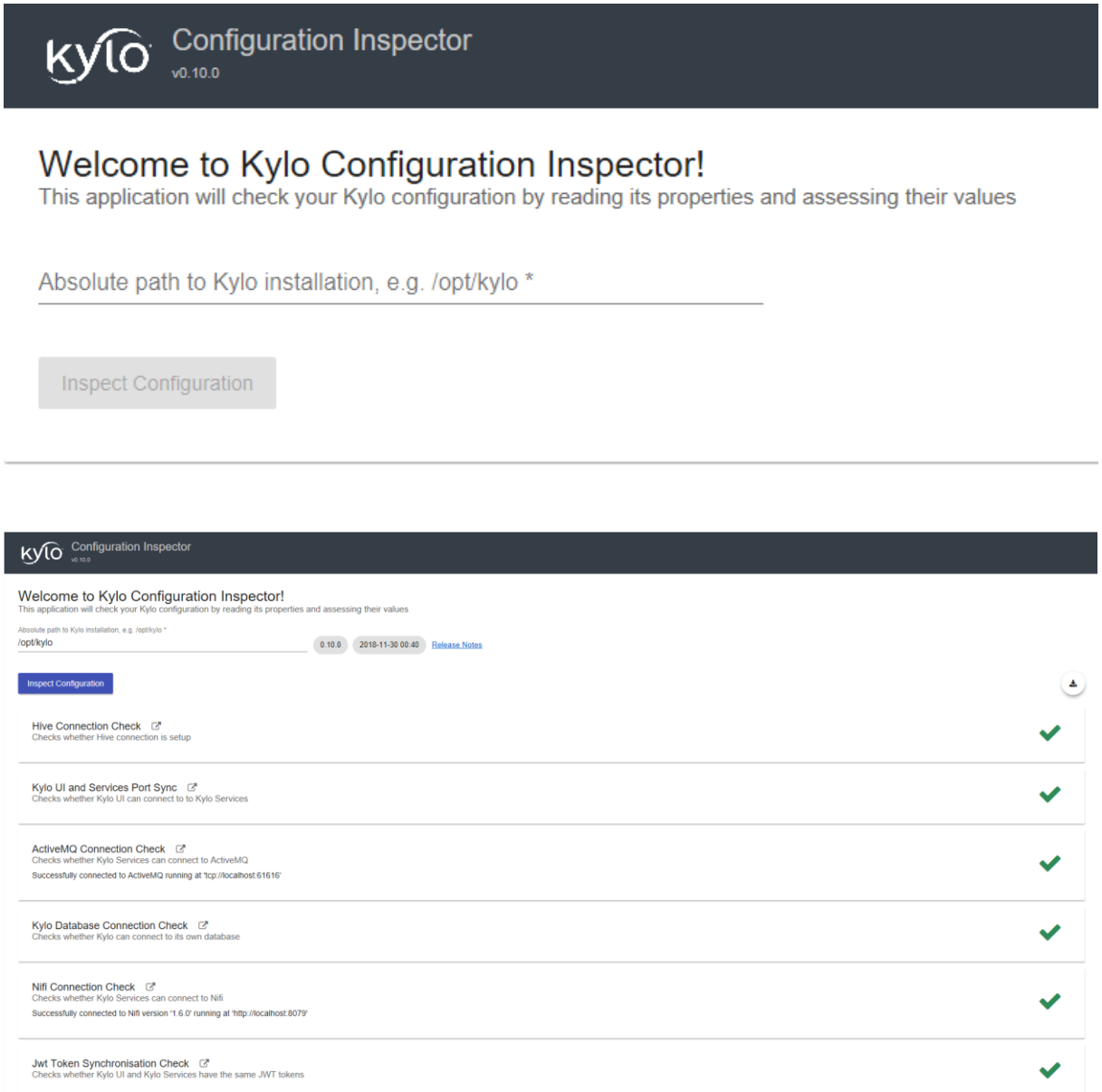


Figure 24. Kylo Configuration Inspector

STEP7. Start Kylo Services

If you configuration is right, the login web page will display, and login with username/password.

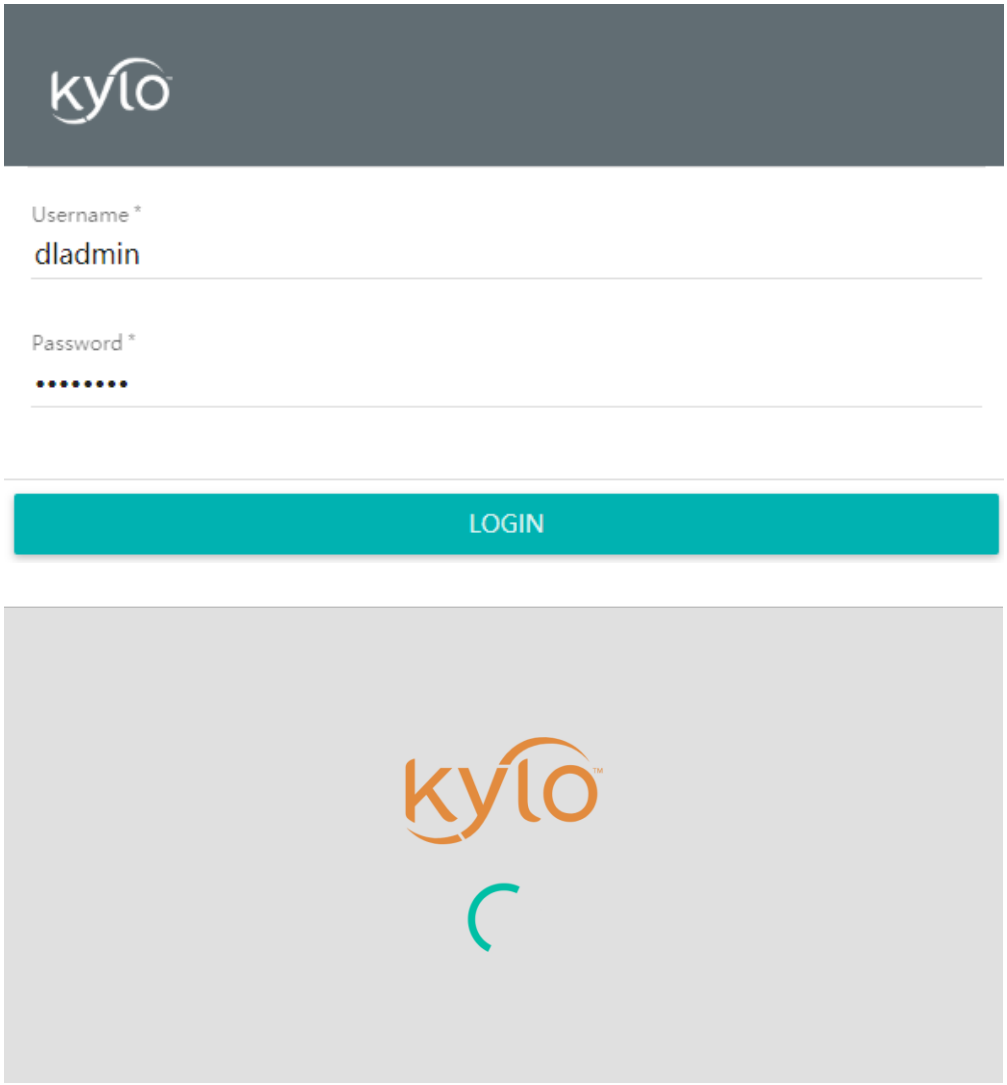


Figure 25. Kylo Login then Loading

After login and loading, this following page will show, which means login successfully. You can try to ingest some data to test.

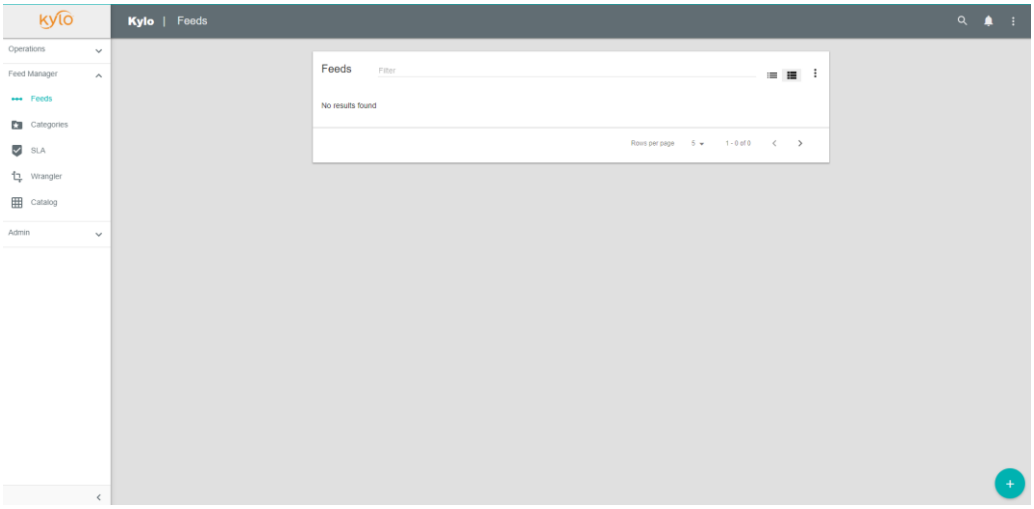


Figure 26. Kylo Home Page

4.2.4 Functions Walk-through

Catalog: In this page, you can add or check your data sources. For example, click HDFS can check the HDFS directories under the path.

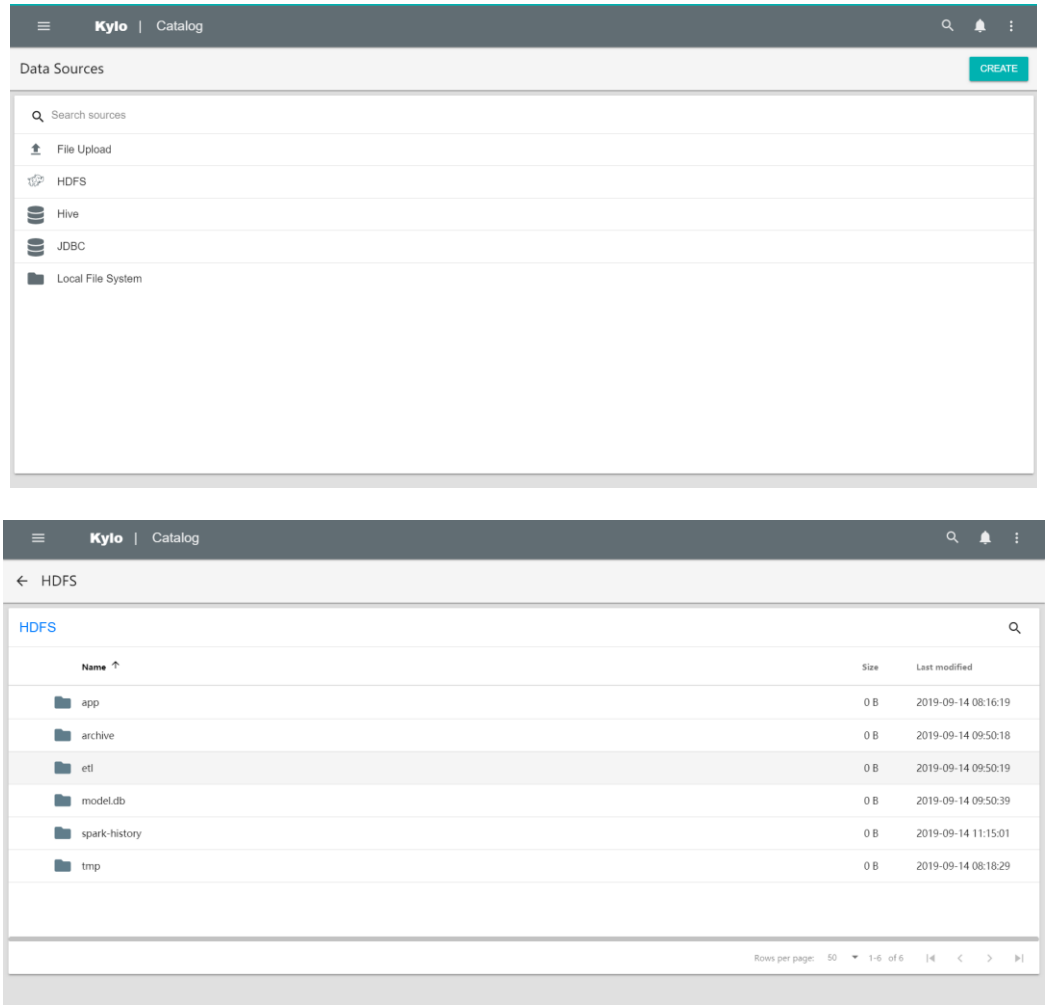


Figure 27. Kylo Catalog

Template: In this page, you can select the NiFi templates to import.

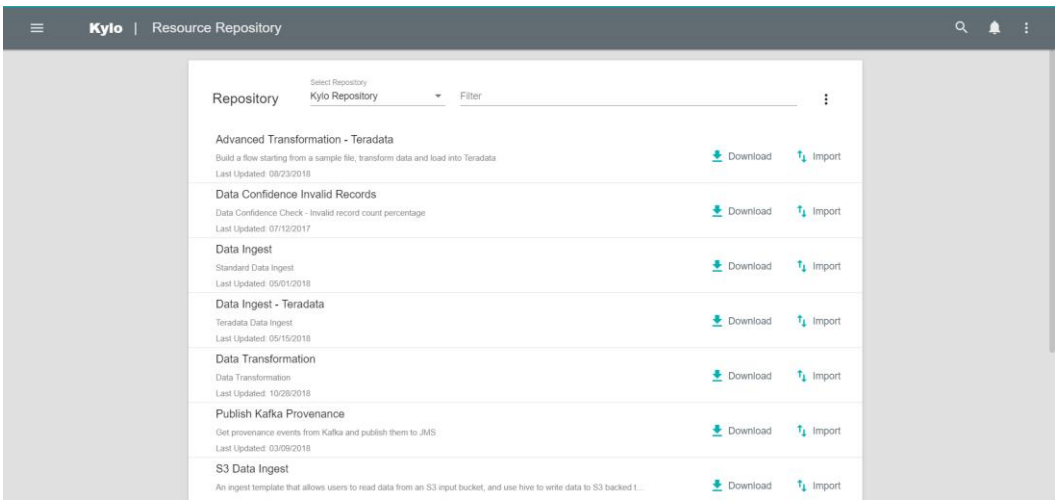


Figure 28. Kylo Remote Repository

Categories: In this page, you can create your data type, classify your data types.

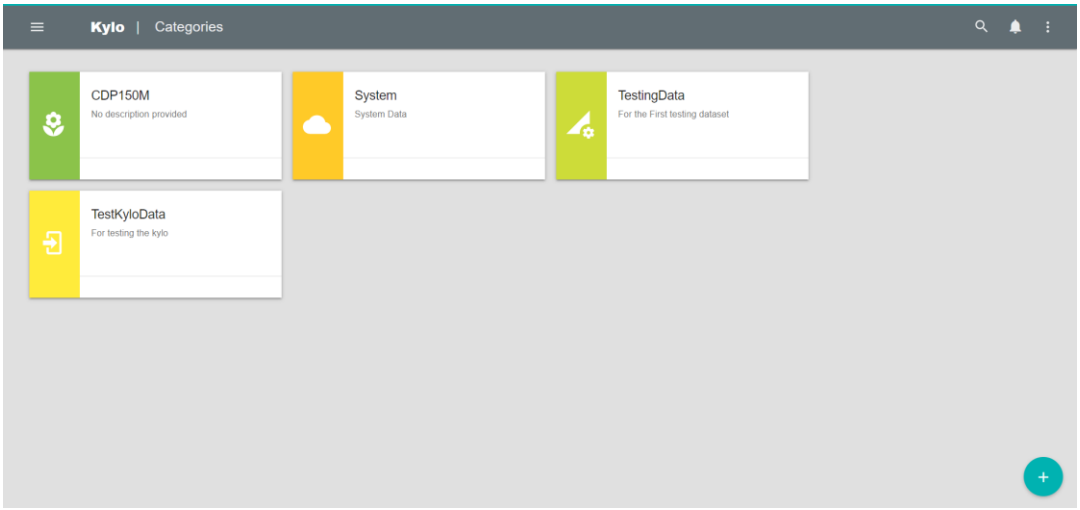
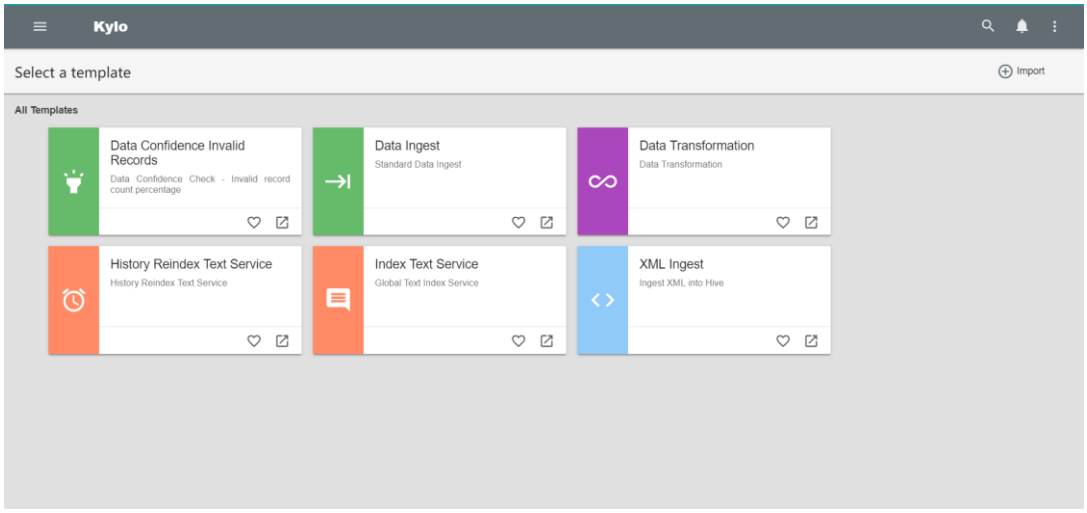


Figure 29. Kylo Categories

Feeds/Feeds Summary: In the following pages, select a template, such as Standard Data Ingest, then fill all your Feed inforamation. Feed represent the key movement of data between a source(a flat file) and sink(e.g. Hive)[1].



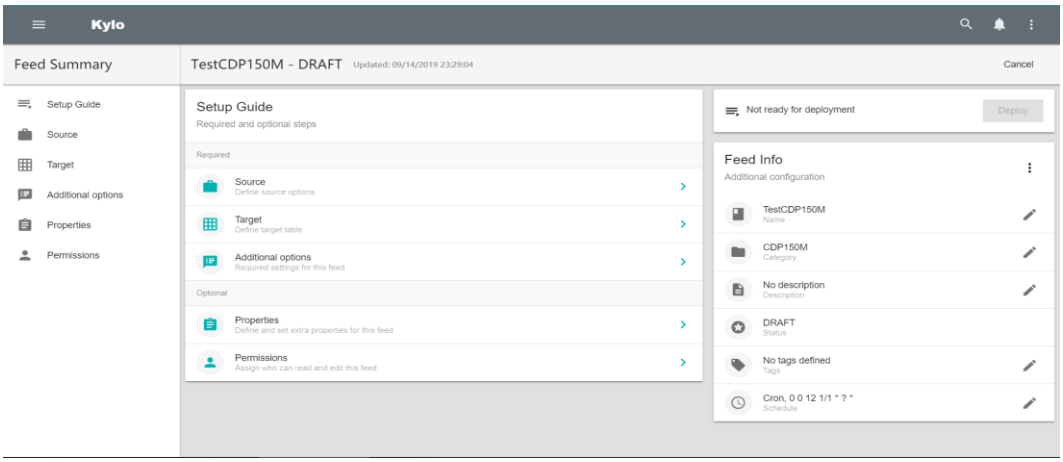


Figure 30. Kylo Feed Draft Generation

Deploy Feed: When you done the draft of Feed, deploy it and start. Click the button beside "Summary", upload file to ingest data lake. After several minutes, you can see the Profile page, it contains Profile Statistics. The Lineage includes show the running stream.

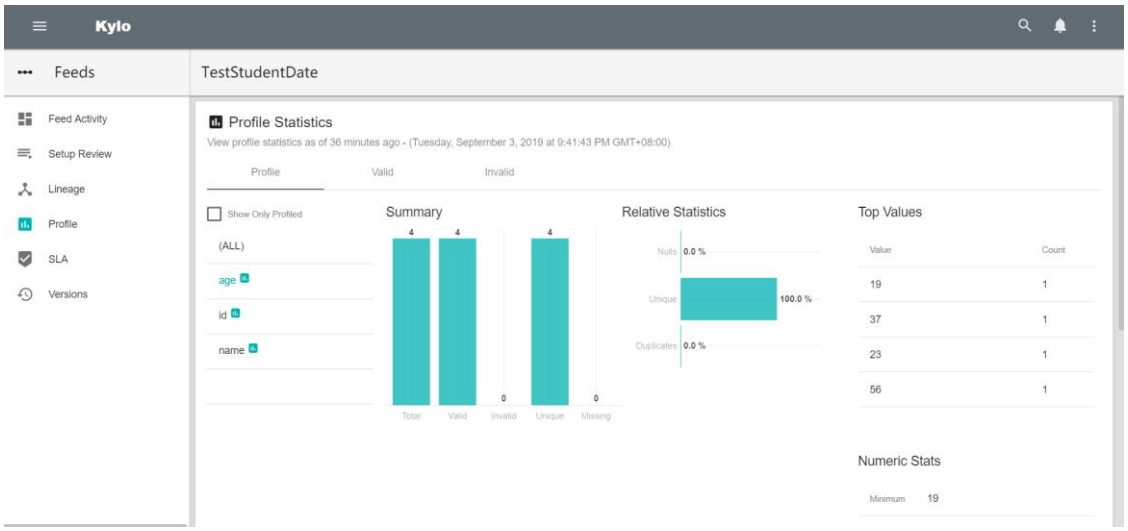


Figure 31. Kylo Feed and Profile Statistics

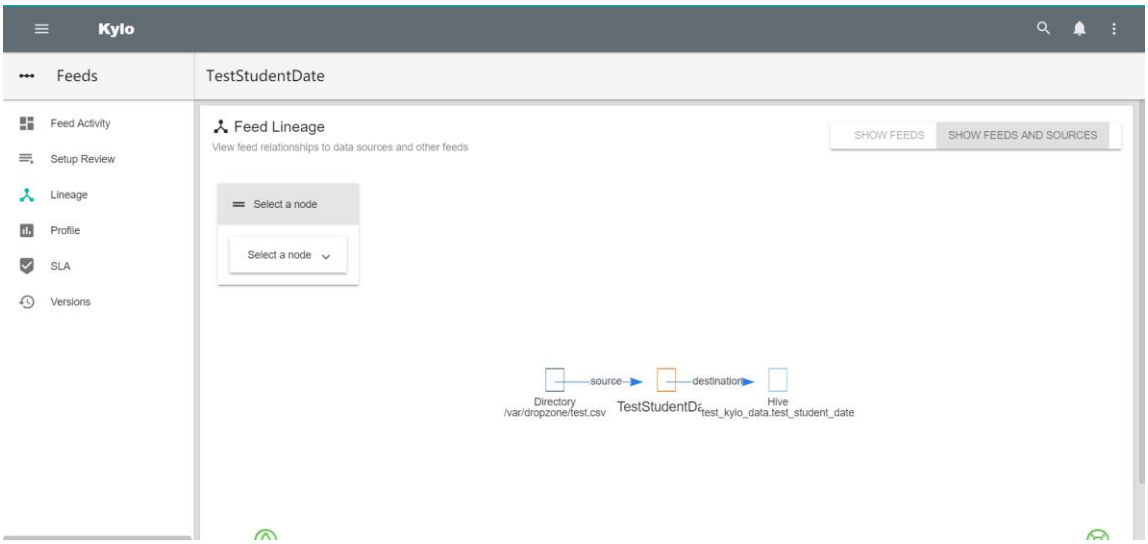


Figure 32. Kylo Feed Lineage

Wrangler: Allow to modify the entire dataset, rename, delete and move columns, even can view the graphical statistics.

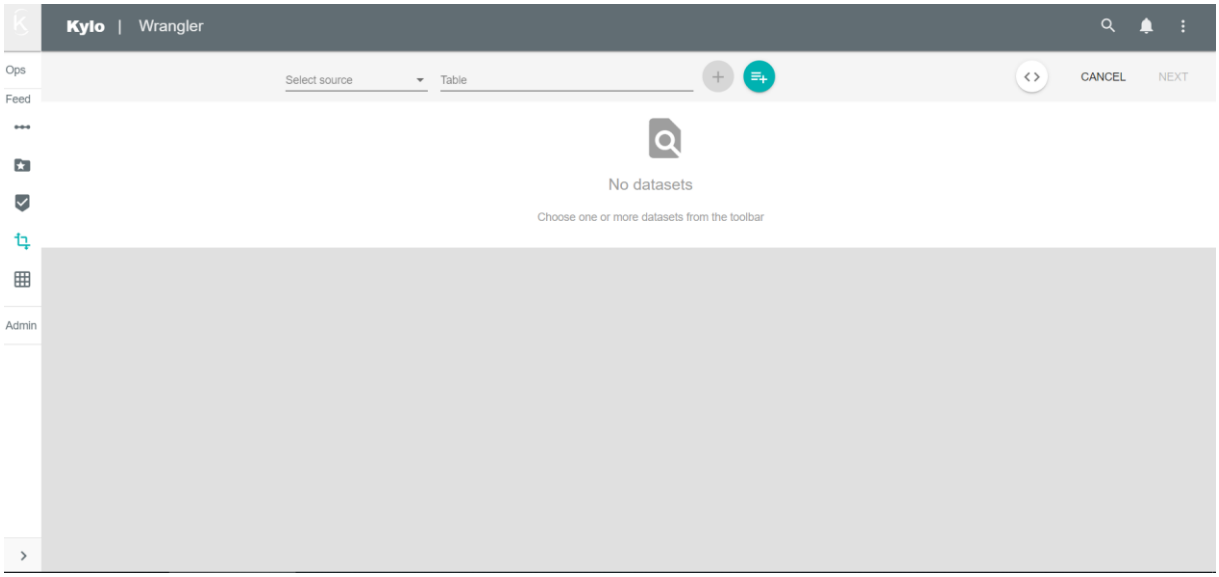


Figure 33. Kylo Wrangler

4.2.5 Running Command Set

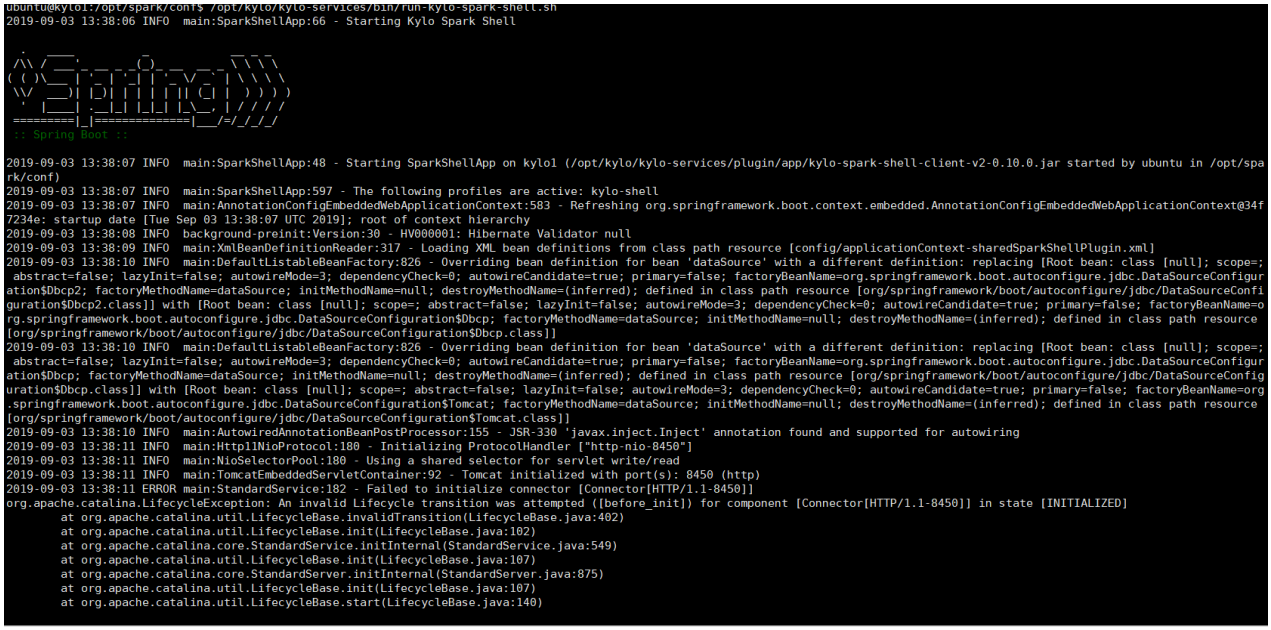
For the convenience of future researchers, I list some of the commands I used:

sudo service mysql start sudo service mysql status sudo service activemq start sudo service activemq status sudo service elasticsearch start sudo service elasticsearch status sudo service nifi start sudo service nifi status hadoop namenode -format /opt/hadoop/sbin/start-all.sh hdfs dfs -mkdir <file_name> hdfs dfs -chown role:role <file_name>	sudo service kylo-install-inspector start sudo service kylo-ui start sudo service kylo-ui status sudo service kylo-services start sudo service kylo-services status /sudo kylo-service strart netstat -tpln

Figure 34. Command Record

4.2.6 Limitation

When I configured the Spark in Kylo, I faced an unexpected exception as followings. And then I open the Kylo_Services_Path/bin/run--kylo-spark-shell.sh, `sudo find / -name kylo-spark-shell-pgrep-marker/com.thinkbiganalytics.spark.SparkShellApp` cannot find these two files. Then I went to website searching, hope they are in the remote maven repository, however, I cannot find anything related to the "/opt/kylo/kylo-spark-shell-pgrep-marker" and "com.thinkbiganalytics.spark.SparkShellApp". Thus the front-end cannot get the timestamp and Job Execution History, the front-end cannot get the data from back-end, the context almost null. That 's an unexpected official issue will influence the Job Status display.



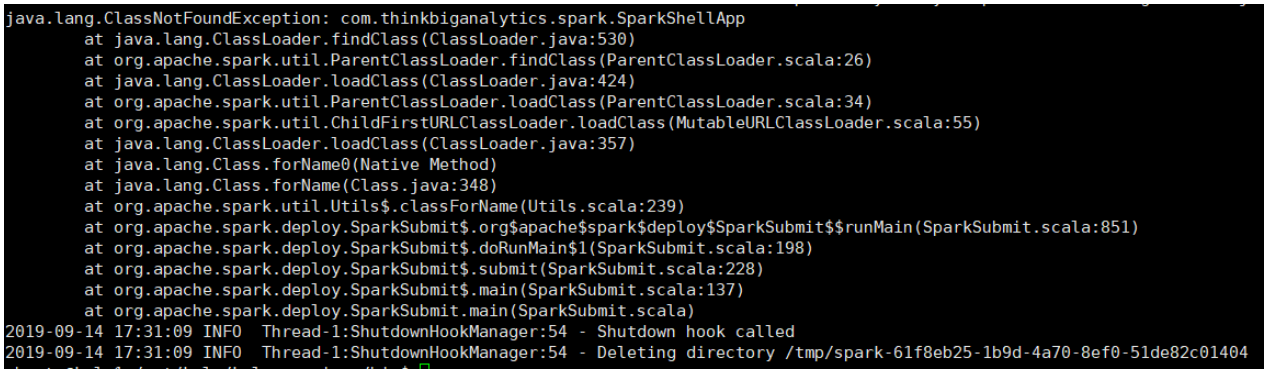


Figure 35. Kylo Spark-Shell Files loss

What is the issue will cause? The most obvious one is the front-end activity and execution history data loss. As the following shows:

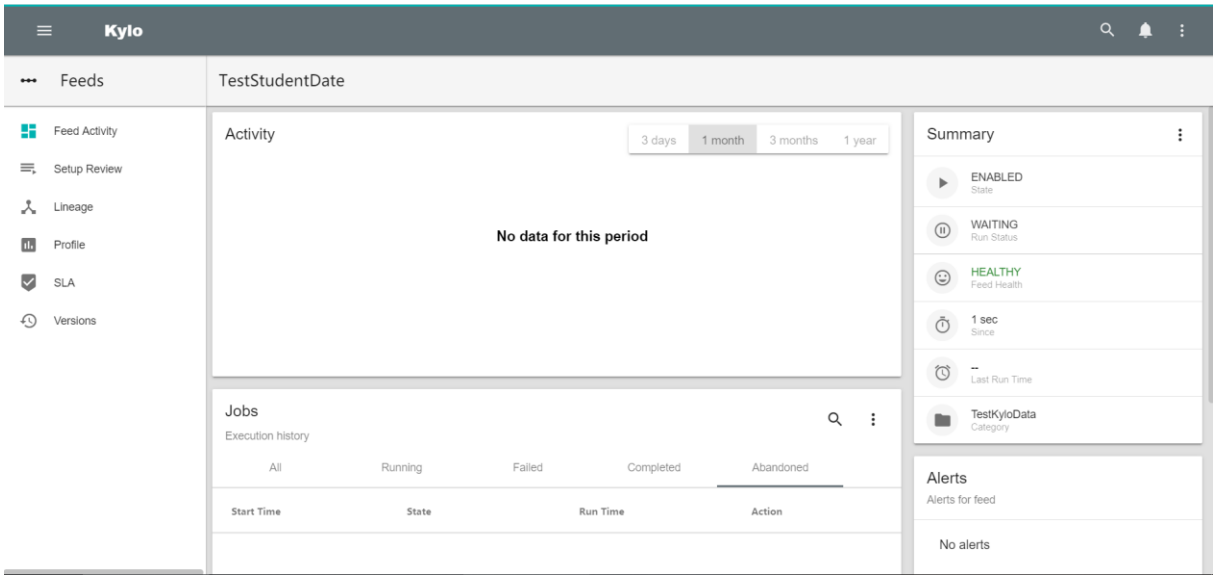


Figure 36. Kylo Front End – Cannot access the history

My alternative solution for this issue is running Spark History Server. You should configure the Spark_Path/conf/spark-default.conf, as the following:

```
# Example:
spark.driver.extraLibraryPath /opt/hadoop/lib/native/
spark.executor.extraLibraryPath /opt/hadoop/lib/native
spark.eventLog.dir hdfs:///spark-history
spark.eventLog.enabled true
```

Figure 37. Spark-default.conf configuration

The default port is 18080, go to the directory Spark_Dir/sbin/start-history-server.sh and run. If successfully configured, you can use the Server_IP:18080 URL in Chrome, it will show the Job History Status (Completed/Incompleted), Started and Completed Time, and the duration as the following:

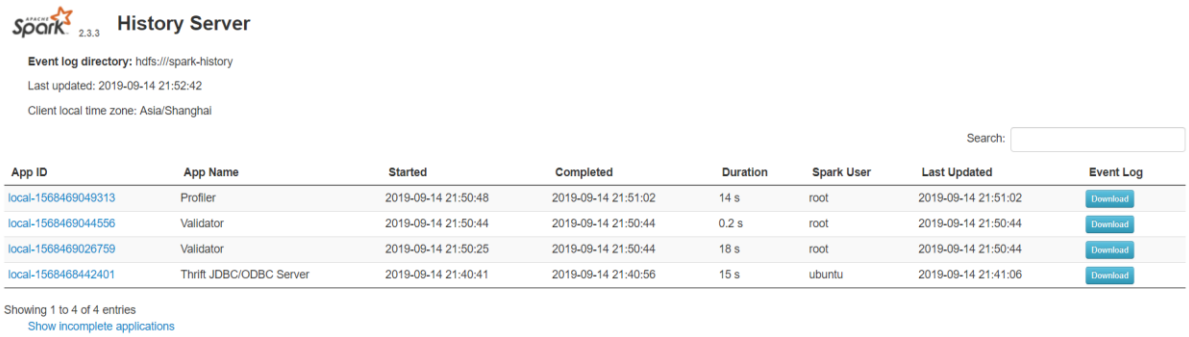


Figure 38. Spark History Server Page

In the history server page, can also search and see the Started time, Completed time and Duration.

4.2.7 Experiment Procedure

This experiment aims at testing the single-node data lake ingestion data size and format. The testing data contains three attributes: ID, Name, and Age. The variables are Data Size, File Format, Number of Uploaded Files, Number of Valid Records, Number of Invalid Records. The dependent variables are Minutes, and the results displaying by Kylo. For example, the 1KB data has 12 records, 10 valid and 2 invalid records.
The following 4 cases aim at testing the ingestion data size, from 1KB to 100M, and the result display as the Figure:

Test Case No	Data Size	File Format	Number of Uploaded Files	Number of Valid Records	Number of Invalid Records	Minutes	Process Result Correct ? (Y/N)	Comment of Result
1	1KB	CSV	1	10	2	8 Min	Y	-
2	10 MB	CSV	1	963340	8917	8Min	Y	-
3	50M	CSV	1	4816412	44670	7Min	Y	-
4	100M	CSV	1	9632824	89340	STUCK	-	-

Figure 39. Test Cases – Data Size

Then the following test cases aim at testing what would happen if upload the multiple files at one time.

Test Case No	Data Size	File Format	Number of Uploaded Files	Number of Valid Records	Number of Invalid Records	Minutes	Process Result Correct ? (Y/N)	Comment of Result
5	1KB	CSV	2	20	4	8 Min	N	0 Valid and 0 Invalid
6	10 MB	CSV	2	1926680	17834	8Min	N	0 Valid and 0 Invalid

Figure 40.Test Cases – Multiple File Upload

And the last step giving a trial of testing the data ingestion in ZIP format, but with different data size.

Test Case No	Data Size	File Format	Number of Uploaded Files	Number of Valid Records	Number of Invalid Records	Minutes	Process Result Correct ? (Y/N)	Comment of Result
7	1 KB	ZIP	1	10	2	8Min	N	0 Valid and 4 Invalid
8	10 MB	ZIP	2	1926680	17834	8Min	N	0 Valid and 107293 Invalid

Figure 41. Test Cases – ZIP Format

4.2.8 Experiment Result

The first case (TC1) requires 8 minutes to finish. In the Kylo Profile page can see the valid and invalid statistics, attributes like 'Age ' as an example showing in below:

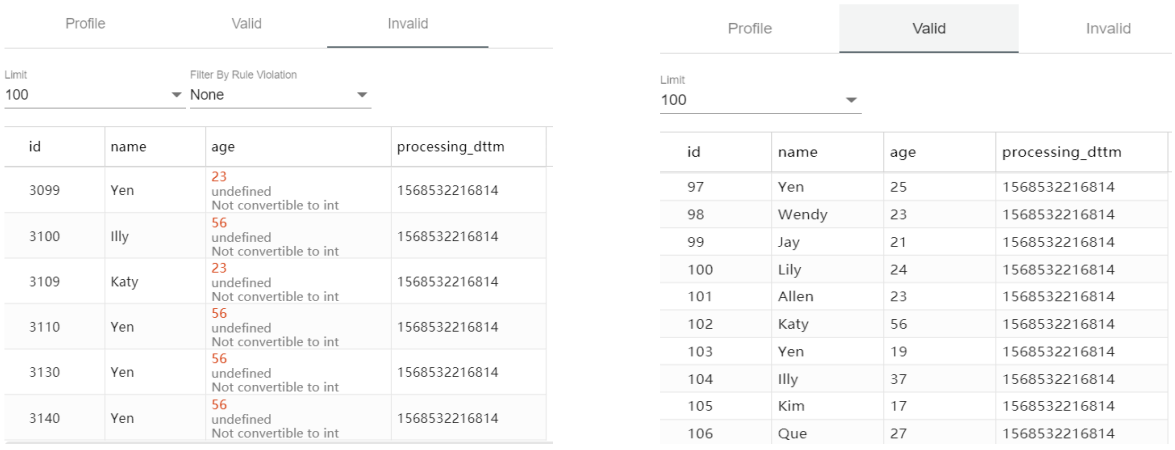


Figure 42. TC1 Valid and Invalid results

If the 'Age' value follows with space, Kylo will classify the value not converted as integer, thus it reports the exception "undefined Not converted to int" as above Figure showing. Kylo will also automatically generate a histogram as the following:

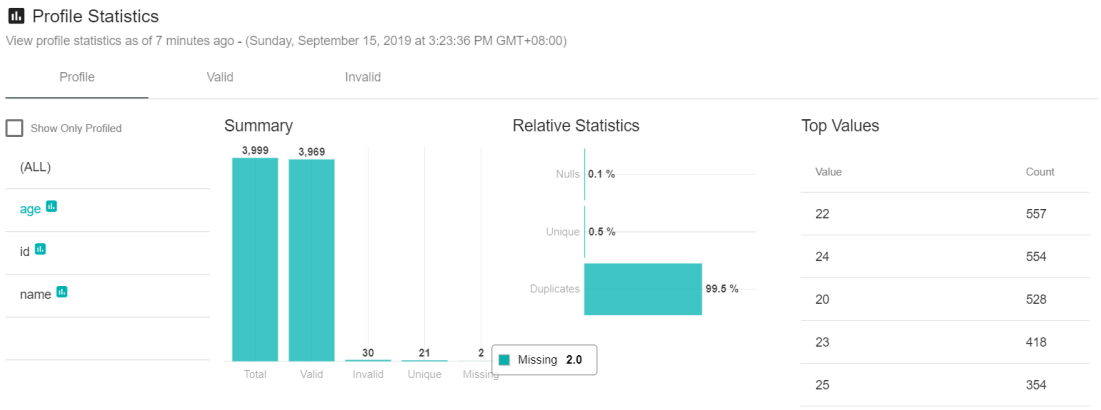


Figure 43. TC1 Result Profile Statistics-'Age'

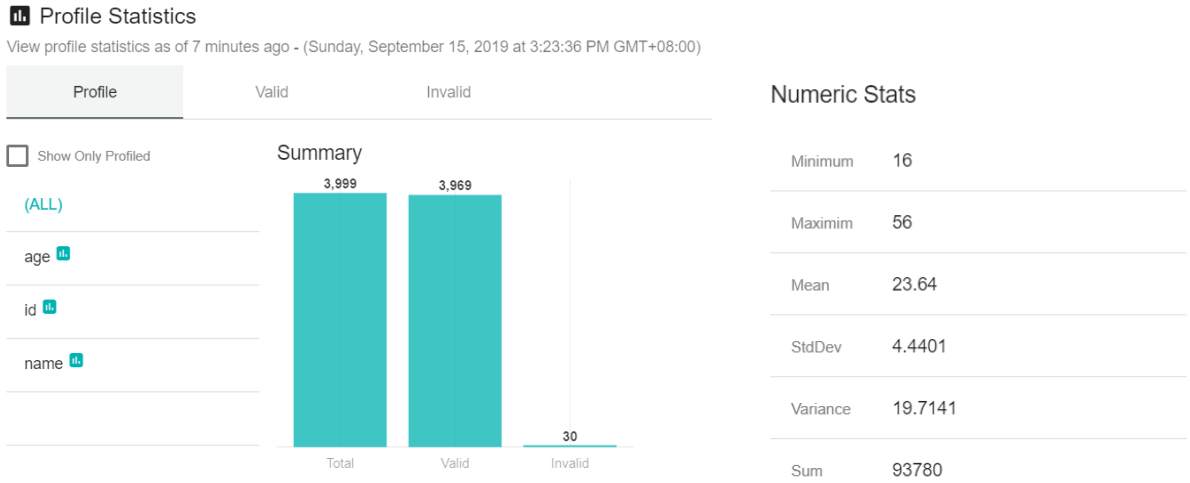


Figure 44. TC1 Result Profile Statistics-'All'

TC2 and TC3 continue increasing the data ingestion size, the result shows no obvious difference of the Minutes, around 7~8 minutes. TC3 settled for testing 50MB data ingestion, it runs well. It can calcluate the valid records are 4816412 and invalid records are 44670 records. The time spent 7 minutes, faster than my expectation. When the TC4 ingested the 100M into Kylo, it got stuck as Fugure 46. Then I decreasing the data size gradually, when reduced to 58M, it can begin to ingest again.

TC5 tried to control the data content and size are the same but change the number of uploaded files, test uploading multiple files in parallel. The result displayed 0 Valid and 0 Invalid rows. TC6 settled to make sure again Kylo is not allowed to upload multiple files in one time. As the result shows, Kylo doesn't support upload multiple files in one time.

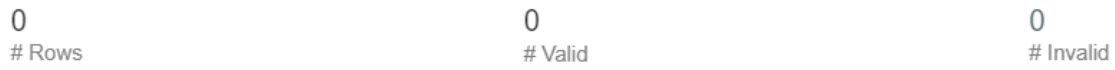


Figure 45. Ingestion with multiple files in one time

TC7 and TC8 designed to test ingest data in ZIP format. TC7 aims at testing when one file under ZIP package, whether the data can be processed successfully. TC8 tried to figure out when containing two files (2 files' content and size are the same). The results show no regarding the number of files under the zip package. The number of the terminated rows is in random. In my person view, the compression procedure has destoryed the original data.

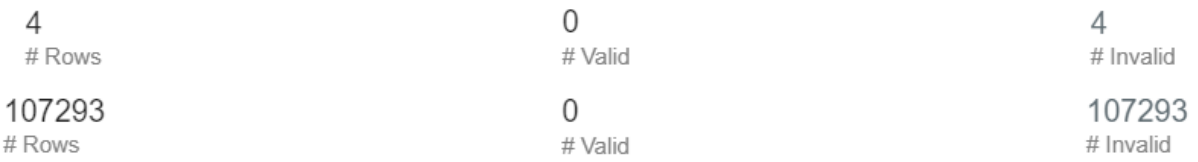


Figure 46.Ingestion in ZIP Format

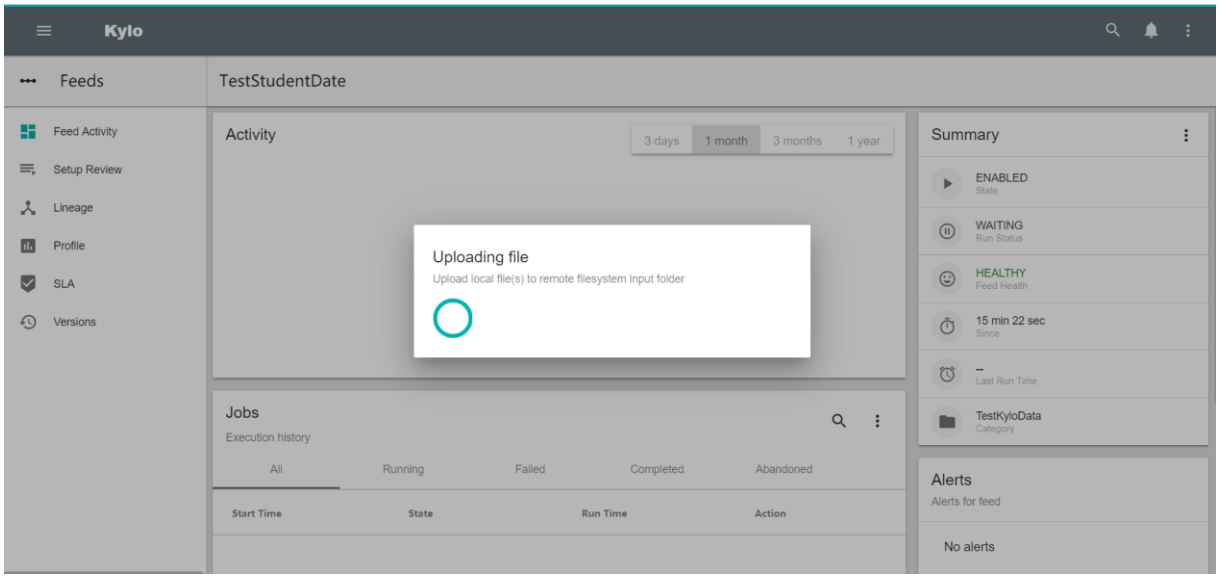


Figure 47. Ingest 100M then stuck

4.2.9 Validity Threat

Internal:
The maximum ingestion of data size is 58M for a single node Kylo data lake. The 58M is much smaller than I expected. Therefore, I summarized the experiment result cannot rule out the following factors:

1. Unexpected issue – the official installation package lost the two Spark execution files of Kylo. The unexpected issue of losing the code, cannot foresee what the functions have been influenced, thus it might be an impact factor;
2. Suppose the 58M is the maximum storage capacity, it might be the storage space of the Single Node mode just so limited ;
3. The official website without the details of installation steps, so it might exist the possibility that I missing any step no configured, even though the 'Configure Inspector' has detected everything has gone well.

External:
The major external threat maybe the operating system. All the testing cases conducted in Ubuntu 16.X. However, the official website declaimed Kylo supports Ubuntu16 and 18.

Chapter 5
Analysis and Discussion

From the Literature Review, we can know data lake is different from traditional database and data warehouse, it can save raw data from multiple sources, wrangle and profile data, provide data analysis and adapt to the machine learning tendency. The data lake is an open and self-service platform, centralized storage of massive volume of data. We build data lake because it can help adjust the organizational structure and reduce the redundancy of "IT" type job. Another reason of building data lake because the machine learning become more and more popular today. Through the trial of the configuration of Kylo data lake, help understand the underlying architectures better. Kylo relies on Hadoop, Hive, Spark, ActiveMQ, NiFi, and Elasticsearch.

The design of the experiment is to test single-node Kylo data lake ingestion capacity. The result shows single node Kylo data lake does not allow to ingest multiple data files in parallel, and not allow to upload files in ZIP format. The maximum ingestion of data size is 58M for a single node Kylo data lake. The 58M is much smaller than I expected. But the result of 58M is instructive. In this experiment, I just installed a simple spark to ingest 58M of data, so in the future improvement of Kylo, the result must greater than the 58M.

In my point of view, Kylo data lake is still an immature data lake platform, the configuration is not easy to get started for data lake novice. And even though the Kylo inspector (8099 port) service monitoring, it doesn't mean Kylo configuration has absolutely done. Therefore, in my point of view, the following aspects should be improved:

1. Kylo relies on many other technologies, NiFi, Hadoop, Hive, Spark and so on. Although the combination is creative, it also means that Kylo is fragile. As long as existing a technical mistake or any execution files are missing during the configuration, the corresponding functions of the data lake cannot be initialized or active.
2. Kylo is not developer-friendly enough. The configuration process is often complex. Personally, suggest the setup-wizard.sh could be improved to realize the real "out-of-the-box".
3. In practice, I found that Kylo's cleanup mechanism for deleting a feed is not automatically configured for the user, which means users should manually import the templates or use NiFi to delete the processors one by one.
4. The most urgent issue should be fixed is the Spark class files are lossing. During the configuration, the installation package cannot find the *kylo-spark-shell-pgrep-marker* and *com.thinkbiganalytics.spark.SparkShellApp* class files.
5. The official documentation should be improved. It supposed should be able to download official installation package from their website links, but actually lose efficacy. The configuration details should be more specific.

Chapter 6

Conclusion and Future Work

Data lakes are getting hotter nowadays, act as an open and self-service data storage platform, it costs low and can do the data processing and analysis, even allows to combine with machine learning. Data lakes emerge as a promising technology, which can bring profit and convenience for the organizations. Especially, deployed data lakes in the Cloud environment can centralized use of data platforms across borders. It helps the team adjust the organizational structure, encourage everyone to understand, analyze data, and reduce the repeat and redundancy of "IT" type work. Compared with the traditional data warehouse and database, the data lake is not curated with the data structure, it tolerates with semi-structured, unstructured data. The data lake is more agility than traditional data warehouse and database, can store large raw data and visualize the result, which can combine with data science and machine learning.

Data lakes emerge promising. However, the open-source data lakes platform provided for academic research is rare to find, thus the underlying architecture is even harder to see. Kylo, is an open-source data lake platform, with rich features and built on Hadoop and Spark, help provide a data lake configuration perspective of this paper. Existing data lakes deployed in Cloud environment commonly, Kylo also allows to deploy in the Cloud. This study chose City Cloud [5] as the public cloud. During the configuration, the underlying technologies of Kylo has been revealed. The Kylo data lake relies on NiFi, ActiveMQ, Elasticsearch, Hadoop, Hive and Spark, etc. The combination of the underlying technologies are really powerful and creative, these new fashion technologies which are commonly used in the big data field. Kylo also allows to integrate alternative underlying technologies, for example, Kafka can instead of ActiveMQ, Solr can instead of Elasticsearch and so on.

However, that's fewer existing papers regards the detail of configuring a data lake, even the official documentation of Kylo missing a lot of details. Therefore, without a clear guideline, it makes difficult for the Kylo data lake novice, the whole configuration procedure is panic and suffering. Hope from this paper, it can provide a practice perspective for readers and avoid the twist of Kylo configuration. As the academic researchers, they can learn the case of Kylo data lake from this paper, then keep on studying the data ingestion performance of Kylo data lake based on this level. As the developers, they can avoid the struggling of the configuration of Kylo data lake, then develop their own projects and integrate with the Kylo. As the business decision-makers, they can consider introducing a data lake to their company and proposing related strategies to help increase business profit.

In this paper, I considered that Kylo data lake platform is not mature enough, at least the open-source production is not stable enough. Introducing too many underlying technologies will make the troubleshooting difficult. If any technical configuration steps are wrong or any other dependent execution files are missing, the corresponding functions of data lake would be disabled to use and the exceptions are hard to position. Therefore, it is suggested that Kylo data lake team could improve and decouple its functional dependencies, simplify the configuration process, make it become real “out-of-the-box” software. The configuration process is not easy to learn, and the official

documentation without sufficient detailed to support, which means the background knowledge has to prepare in advance.

From the set of testing cases, we can know the single node Kylo data lake is not allowed to ingest multiple data files in one time and is also not allowed to upload in ZIP format. The maximum ingestion data size is 58M. The data ingestion capability of a single node Kylo data lake is lower than my anticipated, but the Spark installation and configuration come from the Spark official website in my case, thus the simple Spark allows us to ingest 58M data, the future works can enhance based upon this level. However, the experiment provides a different perspective for future studies.

6.1 Future Work

There is a lot of potential research space for Kylo Data Lake. Apart from the Kylo insufficiencies should be improved, other interesting aspects should intrigue to study in the future.

For example, suppose the scenario that Kylo has completely configured well, can we change the backend storage architectures to test with proper benchmarks. But what is the most proper benchmark? Kylo integrated with several technologies, Hive, Spark, Hadoop. TPCx-BB[21] is a benchmark measuring the performance of Hadoop-based Big Data Systems. Hibench[22] is a benchmark stressing Hadoop systems and goal to test the performance of Hive. And SparkBench[23] is a suite for testing spark performance. However, there is no benchmark for a data lake.

We can also compare how the block and object storage influence the performance of injection and analytics operation in data lakes. And how volumes and ephemeral disks mount of block storage will impact the performance of injection and analytics operations in data lakes. When configured in the distributed mode, whether the performance will be influenced or not?

In addition, the data lake is an open platform, how to manage data permissions and security? How to help users discover data and understand data? If multiple teams use data, how do you share data outcomes (such as portraits, features, metrics) to avoid duplication of development?

REFERENCES

[1] Laskowski, Nicole. "Data lake governance: A big data do or die." URL: http://searchcio. techtarget. com/feature/Data-lake-governance-A-big-data-do-or-die (access date 28/05/2016)(2016).

[2] Miloslavskaya, Natalia, and Alexander Tolstoy. "Big data, fast data and data lake concepts." Procedia Computer Science 88 (2016): 300-305.

[3] Michael Lock. "Angling for Insight in Today's Data Lake" URL: https://s3-ap-southeast-1.amazonaws.com/mktg-apac/Big+Data+Refresh+Q4+Campaign/Aberdeen+Research+-+Angling+for+Insights+in+Today's+Data+Lake.pdf (Aberdeen Group)(2017)

[4] Kylo, https://kylo.readthedocs.io/en/v0.10.0/about/KyloFeatures.html (Teradata Corporation)

[5] City Cloud, https://www.citycloud.com/(City Network Hosting AB)

[6] Apache NiFi, https://nifi.apache.org/docs.html (Apache Software Foundation)

[7] Elasticsearch, https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html (Elasticsearch B.V.)

[8] Wikipedia, https://en.wikipedia.org/wiki/Apache_ActiveMQ (Wikipedia)

[9] ActiveMQ,http://activemq.apache.org (Apache Software Foundation)

[10] Sohu, http://www.sohu.com/a/163001053_163476, 2017-08-07(SOHU)

[11] Spark, http://spark.apache.org/docs/latest/index.html (Apache Software Foundation)

[12] Scala, https://www.scala-lang.org (École Polytechnique Fédérale)

[13] Ambari, http://ambari.apache.org/ (Apache Software Foundation)

[14] Mathis, C. (2017). Data lakes. Datenbank-Spektrum, 17(3), 289-293.

[15] Miloslavskaya, Natalia, and Alexander Tolstoy. "Big data, fast data and data lake concepts." Procedia Computer Science 88 (2016): 300-305.

[16] Llave, Marilex Rea. "Data lakes in business intelligence: reporting from the trenches." Procedia computer science 138 (2018): 516-524.

[17] Farid, Mina, et al. "CLAMS: bringing quality to Data Lakes." Proceedings of the 2016 International Conference on Management of Data. ACM, 2016.

[18] Hai, Rihan, Sandra Geisler, and Christoph Quix. "Constance: An intelligent data lake system." Proceedings of the 2016 International Conference on Management of Data. ACM, 2016.

[19] Miloslavskaya, N., & Tolstoy, A. (2016). Big data, fast data and data lake concepts. Procedia Computer Science, 88, 300-305.

[20] AWS,https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/?nc1=hls (Amazon)

[21] Cao, Paul, et al. "From bigbench to TPCx-BB: Standardization of a big data benchmark." Technology Conference on Performance Evaluation and Benchmarking. Springer, Cham, 2016.

[22] Huang, Shengsheng, et al. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010). IEEE, 2010.

[23] Agrawal, Dakshi, et al. "SparkBench–a spark performance testing suite." Technology Conference on Performance Evaluation and Benchmarking. Springer, Cham, 2015.

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden