A Data Catalog using Semantic Logging

1. Introduction 1

2. Features of a Data Catalog 2

2.1 Discovery and Automated Tagging 2

2.2 Crowdsourcing 2

2.3 Ratings and Reviews 2

2.4 Integration Interfaces 3

2.5 Etosha Data Catalog is More than just Search 3

3. Recap: Hadoop for Big Data Processing 3

4. A Data Driven Approach for Knowledge Management 4

5. Semantic Logging Framework for Knowledge Workers 5

5.1 A simplified data science process model 6

5.2 First Steps: Contextualizers are Building Blocks for Semantic Logging 7

DatasetContextualizer 7

ModelContextualizer 7

ModelpipelineContextualizer 7

FlowContextualizer 7

5.3 Share and Merge 9

5.4 Model Conversion and Multiple Query strategies 10

6. Conclusion and Outlook 10

References 11

# 1. Introduction

In a data driven world we have to navigate along new paths. Big Data is the magic which opens up new dimensions. This can be a challenge besides a huge advantage. Not much was done in the past to map the “data world”. Search technology was driving the usage of the internet but search alone gives us no strategic orientation in the data space. Equally, having paper and ink does not mean that you have a map for your journey already in your hands.

Like a traditional map needs paper and ink a mobile device needs appropriate apps in our modern world – but the content is what matters most, followed by usability aspects. This is why data catalogs should not be evaluated by completeness and accuracy only. Some means of fuzzy search and even fuzzy information is sometimes better than having no information at all.

Let’s look deeper into usability. A data catalog should be open in terms of technology and access to it’s content. Since the catalog contains data about the data - not the data itself - we shouldn’t be too picky. But as always – being careful is better than too much of risk, especially in a very open world. With this in mind, we should say: “The data inside the data catalog should be as open as possible for authorized people and systems.” We have to build up the data catalog without technical limitations which could stop us from exporting and merging multiple catalogs.

Time evolution of the content matters as well! Data sets are usually not static as books are – especially in near real time scenarios (driven by an IoT world) we have to consider the life cycle stages.

The most important design goal is to be independent from a particular programming language and storage technology. Obviously, individual representations of the data model come with particular benefits with regards to accessibility and functionality. The universality of the text based RDF format and the text based wiki/markup languages support our goals. Incorporating ontologies and data transformation tools allows us to integrate into existing data governance infrastructures and to stay independent of any APIs, which change quite fast nowadays.

Since we develop a data driven project it is not a particular software stack which is in our focus, but rather the flexibility to implement the metadata management approach on top of Big Data processing environments, no matter if on premise, cloud based or in hybrid environments.

# 2. Features of a Data Catalog

The Etosha project builds the open source components for open data catalogs. This means that data inspectors and profilers for integration in your own application or simply for interactive use in the Spark-shell build the base of the project. Full-text search, a graph database, and an RDF store build the backbone for multiple access strategies. RDF was chosen as data export and exchange format. Finally, the success of Git technology motivates us to offer a rich set of publication and integration capabilities.

The following section addresses some issues, raised by Todd Goldman in his blog [1]:

## 2.1 Discovery and Automated Tagging

Due to the large amount of information and data which surrounds us we need an automatic approach beside crowd-sourcing.

*Etosha offers data inspection procedures and routines for automatic execution during data ingestion or directly afterwards. Footprints of individual columns of structured data sets are extracted for classification and comparison with future datasets.*

## 2.2 Crowdsourcing

Especially for collaborative projects it is important to share knowledge and to learn from others. This means, integration of knowledge contributed by others is an essential functionality.   
*Etosha allows export and import of metadata and data models using the RDF data format.*

## 2.3 Ratings and Reviews

Automatic data extraction can lead to non accurate or even completely wrong results. Users, especially humans have to evaluate and correct such results. Reviews, ratings provided by humans combined with automatic suggestions and cross-validation procedures provide the quality assurance we need in an open data catalog.

*Etosha uses an open feedback system – integrated into Github. In the future, we want to establish the* ***Global Data Map****: GDM is a specific view on a data catalog (or on multiple aggregated data catalogs) which provides context to data, such as dataset usage, dataset ratings, and user feedback.*

## 2.4 Integration Interfaces

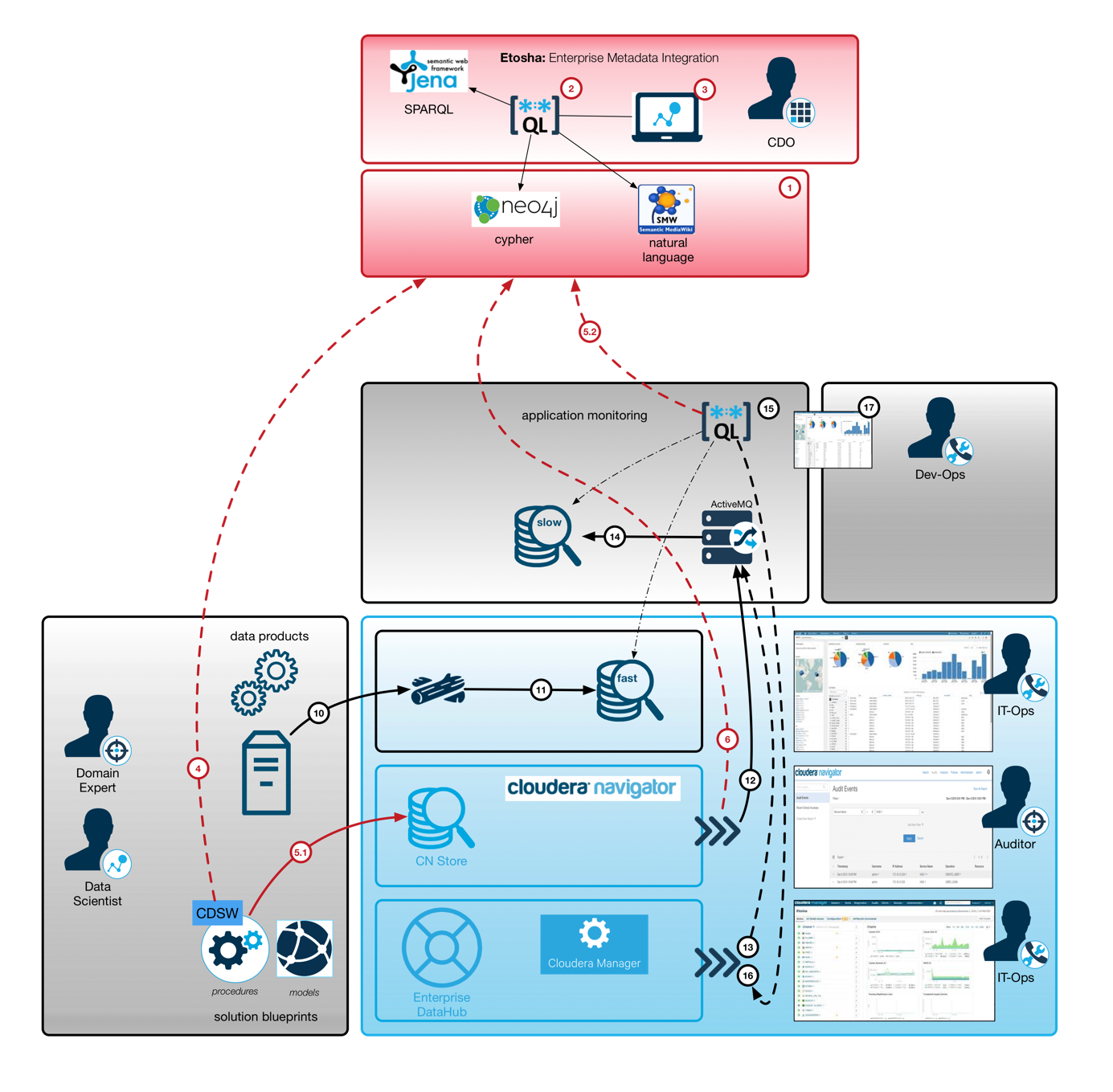
Remember: content is what matters. The catalog provides information and knowledge for a variety of tools in multiple categories. From business glossaries to automatic ETL pipeline generation up to data governance and risk analysis – a lot of potential is already in your data catalog.   
*Etosha tools are focused on providing the best open source data catalog for the data driven business. It is on you to request the information. If it is not there, we have to find a way to* *provide the facts.*

## 2.5 Etosha Data Catalog is More than just Search

*Since search-like interaction is a common feature, we decided to build Etosha around a multi-layer search interface. The search dashboard is the starting point for your work with the Etosha data catalog. We simply replaced the good old menu bar by some stored searches for quick reports, enriched by multiple search facets and visuals. Furthermore, semantic search using SPARQL and the and graph query language Cypher add more flexibility to your search experience. All of this has one goal: make your data catalog accessible.*

# 3. Recap: Hadoop for Big Data Processing

Big Data applications and Data analysis procedures are done in highly integrated environments rather than in individual systems. Nevertheless, we find a coexistence of multiple Hadoop cluster in many companies. The reasons for running multiple clusters are manifold – but out of our current scope. Obviously, there is a demand for another integration layer beyond data flows and process chains. To start with, one can implemented a four-layer metadata management stack as shown in the following figure:



**Figure 1 :** Enterprise data management platform based on CDH (light blue) with customized data products (gray), data science workbench (dark blue), and metadata management tools (red).

Figure 1 shows the Big Data and AI / ML infrastructure which consists of three different types. Cloudera Enterprise offers the Hadoop ecosystem. Specific user frontends provide already a lot of details to administrators, operators and even auditors. Besides those out of the box capabilities, the gray boxes show individual extensions, such as custom applications, data science methods implemented in data products and even the brand new data science workbench. Since those scientific tools are not yet integrated completely into the core monitoring and audit stack we have to provide some additional handlers to grab and organize relevant metadata.

# 4. A Data Driven Approach for Knowledge Management

The cluster related monitoring and auditing is not intended to be a shareable asset – but knowledge about datasets and procedures in general could be such an asset. The Etosha tools solve this problem by forming an **open (meta)data capturing layer**. It can be used for automatic fact extraction, but also in a free style mode by creative data workers. The context of the information in our focus is the organizational and the methodological level rather than physical data representation and security aspects such as encryption and access control. Different aspects needs to be provided to different audiences.

This explains our **multi-layer approach**. The Etosha tools are complementary to existing views. Data can be exported and imported and finally, metadata models from multiple clusters can be integrated into one view. We decided to implement the core data model as an RDF graph to maximize the flexibility and extendibility with at least external dependencies as possible.

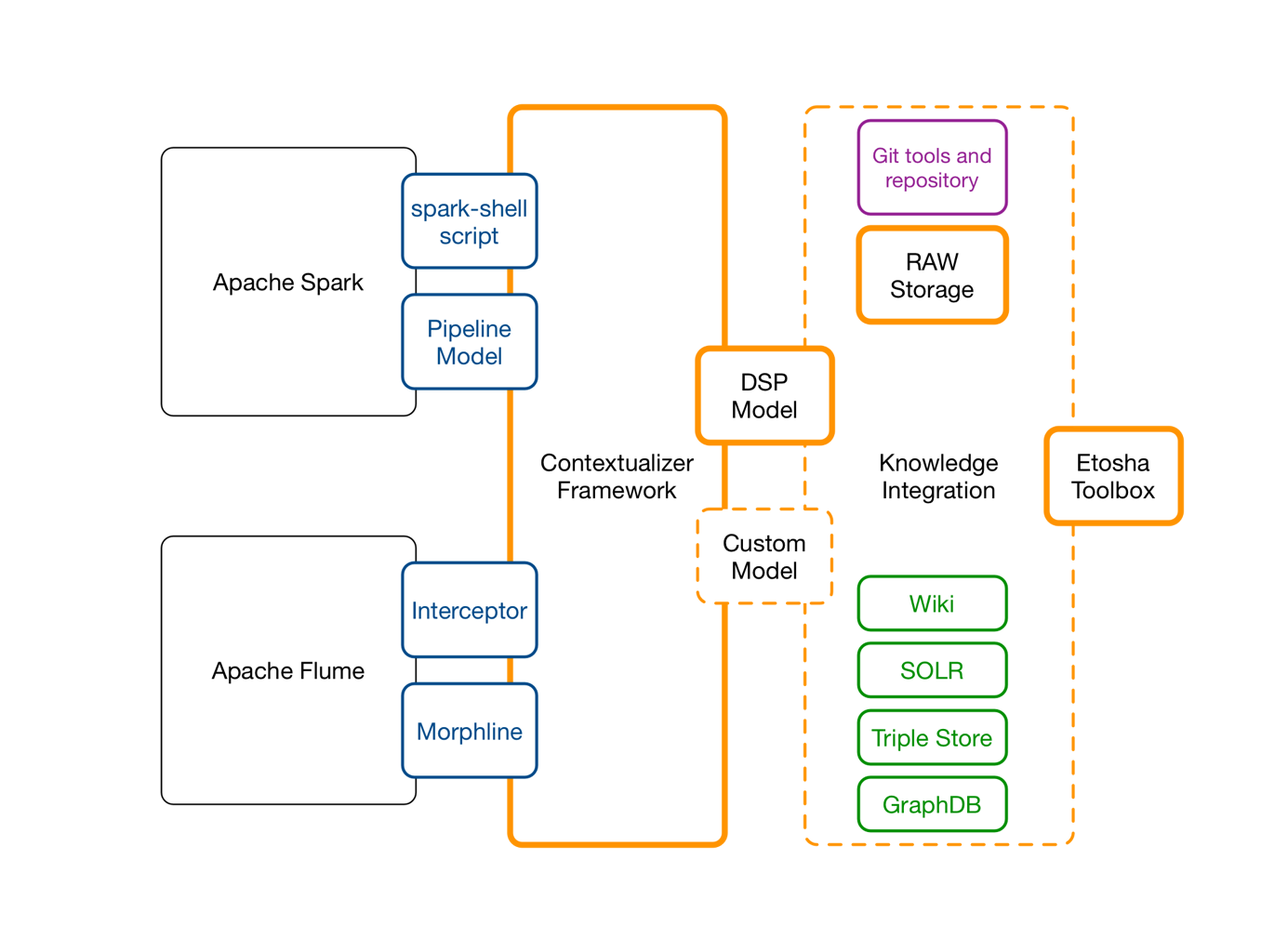
**Semantic logging** allows us to expose structured and unstructured information with a high degree of reusability. Log analysis is common sense today, but a lot of effort goes into log parsing. Standardized log messages are also an important step forward, towards knowledge integration. But the huge amount of technical logs generated during data ingestion or by automatic procedures needs a different handling than models, generated by data scientists. Creative approaches with new algorithms can not be covered by standard log formats but using RDF representations and domain specific ontologies offer such flexibility.

# 5. Semantic Logging Framework for Knowledge Workers

The following figure shows the main components of the Etosha framework (orange). Contextualizers can be embedded into multiple places, e.g., for ad-hoc capturing during data science procedures or for automatic extraction of facts in data ingestion pipelines (black). The components in which semantic logging is used are blue. The core framework offers some software components (orange line) and a methodology (dashed orange line). The knowledge graph is stored in text files using the RDF format. This allows us to handle facts like source code and git is used (see purple box) for sharing and integration of the data. For reliability ans scalability we store raw facts in HDFS – multiple optimization strategies exist and can be applied in a transparent way. Finally, the end user functionality is composed by orchestration of multiple “state of the art” solutions. We use a Mediawiki for user interaction and feedback. SOLR cloud offers facetted search and full-text search capabilities. Apache Jena is used as triple store and SPARQL query engine. The graph database Neo4J offers the cypher query language. SPARQL and cypher are competing and complementary at the same time. Both can be used for analysis and exploration. While SPARQL is a natural fit for semantic projection and integration of multiple ontologies we use cypher on the property graph for topological analysis, data mining and sophisticated analysis on the knowledge graph.

Semantic logging as a data capturing method leads to a continuously growing data set: the knowledge graph. The tooling around it, in this case the Etosha framework is related to it initially but not an inherent part as in traditional application centric solutions.

Traditional IT systems use a very specific, often internal and not completely documented data format. Export is often not easy and sometimes not wanted. An open data format on the other hand requires us individual tool adoption, but it doesn’t dictate the technology which needs to be used to work with the data. This degree of freedom is an important design goal.



**Figure 2 :** Loosely coupled software tools (orange boxes) together with established standard software (black: Big Data environment, purple: data integration and modularization, green: query engines) allow the realization of a continuous data capturing procedure to collect process knowledge.

## 5.1 A simplified data science process model

The following screenshot shows a simplified process model. The green nodes represent users who are also the responsible persons for particular procedures (red nodes) and thus need access to certain datasets (blue nodes). Only such a knowledge graph can show and illustrate the interdependencies between the procedures, datasets and people which all collaborate in one particular context – learning a classification model in this case.



<https://www.youtube.com/watch?v=ZE7Gcanv90s&feature=youtu.be>

**--- Formalizing the approach ---**

Problem:  
 Classification of unknown documents. (Background)

Procedure to solve the problem:

Data ingestion (Flow)  
Docment collection (Dataset)  
Model training (Modelpipeline)

Solution:

LDA or KMeans (Model)

Model evaluation (Flow)

Technical Outcome:

Real time classification pipeline: (Flow)

Business impact:

Inspection and analysis of metadata (Query on Metadata graph)

## 5.2 First Steps: Contextualizers are Building Blocks for Semantic Logging

Contextualizers extract relevant information from datasets (at a given point in time) and data processing components (during a period of time). Furthermore, developers and data scientists use contextualizers to capture information exactly at this point in time, when most facts and context are obvious. This reduces the complexity for data capturing and increases precision dramatically, especially for ad-hoc data capturing during creative procedures. A contextualizer can be compared with a logger component which is widely used in software development. Like in the case of loggers, multiple output paths and output handling strategies exist. We develop contextualizers to capture special facets of an algorithm, a dataset, or a full processing pipeline.

Initial set of Etosha contectualizers:

### DatasetContextualizer

Non technical metadata,

Usage statistics,

Content statistics: column & partition profiles

### ModelContextualizer

Description of model properties and evaluation results

### ModelpipelineContextualizer

Description of arbitrary ML procedures (model creation)

### FlowContextualizer

Data ingestion and streaming processing (model application)

<< show examples of what is captured by a contextualizer and how a contectualizer used >>

|  |  |
| --- | --- |
| Code | RDF output |

## 5.3 Share and Merge

Extracted metadata can be pushed into multiple systems in different formats. To begin, we simply store the facts in a git project. Git offers versioning, merging, and issue tracking functionality together with a decentralized infrastructure. Collaboration across multiple teams and multiple technology stacks becomes easy this way.

<< show model combination >>   
using multiple loads into one triple store or one Neo4J instance ….

## 5.4 Model Conversion and Multiple Query strategies

Individual storage components support specific access patterns. In our case we need a very flexible integration layer and scalable long term storage, combined with analytical power. Thus, we use HBase for raw data and images, and SOLR for keeping multiple indexes. Neighborhood exploration is based on the RDF graph and the property graph (Neo4J). Finally, a human readable representation is provided by a Wiki, such as Confluence or Semantic Media Wiki (SMW). Depending on the project we are able to select the right components and also to change them.

<< show model trafo >>

Indexing to SOLR

Collection to CSV to Neo4J  
XSLT and Velocity

# 6. Conclusion and Outlook

# References

[1] <http://blog.waterlinedata.com/blog/search-is-not-a-data-catalog?utm_content=52869351&utm_medium=social&utm_source=linkedin)>