DELIVERABLE

Project Acronym: E-ARK

Grant Agreement Number: 620998

Project Title: European Archival Records and Knowledge Preservation

---

DELIVERABLE DETAILS

DELIVERABLE REFERENCE NO.	D6.3
DELIVERABLE TITLE	Data Mining Showcase
REVISION	0.1

AUTHOR(S)	
Name(s)	Organisation(s)
Rainer Schmidt, Sven Schlarb, Jan Rörden	AIT Austrian Institute of Technology GmbH
REVIEWER(S)	
Name(s)	Organisation(s)
Janet Delve Zoltán Lux	University of Brighton National Archives of Hungary

Project co-funded by the European Commission within the ICT Policy Support Programme Dissemination Level		
P	Public	P
C	Confidential, only for members of the Consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Submitted Revisions History

Revision No.	Date	Authors(s)	Organisation	Description
0.1	18-10-16	R Schmidt	AIT	Initial document and structure
0.2	07-11-16	S Schlarb	AIT	Standalone software stack, testbeds, demonstrators
0.3	15-12-16	J Rörden	AIT	Natural Language Processing
0.4	15-12-16	R Schmidt	AIT	Data-centric backend
0.5	20-12-16	R Schmidt	AIT	Abstract, Introduction
0.6	13-12-16	R Schmidt	AIT	Text revisions, proof-reading
0.7	23-01-17	R Schmidt	AIT	Review comments incorporated
0.8	24-01-17	J Delve	UoB	English grammar proof-read
1.0	27-01-17	R Schmidt	AIT	Version 1.0

EXECUTIVE SUMMARY

E-ARK has developed the Integrated Platform Reference Implementation (IPRIP) which is a software prototype for package creation, search, and access using data-centric technologies. The underlying concepts and technologies of the system have been detailed in past deliverables (D6.1 and D6.2). The main components of the IPRIP e-archiving environment include a scalable packaging infrastructure for creating OAIS Information Packages and a data-centric repository for searching and accessing data items on demand.

This deliverable focuses on recent developments that support the deployment of the IPRIP in different configurations at E-ARK stakeholder sites. A flexible packaging mechanism combined with a standalone backend implementation enables custom single-server deployments on demand. The scalable, Hadoop-based backend implementation has been ported to the latest CDH (Cloudera Distribution Hadoop) distribution in order to support a recent technology stack, advanced data mining concepts, and enterprise demands. Showcases that dealt with the application of text mining approaches, the extraction and visualization of geographical information, the implementation of a database archiving workflow, and mass document ingest have been implemented and deployed at stakeholder sites in Hungary and Slovenia.

Table of Contents

1. Introduction

E-ARK has developed the Integrated Platform Reference Implementation (IPRIP) which is a software prototype for package creation, search, and access using data-centric technologies. The underlying concepts and technologies of the system have been detailed in past deliverables (D6.1 and D6.2). The main components of the IPRIP include a scalable packaging infrastructure for creating OAIS Information Packages and a data-centric repository for searching and accessing data items on demand.

The IPRIP is an archiving environment that consists of two main components: a Python-based graphical front-end application and OAIS package creation infrastructure, and a backend repository for data storage, search, and access. The frontend and backend systems are loosely coupled through HTTP REST interfaces. This approach has been taken as it provides one with the flexibility to swap different implementations and/or deployments of the frontend and backend components. A detailed description of the IPRIP architecture can be found in the E-ARK deliverable D6.2.

This deliverable (D6.3) focuses on recent developments that support the deployment of the IPRIP in different configurations at E-ARK stakeholder sites. These configurations are specifically targeting the processing of E-ARK Archival Information Packages (AIPs), as detailed in D4.4 Part A. A flexible packaging mechanism combined with a standalone backend implementation enables custom single-server deployments on demand. Work on the single-server deployment has been carried out jointly with WP4 in the context of the E-ARK Web project. For a detailed description of the E-ARK Web SIP to AIP conversion component, the reader is referred to deliverable D4.4 Part B.

The scalable, Hadoop-based backend implementation has been ported to the latest CDH distribution in order to support a recent technology stack, advanced data mining concepts, and enterprise demands. Showcases that dealt with the application of text mining approaches, the extraction and visualization of geographical information, the implementation of a database archiving workflow, and mass document ingest have been implemented and deployed at stakeholder sites in Hungary and Slovenia.

In section 2, we give an overview of the standalone version of the IPRIP that provides a software stack for small scale deployments. This software package, called *E-ARK Web lite*, utilizes a pairtree[1] based storage backend and a container-based deployment model that is based on the Docker[2] platform. In section 3, we discuss work on porting the initial system to a contemporary Hadoop distribution, and the Cloudera Search platform and its implications. Section 4 provides an overview of implemented use-case demonstrations and data-mining showcases including the application of Natural Language Procession techniques on archived data, the extraction and visualization of geographic information (geo-coding) on archived data, and the implementation of automated processes for archiving operational databases. Section 5 provides an overview of testbeds and pilot deployments that haven been evaluated by E-ARK stakeholders, including a public deployment as well as two on-site deployments.

[1] https://wiki.ucop.edu/display/Curation/PairTree
[2] https://www.docker.com/

2. The Standalone Software Stack

The goal of the standalone software stack is to provide an installation package of the Integrated Platform Reference Implementation for small scale environments using an integrated storage backend and a container-based deployment model. The storage is file based using the pairtree algorithm to store Information Packages on the platform. Deployments can be carried out on single machines. The installation is based on Docker containers to ensure a simple installation procedure and support for updating the various components of the system separately. This section describes the storage and deployment mechanisms implemented for standalone deployments of the IPRIP.

2.1. Pair-Tree based storage backend

The standalone version of the IPRIP makes use of a file-based storage backend that uses the pairtree algorithm for file placement. This storage backend provides an alternative to deploying the scalable but also more complex Hadoop-based backend infrastructure which is used for cluster deployments of the IPRIP.

The storage backend used in the standalone software stack makes use of the Python-based *Pairtree File System* implementation[3] of the *Pairtrees for Collection Storage* specification[4]. It is used to store the physical representation of the AIP in a conventional file system.

A p*airtree* is a filesystem hierarchy for storing digital objects where the unique identifier string of the object is mapped to a unique directory path so that the file system location of the object can be derived from the identifier string. Basically, the identifier string is split each two characters at a time and the object folder has by definition more than two characters. Furthermore, the specification defines a mapping of special characters to a set of alternative characters in order to ensure file system level interoperability.

The following example explains how the *Pairtree* storage method is used in the E-ARK Web implementation.

According to the default E-ARK Web configuration, the path to the storage directory in the file system is:

```
/var/data/earkweb/storage
```

The storage folder contains an empty file called "`pairtree_version0_1`" which specifies the version of the *Pairtree* specification.

The storage directory contains the root folder of the *Pairtree* storage:

```
/var/data/earkweb/storage/pairtree_root
```

Let us assume that an AIP has the following identifier

```
urn:uuid:6c496473-4e77-44f5-b387-25bffd362789
```

[3] https://pypi.python.org/pypi/Pairtree
[4] https://wiki.ucop.edu/display/Curation/PairTree?preview=/14254128/16973838/PairtreeSpec.pdf

Following the method defined by the *Pairtree* specification, this identifier is mapped to the following path:

```
ur/n+/uu/id/+6/c4/96/47/3-/4e/77/-4/4f/5-/b3/87/-2/5b/ff/d3/62/78/9
```

and the actual AIP container file is stored in a "data" folder which represents the leaf node of the p*airtree* file system hierarchy.

The leaf node contains one or possibly more sub-directories (5-digits fixed length zero-filled number) for the versions of the AIP.

The full path to the AIP file is as follows (to be read as a single line string):

```
/var/data/earkweb/storage/pairtree_root/ur/n+/uu/id/+6/c4/96/47/3
-/4e/77/-4/4f/5-/b3/87/-2/5b/ff/d3/62/78/9/data/00001/urn+uuid
+6c496473-4e77-44f5-b387-25bffd362789.tar
```

The purpose of the p*airtree* storage backend is to allow a large number of AIPs[5] to be stored in a conventional file system, such as Network Attached Storage (NAS), for example, and allow fast access to individual files by directly reading content streams from the files stored in the TAR-packaged AIP container files.

2.2. Container-based Deployment Model

The IPRIP provides an e-archiving environment that can be deployed at different scales ranging from a single server to a computer cluster hosted within a data centre. Cluster deployments are based on software components that make use of Apache Hadoop[6] and other data-centric software stacks. While the Hadoop-based deployment provides built-in scalability, robustness, and support for data-centric processing, it also comes with increased configuration and maintenance requirements compared to a standalone deployment.

The storage backend described in section 2.1 does not depend on the cluster software stack. It has been designed to support traditional (non-distributed) deployments, which provides an adequate solution for smaller institutions or demonstration purposes, as compared to big data-oriented software environments. The standalone version of the IPRIP e-archiving environment has been made available using a container-based deployment model using Docker[7] in order to support simple and modular installation of the software.

Docker is an open-source engine that automates the deployment of any application as a lightweight and portable container that will run on any platform where the Docker engine is supported.[8] In order to allow deploying IPRIP services on a Docker platform, Docker

[5] No concrete numbers will be given here as to what a "large number of AIPs" means in this context, as the requirements regarding scalability of indexing and access procedures depend on various factors, such as the environment (hardware and network) and the type of collection data to be stored in the repository. It is a matter of evaluating the Standalone Software Stack (cf. section 5) using scalability tests to find out if it can cope with the given scalability requirements, or if the Cluster Software Stack (cf. section 5) is needed.
[6] http://hadoop.apache.org/
[7] https://www.docker.com
[8] https://docs.docker.com/engine/installation

containers for the individual services of E-ARK Web's frontend and backend have been created.

Table 1 shows the software module and the corresponding images used in an IPRIP Docker deployment. The left column describes the software module and the right column indicates the image which is used to create and run a container to provide the corresponding service.

Software module	**Docker image**
MysSQL[9]	earkdbimg[10]
SolR[11]	Solr[12]
RabbitMQ[13]	tutum/rabbitmq[14]
Redis[15]	tutum/redis[16]
E-ARK Web[17]	earkwebimg[18]
Celery[19]	earkwebimg
Celery Flower[20]	earkwebimg

Table 1 The standalone deployment stack packaged as Docker containers

Docker Compose[21] is used to run the components listed in Table 1 as multi-container Docker applications. Docker compose allows one to automatically retrieve or build required images and containers as well as to control start up and shut down of multiple inter-dependent services. Docker Compose also allows one to interlink services making communication between services easier. Figure 1 shows the YAML[22] file for E-ARK Web which defines the service composition allowing different containers to interact in an isolated environment. The dotted lines are used to visually divide the YAML file according to the different services it defines.

[9] http://www.mysql.com
[10] Based on earkweb image created from http://github.com/eark-project/earkweb
[11] https://lucene.apache.org/solr
[12] https://hub.docker.com/_/solr
[13] http://www.rabbitmq.com
[14] https://hub.docker.com/r/tutum/rabbitmq
[15] http://redis.io
[16] https://github.com/tutumcloud/redis
[17] http://github.com/eark-project/earkweb
[18] https://github.com/eark-project/earkweb/blob/master/Dockerfile
[19] http://www.celeryproject.org
[20] https://github.com/mher/flower
[21] https://github.com/docker/compose
[22] http://yaml.org/

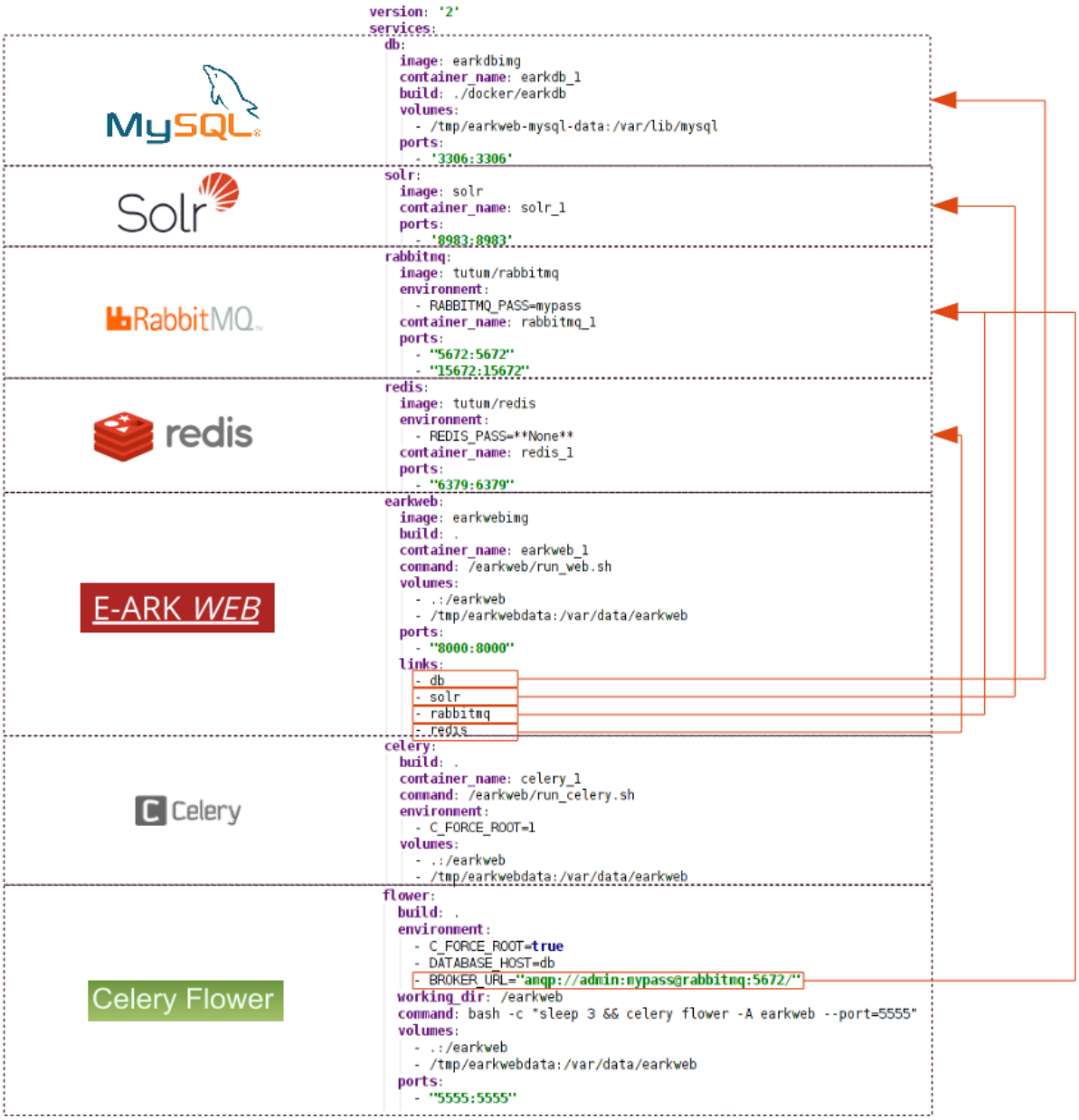

Figure 1 Docker containers included by the IPRIP standalone software stack and the main links between the containers.

For example, the first service, named ‘db’, provides the MySQL database service needed by the E-ARK Web frontend web application to store basic information about the processing status of information packages. The service is defined by the following properties:

- **image**: earkdbimg – The name of the image which is used to provide the MySQL database.
- **container_name**: earkdb_1 – The name of the Docker container.
- **build**: ./docker/earkdb – The Docker file which contains the build instructions to create the Docker image.
- **volumes**: - /tmp/earkweb-mysql-data:/var/lib/mysql – A directory from the Host system (here: /tmp/earkweb-mysql-data) is mounted as the MySQL data directory into the Docker container (here: /var/lib/mysql).
- **ports**: - ‘3306:3306’ – The Port of the application is mapped to the port where the service will be exposed (in this case the standard MySQL port 3306 will be exposed as port 3306 by the container).

The orange arrows in figure 1 show how the services are linked with each other using the “links” attribute of the E-ARK Web container. The attribute references the required service by name. The BROKER_URL environment variable of the flower service is used to link it with to the ‘rabbitmq’ service.

Figure 2 shows the origin of the images which are needed to run the various Docker containers. It shows that the Solr, RabbitMQ, and Redis images are directly retrieved from the Docker Image Library[23], and that the remaining containers are based on Docker container build instructions provided in form of Dockerfiles[24] as part of the E-ARK Web code base.[25]

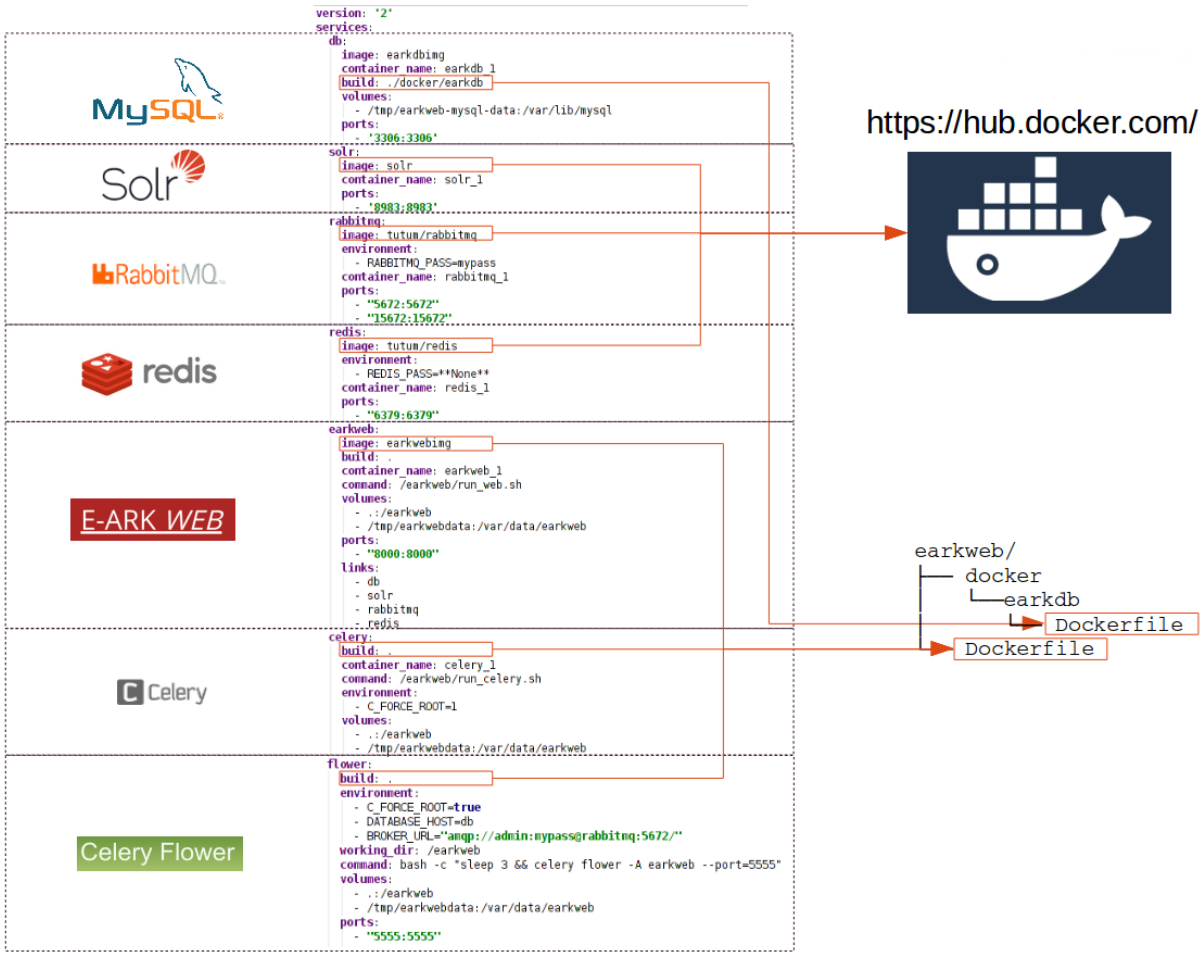

Figure 2 Origin of the images which are the basis to run the Docker containers

[23] https://hub.docker.com
[24] https://docs.docker.com/engine/reference/builder
[25] http://github.com/eark-project/earkweb

Figure 3 shows how various directories of the host file system are mapped to directories of the corresponding Docker containers. On the top right, there are directories of the local file system, and on the bottom right, there are directories of the Docker containers. The colours of the bounding boxes show the relationship between host file system and container directories. For example, the '`/tmp/earkweb-mysql-data`' directory of the host file system is mapped to the '`/var/lib/mysql`' directory of the MySQL database container. It is worth highlighting that for the containers *E-ARK Web*, *Celery*, and *Celery Flower,* the current directory ('.') – which corresponds to the E-ARK Web code base – is mounted to the `/earkweb` container directory. These containers provide the different types of services (the E-ARK Web frontend, the Celery task queue backend, and the Celery Flower task execution monitoring service), based on the same base image using service specific container run instructions respectively.

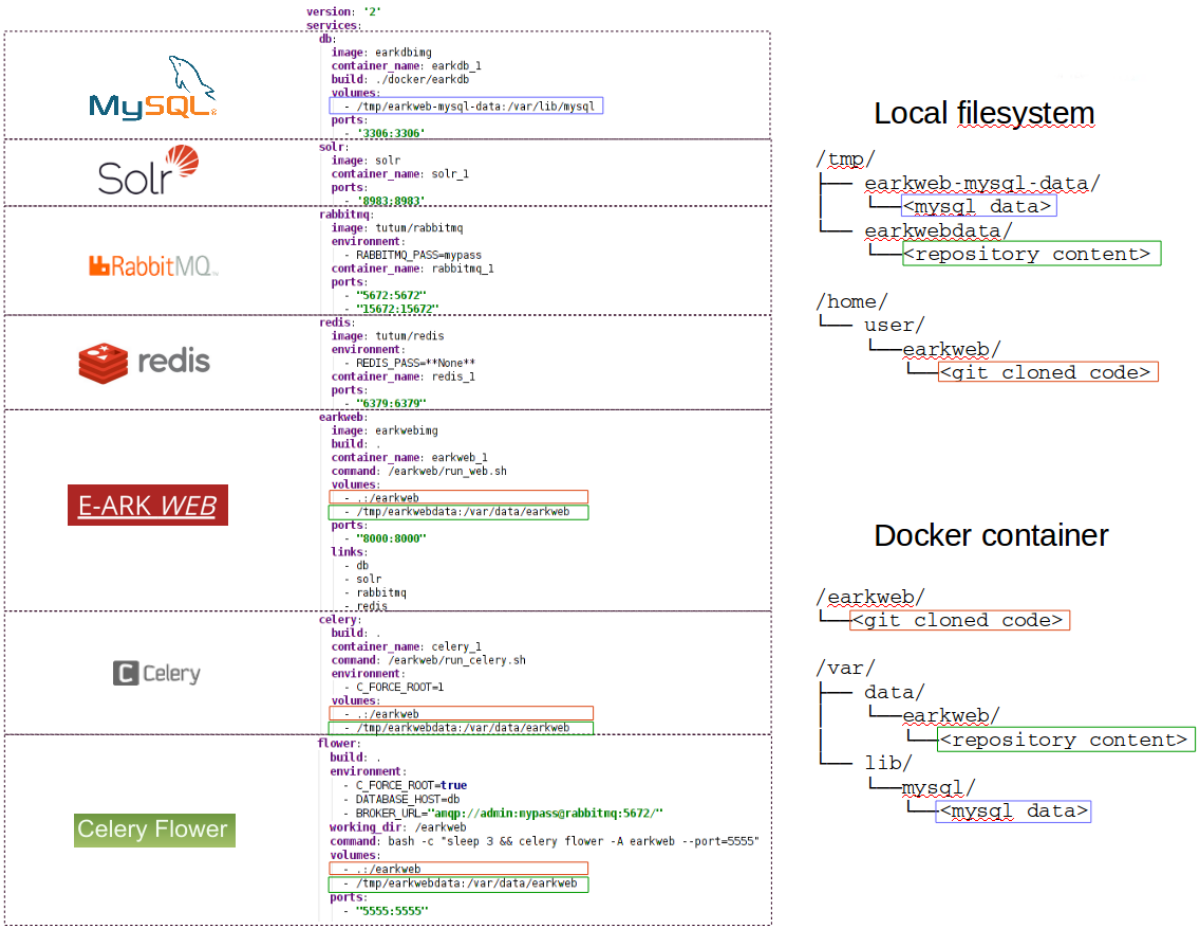

Figure 3 Data directories of the Docker containers and mapping to directories of the host file system

3. Revised Cluster Software Stack

A major part of the work carried out in WP6 in the final project year was the development of a new release of the scalable backend of the IPRIP. The backend is mainly responsible for providing storage, search, and access methods for the ingested information packages. In contrast to the standalone version described in the previous section, the scalable backend relies on data-centric technologies that enable a cluster deployment. As described in the previous WP6 deliverables, the cluster deployment supports scalable storage based on the Hadoop Distributed File System, a distributed and replicated search environment that is based in the SolR cloud platform, and a data-centric repository that provides random access and full-text search to all data items shipped with the ingested information packages. The initial system was based on the CDH4.7 Hadoop distribution and the Lily project, and has been upgraded to CDH5.8 and the Cloudera Search platform in the recent version.

3.1. Motiviation

As the Hadoop eco-system is developing at rapid pace, it is crucial to ensure that applications that build on the Hadoop software stack are kept up-to-date in order to ensure they do not rely on outdated software platforms. One can view the installation of a Hadoop environment as a kind of operating system for the cluster. In order to execute an application, it is required to compile it for the particular software components and APIs installed on the cluster.

It is also important to note that many Hadoop-based applications rely on higher-level frameworks (like Pig, Hive, or HBase) which use Hadoop as their underlying environment. For this reason, cluster administrators typically rely on Hadoop software distributions that include a variety of optional software components that can be deployed on top of the Hadoop base installation. Installing software via a distribution eases the deployment procedure and ensures that the installed components are stable and interoperable.

Apache Hadoop distributions are for example available from Cloudera[26], Hortonworks[27], and MapR[28]. The IPRIP relies on Cloudera's CDH distribution for cluster deployments. It also makes use of the Hadoop-based Lily repository[29] which provides a scalable repository system for storing, searching, and retrieving records and content items. The latest version of Lily was however written for the CDH 4.7 distribution which has passed its end of maintenance date. Besides providing a convenient and powerful repository API on top of HDFS and HBase, a major strength of Lily is its integration of the Apache SolR search platform with the distributed Hadoop environment.

At the time writing this report, the actual Cloudera Hadoop distribution is CDH 5.8. The CDH 5.x Hadoop distributions include the Lily HBase indexer as part of the newly introduced Cloudera search platform which provides a tight integration of Apache SolR with the CDH

[26] http://www.cloudera.com/
[27] http://hortonworks.com/
[28] https://www.mapr.com/
[29] https://github.com/NGDATA/lilyproject

distribution. Compared to CDH 4.x, the CDH 5.x distribution also provides stronger support for advanced data analytics on Hadoop by integrating the Apache Spark platform[30].

As a consequence, WP6 started to port the scalable E-ARK Web backend from CDH4.7, together with the Lily repository, to CDH5.8 and Cloudera Search. The main motivation for this decision was to upgrade the E-ARK Web cluster software stack to a contemporary Hadoop platform, allowing us to maintain and further develop the software on a long-term basis.

3.2. Revised Data-Centric Implementation

The data-centric backend implementation consists of a set of source code projects that deal with data transfer, messaging, ingest, storage and search as well as access and data mining, and are made available through the E-Ark Github repository[31]. As a result of porting the backend from CDH 4.7 to CDH 5.8, it was required to create a new version of the ingest component which is responsible for loading data into the repository. The general functionality of this component, as well as its interfaces, remains, however, unchanged.

To summarize the server-sided workflow: Clients upload information packages to a staging area on the HDFS. During repository ingest, the content and metadata is extracted from the packages. For each content item, a repository record is created and stored on the cluster, where it is indexed and made available through an access API.

In its present version[32], the entire payload for the created repository records is stored within the HBase system. Large binary objects will be written to HDFS only in later versions[33]. As the HBase data model stores all data as uninterpreted byte arrays and due to the absence of the Lily API, it was required to develop a small software layer (the E-ARK repository client) that allows one to store basic data types such numbers, strings, or dates into the database, and to ensure that these data types are also understood by the full-text indexer (provided via Solr).

[30] http://spark.apache.org/
[31] https://github.com/eark-project
[32] presently available through the git branch: cdh5.8
[33] Objects above 50 MB should be stored on HDFS only and not being put into an HBASE cell. The object type is not relevant in this context as HBase stores its cells as uninterpreted bytes arrays.

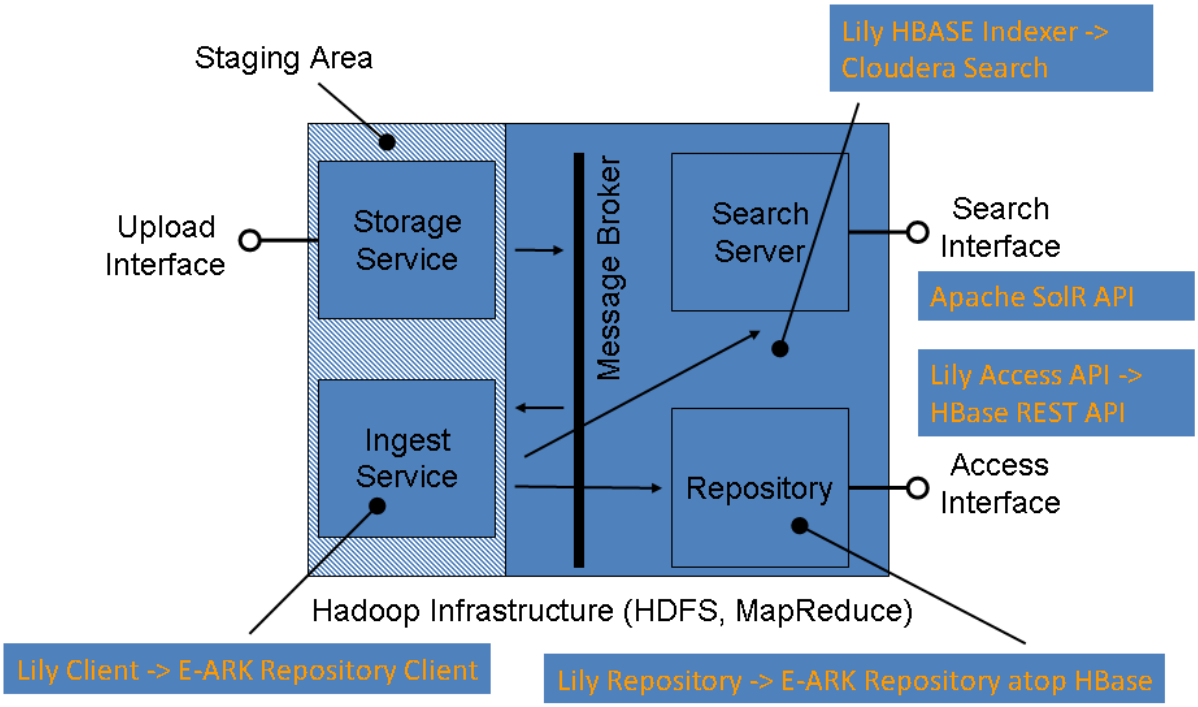

Figure 4 Components of the data-centric backend implementation and migration from Lily to Cloudera Search.

Figure 4 provides an overview of the Hadoop-based backend implementation of the IPRIP. A detailed description of the backend architecture and its components has been provided in deliverable D6.2. Here, we highlight modifications that have been made to the backend system porting the backend from CDH 4.7 to CDH 5.8.

E-ARK Repository Client: The components provide a thin layer that enables the ingest service to store content extracted from the ingested information packages into the repository. In its present implementation, this layer supports storing data items into HBase only. Very large data items are currently ignored and will be stored to the HDFS file system in future versions. The software layer supports a simple typing mechanism that allows the system to take advantage of basic data types like strings, dates, or numbers.

E-ARK HBase Repository: In its present version, data is directly stored into HBase. Each data item (e.g. a file contained within an information package), is stored as a separate column. The fields of each column reflect the structure of the record created by the repository client. The record structure and the structure of the row key have not been changed compared with the previous version, which used the Lily client for record creation, as described in deliverable D6.2.

Indexing and Search Server: Lily's HBase triggering and indexing mechanisms have been integrated with the latest CDH release[34]. In its present version, the data-centric backend makes use of CDH5.8 and incorporates the toolset provided by Cloudera search. This includes the Lily HBase Batch Indexer as well as the Lily HBase near real-time (NRT)

[34] http://blog.cloudera.com/blog/2013/07/cloudera-search-over-apache-hbase-a-story-of-collaboration/

Indexer Service. Cloudera Morphlines are used for the Extract-Transfer-Load (ETL) procedure required to transfer data from HBASE and SolR for indexing.

Search Interface: The IPRIP and its front-end components make direct use of the Solr REST interface and search API, as described in deliverable D6.1. Both, Lily and Cloudera Search (CDH5.8) rely on Solr 4.x as their search server. Consequently, the search interface of the IPRIP remains unchanged after porting it to CDH5.8.

Access Interface: In its current version, the IPRIP stores all data to the HBASE distributed database. After porting the backend to CDH5.8 the HBASE REST API[35] has been used for providing the access interface. In order to support both HDFS and HBASE as storage media for the repository, it will be required to enhance the access interface accordingly.

Finally, it is important to note that the introduced changes integrate well with the graphical environments provided by the IPRIP, and hence are not recognizable by the end-user.

4. Demonstrators and Showcases

In the following, we provide an overview of implemented use-case demonstrations and data-mining showcases implemented in the final project year.

4.1. Natural Language Processing of Archived Content

The Natural Language Processing (NLP) part of E-ARK Web serves as a showcase for text mining that is possible in the context of large-scale archived data. Our goal was to improve and add value to the *Solr*[36] index created from the ingested data, as well as show possible options in terms of access methods. For the reference implementation, we focused on Named Entity Recognition (NER). In the following we will describe experiments carried out for NER as well as additional approaches like geo-spatial search which have been carried out as proof-of-concept implementations. A summary of NLP techniques used in the experiments carried out in this context are provided in figure 5.

[35] https://www.cloudera.com/documentation/enterprise/5-8-x/topics/admin_hbase_rest_api.html
[36] http://lucene.apache.org/solr/

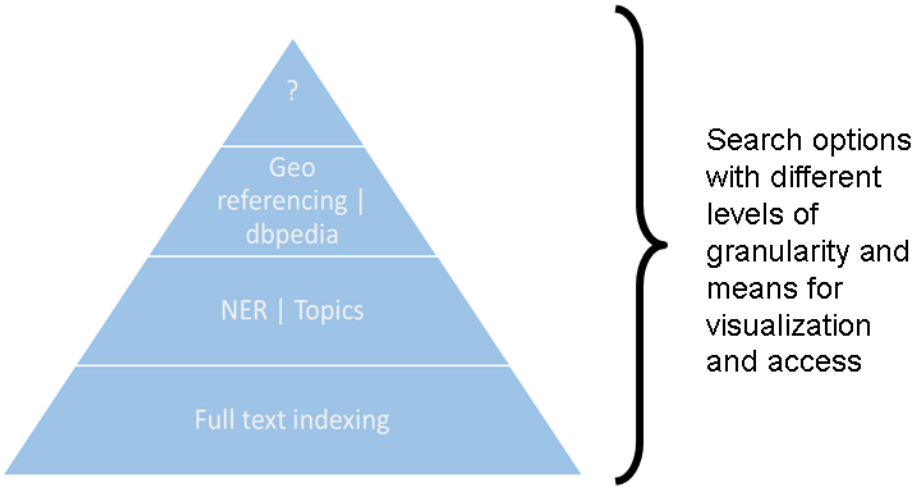

Figure 5 NLP techniques utilized in experiments. Visualization methods and the utilization of discovered implicit information enable the development of advanced search and access methods.

The developed NER component makes use of an already generated *Solr* index as described previously. The Solr index serves as (a) the data source for data mining and (b) as the data sink where results are added. A data set for mining is typically created by formulating a query that selects the data of interest (e.g. data that lies within a certain time frame or within a range of information packages). In its current version, data mining takes place by using the Celery-based task processing infrastructure provided by the E-ARK Web project, as explained in deliverable D6.2 and in section 5 of this document.

The experiments carried out in the context of the data mining showcase make use of the IPRIP search interface in order to receive input data through HTTP requests. The *Solr* index generated by the IPRIP contains, besides a significant set of metadata elements, the content for every text document in plain text. The individual items of a data set are unambiguously identified using identifiers provided by the *Solr* record.

Named Entity Recognition

For carrying out NER, the *Stanford CRF-NER*[37] parser was used in combination with the *NLTK*[38] (Natural Language Toolkit) Python library. These were orchestrated and processed in parallel using one main task queue, and a subtask for every file that should be processed in the Celery environment provided by E-ARK Web.

[37] http://nlp.stanford.edu/software/CRF-NER.shtml
[38] http://www.nltk.org/

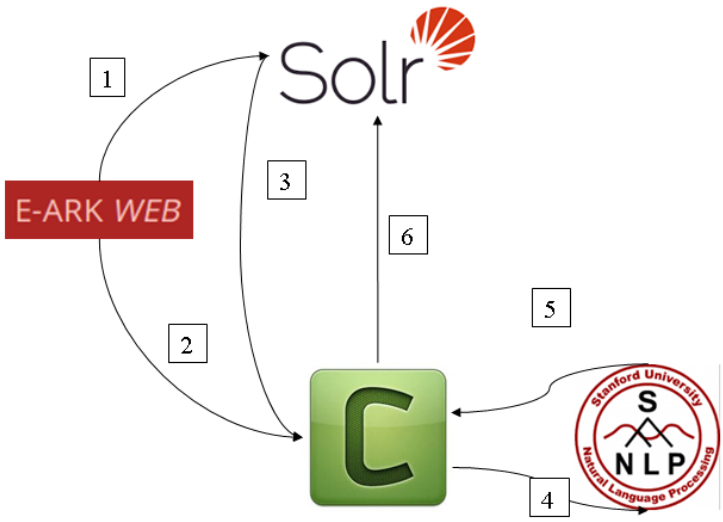

Figure 6 Conceptual workflow implemented for carrying out Named Entity Recognition (NER) using the Integrated Platform Reference Implementation Prototype.

The conceptual workflow and setup for carrying out the NER experiments are shown in figure 6. In the first step (1), the user specifies the criteria for the records to be processed within the E-ARK Web user interface. These criteria are automatically translated into a valid Solr query and sent to the search server; In case matching data is found by the search server, a corresponding Celery task is created (2). This task retrieves the content of every found document from Solr (3). A set of asynchronous subtasks is started to perform NER in parallel (4) using the Stanford NER parser (5).

An example of a tagged sentence can be seen below.

```
Several/O possible/O plans/O emerged/O from/O the/O talks/O
,/O held/O at/O the/O Federal/ORGANIZATION
Reserve/ORGANIZATION Bank/ORGANIZATION of/ORGANIZATION
New/ORGANIZATION York/ORGANIZATION and/O led/O by/O
Timothy/PERSON R./PERSON Geithner/PERSON ,/O the/O
president/O of/O the/O New/ORGANIZATION York/ORGANIZATION
Fed/ORGANIZATION ,/O and/O Treasury/ORGANIZATION Secretary/O
Henry/PERSON M./PERSON Paulson/PERSON Jr./PERSON ./O
```

Optionally, the Celery worker sends a request to Solr, the obtained results are written back to the search server extending the Solr index with information on the found entities for each document or data item (6). The index entry for the individual data items is therefore extended with corresponding fields, as show in the XML snippet below.

```
  <doc>
    <str name="path">1e232ce6-9177-401e-8d83-
4eeb28dc680b/representations/rep-002/data/Charlemagne.pdf</str>
    <str name="contentType">application/pdf</str>
    <arr name="locations_ss">
      <str>Herstal</str>
      <str>Monte</str>
      <str>Cassino</str>
      <str>Aquitaine</str>
      <str>Aquitania</str>
    </arr>
  </doc>
```

Fields are created for extracted entity classes like *location*, *person,* and *organization*. This extension provides additional search terms that allow a user to formulate richer queries, and to find archived documents based on automatically recognized entities:

Examples are:

- Which entities have been found in a particular package/file?
- Which packages/files contain data on a specific location/person/organization?

Here, it is important to note that NER results cannot be considered as 100% accurate, so both false positives as well as false negatives are possible; nonetheless, we believe that this approach offers valuable insights into large amounts of data (within certain margins of error).

Geo-Coding

For implementing automatic geo-coding, an additional experiment was carried out that uses NER results as a basis to add geo information to data items. This was achieved by adding

geo information to documents based on identified locations. Subsequent to named entity recognition, geo-coordinates were retrieved for locations using the *Nominatim*[39] database.

When combining the extracted geo information with a tool like *Peripleo*,[40] the geo-spatial search engine also used by the geo-data demonstrator described in section 4.2, it is possible to offer novel ways to analyse and access text documents. The approach makes it possible to visualize the geographic focus of documents using digital maps, and to access the document with respect to geo information through the same interface. In addition, it is possible to visualize geo-locations and associated data sets over a timely dimension using Peripleo. This allows a user, for example, to visualize and browse documents that are associated with a geo-location (like a city) in different time periods.

Automatic Database Building

In this context, we also included the XML database *eXistdb*[41] as an additional database to store extensive NLP results, and use them as a basis to gather more information about documents. Using such an external data store, we are free to include information that would not be suitable for the Solr index. This covers information such as *tf-idf* scores for every detected entity to rank entities according to their statistical importance and significance in relation to the (sub-)corpus; coordinates (latitude, longitude) for every location; *dpbedia*[42] links for entities etc. Such information can only be found implicitly in the documents – humans make those associations and links based on their knowledge of the world, but they cannot be searched unless they are explicitly added to a database or search index.

Topic Modeling

An experiment on topic modelling has been carried out using the Python machine learning library *scikit-learn*.[43] Using a pre-trained model based on newspaper categories (“Sports”, “Culture”, “Politics” etc.), abstract categories of documents were identified and adding to the Solr index. The idea of this approach is to enable users to quickly identify and select or exclude documents, and to narrow down search queries on large datasets. An important prerequisite for topic modelling, however, is the creation of a trained model that is capable of identifying relevant categories in the processed data collection.

4.2. Visualization of Extracted Geographical Data

The visualization of extracted geographical data is part of the development of the AIP to DIP conversion[44]. As a pre-requisite of this conversion, it is assumed that data is available in the form of data files encoded following the Geography Markup Language (GML) Encoding Standard.[45] Examples of these files are GML representations of the regions of Slovenia in a time series of the years 1994, 1995, 1998, 2002, 2006, 2010, and 2015.[46]

[39] http://wiki.openstreetmap.org/wiki/Nominatim
[40] https://github.com/pelagios/peripleo
[41] http://www.exist-db.org/exist/apps/homepage/index.html
[42] http://wiki.dbpedia.org/
[43] http://scikit-learn.org/stable/
[44] A more detailed description of the AIP-DIP conversion tool can be found in E-ARK deliverable D5.4.
[45] GML XML Schema files: http://schemas.opengis.net/gml/3.2.1/
[46] https://github.com/eark-project/earkweb/tree/master/earkresources/geodata

Regarding the visualization use case, the relevant information available in these GML example files are lists of white space separated coordinate values (each X and Y coordinate pair being comma separated) stored as the `gml:coordinates` element's text value which represent polygons of the Slovenian regions. An example section of the GML XML file about the Slovenian region "AJDOVŠČINA" is provided in the XML snippet shown in figure 7. Only the beginning and the end of the coordinate list is shown to save space. Note that the first coordinate is the same as the last coordinate because the list of coordinates must describe a closed ring by convention.

The *Peripleo*[47] tool from the *Pelagios* project[48] was chosen as the environment for visualizing the geographical data because it is especially designed to visualize, search, and discover sets of geographical data which are created over time. *Peripleo* used an RDF based format where the geographical properties are available as one feature amongst other properties of a gazetteer entity, such as name, and year of a region or point on a map.

[47] https://github.com/eark-project/peripleo
[48] http://commons.pelagios.org

```
<gml:featureMember>
    <ogr:ob_1994 fid="ob_1994.0">
           <ogr:geometryProperty>
                 <gml:Polygon srsName="EPSG:3794">
                       <gml:outerBoundaryIs>
                             <gml:LinearRing>
                                   <gml:coordinates>
                                   432376.312,81964.773
                                   432424.5,81848.507
                                   ...
                                   432206.187,82004.351
                                   432376.312,81964.773
                                   </gml:coordinates>
                             </gml:LinearRing>
                 </gml:outerBoundaryIs>
                 </gml:Polygon>
           </ogr:geometryProperty>
           <ogr:D46_ID>10053609</ogr:D46_ID>
           <ogr:OB_ATR_PE_SIF>OS</ogr:OB_ATR_PE_SIF>
           <ogr:OB_ATR_PE_IME>
           OBCINA - STARA (do 1.12.1994)
           </ogr:OB_ATR_PE_IME>
           <ogr:OB_ATR_D46_ID>1</ogr:OB_ATR_D46_ID>
           <ogr:OB_ATR_D46_IME>AJDOVŠČINA</ogr:OB_ATR_D46_IME>
           <ogr:OB_ATR_N8>0</ogr:OB_ATR_N8>
           <ogr:OB_ATR_POVRSINA>352640500</ogr:OB_ATR_POVRSINA>
           <ogr:OB_ATR_Y_C>415110</ogr:OB_ATR_Y_C>
           <ogr:OB_ATR_X_C>83160</ogr:OB_ATR_X_C>
    </ogr:ob_1994>
</gml:featureMember>
```

Figure 7 XML snippet of a section of the GML file describing one region of Slovenia.

To publish a gazetteer to *Peripleo*, it is required to create a representation of the region in RDF. A portion of the RDF file, which corresponds to the GML encoded region shown in the *Peripleo* RDF[49] snippet, is shown in figure 8*. In the Peripleo RDF, the region is encoded as a WKT string*[50]*. A Python module* was created to allow converting the GML coordinates to WKT.[51] This module can convert a list of position values (element `gml:posList`) into a coordinate tuples list. Furthermore, as the coordinates are in the coordinate system

[49] Full example available at https://raw.githubusercontent.com/eark-project/earkweb/master/earkresources/geodata/ob_1994.gml
[50] http://www.giswiki.org/wiki/Well_Known_Text
[51] https://github.com/eark-project/earkweb/blob/master/earkcore/conversion/peripleo/gml_to_wkt.py

EPSG:3912 (Slovene National Grid)[52], a function from the Python package *pyproj*[53] was used to transform the coordinates to EPSG:4326 (WGS84 - World Geodetic System 1984)[54] used in *Peripleo*.

```
@prefix cito: <http://purl.org/spar/cito> .
@prefix cnt: <http://www.w3.org/2011/content#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix gn: <http://www.geonames.org/ontology#> .
@prefix lawd: <http://lawd.info/ontology/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://127.0.0.1:8000/earkweb/sip2aip/working_area/aip2dip/0fa52d40-77aa-41cf-b4ab-
9222307cf14e/#place/Ajdovščina/1994> a lawd:Place ;
   rdfs:label "Ajdovščina"@si ;
   dcterms:isPartOf
http://earkdev.ait.ac.at/earkweb/sip2aip/working_area/sip2aip/5c6f5563-7665-4719-a2b6-
356ea033c1d/#place/Slovenia> ;
   dcterms:temporal "1994" ;
   gn:countryCode "SI" ;
   geosparql:hasGeometry [ geosparql:asWKT "POLYGON ((
      14.1240482574528 45.87850063597578,
      14.12468538434922 45.87745938243139,
      …
      14.12185129747619 45.87883988684491,
      14.1240482574528 45.87850063597578))" ] .
```

Figure 8 Peripleo RDF snippet description of one region

A set of Celery backend tasks were created which can be used for DIP creation to validate GML data (`class DIPGMLDataValidation`) and to add a *Peripleo* RDF representation to the DIP (`class DIPGMLDataConversion`).[55]

Figure 9 shows an open DIP creation process (process id: 0fa52d40-77aa-41cf-b4ab-9222307cf14e). The DIP creation process used one AIP (urn:uuid:f237c314-757f-4f11-9cb2-

[52] http://epsg.io/3912
[53] https://pypi.python.org/pypi/pyproj/
[54] http://epsg.io/4326
[55] https://github.com/eark-project/earkweb/blob/master/workers/tasks.py

b0cce7399623) with one representation named ‘gmlrep’ (GML files), to create a Peripleo RDF representation named ‘peripleottl’ (TTL files).

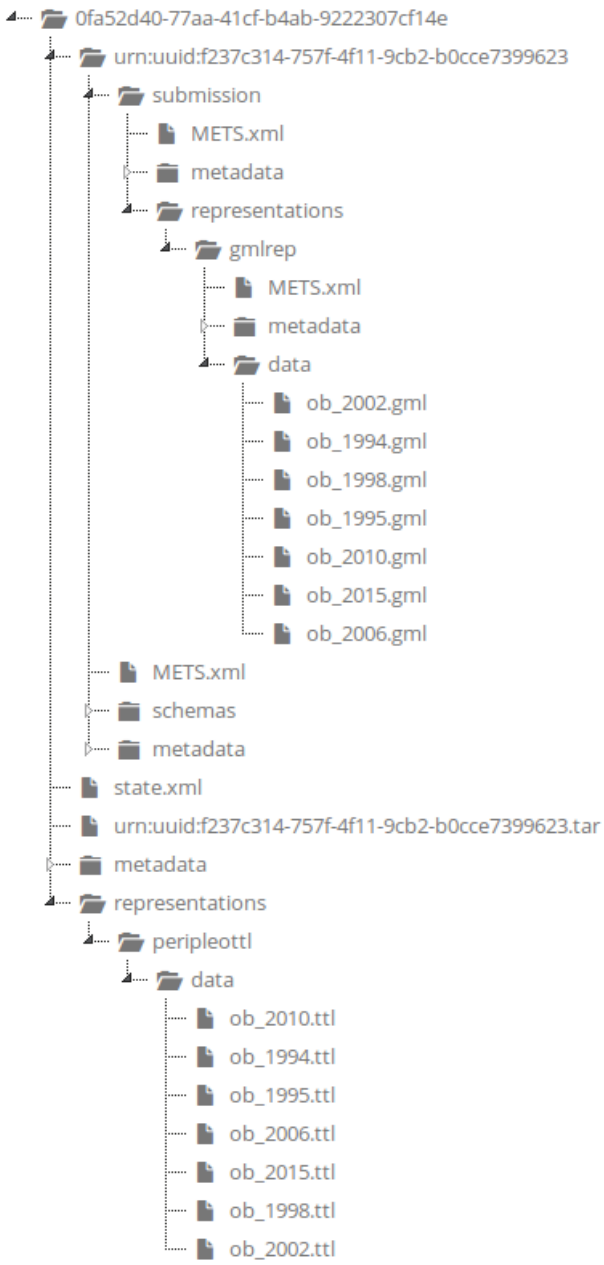

Figure 9 DIP after adding the Peripleo RDF representation to a DIP

Finally, the `DIPPeripleoDeployment` task[56] uses the Peripleo Gazetteer Upload interface to upload the Peripleo RDF files into Peripleo.[57]

[56] https://github.com/eark-project/earkweb/blob/master/workers/tasks.py
[57] Specific changes required in the Peripleo admin API have been published in an E-ARK specific Peripleo branch available at https://github.com/pelagios/peripleo/tree/eark.

Figure 10 gives an example of searching the region “Llubljana” of Slovenia in the time range between 1994 and 2015.

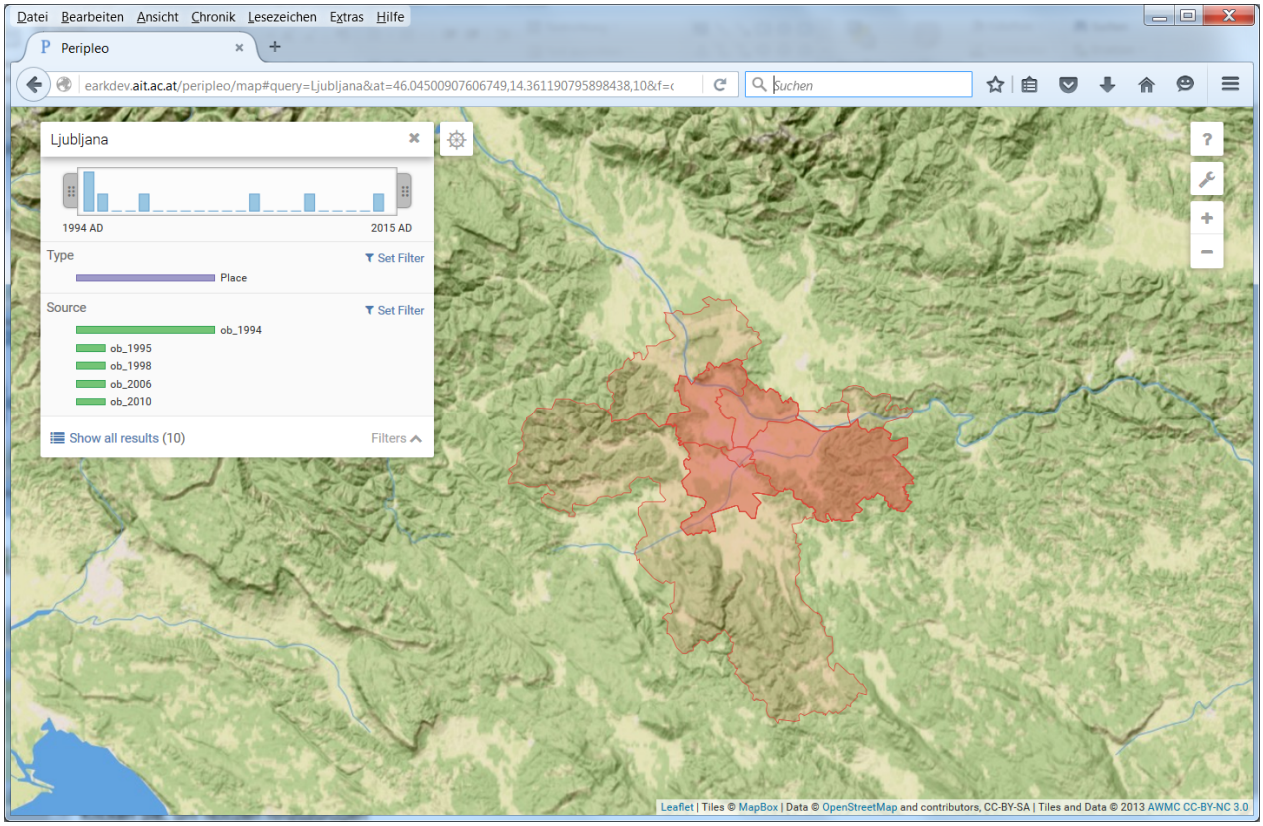

Figure 10 Rendering Slovenian regions on the map using Peripleo.

4.3. Archiving and Accessing Operational Databases

Access of archived databases was implemented as part of the AIP to DIP conversion. As a pre-requisite of this conversion, it was assumed that data is available in form of data SIARD[58] files which have been created as part of the AIP. For this purpose, the ingest workflow can make use of Database Preservation Toolkit[59] to generate SIARD from databases, and create a SIARD representation as part of the AIP.

Celery backend tasks have been created which allow importing a SIARD file into an available relational database management system supported by the Database Preservation Toolkit (`class DIPImportSIARD`), and which allow exporting the database (`class DIPExportSIARD`) as the version of the DIP which is created for access purposes.[60] After the import of the database into an available relational database management system, any required modifications (e.g. deleting parts containing sensitive information) can be performed before exporting the database version for access.

[58] https://www.bar.admin.ch/dam/bar/it/dokumente/kundeninformation/siard_formatbeschreibung.pdf.download.pdf/siard_format_descriptioning.pdf
[59] http://www.database-preservation.com
[60] https://github.com/eark-project/earkweb/blob/master/workers/tasks.py

Figure 11 describes a basic workflow of archiving a database (here a Postgres database) in SIARD format which is then imported into another target database (here Oracle). Finally, further analysis can be performed using specific tools, e.g. Oracle's OLAP tools. [61]

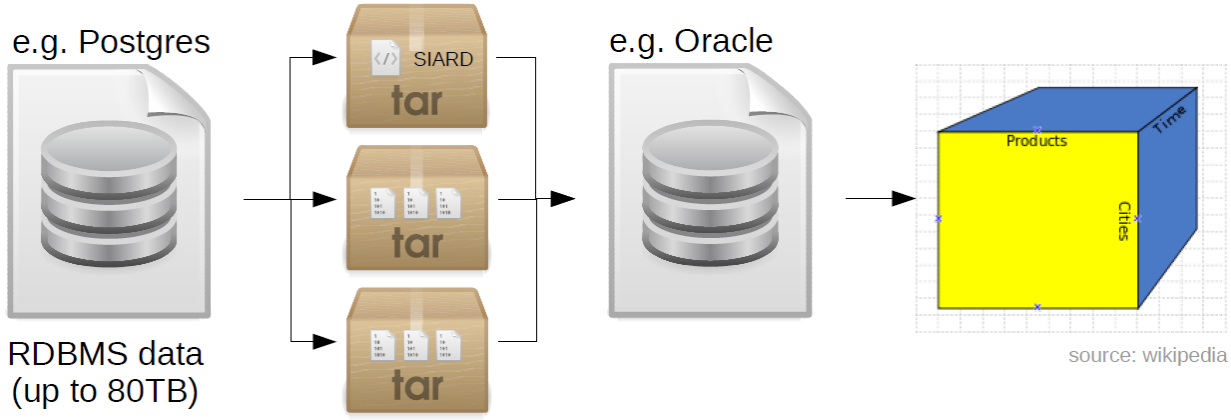

Figure 11 Workflow describing the archiving of a database in SIARD format, the import into another target database, and the subsequent analysis using dimensional tables.

A pilot carried out at the Hungarian National Archive (pilot 7) in the context of E-ARK was dealing with the problem of accessing archived databases. In this context, a workflow was implemented that loads a SIARD-based archived database from an E-ARK DIP into an Oracle-based data warehouse providing convenient access and reporting mechanisms for end-users. The various OLAP technologies utilized in pilot 7 are detailed in E-ARK deliverable D5.4. A general presentation of the pilot is available on Github[62]. In addition, deliverable D6.4 includes a proposal for an E-ARK Standard for Vendor-Independent Archiving of Data Warehouses.

5. Testbeds and Pilot Deployments

The following provides an overview of testbeds and pilot deployments that haven been evaluated by E-ARK stakeholders. The main difference between the different deployments of the IPRIP is between the ones which use the Standalone Deployment Stack shown in figure 12, and the ones which use the Cluster Deployment Stack shown in figure 13. The public demonstration instance (earkdev) falls somewhere between the two because it uses the Cluster Software Stack, but it is actually installed on a single virtual machine (pseudo-distributed Hadoop backend).

[61] http://www.oracle.com/technetwork/database/options/olap
[62] https://github.com/eark-project/Data-Warehouse-and-OLAP

Standalone Deployment Stack	Cluster Deployment Stack
Pilot at the Slovenian National Archives	Pilot at the Hungarian National Archives
Virtual machine available for download[63]	Pseudo-distributed[64] public instance earkdev

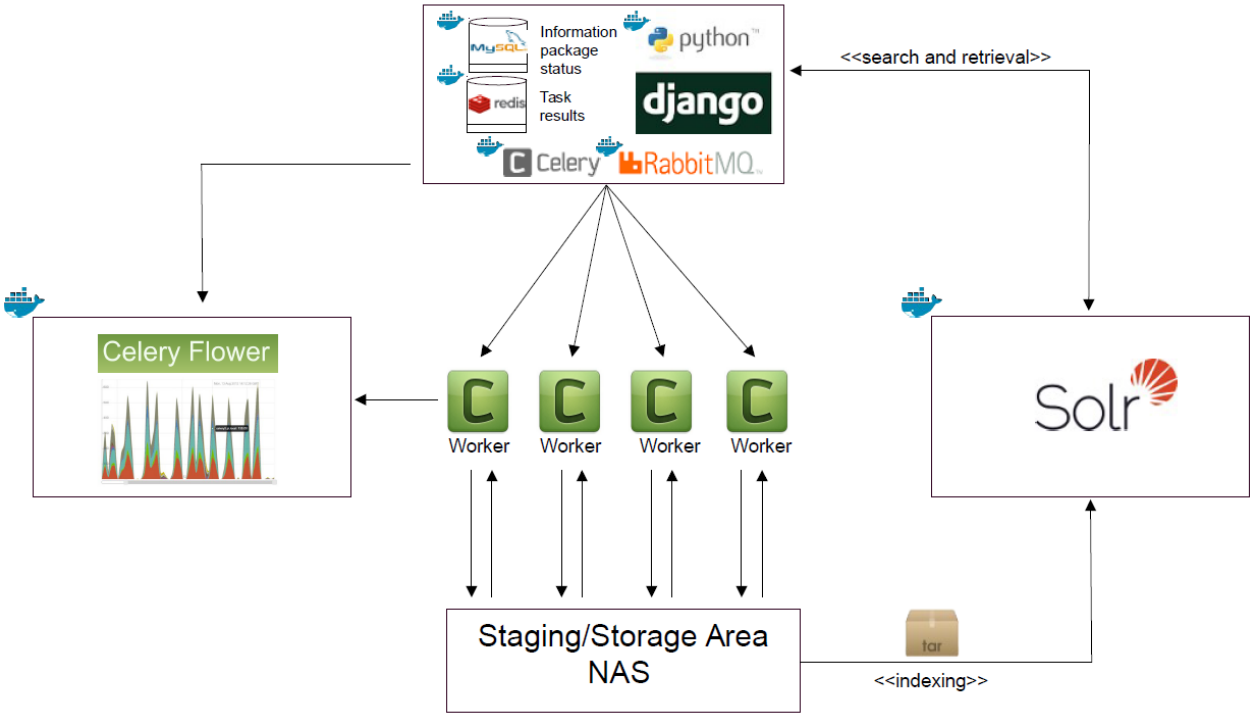

Figure 12 Standalone software stack

[63] http://earkdev.ait.ac.at/eark/pilots/eark-pilot-vm.ova
[64] http://earkdev.ait.ac.at/earkweb/

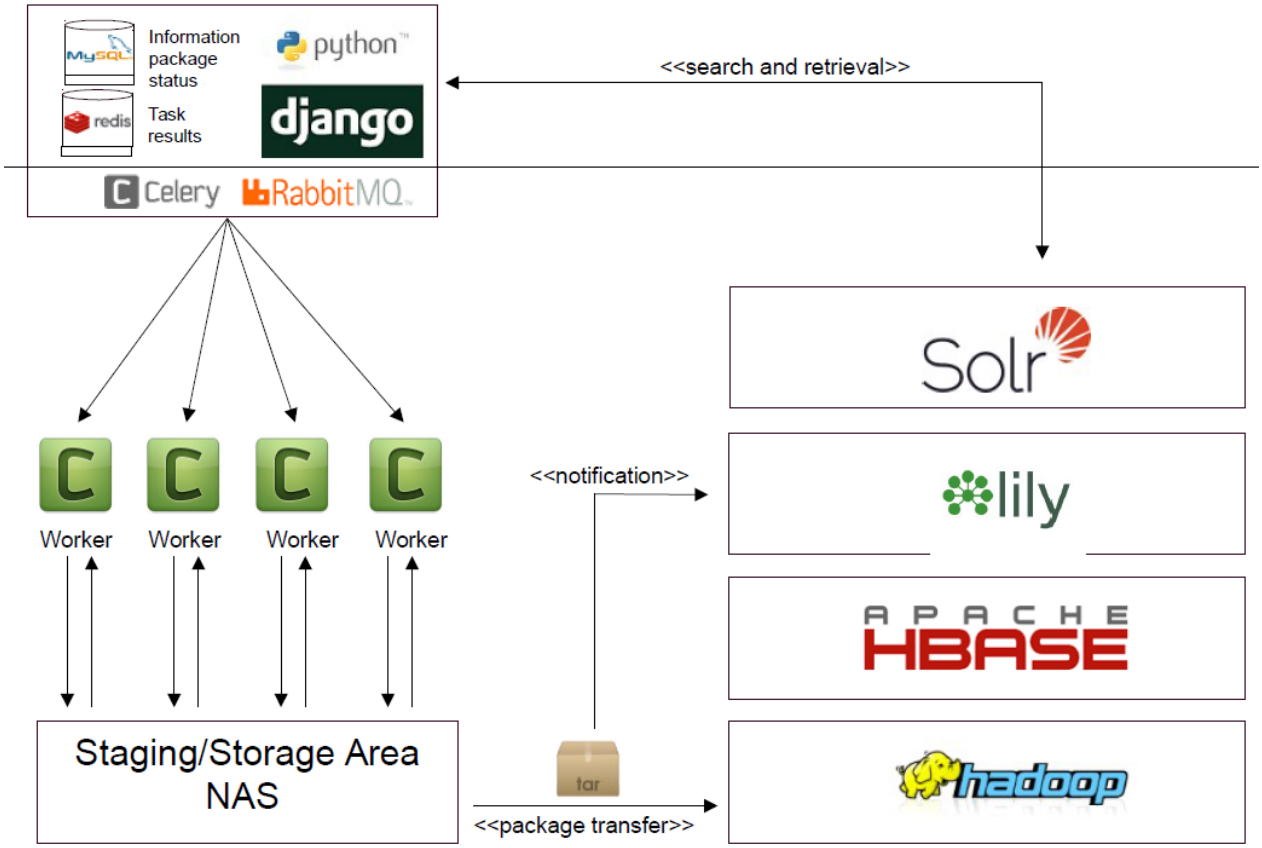

Figure 13 Cluster Software Stack

5.1. Earkdev Public Deployment

Hardware	
Type	Virtual Machine
RAM	8GB
Processors	2 * Intel Xeon CPU 2.50GHz
Operating System	Ubuntu 14.04 Desktop
Deployed Software	
E-ARK Web frontend	WSGI application deployed to an Apache Webserver.
Storage Backend	The storage backend of this instance is a pseudo-distributed Hadoop installation. All Hadoop services, i.e. Jobtracker, Tasktracker, HDFS Datanode, HDFS Namenode, and HDFS Secondary-Namenode, are running on the same

	machine.
Indexing, search and access	Lily single instance installation including SolR for search, Lily access API for accessing content files.
Celery distributed task queue	The Celery backend (distributed task queue) is configured with one single worker which runs 4 processes (using 4 cores) in parallel.
Peripleo for visualizing Geodata	Single instance installation of Peripleo providing open API for Peripleo RDF file deployment.
RabbitMQ	Single instance installation used as message broker by the Celery distributed tasks queue and for information exchange between E-ARK Web frontend and backend.
Redis	Single instance installation used as task execution result backend.
Data Mining Showcases	
Full-text indexing and search	Using typical office documents (Word, PDF, etc.) to test the full-text indexing and search features with document collections in the English language.
Named Entity Recognition	Using typical office documents (Word, PDF, etc.) to test the Named Entity Recognition with document collections in the English language.
Database archiving	Database archiving (creating and restoring SIARD files).
Data	
Electronic Documents	The simplest type of data which can be used to do tests in this instance are document collections (e.g. Word or PDF documents). These can be used to test the full-text indexing and search features.
Stakeholders	
Project partners	All project partners are invited to use this instance for testing new features.
DLM Forum members	The first stable version was made available

	for access to DLM forum members.
Invited external parties	An account can be created on request for any external interested party.

5.2. Standalone Deployments at the Slovenian National Archives

Hardware	
Type	Virtual Machine imported using VMWare.
RAM	-
Processors	-
Operating System	Ubuntu 16.04 Desktop
Deployed Software	
E-ARK Web frontend	Django development server.
Storage Backend	Pairtree Storage (cf. section) 2.1
Search and access	Single instance SolR for search. Packages can be indexed individually or the complete Pairtree storage is indexed. Django access API for accessing individual content streams of TAR files entries.
Celery distributed task queue	The Celery backend (distributed task queue) is configured with one single worker which runs 4 processes (using 4 cores) in parallel.
Peripleo for visualizing Geodata	Single instance installation of Peripleo providing open API for Peripleo RDF file deployment.
RabbitMQ	Single instance installation used as message broker by the Celery distributed tasks queue and for information exchange between E-ARK Web frontend and backend.
Redis	Single instance installation used as task execution result backend.
Data Mining Showcases	
Visualisation of Geodata	Visualization of geographical data as described in section 4.

Data	
Geographical data	Geographical data in GML format as described in section 4.
Electronic Documents	Electronic documents for testing full-text indexing and search.
Stakeholders	
National Archives of Slovenia	The National Archives of Slovenia are evaluating the environment together with Roda-in and ESSArch EPP regarding their suitability to support Geodata archiving and access.

5.3. Cluster Deployment at the National Archives of Hungary

Hardware	
Type	The installation at the Hungarian National Archives is an installation on a real computer cluster using 5 servers.
RAM	Master: 48 Gigabyte Slaves: 13 Gigabyte each
Processors	Master: 2 Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (in total 12 cores and 24 threads) Slaves: 2 Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz (in total 16 cores and 32 threads) each
Operating System	RedHat RHEL 6
Deployed Software	
E-ARK Web frontend	WSGI application deployed to an Apache Webserver.
Storage Backend	The storage backend of is a distributed Hadoop installation. The Hadoop services Jobtracker, Namenode, and Secondary-Namenode are installed on the master node, and Tasktracker, and Datanode services are

	installed on each of the slave nodes. Regarding the distributed database HBase, the hbase-master service is installed on the master node and the hbase-regionserver on each of the slave nodes.
Search and access	Lily distributed installation including a SolR Cloud installation for search, Lily access API for accessing content files. The Lily nodes are only installed on the slave machines.
Celery distributed task queue	The Celery backend (distributed task queue) is configured with one single worker which runs 4 processes (using 4 cores) in parallel.
Peripleo for visualizing Geodata	Single instance installation of Peripleo providing open API for Peripleo RDF file deployment.
RabbitMQ	Single instance installation used as message broker by the Celery distributed tasks queue and for information exchange between E-ARK Web frontend and backend.
Redis	Single instance installation used as task execution result backend.
Data Mining Showcases	
Full-text indexing and search	Using typical office documents (Word, PDF, etc.) to test the full-text indexing and search features with document collections in the Hungarian language.
Named Entity Recognition	Using typical office documents (Word, PDF, etc.) to test the Named Entity Recognition with document collections in the Hungarian language.
Database archiving	Database archiving (creating and restoring SIARD files).
Data	
Geographical data	Geographical data in GML format as described in section 4.
Electronic Documents	Electronic documents for testing full-text

	indexing and search.
Stakeholders	
National Archives of Hungary	The National Archives of Hungary are the main stakeholders.

6. Conclusion

This deliverable summarizes a number of technical developments, experiments, and deployments that have been carried out in the final project year in the context of WP6 - Archival Storage, Services and Integration.

The workpackage has developed a flexible software prototype called the E-ARK Integrated Prototype Reference Implementation (IPRIP). The software architecture and taken design decisions have been described in the previous deliverables D6.1 and D6.2. This deliverable focuses on the application of the software environment in different contexts.

The IPRIP provides an archiving environment that supports the creation of E-ARK Submission Information Packages (SIPs) through the E-ARK Web's package creation environment. The workflow environment enables the system to transfer the created information packages to archival storage as well as to a staging area which is used by the E-ARK search infrastructure and the backend repository. Both components support the E-ARK access interfaces which provide support for full-text search and fast retrieval of data items (e.g. documents) contained in information packages. Specific software components like the E-ARK WP5 order management tool take advantage of these interfaces for the computer aided creation of Dissemination Information Packages.

At present the IPRIP comes with two different backend implementations and deployment modes. The IPRIP can be deployed as a standalone system on a single computer. In this deployment mode the system can be easily installed and all computing and IO processes are handled on the same physical machine. The cluster deployment mode is extremely scalable using Python Celery and Apache Hadoop as underlying platforms. Cluster deployments are naturally more complex to configure and require the availability of networked server hardware. Deliverable D6.3 provides details on the utilized software, its configuration, and explains important recent developments.

A major benefit of the IPRIP architecture is the ability to find, retrieve, and process large-volumes of data. This is achieved by storing the content as well as structured data across computational computer nodes which are equipped with persistent storage media as well as computing elements. Full text search is a common use-case (e.g. provided by Web search engines) allowing users to retrieve a ranked list of relevant documents based on entered query terms. This use-case is also supported by the IPRIP search interface. In year 3, WP6 has additionally carried out a set of so called data mining experiments. The goal of these experiments was to develop and showcase strategies that can provide advanced research and visualization services to stakeholders in the archival domain.

D6.3 describes a number of experiments that combine data analysis techniques such as text mining, geo-coding, and multi-dimensional visualization (e.g. based on time and location). The goal of this work was to improve the user experience and/or to enrich structured and searchable information on the archived data sets. The experiments have been carried out based on specific use-cases and test data-sets. D6.3 also describes an initial set of pilot deployments of the IPRIP (in both cluster and standalone) mode that have been used to ingest, index, and access data at different stakeholder sites.

For future work, we see a large R&D potential in maturing the developed data mining strategies for archives and implementing and piloting them in combination with existing archival information systems at stakeholder sites.