REPEATABLE DEPLOYMENT OF AN OPEN STANDARDS, OPEN SOURCE AND OPEN DATA STACK FOR BUILDING A FEDERATED MARINE DATA MANAGEMENT AND DECISION SUPPORT SYSTEM FOR SOUTH AFRICA

G. McFerren*, R. Molapo and B. McAlister

CSIR Meraka Institute, Meiring Naude Road, Pretoria, South Africa – (gmcferren,rmolapo,bmcalister)@csir.co.za

FOSS4G 2018

KEY WORDS: Open standards and Interoperability, Data Management System, CKAN, CSW, Case Study, Distributed Systems Architecture, Oceans and Coasts, DevOps

ABSTRACT:

The National Oceans and Coastal Information Management System (OCIMS) of South Africa is a large, integrated IT system for enhancing oceans, coastal and maritime governance, and supporting sustainable economic utilisation of ocean and coastal resources. This article is a case study, describing how a range of Free and Open Source Software are deployed to generate the Open Standards based core of this federated system for providing decision support applications in addition to data and information management, access and dissemination services. This article demonstrates the importance of modern software development and deployment approaches in constructing the OCIMS core and easing the integration process with other systems in the federation. Finally, this article discusses some lessons learned and reflects on the lineage of OCIMS architectural choices and how these approaches may need to adapt to changing computing environments

1. INTRODUCTION

The National Oceans and Coastal Information Management System (OCIMS) - https://www.ocims.gov.za - of South Africa is an ongoing government funded effort to integrate heterogeneous IT systems from various organisations into a system-of-systems to support enhanced oceans, coastal and maritime governance and sustainable utilisation goals. OCIMS supports these goals through providing facilities for monitoring of environmental variables and human socio-economic activity, compliance and enforcement support, planning and assessment and information dissemination. Concretely, OCIMS offers 1) IT components for publishing of, discovery of, access to, interaction with and management of data and content services, 2) decision support tools and applications, 3) information dissemination channels and 4) information technology services. Most of these offerings are inherently geospatially oriented.

This article describes the architectural patterns by which open standards for data (and metadata) interchange and service interfaces are used in building the OCIMS federated system. This article demonstrates the importance of modern software development and deployment methods - such as containerisation and service-oriented architectures - that allowed not only the core of OCIMS to be constructed and deployed, but also several of the nodes in the federation. Effectively, the ease of customisation and deployment of containerised FOSS geospatial and data management components, and the isolation provided by this approach to software life-cycle management, allows nodes representing different projects, organisations and sub-organisations to join the OCIMS data and service federation with minimal effort in such a way that their services, data and metadata are interoperable (accessible, available, understandable and harvestable).

This article characterises the primarily Free and Open Source Software software stack that enables the core of the OCIMS federated system and provides much of the functionality. The core portion of the overall system is composed of the data management system Comprehensive Knowledge Archive Network or CKAN (CKAN, 2018), enabled with various spatial search/browse extensions, an OGC Catalogue Service for the Web implementation (pycsw, 2018), and a content management system/ landing page. The OCIMS core preferably harvests metadata from nodes in the federation, but this is not always possible if nodes do not provide harvestable endpoints.

Lessons from initial deployment of the OCIMS are articulated, for example, the difficulty of working with local profiles of metadata standards due to lack of software tooling or the extra steps needed to integrate nodes into the federated system where the owners/maintainers of the node have not planned for interoperability arrangements. Positive lessons are also elucidated, such as the advantages that can be gained from building data processing chains that are configured to emit data products as similar as possible to each other (e.g. NetCDF CF convention structures) and that automatically emit metadata to catalogue services concerning their output products. The main advantage here is the “recipe”-like way that OCIMS ready nodes can be built and deployed.

Finally, this article reflects on the lineage of the architectural choices made in order to construct the OCIMS system, and how systems like OCIMS will likely need to be adapted to the realities of geospatial data and processing increasingly moving into a cloud computing paradigm.

2. OCIMS SYSTEM DESCRIPTION

2.1 High level Description

OCIMS is designed as an integration between heterogeneous systems from various organisations to form a virtual enterprise of stakeholders, data and information producers and consumers concerned with monitoring, modelling, reporting, informing and decision-making in the Oceans and Coastal domains. Organisations contributing to OCIMS display different levels of IT capability and are invested in different technology stacks. The systems joining to this virtual enterprise should enrich rather than disrupt the wider system, while the continued operation of these individual systems should not be disrupted by changes in the National OCIMS. Thus, OCIMS is a loosely coupled system-of-systems - a concept discussed in (Percivall, et al., 2015) - where each component system is designed, developed, tested, operated and owned independent of OCIMS, as illustrated in Figure 1.

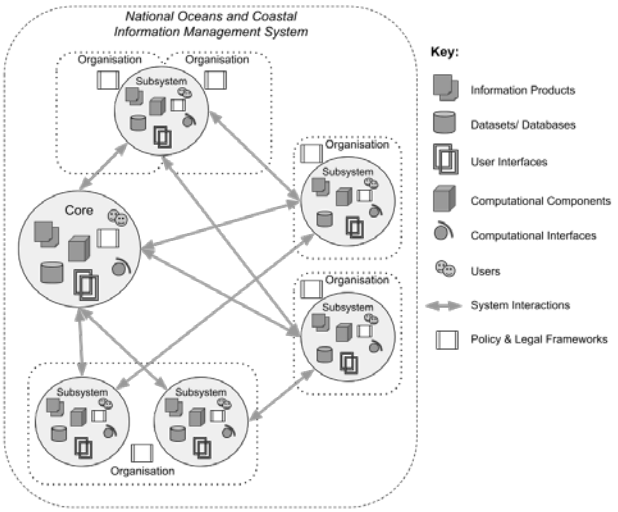

Figure 1. System-of-Systems Interactions

This design requires that inter-system communication ideally happens at well understood interfaces, with well described control signals/protocols exchanging a finite set of data types and encodings between applications and service requests. This kind of a design echoes and is inspired by large undertakings like Infrastructure for Spatial Information in the European Community - INSPIRE (INSPIRE, 2018), Global Earth Observation System of Systems - GEOSS (GEOSS, 2018) and the US open data initiatives such as Geospatial Platform (GeoPlatform, 2018). The latter facility shares some of the system goals of OCIMS and provides an apt description, that of providing "shared and trusted geospatial data, services, and applications for use by the public and by government agencies and partners to meet their mission needs." GeoPlatform and OCIMS share aims of

- Providing access to authoritative data (wherever it may be stored) in support of informed decision making
- Presenting reusable applications and services for multiple stakeholders and users in the governmental and non-governmental spaces
- Acting as a launchpad from which national Oceans and Coastal issues can be informed by data from many spheres of society (government, academia, industry and communities)
- Potentially supplying some shared infrastructure that could host data and applications as a last resort

In essence, OCIMS harnesses international good practice for Spatial Data Infrastructure implementation to build out a national Oceans and Coasts IT system. This is an important point; previous South African Oceans and Coastal IT systems tended to be standalone, largely non-interoperable and difficult to extend via reusable applications. The driving force behind this change is a long term funding plan and political willingness to support OCIMS.

2.2 System Layering

OCIMS can be understood through three layers of functionality and components, named as the Interaction Layer, the Production and Mediation Layer and the Acquisition Layer.

2.2.1 Interaction Layer

Concerned with how the system facilitates the interaction between consumers/ end-users and OCIMS, with capabilities for:

- *Search and discovery* - pursuit of relevant information across collections of various types of documents, data, metadata, messages and other sources; queries are executed via word and phrase search, topic search, structured search over defined fields or facets, spatial search and temporal search.
- *Access* - transfer of data and information from system to system or system to user; may be performed through direct link to a resource, mediated via a service or represented via a portrayal (e.g. a map or graph).
- *Visualisation* - making data and information more comprehensible via visible representations of it; aids in understanding and exploring data, conveying of concepts and summarising of complexities.
- *Query* - perform guided or ad-hoc requests for data or information, often using a query language (e.g. SQL).
- *Publish / Register / Describe* - allow clients to produce data, metadata, information products and tools onto the system
- *Decision-making* - support users in reaching conclusions, make decisions or pursuing a course of action based on data, information, visualisations or reports from the system.
- *Dissemination* - promulgation/ spreading of information, data or reports to an audience; may be broadcast or targeted.

In OCIMS, the software supplying much of this functionality is CKAN (CKAN, 2018) and its dependencies, such as pycsw (pycsw Development Team, 2018) and Solr (Apache Solr, 2018). In effect, the OCIMS Core is largely a CKAN powered data portal for the Oceans and Coastal domain. Wordpress (Wordpress, 2018) provides content management functionality and overall styling. MapStore2 (Mapstore2, 2018) is deployed as the primary means for users to visualise and explore the large amounts of spatial data linked into OCIMS.

The initial set of Decision-Support Tools provided by OCIMS currently utilise a multitude of open-source software components, one key reason being that these components often make effective use of standardised programming interfaces and data encodings, a crucial requirement in distributed interoperable systems.

2.2.2 Production and Mediation Layer

Concerned with general or specific capabilities for processing data and information and may include facilities for discovery of, management/configuration of, invocation of, and result retrieval from:

- Models.
- Simulations.
- Algorithms and Product Processing Chains.
- (Meta)Data Accumulations.
- Mediation services - e.g. data transforming or data cascade/proxying.
- Harmonisation processes - e.g. combining data from different sources into integrated and consistent information products.
- Generation services - e.g. event processing and notification services, reports.

Interaction layer components would typically interface with the components in this layer over a network. To allow this to occur, production and mediation components ideally need to agree to a technical contract with OCIMS that requires the provision of a navigable endpoint for accessing the components and output from this layer; this can take the form of a service endpoint, for example, an OGC W*S service (OGC, 2018), an ESRI ArcGIS Server Map/Feature service (ESRI, 2018) or an OPEnDAP service (OPeNDAP, 2018), or an Application Programming Interface exposing these components and outputs as resources. Such endpoints should preferably conform to open standard protocols or provide detailed data models (e.g. OGC Conceptual Models (OGC, 2018)) and API documentation. It may be the case that components or outputs are not originally designed in this way, or that the component is not meant to share all its data or that an organisation does not have the capability to deploy services or API's; in such situations it may be appropriate to proxy or wrap these components into the service or API endpoint style.

OCIMS itself, and several of the providers to the system, extensively utilise open-source components in this layer. PostGIS (PostGIS, 2018) is usually the primary spatial database, while Geoserver (Geoserver, 2018), ArcGIS Server (ESRI, 2018) and THREDDS Data Server (THREDDS, 2018) provision the web services components. Open source processing components feature strongly in processing chains and algorithms, including GDAL (GDAL, 2018), NetCDF (NetCDF, 2018) and HDF (HDF, 2018) libraries, NASA SEADAS (SeaDAS, 2018) and ESA Sentinel Application Platform (SNAP, 2018). The Python (Python, 2018) programming language is primarily used to develop components and to orchestrate them.

2.2.3 Acquisition Layer

This layer is concerned with the sourcing of data and information products from Databases, Files, Data Streams e.g from sensors or sensor networks, Direct Readout Services, Data Services, and Download Services. The term "sourcing" refers to acquiring access, connecting to the data of information product offering and orchestrating a once-off or continuous query, download or readout of the offering. Further steps of extracting the product, translating it (perhaps into a common data model), and loading it into another system (e.g. an enterprise database) may also be considered as concerns of this layer of the architecture.

OCIMS acquires significant amounts of its data from NASA and ESA/ EUMETSAT remote sensing satellite programmes, but there are numerous other data sources, ranging from vessel position data streams to sensor networks, feature databases (e.g. boundary data) to social media. Each data source can represent integration challenges, and our experience has shown that FOSS tools are crucial in allowing the diversity of data sources to be accessed, translated and utilised effectively..

2.2.4 Layered Architecture benefits

If various components or sub-systems of OCIMS can be described in this manner, a few key insights are available.

Firstly, OCIMS functionality can be decomposed into smaller units of computation, which can be distributed. It is not necessary to build complete vertical or 'stovepiped' solutions, when functionality can be accessed over networks through well known interfaces serving well understood data.

Flowing from this, is the understanding that the development, deployment and management of components can be devolved to decoupled teams within the same or from different organisations. This allows for some efficiencies to be derived from the independence of teams - each can choose the most appropriate technologies - such as FOSS Geospatial tools - and deployment practices in their domain, for example.

These insights echo some of the rationale behind Service-Oriented Architecture and Microservices approaches to building complex systems. Concretely though, these insights encourage the use of innovative software deployment techniques as will be illustrated in section 3.

2.3 System-of-Systems Integration

2.3.1 Service Oriented Architecture

Many of the components in OCIMS will be invoked - that is, a function or method exposed by a component will be called by another component across the Internet; a distributed computing approach known as Remote Procedure Call (RPC). This invocation of components can be done with direct, parameterised calls to component functions typically using HTTP methods and eXtensible Markup Language (XML), a process known as XML-RPC - or, if using JSON, JSON-RPC. Alternatively, remote procedures can be invoked with structured messaging techniques such as Simple Object Access Protocol (SOAP) that encapsulate a protocol for describing the function that needs to be called and the structure of the data payload necessary to parameterise the function. This RPC pattern holds true for data access (particularly for components that provide multi-temporal, multi-dimensional data) but also for access to processing functionality and information dissemination functionality. The OGC Web Services Common standard defines how geospatial data and services can be invoked in these ways (Whiteside and Greenwood, 2010), via remote methods such as "GetCapabilities", "DescribeCoverage" or "GetMap", for example.

Service Oriented Architecture (SOA) is a style of distributed computing architecture that structures computational elements as a set of services that can be accessed and used from a service endpoint (usually an Internet address). Services are interacted with via service interfaces where information objects conforming to well described schemata are interchanged. SOA with a Remote Procedure Call (RPC) style has been, and remains, prevalent in working with spatio-temporal data services (and more generally science data) over the Web. This style has been formalised in various information communities, resulting in the benefit of a wide ecosystem of off-the-shelf open source and proprietary software clients and servers that can be utilised to build highly functional, robust service and application offerings against well established and accepted international standards (such as Open Geospatial Consortium web service standards, using highly structured information models).

2.3.2 Web Application Interface Architecture

In contrast, many modern Web based applications and services are constructed according to architectural styles that do not mimic procedural calls. These styles are often referred to as "RESTful" architectures. In the purest form, Web applications that are built using this RESTful style are concerned with navigating between and altering the state of resources (of a particular media type) at a Web address, using the standardised HTTP or Web protocol interface methods and message structures. This style is data-driven rather than control-driven since the application is formed from a sequence of state changes over resources, rather than calls to procedures or methods that opaquely control what happens to resources. In practice, many systems exhibit incomplete "RESTfulness" but are nevertheless distinct enough from SOA-RPC to be considered as Resource Oriented Architectures, characterised by having a Web Application Programming Interface (API). Web API implementations tend to allow for client applications to create-read-update-delete resources on the Web through a well defined set of possible interactions, often using customised text based (e.g. JSON) data structures and control instructions. Developers may be attracted to this style since it appears to be very clean and direct;

2.3.3 Integration Approaches

It is expected as a minimum that both systems exhibiting SOA-RPC and Web API styles will be joined to the OCIMS and their functions and resources utilised in a compositional way, i.e. an application may be constructed from parts of systems. As an example, Application A may source maps from a SOA-RPC service from System X and data feeds from a Web API at System Y. Application B may source maps from System X and data feeds from a Web API at System Z. This allows reuse of services and resources and is an architectural principle to be aimed for in integrating systems into the OCIMS. Reuse and composition are possible through the existence of detailed, precise and preferably standardised interface definitions, whether provided by documentation or by registration of the service endpoint and interface in a service repository, and agreement on shared media types.

Systems that exhibit these patterns from the outset increase the likelihood of acceptable integration levels into OCIMS. Other systems of importance to the OCIMS may exhibit a more tightly coupled computational interaction approach.

In these cases, applications are delivered vertically - it is difficult, if not impossible to uncouple a client from a specific server and inversely, servers need to communicate with bespoke clients that specifically understand the server interface. In such cases, little reuse is possible without significant software development effort, since a uniform integration interface is not presented. This is not inherently problematic - the tight coupling may be a design feature to ensure security, opacity, performance and simplicity, but systems of this nature will be difficult to integrate into OCIMS.

At least two strategies exist to bring a particular vertical system into the ambit of the OCIMS; namely proxying and linking. Proxying will incur software development effort and is concerned with wrapping an interface around the parts of the system that could be reused, such that this interface could be understood by other components in the OCIMS environment. As an example, a server offering geospatial feature data may need to be proxied as an OGC Web Feature Service (WFS). Such a proxy would receive WFS requests from a client and translate them into custom requests to the server and return the server response to the client according to the WFS protocol. A further example of proxying that has been utilised in OCIMS is the use of Postgres Foreign Data Wrappers via Multicorn (Multicorn, 2018) to draw obscure data formats into the ambit of PostGIS, and thus to the reach of Geospatial Web Server software like Geoserver. Proxying (wrapping, brokering or cascading are other terms used to describe this effort) may be desirable to achieve integrations for certain systems, but is achieved at a cost. Linking, on the other hand, simply refers to the process of leaving OCIMS and interacting with another system separately. This represents a low level of integration and is acceptable only when there is no likelihood or need of reuse, or the system provider has no willingness to provide integration hooks, perhaps for security, political or other reasons.

3. OCIMS SOFTWARE DEVELOPMENT AND DEPLOYMENT PROCEDURES

In order to illustrate the layered architecture and deployment procedures briefly described in section 2.2., we discuss one major part of the OCIMS system, the OCIMS Core. The Core offers a centralised one-stop web portal where OCIMS resources and services are catalogued, sometimes cached, sometimes stored, but where a user can find, explore, launch/view, download or link to distributed resources and services relatively transparently. This process is orchestrated through a rich client user interface, which is responsible for composing and engaging with resources and services. A logical view of the OCIMS Core is provided in Figure 2.

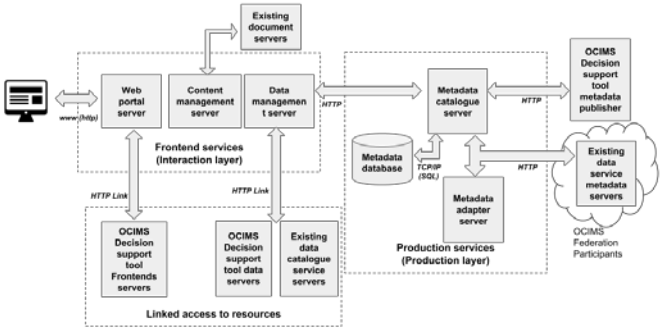

Figure 2. OCIMS Core logical view

The Core system is made up of five major components, a Data Management System, a Content Management System, a database, web servers and a search platform. Each of these components is abstracted from the other and runs inside its own virtual environment. Due to the large number of datasets anticipated, Comprehensive Knowledge Archive Network (CKAN) proved to be the best option in that it is widely used and powers big data hubs and data portals. It allows for easy data publication, discovery, sharing and accessibility (Amorim et al., 2017.), (Kucera, et al., 2012) and (Winn, et al., 2013). One major requirement which was met by CKAN was its adherence to accepted metadata standards and standardised web services. This allowed the core system to interface and exchange data with other service providers through harvesting. CKAN provides various ways in which datasets and metadata can be harvested from different data providers depending on the type of endpoint exposed, including:

- CKAN to CKAN harvesting via the API,
- DCAT harvesting via RDF or JSON,
- CSW through pycsw, data.json etc.

CKAN also provides Solr for indexing any harvested metadata and an API for searching through its catalogue. If the datasets and/or metadata are spatially tagged, CKAN also provides for map based searching on its interface.

In the initial phase of development, CKAN's front-end was customised and branded to meet the necessary look and feel. However, CKAN proved to be limited in meeting certain front-end functional requirements without embarking on significant code development. This prompted the need for a self contained, "user friendly" or intuitive Content Management System, qualities embodied in Wordpress along with its rich functionality and support. Both the IMS (CKAN) and the CMS (Wordpress) use different databases for information storage which are PostgreSQL and MySQL respectively.

The OCIMS core technology stack mentioned above is made up of independent subsystems that reside in their own virtual environment, and only interface with other subsystems through API's and HTTP protocols. These independent virtual environments are made possible via Docker (Docker, 2018), the increasingly popular containerisation software. Docker containers allow for automated deployment and management of applications or services in such a way as to isolate the application/ service from underlying software dependencies. This supports rollout and rollback of these applications or services on a granular level. This method of deployment allows for the decoupling of subsystems, isolation, simplicity and faster configuration.

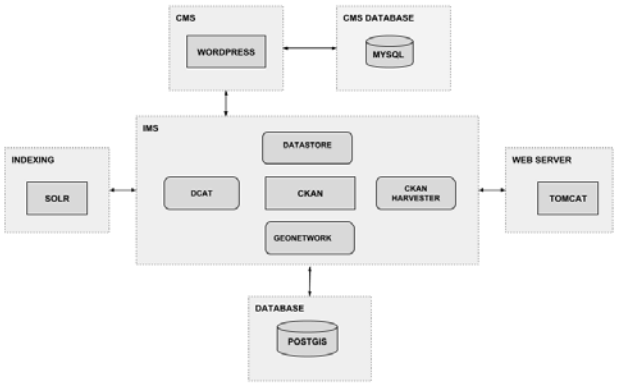

Figure 3. OCIMS Core Docker Containers

Since each Docker container or subsystem is independent, as shown in Figure 3., it can be modified, upgraded or replaced without affecting other components of the stack. It provides a standard method of deployment and portability that allows applications or services to be run transparently on different operating systems or cloud computing environments. Additionally, containers ensure that consistency across various release cycles and deployment stages can be attained. This means that short deployment cycles, high levels of reliability and effortless system duplication are possible.

In a similar vein, some of the Production and Mediation Layer software (described in Section 2) relies on Docker containerisation approaches to deliver robust, re-deployable processing chains that can be easily replaced as new requirements or algorithms emerge.

4. LESSONS AND INNOVATIONS

The groundwork done in building the initial OCIMS implementation as a set of granular and configurable containerised components has led to a significant positive side effect; a "recipe" now materialises for building interoperable data management system nodes that can readily federate with OCIMS or other systems, such as national spatial data infrastructures. This allows for different organisations to build specific parts of OCIMS at different tempos. Given that some organisations may not have the technical capabilities or capacity to construct nodes, it is also possible for the main implementors of OCIMS to offer assistance in deploying the necessary lighweight data management infrastructure shell into an organisation with minimal effort.

It is important to recognise that many of the organisations that may take part in OCIMS have existing data infrastructures. In South African circles, these are commonly based on implementations of various parts of the ESRI suite of software. It is our experience that some of these implementations are left relatively unconfigured or in a default state, with minimal attention paid to interoperability arrangements. Such implementations are somewhat problematic to integrate with CKAN and CSW directly. One solution is to proxy parts of these systems through the FOSS ESRI Geoportal software components (ESRI_b, 2018), configured and deployed as individual Docker containers specific to an organisation.

It is our observation that standardised discovery and usage metadata is often not supplied with OCIMS related datasets and data services. With some programming effort, it is plausible to populate metadata into datasets (such as NetCDF CF Convention elements) and services such as CSW from within scientific data processing chains. To the extent that this is possible for any given dataset, this approach will be pursued in OCIMS, for it lubricates the process of building interoperable solutions with the least hassle.

Much of the architecture of OCIMS is based on international standing good practice such as described by the GEOSS Architecture Implementation Pilots (Percivall, et al., 2015). We believe that much of this method of building interoperable systems remains relevant for the next few years. However, it is not necessarily the case that organisations that contribute to OCIMS will be willing to embark on or continue the processes

of building and maintaining their own systems in-house. The commodification of cloud computing and in particular the types of cloud service offerings available to organisations may change the ways scientific and geospatial data and information are stored, discovered, accessed and integrated into systems. We believe that for subsequent iterations of the OCIMS architecture it will be important to consider emerging approaches (COGEO, 2018), (Tandy, et al., 2017) to building distributed, interoperable geospatial systems in the cloud.

5. ACKNOWLEDGEMENTS

The authors wish to highlight that this article summarises a great deal of work undertaken (primarily by the authors themselves) in providing the OCIMS project with detailed architecture description documentation.

6. REFERENCES

Amorim, R., Aguiar Castro, J., Rocha, J. and Ribeiro, C. 2017. A comparison of research data management platforms: architecture, flexible metadata and interoperability. Universal Access in the Information Society, Volume 16, Issue 4, pp 851-862 . 10.1007/s10209-016-0475-y.

Apache Solr, 2018. Apache Software Foundation. http://lucene.apache.org/solr/ (17 April 2018)

CKAN, 2018. Comprehensive Knowledge Archive Network, CKAN Association https://ckan.org/ (17 April 2018)

COGEO, 2018. Cloud Optimized GeoTIFF project. https://www.cogeo.org (17 April 2018)

Docker, 2018. Docker Software Containerization Platform, Docker.Inc https://www.docker.com/ (17 April 2018)

ESRI, 2018. ArcGIS Enterprise Server: Release 10.6. Redlands, CA: Environmental Systems Research Institute. https://enterprise.arcgis.com/en/server/ (17 April 2018)

ESRI_b, 2018. ESRI Geoportal Server. Redlands, CA: Environmental Systems Research Institute. https://www.esri.com/en-us/arcgis/products/geoportal-server/overview (17 April 2018).

GDAL/OGR contributors, 2018. The GDAL/OGR Geospatial Data Abstraction Software Library http://gdal.org (17 April 2018)

GeoPlatform, 2017. Geospatial Platform http://www.geoplatform.gov/ (17 April 2018)

Geoserver, 2018. Geoserver - open source server for sharing geospatial data: version 2.12, Open Source Geospatial Foundation (OSGEO) http://geoserver.org. (17 April 2018)

GEOSS, 2018. Global Earth Observing System of Systems, Group on Earth Observations http://www.earthobservations.org/geoss.shtml (17 April 2018)

HDF, 2018. The HDF Group. Hierarchical Data Format, version 5, 1997-2018. http://www.hdfgroup.org/HDF5/ (17 April 2018)

INSPIRE knowledge base, 2018. Infrastructure for spatial information in Europe http://inspire.ec.europa.eu/ (17 April 2018)

Kucera, J., Chlapek, D. and Mynarz, J. 2012. Czech CKAN repository as case study in public sector data cataloging, Systemova Integrace, Volume 19, Issue 2, pp 95-107

MapStore2, 2018. Mapstore2, GeoSolutions https://mapstore2.geo-solutions.it/mapstore/ (17 April 2018)

Multicorn, 2018. Multicorn - Postgres Foreign Data Wrapper Development for Python. https://multicorn.org/. (17 April 2018)

MySQL, 2018. MySQL, Oracle Corporation https://www.mysql.com/ (17 April 2018)

NetCDF, 2018. Unidata (2018), Network Common Data Form (netCDF) version 4.4.0.2 [software]. Boulder, CO: UCAR/Unidata. (http://doi.org/10.5065/D6H70CW6) (17 April 2018)

OCIMS, 2018. Oceans and Coasts Information Management System https://www.ocims.gov.za (17 April 2018)

OGC, 2018. OGC Standards and Supporting Documents, Open Geospatial Consortium, http://www.opengeospatial.org/standards (17 April 2018)

OPeNDAP, 2018. OPeNDAP - Advanced Software for Remote Data Retrieval, OPeNDAP Inc. 165 Dean Knauss Dr., Narragansett, RI 02882. https://www.opendap.org/ (April 17, 2018)

Percivall, G. de Lathouwer, B., Nebert, D. and Alameh, N. (eds), 2015. GEOSS AIP Architecture. Group on Earth Observations, 7 bis, avenue de la Paix, Case postale 2300, CH-1211 Geneva, Switzerland https://www.earthobservations.org/documents/cfp/201501_geoss_cfp_aip8.pdf

PostGIS, 2018. PostGIS - Support for geographic objects to the PostgreSQL object-relational database: version 2.4.2., Refractions Research http://postgis.refractions.net. (17 April 2018)

PostgreSQL, 2018. PostgreSQL Global Development Group https://www.postgresql.org (17 April 2018)

pycsw Development Team, 2018. pycsw http://pycsw.org/ (17 April 2018)

Python, 2018. Python Software Foundation. Python Language Reference, version 2.7. http://www.python.org (17 April 2018)

SeaDAS, 2018. NASA Ocean Biology Processing Group, SeaDAS v7.5 https://seadas.gsfc.nasa.gov/ (17 April 2018)

SNAP, 2018. SNAP - ESA Sentinel Application Platform v6.0.0, http://step.esa.int (17 April 2018)

Tandy, J., van den Brink, L. and Barnaghi, P. (eds), 2017. Spatial Data on the Web Best Practices, W3C Working Group Note. https://www.w3.org/TR/sdw-bp/ (17 April 2018)

THREDDS, 2018. Unidata, (2015): THREDDS Data Server (TDS) [software].Boulder, CO: UCAR/Unidata. (http://doi.org/10.5065/D6N014KG) (17 April 2018)

Whiteside, A., Greenwood, J. (eds.) 2010. OGC Web Services Common Standard, OGC 06-121r9, Version 2.0, Open Geospatial Consortium, Inc. http://portal.opengeospatial.org/files/?artifact_id=38867

Winn, J., et al., 2013. Open data and the academy: An evaluation of CKAN for research data management. Univ. Lincoln, Lincoln, U.K.

Wordpress, 2018. Wordpress.org, https://wordpress.org/ (17 April 2018)