21st International Congress on Modelling and Simulation, Gold Coast, Australia, 29 Nov to 4 Dec 2015
www.mssanz.org.au/modsim2015

Tools for enabling rapid deployment of water and energy consumption and supply data services

Yu, J. [a], **Leighton, B.** [a], **Mirza, F.** [a] **and Singh, R.** [b]

[a] *CSIRO Land and Water Flagship, Graham Road, Highett, Victoria*
[b] *CSIRO Land and Water Flagship, GPO Box 1666, Canberra ACT 2601*
Email: jonathan.yu@csiro.au

Abstract: Improving data consistency and facilitating opportunities for cross-domain urban research is important to understanding patterns of urban development and modelling urban growth for a sustainable future. Researchers and policy analysts are confronted with a number of technical challenges when comparing data in the water and energy industries both within and across these sectors. These include variability in data formats, spatio-temporal granularity, access methods and differing semantic definitions. These hinder attempts to access, analyse, interpret, compare and combine the respective water and energy datasets. The AURIN Lens 6/2 project has developed the Water and Energy Consumption and Supply (WESC) data protocol for encoding WESC datasets and tools for consolidated access via standardised geospatial web services. In addition, a number of data agreements across Australian utilities and peak body organisations have developed in order to establish a thematic data hub, called the WESC data hub. The WESC data hub currently hosts relevant datasets and uses the WESC data protocol to provide access to them via the AURIN data portal. In developing the WESC data hub, a number of tools and techniques have been developed to streamline and automate the data extract-transform-load and web service deployment process for each data provider as well as set up continuous testing frameworks to provide quality assurance and performance monitoring of the WESC data hub. In this paper, we will present the technical challenges in delivering the WESC data hub across multiple datasets and data providers, as well as the tools and methodologies used to rapidly deliver data services and its testing framework - namely, Docker, Linux virtual containers, Python and TeamCity. Previously, such deployments required considerable expertise in systems administration, configuration management, and web application development, were time-consuming and not easily repeatable. However, the tools and techniques presented in this paper provides a generic pattern for rapidly developing and deploying standardised geospatial web services and is thus applicable more broadly for other domains and data. The methodology used provides software infrastructure resilience as the data services can be re-deployed easily using Docker. Continuous build and testing tools, such as TeamCity, provide quality assurance as data services are deployed and re-deployed to ensure data integrity as well as performance monitoring which allows the systems to be fine-tuned and made more efficient.

Keywords: *Data services, web services, geospatial data, Deployment technologies, Docker*

1. INTRODUCTION

Understanding urban settlements and patterns of resource consumption in the past, present and into the future can be greatly enhanced with access to consistent and reliable water and energy data. However, comparing and contrasting data in the energy and/or the water sectors presents a number of challenges. Across the water and energy sector similar data is collected on supply and consumption of utilities: water, electricity and gas. While water and energy data is similar it is often represented differently by different agencies collecting and managing the data. Providing a consistent representation of this data along with metadata that provides context and meaning for the data opens up many research possibilities. Therefore, a common data framework (including an information platform) and negotiated access to key water and energy dataset are important to facilitate evaluation, reporting, analysis and scenario planning towards sustainable, resilient and liveable communities.

Partnering with the Australian Urban Research Infrastructure Network initiative (AURIN), CSIRO has developed a nationally applicable data protocol for the recording, storage, access and interoperability of supply and consumption datasets for the energy and water sector. AURIN is a Federal Government initiative that is providing researchers, designers and planners with access to information services and online tools to understand patterns of urban development and to model urban growth for a sustainable future (Sinnott et al. 2012, Sinnott et al. 2015). CSIRO has developed an information platform, the Water and Energy Supply and Consumption (WESC) data hub, to allow access to water and energy datasets to urban researchers, policy makers and managers through the AURIN Lens 6/2 Project (L6/2) (Yu et al. 2015). Working with a number of energy and water service providers across the nation, CSIRO is facilitating data access of water and energy datasets to researchers, government agencies and interested community groups access via the AURIN portal for research and policy analysis. The CSIRO WESC data hub is currently hosting and delivering data contributed by data providers from the major capital cities and a selection of regional centres in Victoria, NSW, ACT and Queensland. The data is delivered out via geospatial web services in a consistent format using the WESC Markup Language (WESCML)[1].

Due to the nature of the project and the multiple organisations involved, here are many challenges in producing a consistent and rich representation of data across the many sources. Each organisation may represent the data differently. For interoperability, these sources require data harmonisation. A web service can deliver data in a standard form. A collection of such services conforming to a common standard can deliver federated data. Queries can be performed across services and the data can be treated as a whole. This provides ready ability to understand data across artificial boundaries imposed by different utility companies or data providers.

The delivery of standardised data through standards-based web data services is well established e.g. OneGeology and GeoSciML (Laxton et al. 2010), GEOSS (GEO 2005), and Auscope (Woodcock et al. 2010). However the configuration of services, Extract-Transform-and-Load (ETL) of data into a common format and maintenance and deployment of the components in the information system stack is time consuming, sometimes prohibitively so. A major barrier for the deployment of distributed orchestrated federated instances of standard web services is the time and expertise required to setup, populate, deploy and maintain these services.

In this paper, we present an approach developed to compose and deploy a configurable spatial information system using automated and well-documented processes. We apply modern DevOps approaches, which are discussed in Section 2, for deploying energy and water data for the AURIN L6/2 project. In Section 3, we present the technical challenges in the AURIN L6/2 project and the tools, approaches and processes used to overcome those challenges, including ETL tools written in Python, continuous integration, and "infrastructure as code" using Docker a means of creating Linux containers. These significantly reduce the effort, expertise and time required to deliver adapt and deploy services. The tools and techniques presented in this paper provides a generic pattern for rapidly developing and deploying standardised geospatial web services and is thus applicable more broadly for other domains and data.

2. DevOps

Reliably deploying developed software systems into operations requires processes that are reproducible and programmatic (Loukides, 2012). Developing such systems is the domain of DevOps. Traditionally many manual steps would be required to deploy an information services stack and populate it with data. Time is spent manually executing commands and configuration files. Knowledge of the deployment process must be

[1] See http://wescml.org

documented, communicated, and understood between team members. Manual processes are fragile and there is a significant risk of mistakes, documentation that is not meticulously maintained and checked risks being incorrect or incomplete, and learning and communicating the details of manual processes is an additional overhead. Manual processes consume significant resources, tend to degrade over time, be less scalable and portable and are high risk due to human error. A key DevOps principle is that infrastructures should be completely described using code. A common epithet of DevOps is "infrastructure as code" meaning that, as much as is possible, infrastructure should be a by-product of executing code that creates, configures and maintains that infrastructure.

Scientific software systems benefit from DevOps practices and tools. Economou et al. (2014) review DevOps in the astronomy community and briefly report on deploying large scale cloud processing at the National Optical Astronomy Observatory using reproducible and infrastructure-as-code approaches. Closely related are scientific high performance compute infrastructures and cloud systems, all of which are areas of current research. Wang et al. (2008) review a range of scientific cloud infrastructure, many of which use virtual machine images as the basis for reproducible deployments, expanding capacity and deploying customised software systems.

DevOps overlaps with and is an extension of Software Configuration Management, which is the practice of reliably and repeatedly developing, configuring, and deploying software systems (Estublier, 2000). DevOps places an emphasis on the deployment of software systems into their penultimate environment, that is, an environment before into operations or production. Software Configuration Management tools are key to DevOps processes and are rapidly evolving in response to DevOps principles and needs. Puppet[2] and Chef[3] are configuration management tools widely used. Puppet and Chef setup software and systems infrastructure by executing code that describes the infrastructure. Typically, Puppet and Chef run agents on foundation physical or virtual machines. These agents configure the machines they run on according to instructions provided from a master service. Different agents may use different parts of the overall code. Furthermore the code may then be customized and form part of a broader system.

A new approach emerging in configuration management is *Containerisation*. Containerisation is the process of building, using and managing containers. Containers are conceptually similar to virtual machines. Containers appear as independent operating systems but actually run within a host operating system. Containers differ from virtual machines in that they are faster to build and run, and do not depend on true virtualisation of underlying hardware. Rather containers exploit the ability of the underlying operating system to partition and isolate system resources at the host operating system level. Most containerisation implementations are Linux-based. Running containers, like virtual machines, can be saved as images. A saved image can then be used to create one, or many, containers with the same specifications container from which it originated. In this way containers can be used to reproducibly create software systems. Docker[4] is one popular containerisation system. In addition to implementing containers and images Docker provides a minimalist and flexible language for describing how images should be built. Thus Docker provides reproducible and programmatic configuration of systems.

3. WESC DATA SERVICES DEVELOPMENT AND DEPLOYMENT

The WESC data hub features a set of services (mainly Web Feature Services using Geoserver implementations) through which the AURIN portal is able to access water and energy supply and consumption data using Web Feature Service client library via authenticated protocols as outlined in the AURIN technical architecture (Sinnott 2012). Figure 1 below shows the components to enable a framework for deploying a standardised WESC information stack. The WESC data standard provides the information model, XML schema implementation and controlled vocabularies used to encode the data and deploy them via Web Feature Service GML encoded data service. The WESC data standard is available online at http://wescml.org/).

This section highlights the processes and tools developed for streamlining the deployment of the data. Section 3.1 presents details around the technical challenges of deploying multiple data services for data custodians. Section 3.2 outlines the data services deployment processes. Section 3.3 discusses continuous testing put in place to ensure the integrity of the development and production deployment environments, which provides a useful real-time feedback tool for the developers and system administrators.

[2] See https://puppetlabs.com/ (Accessed 30 July 2015)
[3] See https://www.chef.io (Accessed 30 July 2015)
[4] See https://www.docker.com (Accessed 30 July 2015)

3.1. Technical challenges

The challenges in building the WESC Data Hub, aside from data licensing, were harmonising multiple datasets from different data custodians, acknowledging governance arrangements for the different data custodians, and being able to efficiently develop and deploy the WESC information services stack for each data provider that is conformant with the WESC data standard. A total of 16 data custodians from across Australia were engaged and agreed to provide data for the purposes of the AURIN L6/2 project - 10 water and 6 energy data custodians. These included water authorities, electricity distributors, peak industry associations, and federal government agencies. They also varied in the regions which they have data holdings, as well as the granularity of the spatial regions they have data, e.g. Yarra Valley Water had water consumption data for 2010-2014 at postcode level, Sydney Water provided data for 2006-2013 at the LGA scale and the Energy Supply Association of Australia provided data from 1995-2014 at the state level.

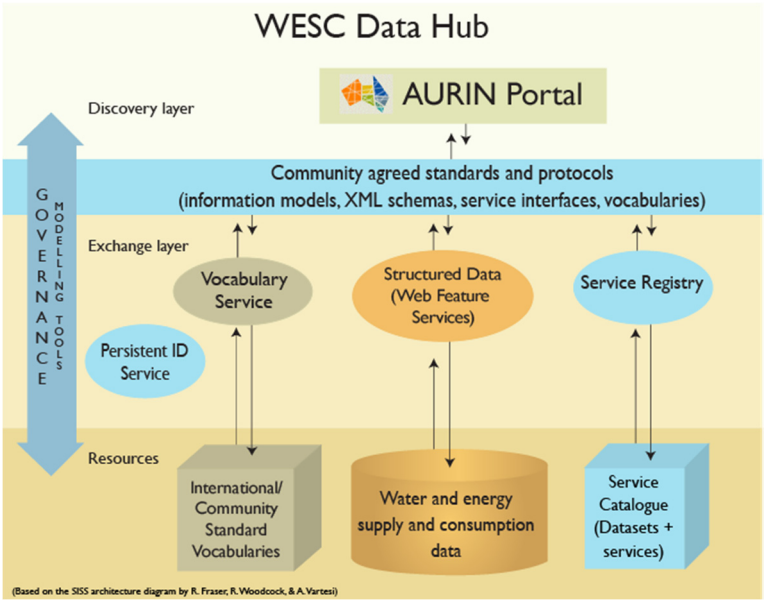

Figure 1. The WESC Data Hub

Each data provider encodes their data in different formats and use different terminologies for the categories of data. In addition to this, some data providers implicitly use financial year to denote the time period, while others use calendar year. These issues were not always declared upfront thus highlighting the need for agile and iterative development and deployment processes. There were also different geographic boundaries and classifications used for each data provider. This meant that each WESC information service stack is required to reference and encode the appropriate geometries for each deployment.

3.2. WESC Data Services Deployment Process

Given the nature of the challenges faced in this project and a short time frame of less than 12 months for deploying all 16 information services stacks (at times requiring redeployment for data corrections), processes were developed to enable consistent and repeatable deployment of WESC information service stacks in order to flexibly develop and deploy versions of a given data service stack. With the governance of each data custodian in mind, separate respective instantiations of the WESC information service stack were deployed.

The WESC data services deployment process involve several components and versioned repositories in order to compose and deploy a single WESC information service stack. Figure 2 gives shows the build and deployment process and the WESC information services stack components. The main components include:

- *Git repositories* (shown in Figure 2 on the left) - source code, files and configurations required for building a WESC information services stack is version-controlled in the Git repositories, including Dockerfile code for installing and configuring services, and running Python ETL code, Python code and for performing ETL, and raw data files from data providers containing electricity, gas and water supply and consumption data used as the input for the Python ETL.
- *Docker build process* (shown in Figure 2 in the middle) - for implementing the build process which involves cloning selected git repositories and then executing the Dockerfile. As the Dockerfile executes it clones further git repositories inside the container and uses Python code and data inside these to populate services.

- *WESC information service stack* (shown in Figure 2 on the right) - which is configured, deployed and populated with data upon completion of the build process. Each WESC stack consists of an Apache web server and a Geoserver application server to deliver data through HTTPS, and a PostGIS database as the backend for Geoserver to store geospatial WESC data.

Each WESC stack is created using a Docker configuration file, which is a machine-readable file that captures dependencies and instructions for building a Docker container with the requisite components in the stack and their configuration (including authentication and WESC data standard configurations). This provides means to customising, deploying and populating the Apache-GeoServer-PostGIS system consistently across datasets and in a repeatable fashion.

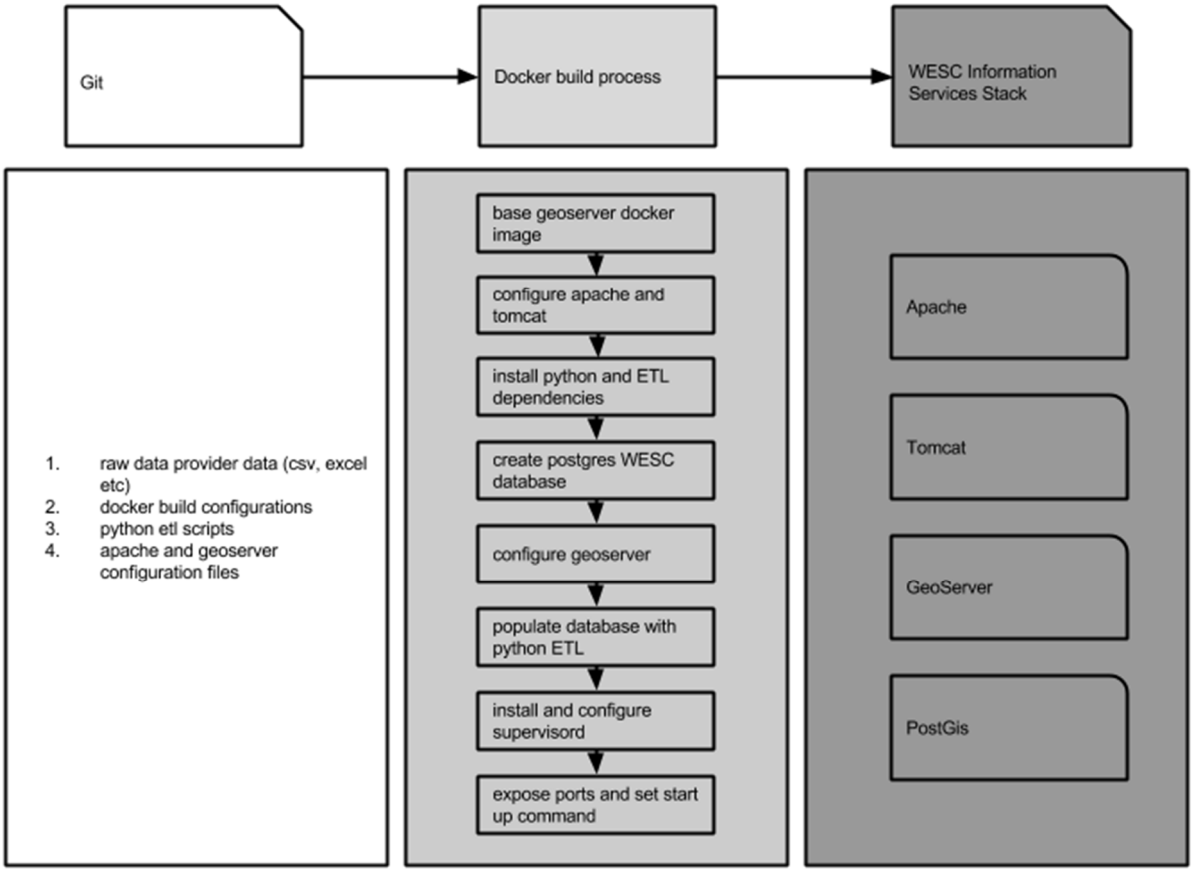

Figure 2. Overview of the deployment process for a given WESC Information Services Stack.

Python is used as part of the Docker build process to perform an ETL process to extract energy and water supply and consumption data from the raw data sources, transform this data to a common WESC database schema, and load the data into a PostGIS instance. Delivering data from a new data provider involves creating a new Python ETL script, configuring a script to point at the new Python ETL and running the existing Docker build process. Using deployment technologies such as Docker allowed the configuration, instantiation and deployment of each WESC web service stack to be largely automated and self-contained. In most cases, the ETL process is automated as a particular CSV data format was used. In other cases, some pre-processing was required, which involved manual inspection and verification with the data custodian to ensure data quality on both our information systems and the data custodian's records.

3.3. Continuous integration and testing

In order to ensure that each WESC information service stack conforms to the WESC data standard, is delivering data for each data custodian with the right spatial features, and is performing efficiently, a continuous testing framework is necessary. For this project, TeamCity was used to implement continuous build and testing of the WESC information service stacks. TeamCity is a continuous integration server which can automate the task of unit testing and build management and can be configured to run at scheduled intervals and/or based on some trigger events, such as, revision control system commits. Some of the unit tests for the services use the metadata about the geospatial features as inputs, such as, the expected number of

features returned for a particular bounding box. The metadata used for the tests are also version-controlled in a Git repository and is updated to match a given dataset.

A unit testing approach was taken to test the information services. Instead of having the user or developer write unit tests for each individual service, code templating techniques were employed to generate unit tests dynamically based on the metadata associated with each service. The templating code, written in Python is integrated with a TeamCity build, but could be run standalone if needed. For the AURIN L6/2 project setup, two servers were used, one for hosting the services under development and other for hosting services in a production environment. To test both the services under development and in production, two different test configurations were defined in TeamCity as shown in Figure 3.

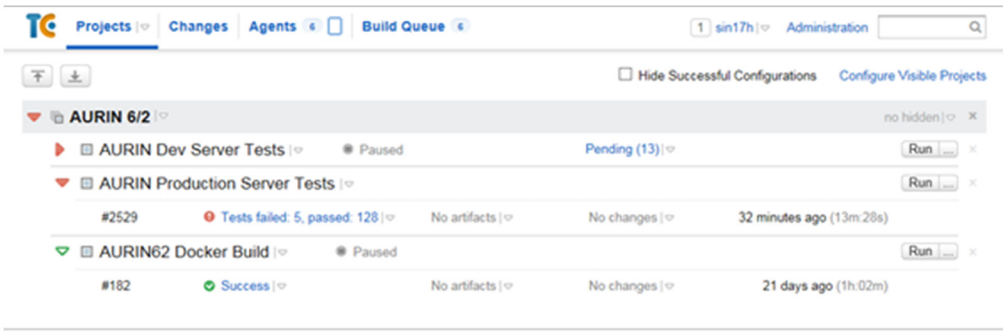

Figure 3. TeamCity AURIN L6/2 configurations.

Each TeamCity test configuration was scheduled at 30-minute intervals and executed the unit tests generated from the metadata. TeamCity would notify the project team when tests for a particular service failed or if the service was down via email as well as via the TeamCity web interface. The advantage of this regular testing at 30-minute intervals is that it gave the team the ability to monitor the deployed services on both development and production environments and respond to services doing down in a timely manner. TeamCity reports the details about the service tests which failed including meaningful assertion messages encoded in the unit tests along with any source code changes that went into this build run so that changes responsible can be inspected. TeamCity reports test duration (see Figure 4) and allows users to view trends over the build history, which can be used to fine tune services whose performance have degraded over time.

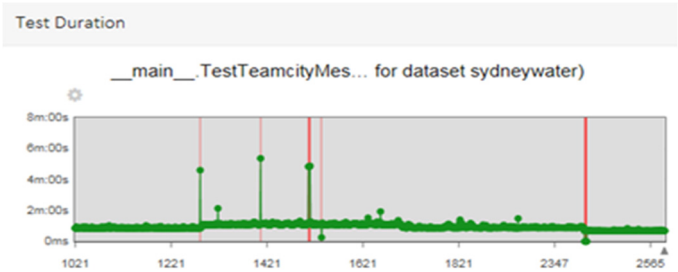

Figure 4. Screenshot of a TeamCity report showing test duration trend for a test run.

4. DISCUSSION

The tooling and methods introduced in this paper streamlines the deployment of information services stacks. Such information infrastructures would previously require considerable expertise in systems administration, configuration management, and web application development to be able to deploy a single instance let alone 16 instances as required in this project. In addition, the shared knowledge between team members would often be isolated and may differ from one person to another. Instructions for deployment may be maintained on separately managed documents and require additional effort to maintain. In contrast, the approach presented in this paper allows the configuration management and deployment documentation to be captured within the codebase itself. During the entire development and deployment process, not only is the source code used for the ETL process versioned using Git, but the configuration and deployment steps captured in Docker configuration files, and metadata and code used for testing (via TeamCity) are also version-controlled. This allowed the project team to implement changes and improvements in a methodical manner to build on previous versions and made available to the entire project team. Thus, reducing the effort and amount of knowledge of the underlying system required to redeploy a given data service stack or add new data services for additional datasets.

The tools and techniques presented in this paper provides a generic pattern for rapidly developing and deploying standardised information services stacks (geospatial and non-geospatial) and is thus applicable more broadly to other domains and data. The approach presented in this paper also provides software infrastructure resilience as the data services can be re-deployed easily using Docker. Continuous build and testing tools, such as TeamCity, provide quality assurance as data services are deployed and re-deployed to fine-tune and improve efficiency, ensure data integrity, and monitor performance.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce tools, processes, and components to efficiently and consistently develop, deploy, and test data services for the WESC data hub as part of the AURIN L6/2 project. Recognising the technical challenges in delivering multiple datasets for many data custodians, the tools and processes used allowed rapid development and deployment of information services stacks configured to deliver data according to the WESC data standard. Using DevOps tools - namely, Docker, Python and TeamCity, the project team were able to address the technical concerns and made deployment of 16 WESC information services stacks feasible in a short time frame. Using Docker files to specify the 'infrastructure-as-code' allowed the deployment process to be iterative, repeatable, adaptive and version-controlled.

The approach we developed enabled efficient and reliable deployment of spatial information services stacks delivering geo-located energy and water supply and consumption data. Continuous integration and testing is used to validate expected behaviour of the systems and reported when errors were made in configuring an information service stack or if services goes down. The benefit of this approach is the lowered effort, expertise and time required to adapt and redeploy such information services stacks. The monitoring of services allowed timely notifications during development and maintenance phases. This enabled greater agility and capacity to deliver the multiple spatial information services stacks efficiently and ensure services adhered to the WESC framework. The approach presented in this paper is a generic pattern for methodical, repeatable, consistent, testable, adaptable development and deployment of information services stacks for a broad range of domains, which will be explored in future work.

REFERENCES

Economou, F., Hoblitt, J. C., & Norris, P. (2014). Your data is your dogfood: DevOps in the astronomical observatory, *Instrumentation and Methods for Astrophysics; Software Engineering,* 7 pp. Retrieved from http://arxiv.org/abs/1407.6463 (Accessed July 17, 2015).

Estublier, J. (2000). Software Configuration Management: A Roadmap, Paper presented at Proc. Intl. Conf. of the future of Software Engineering (ICSE), pp. 279-289, June 4-11, Limrick, Ireland, doi:10.1145/336512.336576

GEO 2005, Global Earth Observation System of Systems (GEOSS): 10-year Implementation Plan Reference Document. ESA Publications Division, ESTEC, 2005.

Loukides, M. (2012). What is DevOps? O'Reilly Radar, http://radar.oreilly.com/2012/06/what-is-devops.html (Accessed July 17, 2015).

Laxton, J., Serrano, J. J., & Tellez-Arenas, A. (2010). Geological applications using geospatial standards–an example from OneGeology-Europe and GeoSciML. *Intl. Journal of Digital Earth*, 3(S1), pp. 31-49.

Sinnott, R.O., Bayliss, C., et al. (2012), A data-driven urban research environment for Australia, Paper presented at Proc. IEEE 8th Intl. Conf. on E-Science, vol., no., pp.1-8, Oct. 8-12, doi: 10.1109/eScience.2012.6404481

Sinnott R. O., Bayliss C., et al. (2015), The Australian urban research gateway, *Concurrency Computat.: Pract. Exper.*, 27, pp. 358–375, doi: 10.1002/cpe.3282

Woodcock, R., Simons, B., Duclaux, G., and Cox, S.J.D (2010). AuScope's use of standards to deliver earth resource data, In EGU General Assembly Conference Abstracts, vol. 12, p. 1556.

Yu, J., Inman, M. and Simons, B. (2015). Protocols to integrate urban water data with energy and other sectors within AURIN, Paper presented at Ozwater'15, Adelaide, Australia, May 12-15. pp. 1-8.