Standardized Big Data Processing in Hybrid Clouds

Ingo Simonis
Open Geospatial Consortium, OGC, 236 Gray's Inn Road, London, U.K.

Keywords: Big Data, Standards, Spatial Data Infrastructure, OGC, Cloud, Container, Docker.

Abstract: There is a growing number of easily accessible Big Data repositories hosted on cloud infrastructures that offer additional sets of cloud-based products such as compute, storage, database, or analytics services. The Sentinel-2 earth observation satellite data available via Amazon S3 is a good example of a petabyte-sized data repository in a rich cloud environment. The combination of hosted data and co-located cloud services is a key enabler for efficient Big Data processing. When the transport of large amounts of data is not feasible or cost efficient, processes need to be shipped and executed as closely as possible to the actual data. This paper describes standardization efforts to build an architecture featuring high levels of interoperability for provisioning, registration, deployment, and execution of arbitrary applications in cloud environments. Based on virtualization mechanisms and containerization technology, the standardized approach allows to pack any type of application or multi-application based workflow into a container that can be dynamically deployed on any type of cloud environment. Consumers can discover these containers, provide the necessary parameterization and execute them online even easier than on their local machines, because no software installation, data download, or complex configuration is necessary.

1 INTRODUCTION

Environmental sciences are witnessing a rapid increase in the amount of available data. They follow the general trend of data generated and shared by businesses, public administrations, numerous industrial and not-to-profit sectors, and scientific research that has increased immeasurably (Sivarajah et al, 2016). A large share of these data is the result of environmental monitoring, either in situ or via remote sensing (Vitolo et al, 2015). At the same time, Big Data Analytics is increasingly becoming a trending practice that many organizations are adopting with the purpose of constructing valuable information from Big Data (Sivarajah et al, 2016).

Data providers are looking for new opportunities to realize revenue with data, or at least to minimize local data storage and maintenance costs by outsourcing elementary services such as storage, cataloguing, or data access to external partners (Sookhak et al, 2017). These constellations led to new partnerships with commercial cloud operators serving earth observation data that was traditionally offered by e.g. space agencies.

A good example is the Sentinel-2 earth observation satellite data, a petabyte-sized data repository. The original data is produced by the European Space Agency (ESA) as part of the Copernicus mission, but storage, maintenance, and access are currently available via Amazon S3 (Amazon, 2018).

There is a growing number of easily accessible Big Data repositories hosted on cloud infrastructures that offer additional sets of cloud-based products such as compute, storage, database, or analytics services. The combination of hosted data and co-located cloud services is a key enabler for efficient Big Data processing. When the transport of large amounts of data is not feasible or cost efficient, processes need to be shipped and executed as closely as possible to the actual data. This paradigm, though postulated earlier, was a complex endeavour before container technologies such as Docker became popular. Locally developed software couldn't be shipped and executed reliably when moved from one computing environment to another.

With the shift from download-and-process-locally to remote storage and remote processing of data, new revenue opportunities arise from selling computing cycles and data analytics services on these cloud platforms. The first are often realized in the form of virtual machines and application container environments combined with data storage, streaming, messaging, or database services. The latter is an

Simonis, I.
Standardized Big Data Processing in Hybrid Clouds.
DOI: 10.5220/0006681102050210
In *Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2018)*, pages 205-210
ISBN: 978-989-758-294-3

emerging field where application developers can offer additional services on top of data analytics offered by the cloud provider. These application developers offer their domain expertise and sell it in the form of optimized applications that can become part of externally controlled workflows and processing routines. Essentially, the necessary environment is not that different from app stores commonly used in the mobile phone sector, where a limited number of vendors offer platforms for software developers to make their applications available to the mass market for further deployment close to the data on various target platforms. Just, the applications addressed here are deployed on powerful machines.

A major challenge with the growing number of Big Data repositories, data sets, and available processing capacities and capabilities is the level of interoperability between and among all involved components. With the criticality of information technology to today's science, information standards are now fundamental to the progress of advancing the knowledge of our world (Percivall et al, 2015). The Open Geospatial Consortium (OGC) is an international not for profit organization committed to making quality open standards for the global geospatial community.

This paper describes standardization efforts to build an architecture featuring high levels of interoperability for provisioning, registration, deployment, and execution of arbitrary applications in cloud environments. Based on virtualization mechanisms and containerization technology, the standardized approach allows to pack any type of application or multi-application based workflow into a container that can be dynamically deployed on any type of cloud environment. Consumers can discover these containers, provide the necessary parameterization and execute them even easier than on their local machines, because no software installation, data download, or complex configuration is necessary.

2 EO CLOUD PROCESSING ENVIRONMENT

The work described herein has been executed in support of the ESA Thematic Exploitation Platforms (TEP), combined with domain specific requirements coming from the agriculture, forestry, and fisheries sectors.

With the goal to leverage the full potential of its earth observing missions, ESA has started in 2014 the Earth Observation Exploitation Platforms initiative, a set of research and development activities that in the first phase (up to 2017) aimed to create an ecosystem of interconnected Thematic Exploitation Platforms for Earth Observation data. In short, an Earth Observation (EO) exploitation platform is a collaborative, virtual work environment providing access to EO data and the relevant information and communication technologies (ESA, 2017). These platforms implement three abstract user scenarios: First, to explore and manipulate the data, second to deploy new applications that work on that data (service development), and third the publication of new results (product development). Here, we concentrate on the second use case, the provisioning of new applications. Goal is to allow any TEP user to develop their own algorithms and workflows on the their local machines, then to upload it to the TEP for further testing and optimization, and eventually to make these algorithms and workflows available to other users in the form of a software container and its complementary metadata, the *Application Package*. These software containers are then deployed upon request by the TEP in cloud environments close the the actual data. All necessary information to deploy and execute a container is provided by the Application Package.

3 CONTAINERIZED APPLICATIONS

An application package (OGC, 2018) needs to contain all information required to deploy and execute the package on the cloud. As illustrated in Figure 1 below, an application package encapsulates the description of the application itself, i.e. the application metadata, a reference to the application software container, metadata about the container itself and its resource types, deployment, execution, and mapping instructions of external data to container-specific locations for input and result data, and auxiliary information such as Web-based catalogues for data discovery and selection. The container itself is located on a Web-accessible hub, but not part of the Application Package.

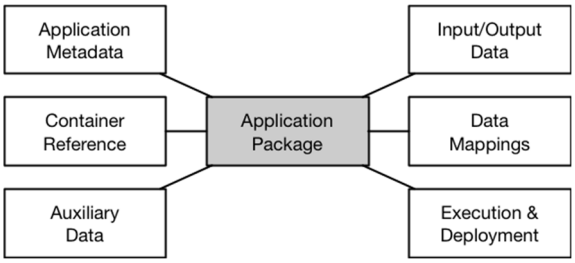

Figure 1: Overview of the Application Package.

The *Application Metadata* describes the application that is encapsulated in the container with sufficient detail, i.e. it provides information about the application's purpose, capabilities, and requirements in terms of input data, parameterization for execution, and result data. It does not contain any detail about the encapsulated software as such, i.e. types and versions of libraries, programming languages, internal workflows etc. This information is entirely opaque.

The container itself is not part of the application package, but referenced from herein (*Container Reference*). Using state of the art container technology such as Docker, the reference would point to a Docker Hub where the container is stored.

The optional *Auxiliary Data* contains any type of references, links, or further information that helps users to execute the application package successfully. It may, for example, contain links to Web-based catalogues that provide (Big) process-ready data that this application can work with.

The *Input/Output Data* section of the application package describes the required input data to run an application, e.g. satellite imagery, reference data, calibration data, etc. and informs what type of output the application produces. Given the volatile nature of many containers, applications will most likely produce output that is made persistent externally to the application container itself.

Some elements of the container package serve administrative purposes only. The *Data Mappings* provide instructions to the execution environment how external data sets need to be mapped and mounted to the container system, and the *Deployment and Execution* instructions provide all information required to deploy the container in a virtual machine environment and to start the actual application process.

4 OGC WEB SERVICE ENVIRONMENT

The information model of the Application Package and its encoding is ideally matching the service and information environment it operates in. The Web service interface standards, information models, and encodings released by the Open Geospatial Consortium (OGC) provide a robust and standardized environment for geospatial data, processing, and visualization and provide all required base-types for Application Package definition and serialization.

To exchange information about geospatial resources and services, the OGC has released the OGC Web Services Context Document (OWS Context Document). Initially developed to allow a set of configured information resources (the so-called service set) to be passed between applications, OWS Context is used today for general exchange of information about geospatial data and services. It supports OGC Web service instances, arbitrary online resources, and inline encoded content. A 'context document' specifies a fully configured set which can be exchanged (with a consistent interpretation) among clients supporting the standard (OGC, 2017).

In order to achieve maximal interoperability, the core context document is largely agnostic of any particular resource type and therefore serves as a solid base for the Application Package. Specific resource types are then defined in individual requirement classes that further specify the handling of that resource offering. The standardized resources types include Web service interface types (e.g. to access maps (Web Map Service, WMS), feature data (Web Feature Service, WFS), or coverage data (Web Coverage Service, WCS)), encodings for in-line content (e.g. Geography Markup Language (GML) or GeoTiff), and storage for external content. The work described in this paper has extended this set with the notion of containers and application parameters to support the deployment of applications in hybrid cloud environments.

Alternatively to the OWS Context Document based approach, process descriptions as offered by the Web Processing Service, WPS, can be used. The WPS is the second essential component within the OGC Web service environment necessary for standardized Big Data processing in hybrid clouds. The service interface allows to execute any type of software process in a standardized way. Recursively, any WPS process can spawn any number of child processes that again call other WPS or data access or processing services. The WPS interface standard defines process descriptions that contain sufficient detail for any type of parameterization in a similar fine-granular structure as the OWS Context Document. The service supports synchronous and asynchronous interaction patterns, out-of-band parameterization, and can be extended with publish-subscribe messaging functionality.

5 STANDARDIZED ARCHITECTURE

5.1 Application Package

OWS Context Documents provide sufficient functionality to support all requirements set by the Application Package definition. It has been used for the definition of the application package, following the OGC Context Conceptual Model (OGC, 2014a) and implementing an ATOM encoding (OGC, 2014b). Experiments with JSON encodings have been started but are not concluded yet.

The Application Package contains mandatory metadata elements such as id, title, abstract, author, and creation date; and a number of optional items that shall help during the initial discovery process. All elements have been mapped to Atom elements.

5.2 Application Container

Any application should be executed as a Docker Container in a Docker environment. The application developer needs to build the container with all libraries and other resources required to execute the application. This includes all data that will not be provided in the form of a parameter setting at runtime. The Docker Container Image itself can be built from a Docker Build Context stored in a repository following the standard manual or Dockerfile-based scripting processes. To allow standards-based application deployment and execution, the application should be wrapped with a start-up script.

5.3 Standardized Service Environment

The goal of the standardized service and application package environment is threefold: First to minimize the overhead for the application developer to make his work available to others. Second the full decoupling of the three players application developer, cloud environment operator, and application user; and third full flexibility for the application user to process his own or referenced data. The following architecture supports these elements by off-loading application deployment and execution tasks from the developer to a service environment, by standardized interfaces between all components, and standards-based parameterization of applications.

5.3.1 Application Development and Provisioning

The application developer uses a local development environment for the application development, configuration, and description. This includes the definition of all software components such as commercial off-the-shelf software, software libraries, scripts, calibration data etc. Once the application runs successfully within the local environment, all elements need to be packed into a Docker container, which then has to be uploaded to a Docker Hub. Figure 2 illustrates this first step (1).

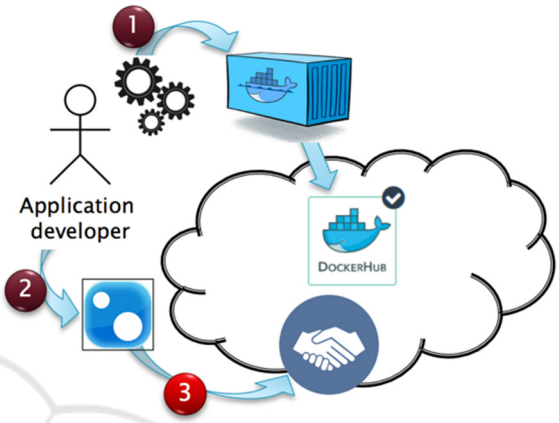

Figure 2: Application developer tasks.

Afterwards, the application package needs to be described with the necessary detail (2), which includes some basic description of the application itself, all required input parameters, the mapping of external sources to internal mount points, type and nature of all resulting products and output locations. In a last step, the application package is made available to the cloud environment. This can be done either by executing the corresponding WPS call, or supported by an application management client web application that façades the WPS and supports the developers with the definition of the application package.

5.3.2 Application Discovery and Usage

The end user discovers the application on the cloud platform (Figure 3 step (4)). The application package provides sufficient detail that can be stored in a catalogue service. If necessary, the user can be pointed to further data catalogues that allow the discovery of input data to be processed by this application (5).

Once all input data references or actual data sets are available, the user provides all required application parameters and executes the application (6). Upon completion of the data, the user can retrieve the final products.

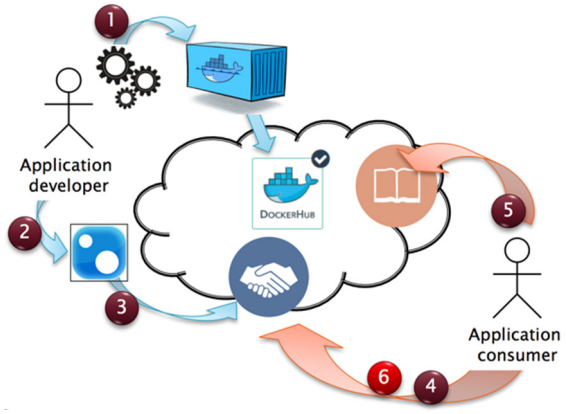

Figure 3: Application user tasks.

5.3.3 Standards based Cloud Environment

The entire workflow, including the provisioning of the application package, the deployment, parameterization and execution of the Docker container, and the provisioning of final products are supported by standardized interfaces. The following diagram illustrates the setup.

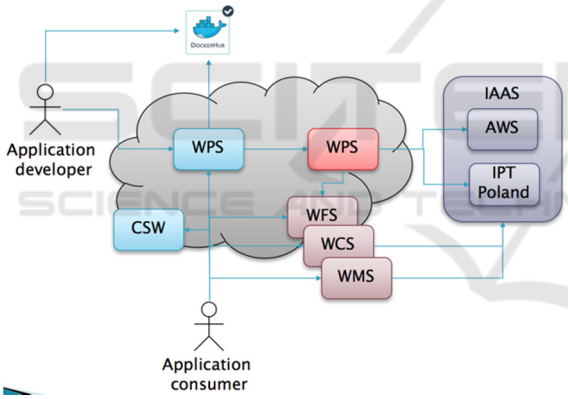

Figure 4: Standards based software architecture.

The application developer makes the application package available at a Web Processing Service, WPS (blue, left), which serves as the application management client. This WPS frontends the actual Thematic Exploitation Platform as described in section 2. It interacts with a second WPS to the right (red), which serves as the Application Deployment and Execution Service (ADES). In order to accept new processes as part of its offered processes portfolio, the application management client WPS needs to support transactions. If the WPS offers a dedicated process for transactions, then the application developer can call this WPS-execute operation directly and provide the OWS-Context encoded Application Package as parameter payload. Alternatively, the OGC WPS Transactional Extension (OGC, 2014c) can be applied, which offers dedicated deploy and undeploy operations.

The application package is described using the ProcessOffering process description mechanism or the OWS Context Document based implementation. Both approaches have been successfully tested. Further research is necessary to identify the ideal solution for dedicated scenarios. To handle input of externally stored data and to save results to external locations, the Application Package defines Application Parameters using the WPS Process Descriptions to cover both process input and process output. Currently, the architecture is limited to pass simple parameters (named WPS Literal Data in the WPS standard) only. These parameters are passed from the Application Deployment and Execution Service (ADES) to the application using either command line parameters or environment variables. Interoperability is assured by re-using parameter names as defined in the WPS Process Descriptions.

If execution is requested by the client, the ADES deploys the Docker container on cloud environments such as Amazon AWS, or, tested here, the EO Cloud Earth Observation Innovative Platform Testbed Poland (IPT Poland). Upon completion, the final data is served back to the client via standardized data access and visualization interfaces such as Web Mapping Service (WMS), Web Feature Service (WFS), or Web Coverage Service (WCS).

6 CONCLUSIONS

The work described herein illustrates results of the OGC Testbed-13 innovation program initiative. Six OGC members were challenged in April 2017 to develop a standards-based integration architecture that allows standards-compliant provisioning, deployment, execution and result access of arbitrary applications in hybrid cloud environments. The resulting developments combined with technical interoperability experiments demonstrate the feasibility of standard-based application handling. Alternative approaches have been tested, as e.g. for the definition of mount points and data mappings, application package definitions based on OWS Context and WPS process descriptions, or transactional WPS extensions vs. dedicated processes. In general, it has been demonstrated that the current standards portfolio requires only minimum extension and profiling work in order to achieve a high level of interoperability. Further research is necessary to identify best practices and to

further enhance the overall architecture with e.g. security concepts and additional functionality, such as application execution quoting and negotiation mechanisms. Dynamic integration of resources and applications from different cloud environments remains a research challenge in terms of cloud federation, security handling, and general performance.

ACKNOWLEDGEMENTS

This paper describes an integration architecture and elaborates the results of Testbed-13, a recent innovation initiative conducted by the Open Geospatial Consortium (OGC). The OGC thanks all participants for their active contributions. The work was further co-sponsored by European Union through the Horizon 2020 research and innovation programme under grant agreement No 732064, project DataBio.

REFERENCES

Amazon, 2018. Sentinel-2 on AWS. Website: http://sentinel-pds.s3-website.eu-central-1.amazonaws.com.

ESA, 2017. About TEP. https://tep.eo.esa.int/about-tep.

OGC, 2014a. OGC OWS Context Conceptual Model. OGC 12-080r2.

OGC, 2014b. OGC OWS Context Atom Encoding Standard. OGC 12-084r2.

OGC, 2014c. OpenGIS WPS2.0 Transactional Extension Discussion Paper. OGC 13-071r1.

OGC, 2017. OGC Web Services Context Document (OWS Context. http://www.opengeospatial.org/standards/owc.

OGC, 2018. Testbed-13: Application Package Engineering Report. OGC 17-035.

Percivall, G., Simonis, I., Idol, T., 2015. Scientific Knowledge from Geospatial Observations. In: IGARRS 2015.

Sivarajah, U., Kamal, M., Irani, Z., Weerakkody, V. 2017. Critical analysis of Big Data challenges and analytical methods. Journal of Business Research 70 (2017) 263–286.

Sookhak, M., Gani, A., Khan, M., Buyya, R. 2017. Dynamic remote data auditing for securing big data storage in cloud computing. Information Science (2017) 380, 101-116.

Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C., Buytaert, W. 2015. Web technologies for environmental Big Data. Environmental Modelling & Software 63 (2015) 185-198.