Dynamic web service deployment in a cloud environment

Marc Kemps-Snijders, Jan Pieter Kunst, Matthijs Brouwer, Tom Visser

Meertens Institute
Joan Muyskensweg 25,
1096 CJ Amsterdam
Netherlands

Sara
Science park 140,
1098 XG Amsterdam
Netherlands

Marc.kemps.snijders@meertens.knaw.nl, janpieter.kunst@meertens.knaw.nl, matthijs.brouwer@meertens.knaw.nl, tom.visser@sara.nl

Abstract

E-infrastructure projects such as CLARIN do not only make research data available to the scientific community, but also deliver a growing number of web services. While the standard methods for deploying web services using dedicated (virtual) server may suffice in many circumstances, CLARIN centers are also faced with a growing number of services that are not frequently used and for which significant compute power needs to be reserved. This paper describes an alternative approach towards service deployment capable of delivering on demand services in a workflow using cloud infrastructure capabilities. Services are stored as disk images and deployed on a workflow scenario only when needed this helping to reduce the overall service footprint.

Keywords: web services, cloud, CLARIN

1. Introduction

A number of e-infrastructure projects have been underway in the past couple of years aiming to provide an integrated and interoperable research infrastructure. These projects must tackle a large number of subjects including legal, organizational and technical challenges to ensure persistent, secure and reliable access to data and services that is easy to use by researchers. Rather than being a one-time effort, building these infrastructures is a continuous process of convergence towards acceptable standards of operations to all parties involved. These project do not operate in isolation but are part of a much wider landscape of related infrastructure projects and initiatives, involving user communities, data services and cross domain activities such as security and curation. For a project such as CLARIN the technological challenges fall into three main categories: data, tools/services and infrastructure. While there is much experience with data management (metadata, publication, persistency, security) that relies on many of the infrastructural services, such as persistent identifiers, the situation for tools/services is less mature. Projects such as CLARIN have seen a large increase in the number of services that are being made available, such as speech recognizers, NLP tools or annotation tools, questions on how to make these available in a sustainable manner over a period of many decades is still a largely unsolved challenge. While the standard method of deploying services on a (virtual) web server is common practice this often is not the most efficient deployment model when considering usage patterns. Many services currently being deployed have not (yet) found a broad user basis or are only used occasionally in a research environment. This paper describes an alternative deployment model that allows services to be available on demand while reducing the amount of resources necessary to make them available to the community. While the focus in this paper is a technical one, other issues must also be taken into consideration when deploying or using web services. In particular legal issues are of importance since in many cases data is migrated across national boundaries. The impact of this is not limited to the use of cloud infrastructures, but is relevant when using any type of web service.

2. Background

In the TTNWW (TST Tools voor het Nederlands als Webservices in Workflow) project a long-standing collaboration between the Netherlands and Flanders, spanning more than 10 years and including projects such as CGN and STEVIN, tools and data for the Dutch language domain are prepared for integration in the emerging national and European CLARIN eScience infrastructure. One of the preconditions for the CLARIN infrastructure is the establishment of centers providing persistent access to stable, highly available services of various types. These centers must free researchers providing data and tools from all non-primary scientific tasks such as additional bureaucratic, administrative and technical tasks associated with centers operation. Goal of the TTNWW project is to incorporate language components in a workflow system using web services, to ensure that all these components are hosted by reliable CLARIN centres and to provide access to these workflows to Humanities and Social sciences (HSS) researchers with little or no technical background. In the TTNWW project at least 9 new services will be made available, including corpus cleanup, tagging, parsing, alignment, NER, coreference, semantic role labeling, spatiotemporal analysis and speech recognition. These services are made available by a range of Dutch and Flemish research institutes. To lower that barriers for using these web services they will be combined in task specific workflows that are offered to the researchers in a web enables user interface.

The BiGGrid project (led by partners NCF, Nikhef and NBIC) aims to set up a grid infrastructure for scientific research. This research infrastructure contains compute clusters, data storage, combined with specific middleware and software to enable research that needs more than just raw computing power or data storage. The project has advanced beyond its original grid infrastructure and has made part of hardware cloud accessible.

3. Description of work

In collaboration with the BiGGrid project a dynamic web service deployment pilot project was conducted where services and workflows are dynamically deployed and executed without the need for maintaining multiple dedicated servers over a long period of time. This technique relies heavily on virtualization and takes advantage of cloud infrastructure capabilities that are delivered by the BiGGrid project. The cloud infrastructure (OpenNebula) provides the possibility to manage virtual machines and images in a cloud environment. This includes assignment of memory usage, number of cpu's, disk and network configuration. Each cloud user is assigned a workspace in which virtual machine images may be uploaded, managed and deployed without interfering with the needs of other users.

In the setup for this pilot project two services were chosen to provide the proof of concept, a corpus cleanup service (TICCLOPS) and a memory-based morphosyntactic tagger and parser for Dutch (FROG[1][FROG]). Both services had already been delivered as part of the TTNWW project and were considered to be stable enough for the operational phase. Both TICCLOPS and FROG are software packages that were originally designed for standalone usage and as such not directly usable in a Service Oriented Architecture (SOA) environment. To make the original software packages available as web services they were converted to web services using CLAM[2]. The CLAM (Computational Linguistics Application Mediator) software package assists in making standalone applications available as a RESTful web service and also makes the original functionality available as a web application. It has successfully been employed in a number of CLARIN-NL projects. CLAM distinguishes a number of discrete operations guiding the execution process of the wrapped software. The basic steps involved in executing a wrapped service consist of:

1. creating a project; a project serves as a container for all artefacts collected during an execution process. It contains input files, output files and is identified by a unique name.
2. uploading the necessary input files,
3. specifying additional parameters; each service may have process specific parameters that must be supplied before execution.
4. starting the process
5. checking process status, a polling mechanism allows for basic monitoring of the process status.
6. downloading the results once the process has finished
7. removal of the project; this includes removal of any input and output files residing on the server

These steps are reflected in a number of web service operations that are exposed to the outside world. After the CLAM wrapping process, each of the resulting services was installed on a separate virtual server and disk images of each of the servers were created. These disk images provide a snapshot of the state of the server at the time it was made and may be used to quickly restore a server's state. These server images were subsequently uploaded into a cloud workspace for further use.

To combine web service operations into a sequence of connected steps a workflow management system was used that monitors and controls the order in which web services are executed. Within the TTNWW project the decision was made to use TAVERNA[3][TAVERNA-1][TAVERNA-2][TAVERNA-3] consisting of a number of tools to support the creation, execution and monitoring process of workflows. Workflows are created using the TAVERNA workbench while execution may either be done locally from the workbench or remotely using the TAVERNA server. For our purposes the TAVERNA server was used making it possible to execute workflows directly without the need for locally installing the TAVERNA workbench. After installation and configuration of the TAVERNA server a disk image was created and uploaded into the cloud's workspace.

For both the TICCLOPS and FROG services 'mini' workflows were created that encapsulate the individual web service steps of CLAM wrapper (project creation, upload, etc.). These workflows are further combined into larger workflows thus allowing for reuse of previously created partial workflows. Configuration parameters and data sources can be specified in such a manner that these may be set as part of the execution process. Normally the web service end points are known in advance. For the purposes of this project the locations of the web services were also externalized to make it possible to execute the workflow without knowledge of the location of the web service at design time.

4. Dynamic web service deployment

To be able to use web services they must be deployed. As indicated above, only the server images are uploaded into the cloud's workspace and workflow specifications have been created that are able to connect the various steps of the execution process. For the deployment process two deployment types were used. The TAVERNA image was deployed as a dedicated server, the TICCLOPS and FROG images were to be deployed as dynamic web services. We wanted to keep the footprint of the combined services as low as possible and only deploy services when they were needed.

In order to achieve the dynamic deployment scenario, i.e. that services are only deployed when they are requested for use as part of a workflow, two additional services were

[1] http://ilk.uvt.nl/frog/
[2] http://ilk.uvt.nl/clam/
[3] http://www.taverna.org.uk/

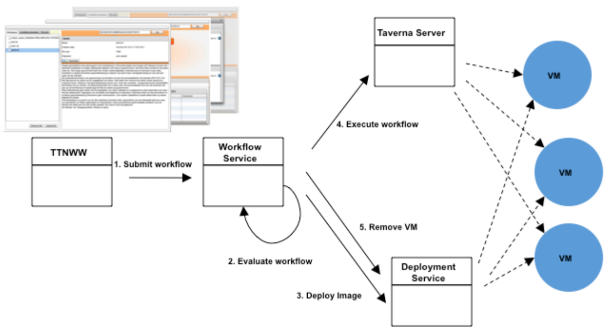
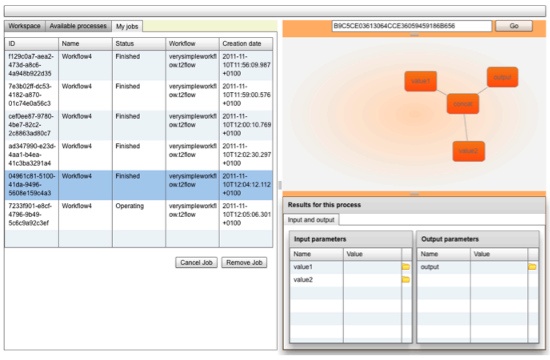


Figure 2: Architecture and interaction patterns during service deployment

created that manage the workflow configuration and manage the image deployment and destruction process. When a workflow is sent to the first service it will assess whether any of the services specified in the workflow process is dynamic in nature. Images of these services are expected to have been stored in the cloud's workspace. In our current setup this was achieved by adding service annotations to the workflow specification document using standard TAVERNA features. Once it is determined that there are indeed services that must be deployed requests are sent to the deployment service which interacts with the OpenNebula interface. Here, the images are loaded from disk and deployed on one of the cloud's compute nodes. Assignment of memory usage, number of cpu's, disk and network configuration are also part of this process. Network configuration ensures that the deployed services are only accessible by processes from within the cloud workspace environment. Once the service(s) are up and running the workflow configuration manager receives the locations for each of the services (IP addresses are dynamically assigned) and forwards this information plus the original workflow specification, configuration parameters and data sources to the TAVERNA server. The TAVERNA server interacts with the previously deployed services and returns the results to the original client who requested the workflow execution. Upon completion all services participating in the workflow process are destroyed.

5. End user experience

The average end user is not interested in where services are located and how they are. As part of the project a web interface was developed encapsulating the intricacies of the SOA and deployment architecture. A small front end web application (see figure 2) was developed allowing users to interact with the services in a transparent manner. To make it easier for users interaction with the web services does not go directly but is mediated through a workflow. Each workflow consists of a number of interlinked services that are designed to complete a specific task. For the purposes of the TTNWW project NLP processing and speech recognition pipelines are created. These perform a specific series of tasks such as tokenization, POS tagging, NER recognition, coreferencing and spatio temporal relation detection. To the end user each of these workflows is represented as a single task. The current interface allows upload of workflow specifications as a whole, which will be replaced in the final interface with a set of menu bar commands that are associated with predefined workflow specifications. In the current user scenarios where novels and year books are analyzed the processing time for executing the individual processing services can be considerable. To accommodate for this the web application will store the results in a user's workspace where they may be retrieved upon the user's return. Given this, startup time of the images in the cloud environment does not significantly reduce user expectancy. Start up of FROG image takes under 30 sec. Moreover, in the current setup all services are deployed at startup of the workflow. As a result services are generally ready to go once the workflow engine requests access (with the exception of the first).



ID	Name	Status	Workflow	Creation date
f129c0a7-aea2-473d-a8c6-4a948b922d35	Workflow4	Finished	verysimpleworkflow.t2flow	2011-11-10T11:56:09.987+0100
7e3b02ff-dc53-4182-a870-01c74e0a56c3	Workflow4	Finished	verysimpleworkflow.t2flow	2011-11-10T11:59:00.576+0100
cef0ee87-9780-4be7-82c2-2c8863ad80c7	Workflow4	Finished	verysimpleworkflow.t2flow	2011-11-10T12:00:10.769+0100
ad347990-e23d-4aa1-b4ea-41c3ba3291a4	Workflow4	Finished	verysimpleworkflow.t2flow	2011-11-10T12:02:30.297+0100
04961c81-5100-41da-9496-5608e159c4a3	Workflow4	Finished	verysimpleworkflow.t2flow	2011-11-10T12:04:12.112+0100
7233f901-e8cf-4796-9b49-5c6c9a92c3ef	Workflow4	Operating	verysimpleworkflow.t2flow	2011-11-10T12:05:06.301+0100

Input parameters

Name	Value
value1	
value2	

Output parameters

Name	Value
output	

Figure 1: User interface showing execution of several complex tasks and the structure and results of one of them.

6. Relation to metadata

The possibility of using disk images also opens alternative approaches towards web service publication. Traditionally, web services are associated with WSDLs or WADLs containing the service end points. Although efforts have been made to make full service descriptions available, for example using UDDI and EbXML, none of these approaches have been able to meet the requirements from an e-infrastructure perspective. In CLARIN services are therefore described using CMDI meta data[CLARIN D2R-6b][CLARIN D2R-7b] that is also used for describing (language) resources. This provides not only a shared technological basis for publishing both data and services but also provides a high degree of flexibility capable of describing the service's capabilities. Each service must announce its capabilities and whereabouts using the same component based mechanisms that also apply to metadata publication of data resources. Similar to resources, the location of a web service can be described by pointing to a dedicated web server location or may be made flexible by pointing to a deployment agent with the appropriate image specification. Also, the image itself may be published allowing users to download the image and run the service. The latter option is particularly useful for organizations who do not have the capabilities or desire to maintain a dedicated web server but also have sensitive data that is not expected to leave institutional grounds.

Since the workflow specifications are XML documents they can also be published in the same manner using CMDI. Workflow specifications have similar characteristics as data resources except that they also specify input and output characteristics comparable to a service's input and output parameters. Using the CMDI mechanisms it is thus possible to publish (partial) workflows as well as the data resources and services that are used in the execution chains.

7. Discussion

This approach of dynamic service deployment has a number of advantages. Firstly it reduces the amount of necessary resources to keep a set of services available over a period of time. Particularly for services that are seldom used or in the early stages of adoption of the process this presents an interesting possibility of gradual scaling. Time needed to deploy a service depends upon the size of the image. This method of service deployment will also, as a side effect, ensure that any files uploaded to the service or generated by the service are automatically destroyed once the service is destroyed. While the CLAM service wrapper also provides the possibility to remove all uploaded and generated files this must be explicitly initiated by the client. No residual data is thus left behind on the service instance.

To publish the service's capabilities standard CMDI mechanisms may be used. Service descriptions can be extended by announcing the service's dynamic deployment end point, i.e. the deployment service and the location of the disk image. This provides capabilities of flexible remote or local execution scenarios. Workflow specifications that chain the services can be published and reused in this manner as well, thus making it possible to make data resources, services and workflows widely available and reusable throughout the infrastructure.

8. Acknowledgements

We would like to express our gratitude to the Stichting Nationale Computer Faciliteiten who provided the necessary computing and storage facilities for this project.

9. Bibliographic References

[CLARIN D2R-6b]Adam Funk , Nuria Bel, Santi Bel, Marco Büchler, Dan Cristea, Fabienne Fritzinger, Erhard Hinrichs, Marie Hinrichs, Radu Ion, Marc Kemps-Snijders, Yana Panchenko, Helmut Schmid, Peter Wittenburg, Uwe Quasthoff, Thomas Zastrow. Requirement specification web services and workflow systems. *Available at:* http://www-sk.let.uu.nl/u/D2R-6b.pdf

[CLARIN D2R-7b] Marc Kemps-Snijders. Web services and workflow creation. *Available at:* http://www-sk.let.uu.nl/u/D2R-7b.pdf

[TAVERNA-1] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, "Taverna, reloaded," in *SSDBM 2010, Heidelberg, Germany, 2010.*

[TAVERNA-2] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services.," *Nucleic Acids Research, vol. 34, iss. Web Server issue, pp. 729-732, 2006.*

[TAVERNA-3] T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, "Taverna: lessons in creating a workflow environment for the life sciences," *Concurrency and Computation: Practice and Experience, vol. 18, iss. 10, pp. 1067-1100, 2006.*

[FROG] Antal van den Bosch, Bertjan Busser, Sander Canisius, Walter Daelemans. An efficient memory-based morphosyntactic tagger and parser for Dutch. *Available at:* http://ilk.uvt.nl/downloads/pub/papers/tadpole-final.pdf