SeaClouds Project

D2.3.2 - Periodic Standardization report

Project Acronym	SeaClouds
Project Title	Seamless adaptive multi-cloud management of service-based applications
Call identifier	FP7-ICT-2012-10
Grant agreement no.	610531
Start Date	1st October 2013
Ending Date	31st March 2016
Work Package	WP2 Requirements Analysis, overall Architecture and Standardization
Deliverable code	D2.3.2
Deliverable Title	Periodic Standardization report
Nature	Report
Dissemination Level	Public
Due Date:	M18
Submission Date:	March 31, 2015
Version:	1.0
Status	Final
Author(s):	Andrea Turli (Cloudsoft), Jose Carrasco (UMA), Javier Cubo (UMA), Elisabetta Di Nitto (POLIMI), Francesco D'Andria (ATOS), Marc Oriol (UPI)
Reviewer(s)	Francesco D'Andria (ATOS), Ernesto Pimentel (UMA)

Dissemination Level

Project co-funded by the European Commission within the Seventh Framework Programme		
	Public	X
	Restricted to other programme participants (including the Commission)	
	Restricted to a group specified by the consortium (including the Commission)	
	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	09/03/15	First ToC	Andrea Turli
0.2	18/03/15	First contributions	Jose Carrasco, Javier Cubo, Andrea Turli, Francesco D'Andria (ATOS), Marc Oriol (UPI)
0.3	24/03/15	Second contributions	Javier Cubo, Elisabetta Di Nitto, Andrea Turli
0.4	25/03/15	Version to be revised	Andrea Turli
1	31/03/2015	Final version	Andrea Turli and Francesco D'Andria

Table of Contents

Executive Summary

The D2.3.1 Periodic Standardization Report summarised the main standards relevant to our work in SeaClouds, in particular outlining the areas where we are consuming the standards, contributing to the standards, or fostering adoption of the standards.

This deliverable, D2.3.2, is an update on some of the topics described in D2.3.1

In order to have a lightweight document, some sections will contain simply a reference to the previous version of the doc.

The structure of this document is the following:

- Section 1: This section outlines why we view standards as important to our research and to the wider community
- Section 2: This section outlines the key standards in four areas:
    - Application Topology
    - Infrastructure-as-a-Service
    - Platform-as-a-Service
    - Service Level Agreements

2. The Importance of Standards in SeaClouds

This section is an integration of what already discussed at D2.3.1 “The importance of Standards in SeaClouds”.

Relevant Standards and Open Source Systems, and Our Work in SeaClouds

Application Topology

2.1 TOSCA

OASIS TOSCA — Topology and Orchestration Specification for Cloud Applications — describes how an application should be deployed. In its original form it specifies an XML syntax describing nodes, services, and relationships, along with the type definitions and implementational details.

TOSCA Simple YAML Profile

TOSCA Simple YAML Profile proposes a YAML-based modeling language which permits to specify an application’s structure as a topology graph, and the management tasks as plans. More precisely, TOSCA YAML aims at providing a language to express how to automatically deploy and manage complex cloud applications by requiring developers to define an abstract topology of the application and to create plans describing its deployment and management.

TOSCA YAML addresses the portability and interoperability by providing a standardized way to describe the topology of multi-component applications in an interoperable and reusable way.To this end, TOSCA YAML abstracts from messages and protocols details, and it permits to describe the dependencies between application components in a reusable manner.

For more details, please have a look at [3]

TOSCA monitoring subgroup

Since January 2015, the technical committee has created a new subgroup that is focusing on extending the TOSCA specification in order to describe monitoring aspects of an application.

One of the partners of the consortium, Polimi, is attending the conference calls being held within this working group. The current discussion is around the development of specification pieces that define the information that are relevant for a monitoring metric (the way it is obtained, its type, the way it can be aggregated, e.g., but computing an average on a certain time window) and for the way a monitoring metric is connected to a monitored component.

2.2 Why Relevant

The description of an application’s topology is essential as input to SeaClouds and useful as intermediate representations and presentation back to a user. By following open standards, we increase the potential for SeaClouds to interoperate with other tools on the inbound and outbound sides.

The definition of a monitoring specification in terms of potentially gathered monitoring metrics and in terms of monitoring rules that can be enacted is relevant to the development of the SeaClouds runtime.

2.3 Our Contributions

Some SeaClouds members have been involved with CAMP and/or TOSCA technical committees, suggesting improvements to the specifications based on our activity. Many of the SeaClouds members have been contributing to Apache Brooklyn which provides an implementation of CAMP (earlier version as of this writing) and is starting to develop an implementation of TOSCA.

Location

The work done to connect the Planner and Deployer components has derived in a proposal to slightly improve the TOSCA expressiveness.

Although we have mentioned the TOSCA expressiveness (in previous documents where TOSCA was described), this standard does not define in the topology any property to specify the target provider to deploy the application modules.

Instead, it defines an orchestration plan and the implementation artifacts to specify the deployment operations. These artifacts point to the specific providers used for the distribution of the components implementing the logic to perform the deployment. But, SeaClouds focuses on the TOSCA descriptive mechanisms and the artifact management features is not used, so we avoid to implement a robust logic to infer the distribution information from the artifacts. Then, we take advantage of TOSCA expressive capability to describe the application modules and how they are involved, but it is unable to specify the target clouds to carry out the application distribution.

Moreover, the definition and maintenance of an orchestration plan is a complex and error-prone task. The plans have to define each necessary step to deploy and configure the application taking into account all the properties and requirements of the providers. Also, according to the current mechanism proposed, the plan's operations should be modified in case of changing the cloud providers in which modules are deployed. This is because the modification of the providers is performed by substituting the artifacts that implement the deployment operations.

In order to tackle this complex task, we propose to enrich the TOSCA specification to describe explicitly the providers in which each component will be deployed [1].

We define a new tag, <location>= "xs:string"</location>, in the Node Template specification, as shown in the next code which indicates that the JBoss component (of the example introduced in Section II-B) will be deployed on an Amazon AWS, using Oregon's clusters (as we can see in Table LOCATION-CODE).

With this extension, not only the expressivity of the standard is increased and clarified, but also this information can be extracted during the interpretation of the component's definition, avoiding to bind it to the execution of the artifact implementation.

With the purpose of expressing in an unambiguous way the artifact location, we propose to add a set of identifiers referencing to the cloud provides to be used, using a pair <Key-Value> (e.g., tosca.location.named.AWS Oregon = aws-ec2:us-west-2).

```
<NodeTemplate id="JBossMainWebServer"
 name="JBoss Main Web Server"
 type="JBossWebServer">


  <Properties>
    <ns2:JBossWebServerProperties>

      <httpPort>80</httpPort>

      <location>aws-ec2:us-west-2</location>

    </ns2:JBossWebServerProperties>
  </Properties>
  <Capabilities>

    <Capability
     id="JBossMainWebServer_webapps"
     name="webapps"
     type="JBossWebAppContainerCapability" />

  </Capabilities>
</NodeTemplate>
```

Table 1: Location Code

Note the location definition could be modeled as a property in the Node Templates. This could be useful to avoid a large negotiation of the consortium to approve the standard extension and providers could feel free to support this feature implementing the necessary mechanisms in their platforms.

However, although the mentioned approach has several advantages, our goal goes beyond, by defining an extension of the standard to ensure the multi-deployment performance and allow the correct definition of the providers.

3. Infrastructure-as-a-Service

This section is an integration of what already discussed at D2.3.1 “Infrastructure-as-a-Service”.

3.1 Our Contributions

SeaClouds is actively using Apache Jclouds via Apache Brooklyn to support *all* of the systems identified above. SeaClouds members have been contributing directly to Apache Jclouds and to Jclouds support in Apache Brooklyn.

Recently, SeaClouds members have been contributing actively on adding the support for docker and Azure compute to Apache jclouds.

4. Platform-as-a-Service

4.1 The State of the Art and the Need for Standards

This section is an integration of what already discussed at D2.3.1 “Platform-as-a-Service”.

Cloud Foundry

Cloud Foundry (http://cloudfoundry.org/) is rapidly emerging as the leading player in the PaaS space. Cloud Foundry is an open source PaaS, proposed by VMWare (http://www.vmware.com/), spun out as the company Pivotal, and now brought to a new foundation, the Cloud Foundry Foundation.

The Cloud Foundry Core defines a baseline of common capabilities to promote Cloud portability across different instances of Cloud Foundry. We have mentioned that this approach is a cloud computing PaaS, however it needs an IaaS layer over deployed applications. Currently, Cloud Foundry supports AWS, OpenStack (mentioned above), and VMware clouds [4], through the BOSH Ruby tool chain.

So, it is very easy to define an application and its dependencies using the aforementioned Build Packs and to deploy it in a **portable way** over the IaaS supported, so long as Cloud Foundry and the Build Packs are supported in that IaaS. (Note that Cloud Foundry can be used over public, private and hybrid cloud.)

In this sense, Cloud Foundry provides an abstraction of the IaaS infrastructure through the PaaS level.

4.2 Our Contributions

In the context of Seaclouds requirements to deploy against PaaS locations, some developers in Atos has identified a gap in the official *cloudfoundry-client-lib*, a Java library that provides a Java language binding for the Cloud Foundry Cloud Controller REST API.

Cloudsoft team has contributed[1] to the cloudfoundry-client-lib to fill that gap.

5. Relevant Research and Interoperability Projects

This section is an integration of what already discussed at D2.3.1 “Relevant Research and Interoperability Projects”

SeaClouds is going to release the components of the platform not only as the whole platform, but also as individual components could be used.

Thus, for example, the designer or the deployer components could be used for other projects or tools, such as Alien4Cloud [4], which have already shown the interest on the SeaClouds components (mainly as regards the Deployer component based on Brooklyn and TOSCA, CAMP standards).

Considering also the related projects previously mentioned in D2.3.1 Relevant Research and Interoperability, here a list of additional projects related to SeaClouds at different levels:

CloudWave. European Project. http://cloudwave-fp7.eu

CloudWave aims to significantly increase the competitiveness of the European cloud services industry by introducing breakthrough technologies for developing and deploying high-quality, adaptive cloud applications for a converged IoS/IoT ecosystem.

With CloudWave, service providers will be able to rapidly design and deliver innovative, sustainable digital services for consumers, at low cost and high quality; and service

[1] https://github.com/cloudfoundry/cf-java-client/commit/8be775dac4e543cb281e57b5eba13e1817bc75e3

consumers will benefit from higher service availability, quality and dependability across all areas of life – including business, science, leisure activities and government operation.

At some points, SeaClouds could convergence with Cloudwave, mainly in the sense distributed algorithms and data models that enable cloud infrastructures and applications to take actions in response to the dynamic changes in their environment, by coordinating the adaptation. Also, it is worth mentioning SeaClouds and CloudWave are joining efforts and commutities in order to organize a joined workshop in the European Conference on Service-Oriented and Cloud Computing (ESOCC 2015): http://esocc2015.unime.it

ARTIST. European Project: http://artist-project.eu

The project creates tools to assess, plan, design, implement and validate the automated evolution of non-cloud software to SaaS and the Cloud Computing delivery model. By focusing on reusability during this transition, the methods and tools are generic enough to cover future shifting efforts, e.g. deployment to future platform delivery paradigms.

Members from both consortium have been discussing about some technical aspects. For example, related to standards such as TOSCA. Also, SeaClouds could use some of the tools created in the ARTIST project, mainly related to the migration mechanisms, which in SeaClouds are also required.

REMICS. European Project: http://www.remics.eu/consortium

To develop advanced model driven methodology and tools for reuse and migration of legacy applications to Interoperable Cloud services.

SeaClouds will review the REMICS paradigm for migrating legacy applications on the cloud in order to potentially apply and/or use it for designing the migration strategies. Although REMICS methodology focuses on legacy applications, SeaClouds shifts the migration paradigm to the cloud services.

RESERVOIR. European Project: http://www.reservoir-fp7.eu/

To develop an innovative service–oriented infrastructure that will allow the dynamic interoperability of Cloud providers for the reliable delivery of services. To this end, it separates the roles of service provider and infrastructure provider.

As SeaClouds, RESERVOIR bases on a SOA foundation, but more focus on the dynamic interoperability from the infrastructure view. In comparison, SeaClouds focuses more on the platform level (although also addressing some issues in the infrastructure one).

CumuloNimbo. European Project: http://www.cumulonimbo.eu/node/8

To provide a scalable PaaS Service which will enable secure and un-partitioned data transactions resulting in consistent applications and at the same time ensuring the independent and optimized use of resources at a minimum cost.

It's centred on data consistency and they propose their own full-blown PaaS stack, with special emphasis in consistent storage service. SeaClouds goes beyond, not only with data synchronization, but also reconfiguration, migration process, although SeaClouds is not considering security at the same level of CumuloNimbo.

Cloud-TM. European Project: http://www.cloudtm.eu/home/consortium

To define a programming paradigm to facilitate the development and administration of Cloud applications.

It is focuses on self-optimizing middleware platform aimed at simplifying the development and administration of applications deployed on large scale Cloud Computing infrastructures, while SeaClouds administrates complex applications but tackling the distribution and migration issues.

ConPaaS. European Project: http://www.conpaas.eu/

It provides a handful of services to ease the development of new SaaS. It offers an alternative PaaS, and a new lock-in pit.

In SeaClouds the orchestrators services developed could also be used like new SaaS, by performing the distribution of modules of a complex application in multiple Clouds, what is not address by ConPaaS.

NEFFICS. European Project: http://neffics.eu/

Networked Enterprise transFormation and resource management in Future internet enabled Innovation CloudS.

NEFFICS gives value-added in the sense of which networks will be more open, flexible, adaptive, participatory and peer-to-peer. Although it is mainly focused on the resource's management (including processes, products, services and persons); and not on the orchestration of services in multiple Clouds as is the foundation of SeaClouds.

SHAPE. European Project: http://www.shape-project.eu/

Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering. SHAPE promotes a new development paradigm with a higher degree of involvement of joint users and development communities through minimising the gap between business and system modelling.

But SeaClouds goes beyond, addressing a life-cycle to model, plan and control the distribution and migration of modules of cloud-based applications.

MODAClouds. European Project: http://modaclouds.eu

MODAClouds aims at developing a model-driven DevOps approach to support the development of multi-cloud, quality aware applications.

Differently from SeaClouds, MODAClouds does not support unconstrained distribution of application components on multiple clouds. Conversely, it allows replication of applications on multiple clouds to support fault tolerance and to increase availability. Also, MODAClouds does not support full-fledged automatic planning of application configuration but it offers mechanisms to support SoQ-based optimization of configurations defined by the users.

SeaClouds is incorporating in its infrastructure the multi-cloud monitoring platform offered by MODAClouds as well as its mechanisms to support data replication and synchronization on different Databases as a Service.

6. Service Level Agreement Languages

The main challenges addressed within the SeaClouds project also include the enhancement of existing Service Level Agreement (SLA) description models as well as respective negotiation strategies and protocols to enable the an end-to-end Service Level Agreements creation, the development of monitoring and feedback mechanisms to observe the commitments met by an SLA, and the development of adaptation strategies to mitigate the effects of possible SLA infringements.

All these aspects have been analyzed both on a design and implementation level. SeaClouds looked at several relevant standardization bodies and working groups (e.g. ETSI, OGF GRAAP, OGF OCCI or DMTF Open Cloud Standards Incubator) in the evolution of a standardized model of end-to-end Service Level Agreement procedures for Clouds, which will allow precise description of Quality of Service (QoS), an effective governance and audit processes, and lifecycle management of complex systems in heterogeneous and multi-cloud environments.

Finally SeaClouds decided to adopt and extend the SLA description model defined by WS-Agreement Specification.

On one hand the extension to the model has been proposed to adapt WS-Agreement to the specific SeaClouds requirements, on the other hand the model is full compatible with the whole SeaClouds approach.

6.1 Why Relevant

SeaClouds requires a language for expressing quality of service, to be embedded in the abstract model supplied by the user and passed to the monitoring systems. Thus an understanding of the standards and available systems is very relevant to our work.

6.2 Our Contribution

SeaClouds will primarily be a consumer of these standards. We do not envision attempting to advance them, although we see some promise in using the YAML lightweight expressive style used by CAMP [2] and TOSCA [3] to formulate the semantics of these standards, and are open to the opportunity to bring the SLA communities and the Application Topology communities closer together.

7. References

[1] J. Carrasco, J. Cubo, E. Pimentel. Towards a flexible deployment of multi-cloud applications based on TOSCA and CAMP. ESOCC 2014 Workshops (To appear).

[2]: http://docs.oasis-open.org/camp/camp-spec/v1.1/cs01/camp-spec-v1.1-cs01.html

[3]:http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csd01/TOSCA-Simple-Profile-YAML-v1.0-csd01.html

[4]: http://alien4cloud.github.io