HORIZON2020 FRAMEWORK PROGRAMME

TOPIC EUK-03-2016

"Federated Cloud resource brokerage for mobile cloud services"

D3.6
BASMATI Server- and Client- side Applications Adaptation and Reconfiguration: Software Prototype

Project acronym: BASMATI

Project full title: *Cloud Brokerage Across Borders for Mobile Users and Applications*

Contract no.: 723131

Workpackage:	WP3	Users and Applications Modelling and Analysis	
Editor:	Emanuele Carlini		CNR
Author(s):	Emanuele Carlini, Vinicius Monteiro De Lira, Patrizio Dazzi, Cristina Muntean		CNR
Authorized by	Young Woo Jung Konstantinos Tserpes		ETRI ICCS
Doc Ref:	D3.6		
Reviewer	K. Tserpes		ICCS
Dissemination Level	Public		

Document History

Version	Date	Changes	Author/Affiliation
1	18/04/2018	Document content	CNR
1_KT	24/04/2018	Review version	ICCS

Executive Summary

This deliverable describes the software prototype of the BASMATI component Application Repository. The Application Repository is the common repository of the BASMATI Enhanced Application Model (BEAM). The BEAM has been purposely designed in BASMATI to support application reconfiguration and adaptation, by providing a structure for extending a classical TOSCA-based application topology with a set of companion document. This deliverable goes through the requirements and installation of the component and provides an example of usage via its administration APIs.

BASMATI Glossary

Term/Acronym	Definition
Mobile cloud services	Online services offered by cloud resources to support mobile apps. The backend of the mobile apps.
CP	Cloud Provider. The actor that provides the cloud infrastructure/resources, such as VMs
CSP	Cloud Service Provider. The actor that provides cloud services on top of a rent infrastructure from a CP
Cloudlet	Limited capacity infrastructures with virtualization capabilities, often used to support a limited amount of users or perform a limited set of operations on behalf of the central cloud infrastructure that hosts the complete application
Edge resources	Resources aimed to operate specialized functionality, located at the "edge" of the network infrastructure, thus, closer to the end users. Examples are (clusters of) RaspberryPis or cloudlets
BUDaMaF	BASMATI Unified Data Management Framework
KE	Knowledge Extractor
DM	Decision Maker
RB	Resource Broker
MVD	Mobile Virtual Desktop
DASFEST	An 3-day long music festival taking place in Karlsruhe, Germany every July
ACE	Amenesik Cloud Engine. The cloud service deployment tool through which actual federation is achieved
BEAM	BASMATI Enhanced Application Model. An extension of the TOSCA specification
ASP	Application Service Provider. A Federation user that rents resource services in order to provide an Application services to End-users
Brokering	The matchmaking support provided by BASMATI platform to decide about the best cloud resources to exploit for the execution of the back-end of BASMATI applications. This activity regards the placement of the services or data on computational resources and storages belonging to the cloud data centre and the cloudlets within the federation.
End user	A user who benefits the various application and infrastructure services provided by the Cloud. Within BASMATI, the most typical example is exploiting the Cloud federation via a mobile device (possibly a laptop) using specialized apps or a web browser.
Offloading	The ability of BASMATI platform supporting the runtime placement of the components composing the front-end of BASMATI applications on edge resources available nearby the end user. This activity takes place both when

	edge and mobiles exchange one each other their own workload or when such devices transfer some workload to the clouds or cloudlets. In BASMATI we often distinguish Front-end offloading, related to the mobile part of application, from Back-end offloading, concerning the server side of applications. The latter roughly translates to the known concept of Cloudbursting.
QoE	Quality of experience. It is a measure of a customer's experiences with a service. It may be related to some aspects of the QoS and QoP, but can also take into account other metrics.
Service handover	Service handover refers to the activity of transferring an active service between two computational resources (e.g. Cloudlets) with minimal or no disruption on the availability of the service. Ideally, service handover is transparent with respect to the user.
Situational Awareness	The ability of the BASMATI platform to recognise the “situation” characterising the actual combined status of users, applications and resources, aimed at achieving an effective and efficient management of applications and resources.

Table of Contents

1 Application Repository

BASMATI extends a classical TOSCA application model with other pieces of information that drives the placement of the application in cloud resources belonging to the cloud federation. The model exploited is the BASMATI Enhance Application Model (BEAM). From a practical point of view, a BEAM archive is a collection of information, materialized as documents, which describes the application as a collection of services and defines hints about the resources needed, its placement and deployment (for all details please see Deliverable 3.5 [1]).

The Application Repository is the component in charge of storing the BEAM archives. Each "basmatized" application has its own BEAM documents; no documents are shared between different applications. The Application Repository (AR) is a passive component, serving the requests coming from the other components; it serves as common repository for the BEAM archives. In the following we describes the requirements and the installation process to run the application repository, as well an example of its usage.

2 Requirements and Installation

The Application Repository has been developed and tested under Linux Ubuntu 16.04. However, other unix-like operative systems, as well as Windows OSs, can be suitable to run the AR. The examples in the following are provided for Linux Ubuntu. The Application Repository is written in Python 3.0, specifically using the Django Framework[1] and the Django RestFul[2]. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design of web applications. As a consequence, the setup for the Application Repository requires the installation of both libraries.

The source code of the Application Repository can be downloaded from the Basmati Git Lab repository, located at the following URI:

```
basmati.amenesik.com/code/platform/modules/application_repository.git
```

To execute the AR, the bash file need to be launched:

```
application_repository/application_repository/run.sh
```

[1] https://www.djangoproject.com/
[2] http://www.django-rest-framework.org/

The Application Repository will start and will be listening at the address: http://[localhost]:8005/. It is possible to change the network port by modifying the file run.sh, where the port is indicated as a parameter at the end of the command.

3 Usage Example

In this section we show the capabilities of the AR, and perform the insertion of a sample application. As explained in Deliverable 3.5 [1], there are two ways for manipulating and retrieving data from/to the AR:

1. The admin Web-Interface, through which an admin user can manage all the repository data using a web user-friendly interface.
2. REST APIs providing POST, GET, DELETE and PUT for all the stored entities

In the remaining of this section we use the admin web interface. To access the admin area, it is required a username and password (default: basmati and basmati2018). Figure 1 exhibits the web interface for management of the content inside the AR.

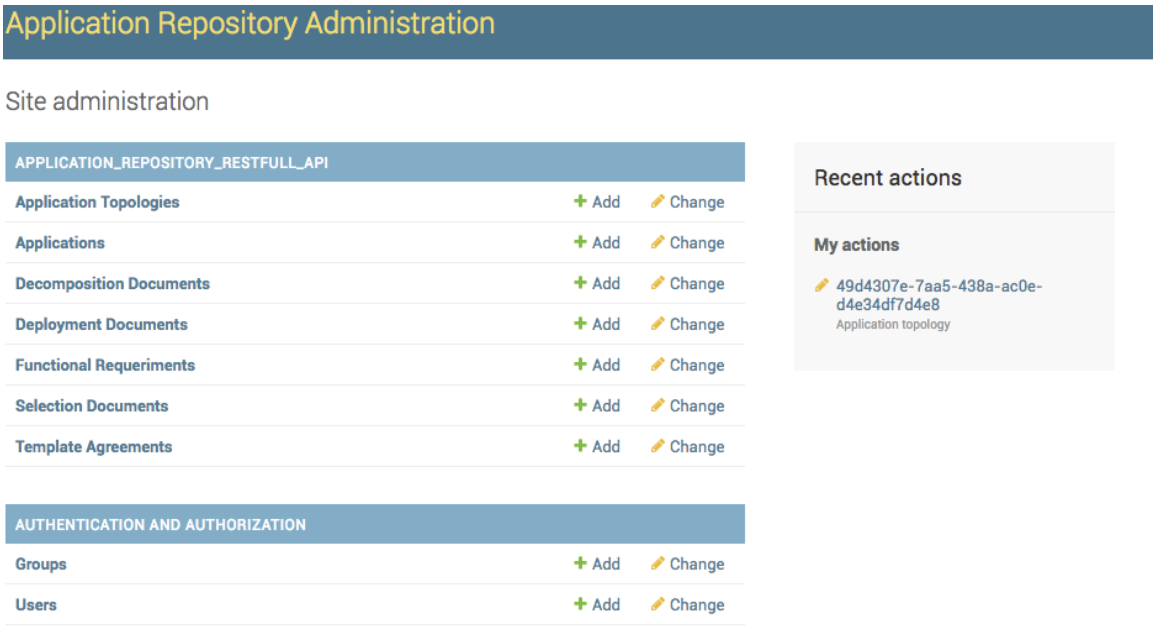

Figure 1. Admin home

To create a new application BEAM inside the AR, it is sufficient to add an entity in the "Applications" resource. In this case, we add a "Demo Application" and we provide a simple description (see Figure 2).

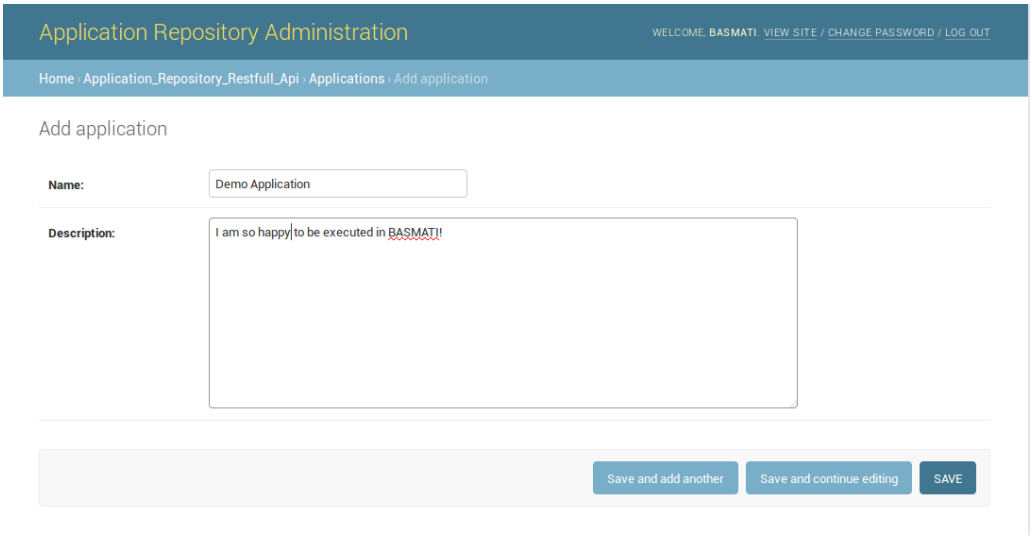

Figure 2. Adding an application

Upon insertion, the application is associated with a UUID given by the AR (see Figure 3). The UUID identifies the application in all REST call, and during all its life-cycle within the BASMATI platform.

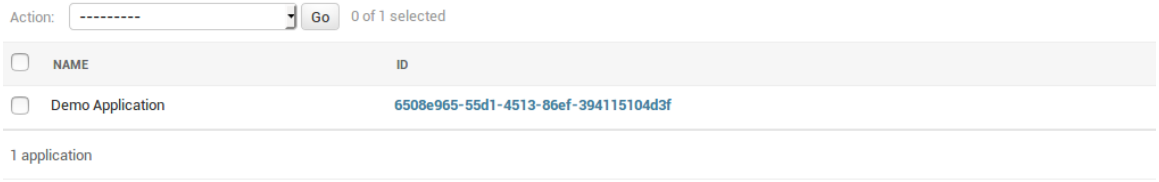

Figure 3. Applications List

The BEAM is composed by multiple documents. The first document that needs to be loaded (actions typically done by the application owner) is the Application Topology, which we add by populating with a new item the respective category (see Figure 4). Note that the topology is associated to the UUID of the application we just created. Similar document associated to the application can be added with a similar procedure using the web interface and, equivalently via the REST API. In fact, the rest API is what is used by the other components to populate the information of the BEAM during the "basmatize" (see Deliverable 3.5 [1]) process.

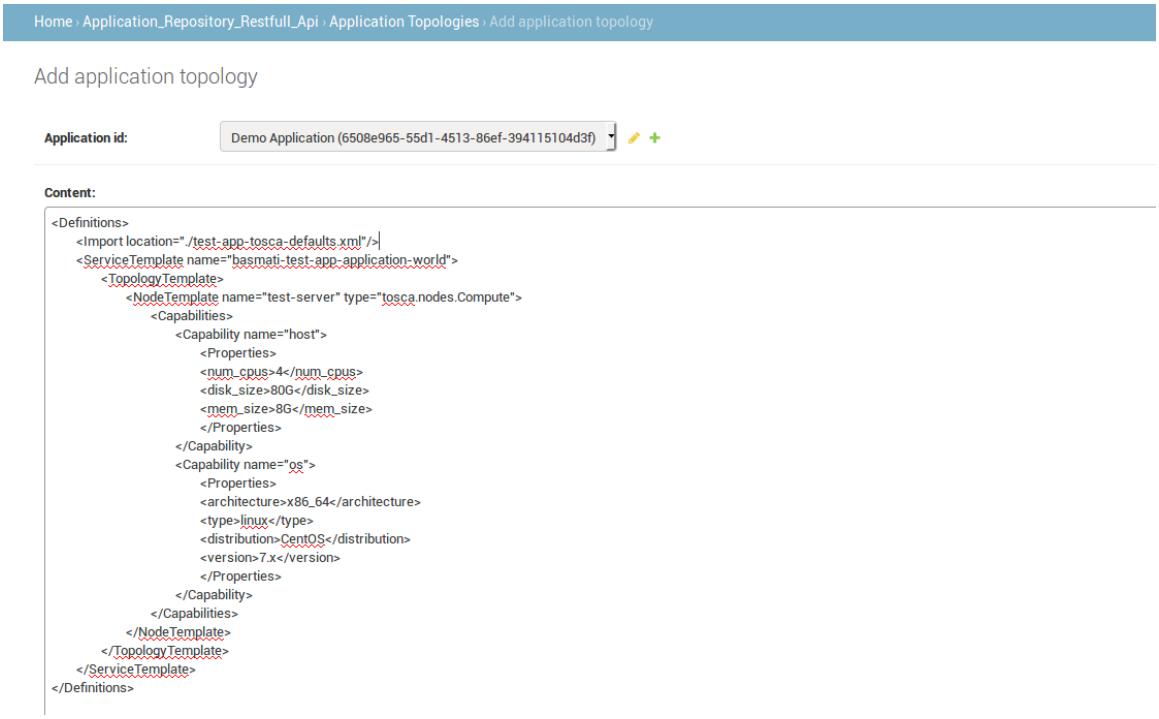

Figure 4. Adding application topology

Since the BEAM is designed to support application dynamicity, the AR is developed to accommodate multiple version for each kind of the document stored. Practically this means that, for example, more than one application topology can exist for a single application. To specify which is the current valid document at any given time, a boolean field "current application setting" is integral part of the data model of the application repository. For example, in Figure 5 the second application topology is the one marked as currently used.

Figure 5. List of available application topology

4 References

[1] "Deliverable 3.5: BASMATI Server- and Client-side Applications Adaptation and Reconfiguration: Design and Specification," 2018.