International Journal Of Scientific And University Research Publication

ISSN No 2364/2018

Listed & Index with
ISSN Directory. Paris

Multi-Subject Journal

Volum : (3) | Issue : 211 |

INTERNATIONAL JOURNAL OF SCIENTIFIC AND UNIVERSITY RESEARCH PUBLICATION

Research Paper

APPLIANCE MANAGEMENT FOR FEDERATED CLOUD ENVIRONMENTS

Mohammed Airaj || researche

ABSTRACT **Cloud infrastructures provide compelling features for scientific and engineering applications. Federated clouds additionally promise improved scalability via access to a larger pool of resources and improved service availability through geographically distributed redundant servers. Effective use of federated clouds requires the creation of portable appliances and consistent appliance management techniques. The StratusLab Marketplace, a platform-agnostic appliance registry, facilitates appliance management in a federated environment. This paper describes the Marketplace design goals, implementation, and security concerns. It also covers the planned improvements based on our experience of running this service in production for more than two years.**

KEYWORDS :Rapid resource provisioning, dynamic scaling,customized computing

INTRODUCTION

Rapid resource provisioning, dynamic scaling, and customized computing environments make cloud infrastructures a compelling choice for a wide range of scientific and engineering applications. Federated cloud infrastructures potentially offer further advantages such as improved scaling via access to a larger pool of resources and improved service availability through geographically distributed redundant servers.

Users, however, will only use federated cloud infrastructures if the advantages outweigh the additional overheads. Existing and well tested techniques for federated identity management, used in cluster, grid, and commercial services, can provide unified access to the federated clouds. Specifically for clouds, much work has been done on standardizing the cloud management interfaces (e.g. CIMI [1] and OCCI [2]). Unfortunately, the critical areas of appliance portability and management, required for consistent computing environments across cloud infrastructures, have received much less attention.

A distinguishing feature of the StratusLab [3] cloud distribution is its Marketplace, a platform-agnostic appliance registry that facilitates sharing of appliances and their use on multiple cloud infrastructures. This service and the associated appliance management techniques lower barriers for users in both federated and non-federated cloud environments. StratusLab provides a complete, open-source solution for deploying public or private “Infrastructure as a Service” (IaaS) cloud infrastructures and is designed to be both simple to install and simple to use. In addition to the Marketplace, it provides services similar to those in the more widely known distributions like OpenStack [4], OpenNebula [5], and CloudStack [6].

This paper first discusses (Sec. II) what is required for creating portable appliances. It then highlights the required appliance management features by describing the primary use cases (Sec. III) and provides examples of how these features are implemented in StratusLab and other cloud distributions (Sec. IV). The core of the paper (Sec. V–VII) describes the Marketplace: its design, implementation, and associated security concerns. A discussion of our experience in running this service and planned improvements based on that experience is then given (Sec. VIII). A description of a Marketplace deployed to support aparticular user community (i.e., bioinformatics) concludes the paper (Sec. IX).

II. PORTABLE APPLIANCES Before one can talk about sharing appliances, the appliances themselves must be portable [7] technically capable of running on different cloud platforms and generic enough to appeal to a number of users. The technical issues relate to the appliance format and contextualization.Appliance Format: Nearly every hypervisor uses a different, native appliance format; fortunately, tools exist to easily convert appliances between the common formats. Users can take advantage of these tools to generate alternate formats of their appliances. However to reduce the maintenance burden for users, cloud infrastructures (and/or hypervisors) ideally would accept all formats and make the necessary conversions automatically. StratusLab, for example, automatically converts appliances to the “raw” format it uses internally.

Appliance Contextualization: This allows an appliance to discover its “context” and to automatically configure services to work properly on a given infrastructure. Common mechanisms provide contextualization information via a disk (CDROM or floppy) attached to a virtual machine instance or

c 2013 IEEE via web server at a predefined local address. Image creators can make portable appliances that detect and use multiple contextualization mechanisms, although this requires more work. Cloud distributions can minimize the additional work by supporting multiple contextualization mechanisms. StratusLab supports CloudInit [8], HEPiX [9], and OpenNebula [10] contextualization. Fortunately, CloudInit is becoming a de facto standard and will simplify both appliance creation and contextualization support by cloud distributions.Creating portable appliances would be simpler if there were one standard format and one contextualization mechanism. Nonetheless portable appliances can already be created by detecting and using multiple contextualization mechanisms and by automated conversion of appliances between common formats. StratusLab, in fact, provides portable, minimal appliances for most popular Linux distributions.

III. ACTORS AND USE CASES Three core use cases—publishing an appliance, using an appliance, and authorizing an appliance—expose the required features and the actors for appliance management in both federated and non-federated environments. The primary actors and their roles are:

Creator Makes a new appliance and desires to publish the image for her own or someone else’s use. EndorserValidates an appliance against certain criteria and issues an endorsement to this effect. User A scientist or engineer who wants to find and to use existing appliances. Admin The cloud administrator responsible for maintaining the cloud services and the security of the platform.

A. Publishing an Appliance Despite automation, appliance creation is a tedious, errorprone, and lengthy process. Creators with the expertise to create new appliances often want to share their appliances with a larger community for wider use and better testing. Doing so requires publishing the appliance.To publish the appliance, the creators must make the appliance’s contents and metadata available. At a minimum, the metadata needs to include information about the appliance’s operating system, service configuration, and access parameters. The appliance’s contents and metadata may be provided either as separate files or as a single file.

B. Using an Appliance Users want to avoid the effort required to create an appliance by reusing an existing one, if possible. A central registry allows users to discover appropriate appliances based, for example, on the operating system, what services are enabled, etc. Complete appliance metadata is crucial for finding appropriate appliances.Users also want to verify the origin of the metadata and

the integrity of the appliance's contents. An endorsement of the appliance is critical for establishing trust in the appliance itself. The creator is usually also an endorser of an appliance,but appliances can and often are endorsed by multiple people. This allows, for instance, third party certification of appliances.Once users find a suitable appliance, they want to run an instance of that appliance on a cloud infrastructure. This requires transport of the appliance's contents to the cloud infrastructure. Although users can do this manually, cloud infrastructures should handle the transport transparently given the appliance identifier.

C. Authorizing an Appliance Most users of cloud infrastructures have little or no experience with system management. They are unfamiliar with best practices and techniques for securing machines, for example limiting SSH access and configuration of firewalls. Consequently, cloud administrators have a strong interest in ensuring that users run appliances that have been built with these best practices in mind.Before allowing a user to start an appliance, cloud administrators will want to authorize that particular appliance. Based on appliance metadata, administrators can define a policy that suits their needs, ranging from very restrictive policies, which allow only appliances endorsed by one particular person, to open policies, which permit all appliances except those with known security problems.

IV. EXISTING APPLIANCE MANAGEMENT TECHNIQUES

Based on the previous generic use cases, it is clear that all cloud distributions must provide tools or mechanisms for the:

- Generation of appliances,
- Storage of appliances,
- Efficient appliance transport, and
- Management of appliance metadata.

This section describes the approaches used to implement these features in StratusLab and other cloud distributions.

A. Appliance Factories (Generation) As discussed earlier, the manual creation of portable appliances remains an error-prone, time-consuming process. In response to this, it is not surprising that tools and services have appeared to simplify this task.

Services like Bitnami [11] are appliance factories that provide pre-built, ready to deploy appliances. These can be used as is, or treated as base appliances that a user can customize with their own software and services.There are,however, a larger number of command line tools that automate the process of creating an appliance. In general these follow a common pattern of installing a specified base operating system on a newly created disk image, then customizing that image. The customization methods vary, using predefined, user-generated templates or command line options. Examples of these tools include VMBuilder [12], VeeWee [13], and BoxGrinder [14].StratusLab follows this common pattern. "Image recipes" automate the creation of minimal appliances for most common Linux distributions, using the distribution's standard configuration mechanism (e.g. Kickstart for CentOS). These recipes are used to update automatically the base StratusLab appliances, which can subsequently be customized by users.

The StratusLab client includes a tool that automates the customization of the base appliances. The tool installs packages specified by the user and executes a user-defined script to configure the newly generated appliance. The new appliance is made available in the StratusLab storage service.

B. Appliance Repositories (Storage) No matter what cloud is used, a copy of the appliance contents must exist on the cloud before virtual machine instances of that appliance can be started. Cloud distributions store appliances in a variety of ways.The OpenStack project provides appliance discovery, registration and delivery via its "Glance" service. The appliances can be stored in a simple filesystems or object-storage systems like "Swift". Both metadata about registered appliances and the appliances themselves are exposed via the Glance API.Eucalyptus [15], an open-source Iaa S cloud distribution, provides an Amazon S3 interface to its "Walrus" storage service. Virtual machine images are stored/retrieved using HTTP put/get.OpenNebula uses the concept of a "Datastore" for storing appliances. Multiple datastores can be created backed by one of a selection of supported filesystem types. The method used to store and retrieve appliances depends on the type of datastore used (e.g. filesystem, iSCSI, Ceph). Sets of "transfer manager" scripts that handle the interaction with the storage backend are provided for each of the types.

For StratusLab, appliances can be stored in any web accessible location. The location of the appliance is contained in the metadata published in the Marketplace. This concept makes it possible to share appliances between StratusLab users, and also between users of different cloud infrastructures, because of the open accessibility and portability of StratusLab appliances (Sec. II).

C. Appliance Transport In a federated cloud environment, the appliances usually are made available outside of a particular cloud infrastructure and must be transported to the cloud before use. Most cloud distributions require the user to do this manually, but some tools exist to facilitate automated transfers.The vmcaster/vmcatcher tools developed within the HEPiX Virtualization Working Group [16], use the concept of subscriptions to an appliance list. It makes the download and transport of an appliance from the appliance list similar to using a system package manager. The downloaded appliances are verified against their X509 signatures and cached.In StratusLab, appliances are transported from a web server or from cloud storage. Based on the appliance identifier in the Marketplace, the transport of the appliance is done transparently by the cloud infrastructure. The downloaded appliances are then verified and cached in the persistent disk storage,

ensuring that the transport of a given appliance is done only once.Making the transfers transparent to users, requires that these tools be integrated with the cloud's appliance management workflows.

D. Appliance Registry An appliance registry allows users to search for existing appliances based on certain criteria. Any cloud distribution that allows users to share appliances must have a registry of some sort. Most distributions do provide such a service, although in many cases it is integrated with the appliance repository.

The Eucalyptus Image Store provides a set of base appliances that can be downloaded and imported into a local Eucalyptus cloud using a command-line client.The OpenNebula AppMarket allows registered appliance developers to upload appliances that users can then download to use in their local cloud. This is done using the OpenNebula command-line client, or through the "Sunstone" GUI.The European Grid Initiative (EGI) Applications Database (AppDB) [17], is a central service to which appliance lists can be published. This is achieved through integration with the vmcaster/vmcatcher (Sec. IV-C) tools, which are used to produce, sign, and upload a list. By using vmcatcher to subscribe to a published appliance list, appliances can be automatically downloaded.The StratusLab Marketplace is at the center of the appliance handling mechanisms in the StratusLab cloud distribution. It contains metadata about appliances and serves as a registry for shared appliances. In order to use and/or share an appliance, its metadata must be registered in the Marketplace.Once an appliance is created, StratusLab provides simple tools for building, cryptographically signing with a valid certificate, and uploading the metadata to the Marketplace. The Marketplace validates the metadata entry and verifies the email address of the endorser. If all the checks pass, the metadata will then be visible in the Marketplace and other users can search for the entry.

V. DESIGN AND REQUIREMENTS

The Marketplace provides a database of available appliances, allowing users to find appliances of

interest and administrators to validate those appliances. To make the database easily accessible it is implemented as a web service permitting both programmatic and browser-based access. The design and implementation are agnostic with respect to the cloud distribution, allowing any cloud distribution to interface to the Marketplace.Table I contains a detailed list of requirements for the Marketplace as an appliance registry and for the appliance metadata.1 The core requirements have been derived from the primary use cases and feedback from the StratusLab users and administrators.

1References prefixed with 'RU' or 'RS' in the following text refer to the requirements in this table.

T1: Original creator (A) endorses appliance providing basic information.

T2: Third party (B) endorses appliance, optionally providing complementary information.

T3: Endorser A updates information about the appliance.

T4: Endorsement from Endorser A expires; other endorsements still valid.

T5: An administrator (C) encounters problem with appliance and deprecates it.

T6: Endorser B also deprecates appliance.

A. Timeline A core concept within the Marketplace is the appliance timeline, a complete history of all endorsements related to a given appliance (see Fig. 1). An appliance can be endorsed by more than one person to allow for third party validation and approval of appliances. [RU3] Moreover, a particular endorser may change her endorsement over time, updating the metadata data associated with the appliance or explicitly deprecating the appliance. [RU2] Endorsements are also timelimited, containing an explicit validity period.

Users of the Marketplace can retrieve the full history, for example to conduct an audit on why a particular appliance was authorized at a particular point in time. Normally however, the Marketplace users only want to see the current endorsements for an appliance, that is the list of the latest, non-expired endorsements from all of the endorsers of an appliance. This allows users and administrators alike to decide if an appliance is currently valid.

B. Separation of Metadata and Appliance Contents A conscious design decision was made to separate the storage and transport of appliance contents from the Marketplace implementation. Storing the appliance contents outside of the Marketplace makes it easier to:

• Scale the Marketplace implementation,

• Create mirrors of the Marketplace,

• Ensure the implementation is independent of the transport protocol

• Allow owners of the image to control access to the appliance contents, and

• Relieve the operator of the Marketplace from copyright and licensing concerns.

It also allows the StratusLab distribution to take advantage of standard web servers or other cloud storage services for the appliance contents.

VI. IMPLEMENTATION The Marketplace implementation uses standard web technologies to create a service accessible programmatically and via a web browser. For programmatic access, the service exposes an interface over HTTP(S) using RESTlet [18], a Java

TABLE I. REQUIREMENTS

RU1 Anyone with a valid email address can upload metadata descriptions to the Marketplace.

RU2 Users can "replace" existing metadata descriptions by uploading a new signed description(s).

RU3 Multiple metadata entries may be associated with a particular appliance, allowing third parties to endorse images.

RU4 All validated descriptions uploaded to the site must always be available to provide a timeline of the metadata evolution.

RU5 Users must be able to search the metadata database on a reasonable subset of the possible keys, for example the image identifier and the endorser's email address.

RU6 The registry should allow the metadata to be downloaded in alternate formats, notably JSON and HTML.

RU7 The service must be easy to access from all programming languages (including scripting languages) and usable from a web browser.

RU8 The underlying schema for the metadata entries must be flexible and extensible, to account for different and evolving needs.

RU9 Entries should contain at least one location from which the appliance can be obtained; entries without a location are appropriate only for deprecated appliances.

RS1 Metadata entries must be cryptographically signed with the endorser information matching the information in the certificate itself.

RS2 Metadata entries must contain a valid email address, which is confirmed for each entry upload.

RS3 Users must be able to download the original signed metadata in the RDF/XML format from the registry for verification.

RS4 All entries must contain a creation date for the endorsement. The server must only accept descriptions with a creation date more recent than the current latest.

RS5 It must be possible to unambiguously associate an entry to an appliance and to verify the integrity of the appliance.

framework for RESTful [19] services. HTML representations and browser interactions are provided with a combination of FreeMarker [20] (a Java template engine library), CSS, JavaScript, and JQuery [21].

The implementation allows any cryptographically signed metadata entry with a valid email address to be uploaded to the Marketplace. This allows open, but not anonymous, posting to the service. All of the validated entries can be read without authentication. [RU1]

A. Identifiers The separation of the appliance metadata and the appliance contents requires an unambiguous mechanism for matching the two. StratusLab uses the SHA-1 hash of the appliance contents to generate an unambiguous, intrinsic identifier for the image. The identifier is the 27 character string generated by encoding the SHA-1 checksum with the base64url encoding. [RS5]

B. Metadata Semantic web technologies were designed to manage metadata about (third-party) resources identified with a URI.

Consequently, they are ideally suited to this situation in which the Marketplace must manage metadata about appliances. These technologies already provide standard formats for the metadata (RDF/XML [22], [23], [24]) and query languages (SeRQL [25], SPARQL [26]). The Marketplace implementation makes use of the OpenRDF Sesame [27] framework to provide search capabilities over the metadata database. [RU5] Working with RDF also requires an agreed vocabulary to ensure a common semantic meaning of the metadata tags. The Dublin Core Metadata Initiative has published a vocabulary [28] that can be used for much of the appliance metadata descriptions. This is complemented by a vocabulary specific to StratusLab, which includes, for example, the location URL(s) of the appliance. [RU9] Using RDF also allows additional metadata fields to be specified (in separate namespaces) to complement the standard fields, making it possible for users to extend the schema with application-specific metadata. [RU8] RDF with Dublin Core maintains a good balance between machine and human readability. (See Fig. 2 for an abbreviated example for a CentOS appliance.)As the overall aim is to provide a high-level description of an appliance these metadata standards are more suited than something more heavy-weight, such as the Open Virtualization Format (OVF). OVF [29] describes the packaging and distribution of a full virtual machine rather than just an appliance, and so would contain a large volume of

additional information that is not particularly relevant for cloud users and administrators. It should be noted however, that the RDF metadata descriptions are easily extensible and could include the OVF metadata if necessary. Similarly, the use of OVF to package the appliance itself is not precluded.To validate the metadata associated with a particular appliance, it is necessary to sign individual entries cryptographically. As the raw format used for the metadata entries is XML, the XML Signature [30] specification is reused. Conveniently, modern Java runtime environments include this as a standard part of the API. [RS1]

C. REST Resource URLs REST over HTTP provides convenient access to the service via a browser and facilitates programmatic access from all programming languages. The mapping between URLs and service resources essentially defines the API of the service. [RU7]
Table II provides the URL mapping for the Marketplace along with the actions associated with the given HTTP verbs. (The DELETE and PUT actions are not supported by any URLs.) Within the table "identifier" refers to the 27 character image identifier, "email" refers to the endorser's email address, and "date" refers to endorsement date written in the format yyyy-MM-ddThh:mm:ssZ. All of the URLs support XML and HTML representations. Individual metadata entries also provide a JSON representation. [RU6]

D. Storage and Query of Metadata
Two copies of successfully validated and confirmed metadata are stored on the filesystem. The original uploaded file is saved unmodified, while a second copy stripped of the XMLTABLE

II. CORE REST RESOURCES GET redirects to /metadata resource/endorsersGET list of endorsers in database
OPTIONS number of endorsers; last update time /endorsers/hemaili GET statistics about particular endorser OPTIONS number of entries; last update time /metadata/?hqueryi GET list of identifiers and selected fields (query terms of (identifer, email, and created can be used to refine list)
POST create new metadata entry OPTIONS number of entries; last update time/metadata/hidentifieri/hemaili/hdatei GET unique metadata entry/query
GET form for simple query of service
POST submit query/upload
GET form for browser upload of metadata entry
POST create new entry via post to /metadata
signature is added to the Sesame RDF repository. Storing the metadata in the repository allows SPARQL queries to be easily supported. A request for a specific metadata entry in XML format returns the original signed file. [RS3]

VII. SECURITY CONSIDERATIONS The Marketplace and the information contained within the Marketplace play key roles in maintaining the security of the cloud infrastructures. However, the security policies both for the users and for the cloud administrators can vary widely and consequently, the Marketplace itself does not define or enforce any security policy. It instead provides the appliance metadata allowing both users and cloud administrators to make informed decisions about the appliances.
To maintain confidence in the information provided by the Marketplace, it must securely provide complete, accurate information about the appliances. We have identified a number of security concerns and describe how the Marketplace solves them.

A. Altered Appliances Because the appliance metadata is separated from the appliance contents, there is a danger that the appliance contents could be altered, either accidently or maliciously. As described above, the appliance identifier is based on the SHA-1 hash of the appliance contents ensuring a very reliable link between the metadata and the appliance contents. [RS5] Although modifying an appliance while maintaining the SHA-1 hash is difficult, it is a remote possibility, allowing an altered appliance to masquerade as the original. To minimize this possibility, additional information is provided in the metadata descriptions: the size of the file in bytes and the MD5, SHA-1, SHA-256, and SHA-512 hash values. The likelihood that someone can create an altered appliance with exactly the same length and multiple checksums is negligible. [RS5]

B. Verification of Uploaded Metadata When new metadata entries are uploaded to the Marketplace, the service validates the entry before accepting it. To validate an entry, the server:
• Verifies that the metadata is a valid, signed RDF/XML file, following the defined schemas and conventions,
• Accepts only entries endorsed after the most recent entry for a particular appliance, [RS4] and
• Confirms the email address of the entry. [RS2]
Only validated metadata entries are visible from the standard Marketplace interface.

C. Altered Appliance Timeline The Marketplace must ensure that all of the data concerning an appliance is available. By design, the Marketplace never removes metadata entries—the entire appliance timeline is always available. [RU4] By default however, only the current endorsements are provided as these are the entries needed to make decisions about the validity of an appliance.
The Marketplace does not allow the history of an appliance to be altered. As described above, the Marketplace does not accept entries with a timestamp earlier than the latest entry in the timeline. It also validates the endorser to avoid one endorser from impersonating another.
The Marketplace must ensure that the data transmitted to users is not altered, for example by a third-party removing deprecation notices from the returned information. To ensure the integrity of the returned information, the Marketplace only transmits information over a secured communication channel.

D. Compromised Marketplace Server If someone were to take control of the Marketplace server, he could not alter individual metadata entries as those are signed by the endorsers' private keys which are not stored on the server. However, he could delete entries making, for instance, deprecated images appear valid.
This is a significant risk and the server must be operated according to modern best practices to avoid this. In addition, backups of the information should be kept and periodically compared to the current information to detect any such attack.

VIII. PLANNED IMPROVEMENTS In parallel with its software development, the collaboration operates a federated cloud infrastructure with sites in Orsay, France and Athens, Greece. These sites share a common user authentication framework and Marketplace allowing users to allocate resources and to use appliances on either site. The collaboration uses this production cloud to validate its software in real world conditions.
As a core service, the Marketplace is accessed frequently by users to find appliances and by the cloud infrastructures when authorizing requests for new virtual machine instances. Figure 3 shows statistics and the frequency of requests for the relatively light period in August. The number of requests endorsers 44 current appliances 105 deprecated appliances 170 expired appliances 832 Weekend days are shaded. The current number of appliances and endorsers are also shown.
will scale with the number of users and the number of virtual machines being started.
Overall the service has performed well, with minor problems being addressed as the software evolves over time. Some outstanding issues and potential solutions are described below.

A. Availability Having a central Marketplace instance allows users to easily find all of the appliances from a single location. Similarly, it allows image creators to upload the metadata just once. However, the Marketplace is consulted every time a new machine instance is launched to check if an appliance has been deprecated. Consequently if the Marketplace is not available, new instances cannot be created on any cloud relying on the Marketplace. Future iterations of the Marketplace must provide redundancy and high-availability of the Marketplace service.To provide for this, a replication scheme will be implemented that allows for multiple Marketplace instances to be deployed, each maintaining a local copy of the metadata entries. As all the information required to rebuild the metadata index stored in Sesame is the set of raw metadata files, it is only these that need be replicated. A potential solution would be to use a Git repository as the core 'database' for the metadata entries, with each Marketplace updating its index periodically from a local clone of the global repository.

B. Data Protection By design the appliance metadata is considered public. In reality, however, both users and administrators would like to restrict the visibility of the appliance metadata for certain appliances. Many cloud administrators would like to run a "private" Marketplace to limit the visibility of certain appliances while still taking advantage of the central, public Marketplace instance.
There must be a mechanism for federating Marketplace instances in the future and the move to using Git for metadata file management may also facilitate the federation of different Marketplace instances.

C. Appliance Quality Although the metadata contains a significant amount of information about an appliance, it does not contain information about how well the appliance functions for users. A common
request has been to add social features to the Marketplace to allow users of an appliance to leave comments and to signal problems with the appliance itself. A possible approach to add these features without overly complicating the Marketplace implementation would be to make use of an external service such as Disqus [31].

D. Appliance Evolution Appliances naturally evolve as operating system updates are applied and new services are added. However each time an appliance is updated, the SHA-1 hash and the corresponding appliance identifier change. This makes it difficult for users to track the evolution of an appliance and impossible to use a stable identifier for, for example, the latest version of the CentOS appliance.
Recently a 'tag' feature has been added to the Marketplace. This allows an endorser to provide a simple label for a series of appliances, where the tag (namespaced by the endorser's email) will always resolve to the latest appliance identifier. By using the tag, users can always use the latest version of an appliance without having to find the associated identifier manually.
A complementary and more rigorous solution would be to make use of the Dublin Core terms replaces and isReplacedBy. This would provide a link in each metadata entry to the previous and next entries in the evolution of the appliance. The StratusLab tools must be updated to simplify the use of these terms to ensure that they are widely used.

IX. BIOINFORMATICS APPLIANCE METADATA In addition to the reference cloud infrastructure, a cloud infrastructure devoted to biology (IDB cloud [32]), with a separate Marketplace instance, has been deployed at IBCP2. IDB cloud is the first brick of the future federated cloud infrastructure of the French Bioinformatics Institute. Biologists and bioinformaticians frequently combine multiple software packages (from the thousands available in the international community) to study their data with their own or public analysis pipelines. With the advent of the cloud, experts now create customized appliances, but these are often not adequately described or easily located. Helping scientists easily identify suitable appliances containing the required software packages is essential.
Operating a separate Marketplace with a limited thematic scope already allows users to find appropriate appliances more easily. It also reduces "noise" in the central Marketplace from iterative attempts to create working appliances. More importantly, however, it allows visibility contraints for the bioinformatic appliances, such as confidentiality for specific projects, to be respected.
To further facilitate searches for appropriate appliances, the generic appliance metadata schema (Fig. 2) has been extended with additional elements related to bioinformatics tools (Fig. 4). These metadata can be used to select suitable bioinformatics appliances containing the required tools by searching for the tools themselves (e.g. BLAST or ClustalW2) 2Institut de Biologie et Chimie des Prot´eines in Lyon, France Appliance creators enhance the descriptions by appending 'bio:tool' entries in the appliance metadata (see Fig. 5).As with all metadata, the biotool information is indexed by the Marketplace, allowing bioinformaticians and biologists to search the Marketplace with a SPARQL query to find an appropriate appliance. SPARQL, however, operates at a rather low level, so to further simplify searches, the Marketplace was linked with the IDB bioinformatics web portal. The portal is synchronized (manually at this point) with the list of suitable appliances from the Marketplace. The portal provides users with popup menus to filter appliances according to the above criteria. With the Marketplace, additional metadata, and the portal, it is trivial for users to find and run the right appliance.

X. SUMMARY The StratusLab Marketplace has been operated in a federated cloud environment for more than two years. Over that time, it has evolved, taking into account operational and user concerns. A roadmap has been defined to make the service even more reliable and functional. With its platform agnostic design, we look forward to interfacing the service with another IaaS cloud distribution.

ACKNOWLEDGEMENTS StratusLab was co-funded by the European Commission through the 7th Framework Programme (Capacities), contract number INFSO-RI-261552 from June 2010 to May 2012. The authors also gratefully acknowledge support from the other members of the StratusLab collaboration and their institutes.

CONCLUSION

StratusLab was co-funded by the European Commission through the 7th Framework Programme (Capacities), contract number INFSO-RI-261552 from June 2010 to May 2012. The authors also gratefully acknowledge support from the other members of the StratusLab collaboration and their institutes.

ref_str

1. **Distributed Management Task Force,** Inc., "Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol," http://dmtf.org/sites/default/files/standards/documents/DSP0263 1.0.1.pdf.
2. **Open Grid Forum**, "Open Cloud Computing Interface - RESTful HTTP Rendering," http://www.ogf.org/documents/GFD.185.pdf.
3. **C. Loomis**, M. Airaj, M.-E. B´egin, E. Floros, S. Kenny, and D. O'Callaghan, "Stratuslab cloud distribution," in European Research Activities in Cloud Computing, D. Petcu and J. L. V´azquez Poletti, Eds. Cambridge Scholars Publishing, 2012.
4. **The OpenStack Foundation,** "OpenStack," http://www.openstack.org.
5. **OpenNebula Project,** "OpenNebula," http://opennebula.org/.
6. **Apache Software Foundation**, "CloudStack,"

http://cloudstack.apache. org/.
7. **D. Petcu**, "Portability and interoperability between clouds: challenges and case study," in Lecture Notes in Computer Science, vol. 6994, 2011, pp. 62–74.
8. **Scott Moser,** et al., "CloudInit," https://launchpad.net/cloud-init/.
9. **O. Synge**, et al., "Contextualisation," in HEPIX Virtualisation Working Group, 2012. [Online]. Available: http://grid.desy.de/vm/hepix/vwg/ doc/pdf/Book-a4.pdf
10. **R. Moreno-Vozmediano**, R. S. Montero, and I. M. Llorente, "Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures," Computer, vol. 45, no. 12, pp. 65–72, 2012.
11. **"Bitnami**," http://bitnami.com/.
12. **"Virtual Machine Builder**," https://launchpad.net/vmbuilder.
13. **"veewee,"** https://github.com/jedi4ever/veewee.
14. **"BoxGrinder,"** http://boxgrinder.org.
15. **Eucalyptus Systems**, "Eucalyptus," http://www.eucalyptus.com/.
16. **O. Synge**, et al., "Virtual machine image transfer," in HEPIX Virtualisation Working Group, 2012.
17. **EGI Applications Database**, "AppDB," https://appdb.egi.eu/.
18. **"RESTlet–RESTful web framework for Java,"** http://www.restlet.org/.
19. **Roy Thomas Fielding**, "Architectural Styles and the Design of Networkbased Software Architectures," Ph.D. dissertation, University of California, Irvine, 2000.
20. **"FreeMarker–Java Template Engine Library,"** http://freemarker. sourceforge.net/.
21. **Eric Sarrion**, jQuery UI. O'Reilly Media, 2012.
22. **"RDF/XML Syntax Specification (Revised)**," http://www.w3.org/TR/ 2004/REC-rdf-syntax-grammar-20040210/.
23. **"RDF Primer,"** http://www.w3.org/TR/2004/ REC-rdf-primer-20040210/.
24. **"RDF Vocabulary Description Language 1.0: RDF Schema,"** http: //www.w3.org/TR/2004/REC rdf-schema-20040210/.
25. **"SeRQL–Sesame RDF Query Language,"** http://www.openrdf.org/doc/ sesame2/users/ch09.html.
26. **"SPARQL Query Language for RDF,"** http://www.w3.org/TR/2008/ REC-rdf-sparql-query-2008011.
27. **"Sesame–Open source framework for storage,** inferencing and querying of RDF data." http://www.openrdf.org/.
28. **DCMI Usage Board**, "DCMI Metadata Terms," http://dublincore.org/ documents/2010/10/11/dcmi-terms/.
29. **Distributed Management Task Force**, Inc., "Open Virtualization Format Specification," http://dmtf.org/sites/default/files/standards/documents/ DSP0243 2.0.0.pdf.
30. **Mark Bartel**, et al., "XML Signature Syntax and Processing (Second Edition)," http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/.
31. **"Disqus,"** http://www.disqus.com. [32] "IDBcloud," https://idee-b.ibcp.fr/cloud.html.

IJSURP Publishing Academy

International Journal Of Scientific And University Research Publication

Multi-Subject Journal

Editor.

International Journal Of Scientific And University Research Publication

+965 99549511

+90 5374545296

+961 03236496

+44 (0)203 197 6676

www.ijsurp.com