# The Design and Implementation of the Service Hosting and Delivery Platform Based on Cloud

Yanqiao Sun[1, a], Yanlei Shang[2, b]

[1] The State Key Lab of Networking and Switching Beijing University of Posts and Telecommunications, Beijing, 100876, China

[2] The State Key Lab of Networking and Switching Beijing University of Posts and Telecommunications, Beijing, 100876, China

[a]email: yanqiao_sun@163.com, [b]email: shangyl@bupt.edu.cn

**Keywords:** Service Hosting; Service Delivery; Service Catalog; Cloud Computing

**Abstract.** With the arrival of big data and Cloud computing era, it is a huge issue of how to organize the application services. Organizations and companies usually have plenty of applications range from web, mobile or middleware, etc. Because the quantity and diversity of services, they are delivered through different platforms. The proposed Service Hosting and Delivering Platform is a multi-application platform to deliver and consume the services in a loosely coupled manner, which is hosted on OpenStack, thus solving the problem in cross-platform service delivery. In addition, it supports the service lifecycle management and we also provide user-friendly web interfaces. From the end-users' perspective, they can consume the services on the cloud with the push of a button expediently.

## Introduction

Along with the development of science and technology, especially the development of distribute computing, and virtualization, the concept of cloud computing becomes more and more popular. Cloud computing is a method of sharing infrastructure which can manage plenty of physical resources uniformly [1]. Cloud computing [2] is a significant trend with the potential to increase agility and lower costs.

Meanwhile, there is a great increase in the quantity of applications as well as variety. Enterprises usually need to deliver various applications across platforms, which can include such things as web applications, mobile apps, even system middleware services. In general, we deliver web applications, mobile apps and middleware applications through portal sites, app store or market place, and PAAS (Platform as a Service) [3] respectively. It can be time-consuming.

However, to date, almost all the solutions to deliver services only support a single type of application. For example, app store [4] is a market place only for mac apps; Google play [5] serves as a store for Android operating system; Salesforce [6] only supports CRM (Customer relationship management) products.

In this paper, we proposed a Service Hosting and Delivery Platform. For higher resource utilizing efficiency, we implemented it based on OpenStack [7]. All the resources and services are hosted on the cloud. In addition, it support delivering and consuming multiple application services.

## Architecture of Service Hosting and Delivery Platform

The Service Hosting and Delivery Platform mentioned in this paper adopts B/S architecture which is shown in Figure 1. Exposing RBAC (Role Based Access Control), Service Catalog and Lifecycle management functions through REST API, it's quite easy to use and integrate with other modules.

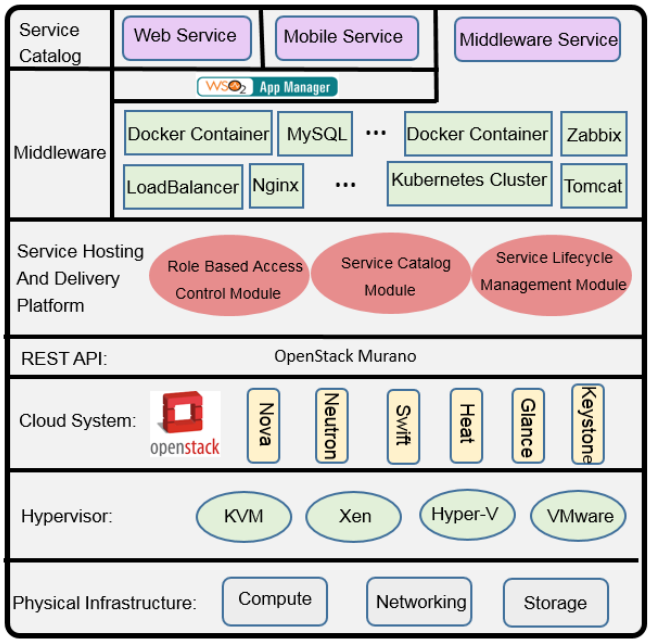



Fig.1. The Architecture of the Service Hosting and Delivery Platform

The bottom layer is mainly responsible to realize the unification of physical resources allocation and management through OpenStack, which is a cloud operating system that controls large pools of compute, storage, and networking resources. It supports various virtualization technologies as its Hypervisor, such as KVM, Xen, Hyper-V, VMware, and so on. In addition, OpenStack has several core components to manage different resources and services. For example, Nova manages the compute resources, Neutron manages the virtual networks, Swift is the Object storage service, heat is the main project in the Open Stack Orchestration program and Keystone is the service which is responsible for authentication.

The upper layer is our Service Hosting and Delivery Platform based on OpenStack. The Murano Project introduces an application catalog to OpenStack, enabling application developers and cloud administrators to publish various cloud-ready applications in a categorized catalog [8]. It gives a solution to publish applications and services, deployment rules and requirements included. Then WSO2 App Manager is a solution to provide controlled access to several applications for many users in an organization [9], which supports publishing Web apps as well as iOS, Android, Hybrid and Web types of mobile apps with app versioning and lifecycle management. We integrate OpenStack Murano and WSO2 App Manager to fulfill our platform. With Murano, we can offer any types of middleware applications, such as LoadBalancer, Tomcat, MySQL, etc. With WSO2 App Manager, we can support multiple web and mobile applications as our services. The both of them provide a complete REST API to help users operate resources and services conveniently. The proposed platform realized three main modules: Role Based Access Control (RBAC) module, Service Catalog Module and Lifecycle Management module. The RBAC module is responsible for managing users and their authorization levels. The service end-user can only access services tied to his user-role and publicly available. With the help of the module, we can make centralized control over many users and various applications. Service Catalog module, an app store where users can create, publish, browse, choose, pick and request apps for them to do their jobs more efficiently that you make available to them. Then the goal of the Lifecycle management module is to manage service lifecycle from cradle to grave: create, publish, subscribe, delete, retire, etc.

The Service Hosting and Delivery Platform proposed in this paper supports publishing and consuming middleware services, web services as well as iOS, Android, Hybrid and Web types of mobile apps. To the one whose customized requirement is not high, he can use the web and mobile services directly. On the contrary, he can develop a new application using the middleware services, such as Tomcat, MySQL, PHP, etc.

The platform provides users a rich service catalog including catalog management. From the application creator’s perspective, the platform will support creating almost all types of services,

thus making him need not to cross platforms. From the publisher's perspective, it provides a way to publish applications and services, including access rules and requirements, suggested configuration, etc. From the subscriber's perspective, it is able to access all the services in a centralized single location just like an application market.

## Implementation of Service Hosting and Delivery Platform

The platform using B/S architecture described in the last section is implemented by the several modules: RBAC Module, Service Catalog Module and Lifecycle Management Module. We will make a brief introduction of them in the following.

**Role Based Control Access Module.**

The module provides the ability to manage users and roles in the system. Users are assigned to one or more roles which determine the user's access or privileges to system resources and the actions he can perform. The Service Hosting and Delivery Platform involves three major roles: Creator, Publisher and Subscriber. All the user roles in this system are shown in Table 1.

Tab. 1. User Roles Summary

| Role | Description |
|---|---|
| Administrator | Have full power over the system and can do absolutely everything. |
| Creator | Create apps and to submit them for reviewing before they are published into the Application Catalog. |
| Publisher | Manage and maintain the lifecycle of an application by approving/rejecting, publishing/unpublishing, and deleting them. |
| Subscriber | Have access permission of the Application Catalog to subscribe to published applications, view the documentation. |

Users with Administrator's role can create more Administrators, add or remove existing users and change the user roles. Then it can be written in the database. We use MySQL instead. They have complete control over users as well as services. Nothing is off-limits for them, including deleting the entire system. By default, the admin user is assigned with Creator, Publisher and Subscriber roles. Therefore, considering security requirements, we have only one administrator in our system and will give usual users the minimal set of privileges required to ensure a system of least privilege.

**Service Catalog Module.**

The main function of the module is to provide end-user to browse, search and consume application services. Meanwhile, the service can be various. In addition, it supports complete service management, such as creating, publishing and managing all aspects of an application service. Then, we will describe its main functions respectively.

*Multiple services.*

The platform is aimed at building a service eco-system, which supports these three types of service.

1) System Middleware Service

The services may be a LoadBalancer, an Apache Tomcat server, a MySQL server, REDIS, Kubernetes or a cluster composed with these application services. It can offer user an environment with some functionality with auto-scaling and self-healing.

2) Web Application Service

Any type of application works in the browser can be imported in the system.

3) Mobile Application Service

The platform supports iOS, Android, Hybrid and Web types of mobile apps. They all can be uploaded, subscribe, and installed or browsed on your device.

*Application Service Introduction Process.*

The multiple services described above can be introduced into the Service Catalog. The process of an application service introduction consists of three steps:

1) Upload application service:

Creator can add new application service and submit a review request.

2) Review and publish the uploaded service Module:

Publisher will receive the request and then evaluate the service to decide whether approval or reject the request. Only when the request is approval and publisher has published the service, can it be browsed in the service catalog.

3) Subscribe service Module:

Subscriber can use service catalog to find services quickly and easily. The services are imported grouped by categories for easier navigation. It can be subscribed as self-application.

*Service Lifecycle Management.*

Service Lifecycle Management main focuses on exposing the actual IT functionality, which is one or more aggregated resources exposed as a single unit of management. We call the REST API of Murano and WSO2 App manager to manage throughout its lifecycle from cradle to grave through managing the application service. In this context, the term resources refer to any type of application service that can be encountered in a datacenter environment. This can include such things as servers, storage, networks, operating systems (OSs), applications, or middleware. The service lifecycle is shown in Figure 2.

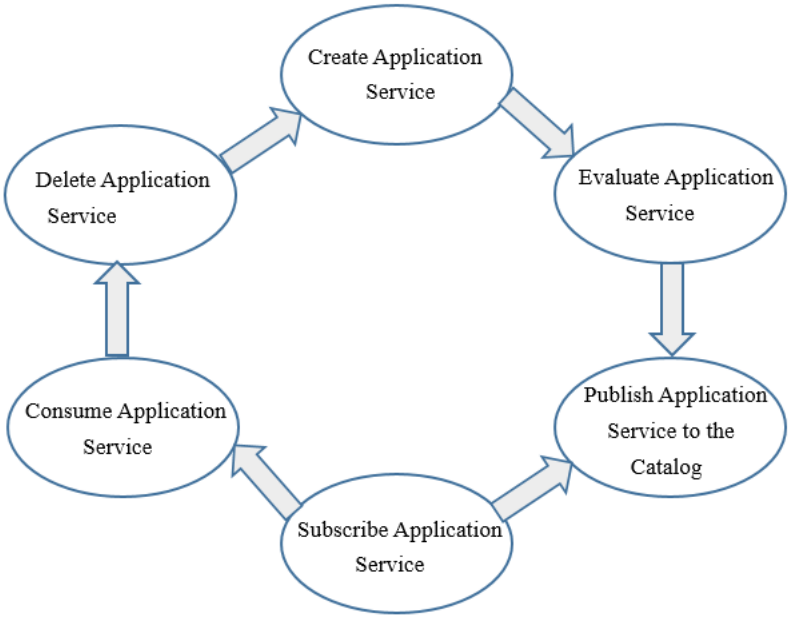



Fig.2. Lifecycle of an application service

For all the functions, we have implemented user-friendly web interfaces for users to manage and consume services.

## Experiments

After setting up a cloud computing environment based on OpenStack and deploying our system into the Tomcat container, we have conducted a series of experiments. We evaluate it in three aspects: RBAC Module, Service Catalog Module and Lifecycle Management Module. Our goal is to validate that the system can work well and it is quite stable.

**Experiment of RBAC Module.**

Users with Administrator's role can create more Administrators, add or remove the existing users and change the user roles. Therefore, we login the system with admin user and add three users first, the user list is shown in Figure 3.

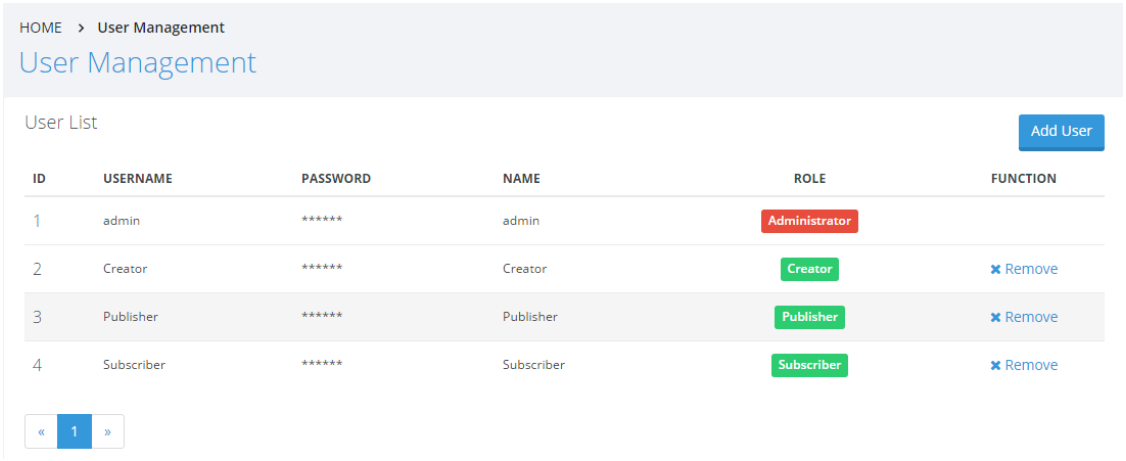



Fig.3. Users and roles

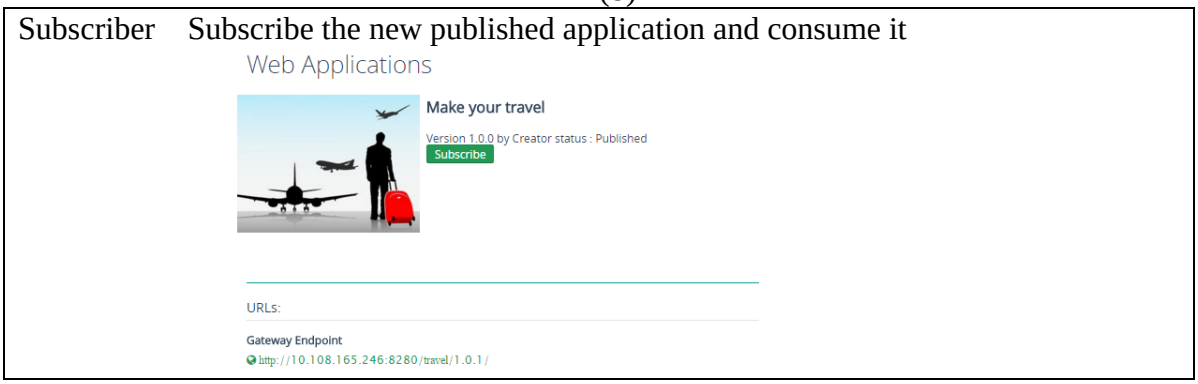
In this part, we can only simply test the user management. It is the basis of the entire system. So we create users first, and the functions will be shown with the following experiments.

**Experiment of Service Catalog module.**

The main purpose of this part of test is to verify the Service Catalog module in three user roles: creator, publisher and subscriber. To clarify, different roles have different permissions. As we have mentioned above, creator is responsible to upload an application which can be a Murano package, web URL, Android, iOS or some hybrid mobile apps as a service; publisher need to approval or reject the uploaded service through some evaluations; subscriber can subscribe the service and consume it expediently. We use the users created in above step and take web application as an example, the process is shown in Figure 4 (a), (b) and (c).


(a)


(b)




(c)

Fig. 4. The Full Process of service Introduction

From the test result, we can see it works fine. In addition, when we send a request to introduce

and consume a middleware application and mobile apps, we also get the similar information as above. Hence we can make sure that the Service Catalog Module is working smoothly.

**Experiment of Lifecycle Management module.**

The main purpose of this experiment is to test the Service Lifecycle Management Module. Only publisher can access the module. He can reject, approval, publish, unpublish and terminate the service. We have done some tests based on the above steps, it shows that with the Lifecycle Management Module, all resources associating the services can be managed uniformly by publisher. Of course, if a middleware application is no longer needed or the reservation for its resources expires, it will be deleted by publisher. And all resources that were allocated by the middleware service instance are returned to their respective pools such that they can then be reused by other cloud service instances.

## Conclusion

With the rise of the Cloud Computing and big data, the method to provide network services and the demands for services has changed greatly. We usually want to provide services easily instead of crossing different platforms. And more often, we need to figure out the necessary applications and services in the vast amounts of data. Under this background, we design and implement a Service Hosting and Delivery System hosted on OpenStack for higher resource utilizing efficiency. The system supports centralized control access and lifecycle management of services, which provides a new solution to deliver and consume services. It should greatly facilitate the management and lessen the burden of IT and developers. Maybe it can change the way on the use of services.

## Acknowledgement

This research is supported by the National Key Technology Research and Development Program of China (Grant No. 2012BAH94F02); National Natural Science Foundation of China under Grant No. 61132001); National High-tech R&D Program of China (863 Program) under Grant No. 2013AA102301; Project of New Generation Broad band Wireless Network under Grant No. 2012ZX03005008-001.

## References

[1] Cloud computing. https://en.wikipedia.org/wiki/Cloud_computing

[2] Michael R. Head, Anca Sailer, Hidayatullah Shaikh, Mahesh Viswanathan “Taking IT Management Services to a Cloud” 2009

[3] https://en.wikipedia.org/wiki/Platform_as_a_service

[4] https://en.wikipedia.org/wiki/App_store

[5] https://en.wikipedia.org/wiki/Google_Play

[6] https://en.wikipedia.org/wiki/Salesforce.com

[7] http://www.OpenStack.org/

[8] https://wiki.OpenStack.org/wiki/Murano

[9] http://wso2.com/products/app-manager