Universal API for Managing Multiple Cloud Platforms

Department of Computer Science and Engineering
School of Engineering and Computer Science
BRAC University

Advisor : Dr. Md. Haider Ali
Co-advisor : Md. Abdur Rahman Adnan
Co-advisor : Imran Hossain Shaon

Syeda NoorJaha Azim 12101064
Tajul Islam 12101067
Abu Sufian 12101048

DECLARATION

We hereby, declare that this work was carried out by us under the guidance and Supervision of Professor **Md. Haider Ali,** Professor, Department of Computer Science and Engineering, BRAC University and **Md. Abdur Rahman Adnan** and **Imran Hossain Shaon,** Lecturer, Department of Computer Science and Engineering, BRAC University. We declare that this work has not been submitted anywhere else for the award of any other degree.

Signature of Advisor

_____

Dr. Md. Haider Ali
Professor, Department of Computer Science and Engineering,
BRAC University

Md. Abdur Rahman Adnan
Lecturer, Department of Computer Science and Engineering,
BRAC University

Imran Hossain Shaon
Lecturer, Department of Computer Science and Engineering,
BRAC University

Signature of Authors

_____

Tajul Islam
ID: 12101067
tajulislam.glab@gmail.com

_

_____

Abu Sufian
ID: 12101048
ynor7005326@gmail.com

_____

Syeda NoorJaha Azim
ID: 12101064
synjaz@gmail.com

TABLE OF CONTENTS

ABSTRACT

Cloud computing emerged and acquired huge popularity. Many Cloud Service Providers (CSPs) like, AWS, Rackspace, Azure and many more are facilitating clients with common services (storage, computing, networking, etc.) and thriving for more client by adding new features. So, Cloud Consumer (CC) falls under critical situation to adapt new API and quickly switch to CSP that best suits their business requirement. To solve this perplexity of CC, we introduce a prototype which integrates different Cloud API to an abstract API platform and developed a Proof of concept (PoC) for our platform. Using our single point API CC can quickly switch and get facilitated with the services of different CSP without individual API knowledge.

ACKNOWLEDGEMENT

We express our deep sense of gratitude to Chairperson of CSE dept. of BRAC University, Dr. Md. Haider Ali, for encouraging research and development. Also we would like to express our sincere thanks to Md. Abdur Rahman Lecturer of CSE Dept. of BRAC University, for his precious suggestions in refining the research work. We specially thank Md. Imran Hossain Shaon, Lecturer of CSE Dept. of BRAC University, for tool related support.

Chapter 1

1. Introduction

Cloud Computing is a model where all the resources are available in the Internet. Just like cooked food, cloud services are prepared dish where all the required ingredients are composed for you. There are three renowned categories of cloud services- Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). SaaS allows accessing the functionality of particular software without worrying about the storage or other issues [1].

IaaS provides the networked computer, along with hardware and virtualized Operating System (OS), running in a hosted environment. Some well-known IaaS Cloud providers are Amazon EC2, Rackspace and Google Computer Engine. PaaS is somewhere in the middle of the other two services, it adds support for the development environment. Some examples of the popular PaaS providers are: Amazon AWS Elastic Beantalk PaaS (supports Java, PHP, Python, .Net and Node.js); Google App Engine (supports a subset of common Java environments as well as Python and Go), Cloud Foundry (supports Java, Ruby, Node.js and Scala), Engine Yard

(supports Ruby on Rails, PHP and Node.js) [2] What is supported by the platform will obviously impact CCs decision on which platform to use.

All CSPs has its own cloud OS to provide the services, such as Eucalyptus which is an open source project for cloud computing. The virtualized OS of CSPs have common orchestration task like storage, compute and network for configuration management; virtual machine (VM) and application instances for provisioning; IT automation and DevOps, security and compliance assessment, monitoring and reporting [3].

Though all the orchestration task of different CSP has same purpose, still there are variations in the service's feature that distinguish one CSP to others. These differences in the services influence CC‟s choice of cloud vendor. For example, some cloud vendors have non-negotiable contracts for the CCs but there are certain features of the service provided by those cloud vendors that are essential for the CCs business.

In this case the CC prefers to consume two or more cloud vendors‟ services at a time but there is no such platform where more than one cloud vendor's services can be operated. This paper introduces a prototype of an API that provides an integrated common platform of multiple clouds API from where CC can consume service from numerous CSPs; without being distressed about maintaining many cloud computing vendors. Users of this

API are more relaxed and can put their full effort in leveraging the enterprise goal. According to Steve Jobs “I’m convinced that about half of what separates the successful entrepreneurs from the non-stressful one is pure perseverance.” [4]. Therefore when all the cooked food is served on a single table than it is more appropriate to give attention in eating the food enjoyably.

1.1 Motivation

When we started our studies regarding cloud services we saw that different cloud services have different features. There are few things which are unique, say for example: Eucalyptus has AWS Compatibility, Scalable Object Storage, and Network Options for growth and Cloud account administration whereas OpenStack has ability to manage local area networks, Role-based access control, VNC proxy through a Web browser, Virtual machine image management. So if someone use Eucalyptus they do not have the flexibility to use the features of OpenStack. Thus,

We took the initiative to develop a Universal API which will solve this issue by using the request from Client and convert them as cloud specific requests. So there will be flexibility for the user of using different cloud service software at a time. Besides this will be beneficial for Cloud service model because IaaS is related to PaaS and SaaS.

1.2 Key Features and Impact

- Don’t need to learn more API of different cloud platform
- Double authentication process which is more secured then before
- Dynamically switch platform
- User friendly tutorial to habituate the system

Chapter 2

2. Background Study

Cloud computing emerged and acquired huge popularity. Cloud Computing is the delivery of on-demand computing resources everything from applications to data centers over the Internet on a pay-for-use basis.

Cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. When we store data on or run programs from the hard drive, that's called *local* storage and computing. When we store data or run programs over the internet, that’s called cloud storage and cloud computing.

2.1 Cloud Computing Architecture

Figure 1: *Cloud Computing Architecture*

When we talk about cloud computing architecture, it comes the two side front end and back end. They connect each other through internet. The front end is the side the computer user, or client, sees. The back end is the "cloud" section of the system.

2.2 Cloud computing deployment models

Cloud computing deployments models mainly distinguish based on size and access. It tells about the purpose and the nature of the cloud. Most of the organizations are willing to implement cloud as it reduces the capital expenditure and controls operating cost.

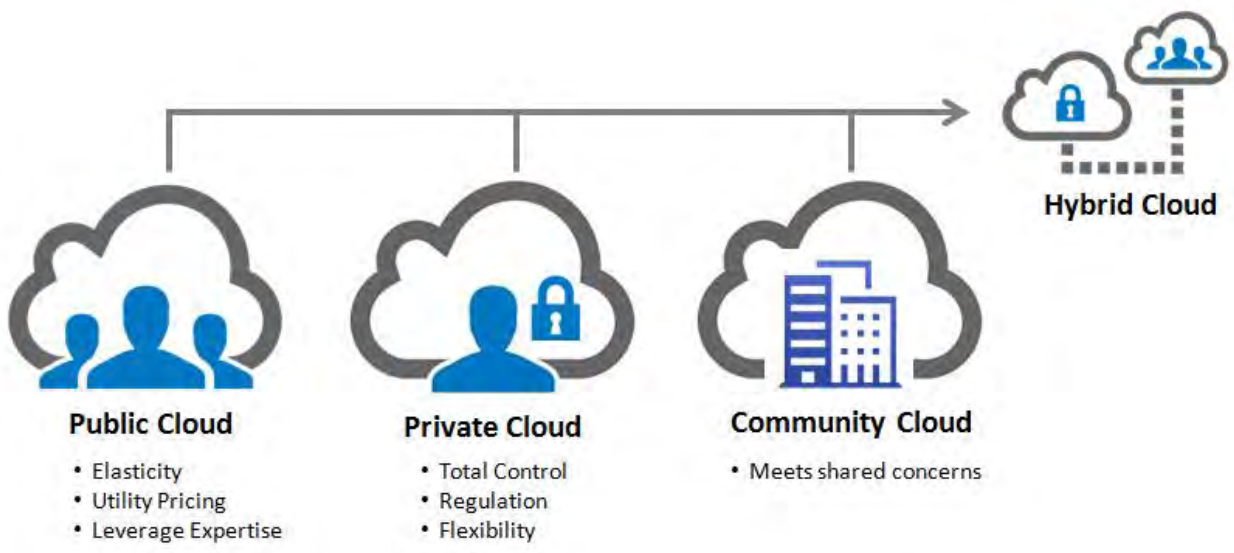


Figure 2: *Cloud Computing Deployment model*

Public cloud: It is a type of cloud in which the cloud services are delivered over a network which is open for public usage. The customers do not have any control over the location of the infrastructure. For example email public cloud is less secure because of its openness.

Private cloud: It is s also known as internal cloud; the platform for cloud computing is implemented on a cloud-based secure environment. Private cloud as it permits only the authorized users. It allows system and service accessible within organization. It increased security because of its private nature [5].

Hybrid cloud: It is a type of cloud computing which is integrated. It can be an arrangement of two or more cloud servers, i.e. private, public or community cloud .Consider an e-commerce website, which is hosted on a private cloud that gives security and scalability, since security is not a prime concern for their brochure site it is hosted on a public cloud which is more economical as compared to a private cloud. Businesses that have more focus on security and demand for their unique presence can implement hybrid cloud as an effective business strategy [5].

Community Cloud: Community cloud is a type of cloud in which setup is shared between many organizations that belong to a particular community. The main intention of these communities is to achieve their business related objectives. A community cloud may be internally managed or it can be managed by a third party provider. It can be hosted externally or internally. The cost is shared by the specific organizations within the community, hence, community cloud has cost saving capacity [5].

2.3 Cloud Computing and Its Services

Cloud computing has grown in popularity. Each type of cloud service, and deployment method, provides cloud consumers with different levels of control, flexibility, and management system.

Infrastructure as a Service (IaaS): Infrastructure as a Service provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today. For example Amazon Web service, Eucalyptus.

Platform as a Service (PaaS): platform as a service allow you to focus on the deployment and management of your applications. You don't need to think about software maintenance or any other things that running on your computer. It refers to the delivery of operating systems and associated services over the Internet without downloads or installation. For example google app engine.

Software as a Service (SaaS): Software as a Service provides complete product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user

applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on [6].

2.4 Different type of Cloud Computing Platform:

2.4.1 Eucalyptus

Eucalyptus is an open source software framework for cloud computing that implements what is commonly referred to as Infrastructure as a Service (IaaS). Eucalyptus software provides users with the ability to run and control isolated collections of virtual machine instances with many EC2/S3-compatible tools. Eucalyptus uses the Xen and KVM as the hypervisor of choice for Virtualization.

Eucalyptus components: Logically the Eucalyptus cloud consists of four components: the client, the cloud controller (clc or "cloud"), the cluster controller (cc), and the node controller (NC). The client is solution dependent, but may come in the form of a browser script, a user space

program, or even a kernel module. The cloud controller is the client's interface into the cloud and provides the logic decision of the cloud. The cloud controller runs services that authenticates the client, and then translates the client requests into transactions. The cluster controller is a collection of node controllers. It is responsible for state information and interaction with the virtual machines availability. However, the nodes themselves are responsible via the hypervisor (Xen or KVM) to online and offline virtual machines. A node controller exists as a single instance on a single machine; however, multiple node controllers make up the particular cloud. The figure below shows these components and their hierarchy relationship.

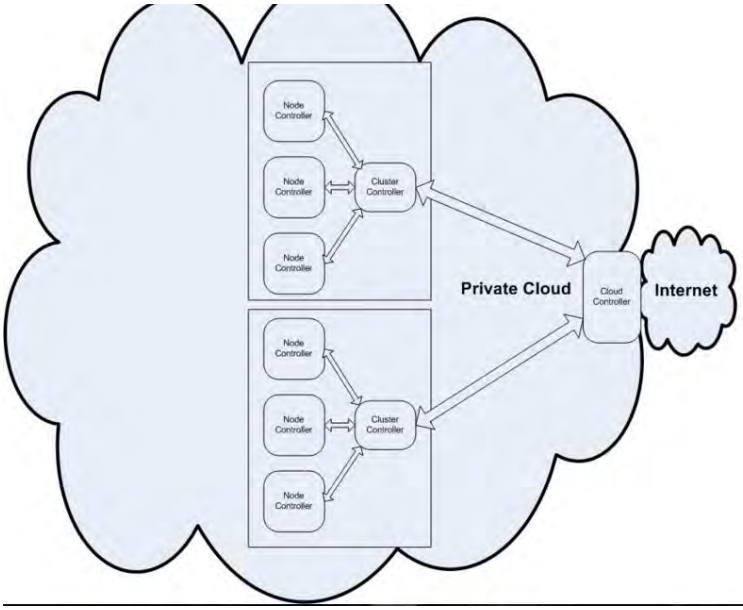


Figure 3: *Eucalyptus Components*

Cloud Controller (CLC): Cloud controller is responsible for entire system management. It is the main entry point into the Eucalyptus cloud for all type of clients. The CLC is responsible for passing requests to the right component, collecting them, and sending the responses from the components back to the client.

Functions:

1. Monitor the availability of resources on various components of the cloud infrastructure, including hypervisor nodes that are used to actually provision the instances and the cluster controllers that manage the hypervisor nodes.
2. Resource arbitration – deciding which clusters will be used for provisioning the instances.
3. Monitoring the running instances.

Cluster Controller (CC): Requests are communicated to the CC using the SOAP or REST-based interface. The CC maintains all the information about the Node Controllers. Cluster Controller decide which Node Controller should be used to run instances.

Functions:

1. To receive requests from CLC to deploy instances.
2. To decide which NCs to use for deploying the instances on.

3. To control the virtual network available to the instances.
4. To collect information about the NCs registered with it and report it to the CLC.

Node Controller (NC): It controls the host operating system and the corresponding hypervisor. It controls life cycle of each instances. It interacts with the OS and the hypervisor running on the node on one side and the CC on the other side.

Functions:

1. Collection of data related to the resource availability and utilization on the node and reporting the data to CC.
2. Instance life cycle management.

Storage Controller: Storage Controller (SC) implements block-accessed network storage (e.g., Amazon Elastic Block Storage -- EBS) and is capable of interfacing with various storage systems (NFS, iSCSI, etc.). An elastic block store is a Linux block device that can be attached to a virtual machine but sends disk traffic across the locally attached network to a remote storage location.

Functions:

1. Creation of persistent EBS devices.
2. Allowing creation of snapshots of volumes.

Walrus (W)/Scalable Object Storage: It manages access to the storage services within Eucalyptus. It allows users to store data, organized as buckets and objects. It allows users to create, delete, list buckets, put, get, and delete objects, and set access control policies. Walrus interface compatible with Amazon's S3, and it supports the Amazon Machine Image (AMI) image-management interface, thus providing a mechanism for storing and accessing both the virtual machine images and user data.

Management Platform: Management Platform provides an interface to various Eucalyptus services and modules. These features can include VM management, storage management, user/group management, accounting, monitoring, SLA definition and enforcement, cloud-bursting, provisioning, etc. [7].

OpenStack and its Services

OpenStack began in 2010 as a joint project of Rackspace Hosting and of NASA. It is a free and open-source software platform for cloud-computing, mostly deployed as an infrastructure-as-a-service (IaaS). OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API.

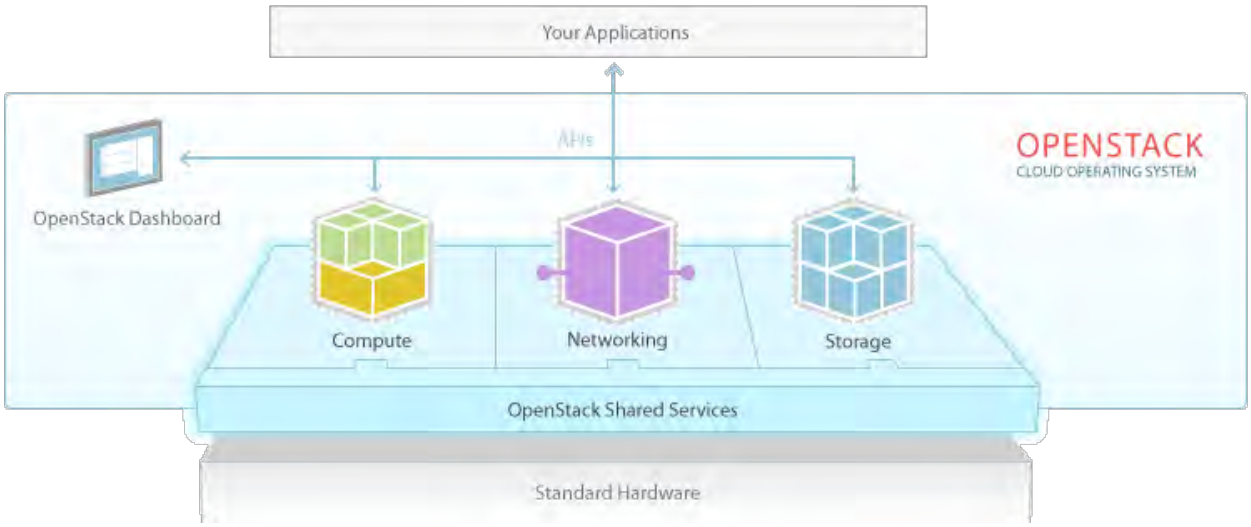


Figure 4: *OpenStack Operating System*

OpenStack provides an Infrastructure-as-a-Service (IaaS) solution through a variety of complemental services. Each service offers an application programming interface (API) that facilitates this integration. The following table provides a list of OpenStack services:

Dashboard: Through the project Horizon Openstack provides the dashboard service. Horizon provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.

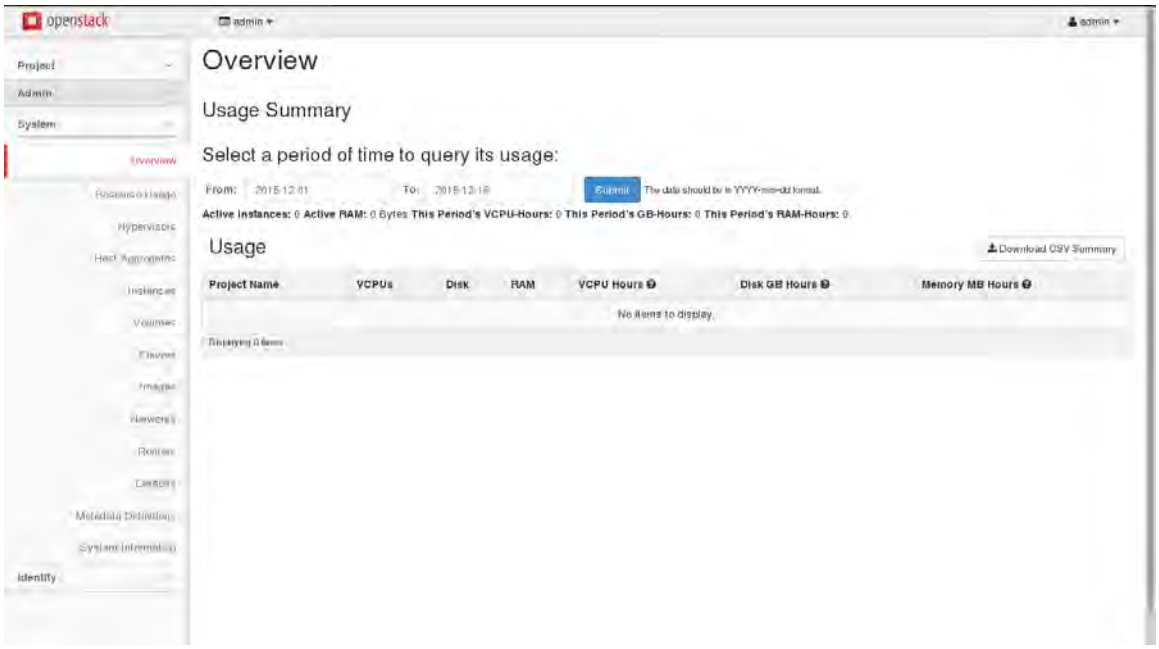


Figure 5: *OpenStack Dashboard*

Compute: Nova project manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of virtual machines on demand.

Networking: Neutron project enables Network-Connectivity-as-a-Service for other OpenStack services, such as, OpenStack Compute. Neutron provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.

Storage Service: OpenStack facilities with two type of Storage services Swift for Object Storage and Cinder for Block Storage. Swift project stores and retrieves arbitrary unstructured data objects via a

RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories. On the other hand, Cinder project provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.

Shared Services: Identity Service (Keystone) provides an authentication and authorization service for other OpenStack services. Keystone provides a catalog of endpoints for all OpenStack services.

Image Service (Glance) stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning. Telemetry (Ceilometer) monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes [8].

2.5 Difference between Cloud Platform services

Eucalyptus and OpenStack both give many similar services but with different features and facilities. Storage service is one of them. Storage service is really a big concern for any cloud platform. There are possibilities of giving the same storage service but in different process. Object storage is one of the types of storage service. It keeps the data as object and it can store unlimited object in bucket. For retrieving object there is unique developer

assigned keys. Most important advantage of it to access objects from anywhere of the bucket and secured from unauthorized access. For maintaining unstructured data or flat structure, web content, documents, sensor data, databases and log files Object storage is really a great option. But for random access data or relational database could not possible in Object storage which is one of its' weak point. Instead of its Object storage have scalable capacity as well as scalable performance. Even it is durable with low cost and simplified management process.

On the other hand there is another storage service called Block storage. Block storage is usable almost any kind of application including file storage, database storage. Heavy random data access can easily be done in this storage service. Block Storage is persistent storage organized into unstructured "blocks", each the same length. An ordinary disk drive, RAID array, or USB storage key are examples of locally attached "block storage". In Block storage, files are split into evenly sized blocks of data, each with its own address but with no additional information (metadata) to provide more contexts for what that block of data is. In a nutshell, key phrases associated with block storage are granularity, great performance, little or no metadata, and local use. Even Block storage is ideal for databases, since a DB requires consistent I/O performance and low-latency connectivity.

Both of these Storage service have some unique features and the usability of the features varies from user to user. To select a cloud provider or technology, user should understand their requirements in order to list the needed features. All of the quality of storage services is not solely served by either Eucalyptus or OpenStack. So problem arises when user wants to use some combined feature which Eucalyptus or OpenStack won't be able to fulfill it individually. Previously Customers were not able to use cross platform for multiple features but now through this Universal API there is a possibility for manipulating any features for cross cloud platform which is an undoubtedly a great innovation for upcoming cloud users as well as for providers.

Customer end will always be beneficiated by using Universal API in their application. They will use multiple features from multiple cloud platforms with the same cost just like before.

According to the Google Search trend the rate of Cloud platform search rises steeply after 2011. Figure 6 below shows the graph of average search of popular Cloud platforms. From the graph we can infer that Amazon Web Service (AWS) search rate is consistently increasing as the years goes by. Eucalyptus and Openstack are very competitive and after 2011 the graph shows Openstack has gain more reputation than Eucalyptus.

We know that all Cloud platforms provide same computing services but the graph shows fluctuation, which means though they provide the same service still there are certain difference in the service features which makes one Cloud platform more favorable to a business case than other. Table 1 [8] represents a Comparative study of IaaS solution („+" sign means better solution).

Table 1: *Comparative study of IaaS solution ('+' sign means better solution)*

	Solutions cloud computing IAAS			
	Eucalyptus	OpenNebula	Cloudstack	OpenStack
Storage	+ + + + +	+ + +	+ + + + +	+ + + + +
Network	+ + + +	+ + + +	+ + + + +	+ + + + +
Security	+ + + +	+ + +	+ + + +	+ + + + +
Hypervisor	+ + + +	+ + +	+ + + + +	+ + + +
Scalable	+ + +	+ + + +	+ + + + +	+ + + + +
Installation	+ +	+ + +	+ + + + +	+ + + + +
Docs	+ + +	+ + +	+ + + + +	+ + + + +
Code Openness	+ + +	+ + + + +	+ + + + +	+ + + + +

Therefore companies prefer to run their business in multiple Cloud platform to assure their resources are more secure having saved them on two secure storage; moreover companies are facilitated with different flavor of the same services and are not locked into one vendor.

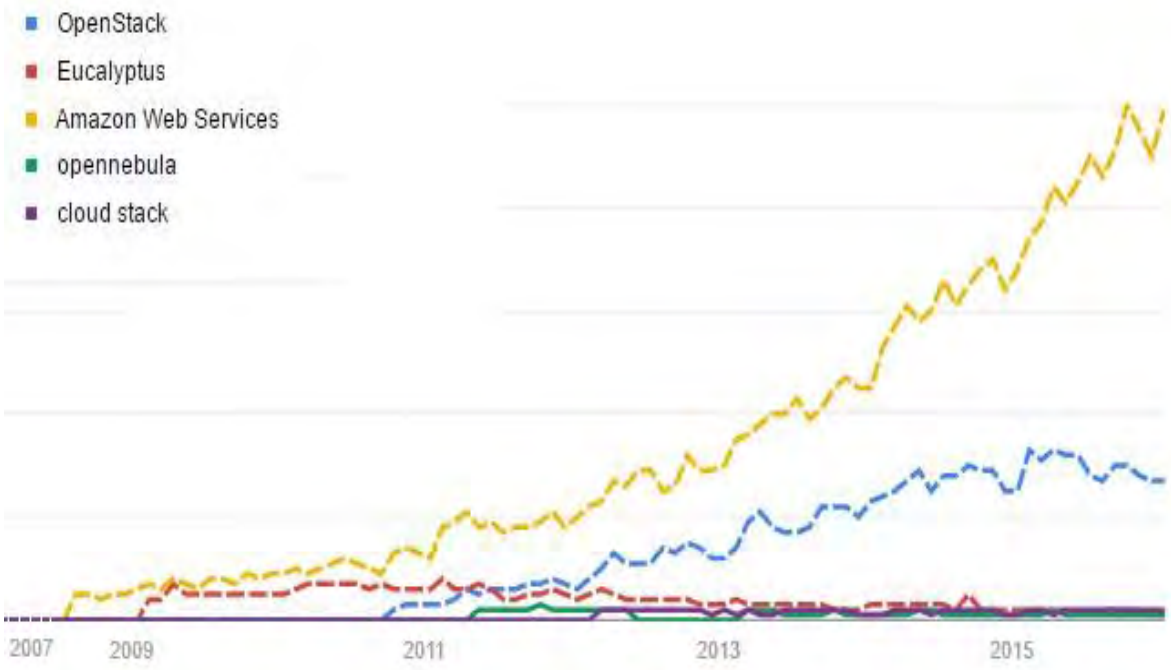


Figure 6: *Google Search Trend of Cloud platforms*

2.6 Case Studies

2.6.1 Downtime brings huge debt

Any business depending on a Cloud computing cannot absorb prolonged bout of frequent outages or slowdown. Though CSP is expected to be immune to service outages still more than few times such incident already happened. In 2014 Dropbox faced an outage for as long as two days [5]. Thus all the applications running on internet dropped offline and the business has to face huge debt. A backup Cloud platform for the online applications could be a great help for Dropbox at that situation.

2.6.2 Lack of Security and Privacy can put an end to enterprises

To rob a company it is not always necessary to break into an office building late at night, demanding a ransom, and then throwing grenades into the data center if the demands were not fulfilled. A company is under a security threat when an attacker gets access to its only CSP‟s control panel and removes all its precious resources stored in the server. Code Spaces is the unfortunate company that has to close door because of an attacker. It was a company that offered developers source code repositories and project management services using Git or Subversion, among other options. According to the information on the Code Space website, the company's AWS control panel was hacked by an attacker and demanded money in exchange of giving control back to Code Space. When the Code Space tried to regain the control, the attacker started deleting resources. Code Space could not save all its EBS snapshots, S3 buckets, all AMIs and some EBS instances. A seven year running company with no shortage of customer would have survived if it had saved its treasured resources on any back up Cloud platform without the burden of maintaining several cloud vendors.

2.6.3 Cloud Computing Platform Dependencies

The new breed of extremely lightweight company like Coupa is not only totally run in the cloud, but also is a cloud based provider itself. Coupa is a C2C business, acting as a broker of services assembled from offering supported by third-party providers. It provides online procurement software and services to the business community. For online services its servers are provisioned at Amazon Web Services, while its customer relationship management runs on Salesforce.com and Google handles its Email. [7] For such companies that utilize cloud services from different providers, would amplify its business growth if they could meet all the Cloud platforms on a common platform.

2.7 Related Works

Cloud computing means storing and accessing data and programs over the Internet instead of your computer's hard drive. When we store data on or run programs from the hard drive, that's called local storage and computing. When we store data or run programs over the internet, that’s called cloud storage and cloud computing. In the recent years Right Scale have launched a platform to manage different cloud platform. RightScale Cloud Portfolio Management is an integrated software suite that includes Self-Service,

Cloud Management, and Cloud Analytics. Built on a Multi-Cloud Platform, RightScale supports Infrastructure-as-a-Service resource pools across public clouds, private clouds, and virtualization [10] .

Chapter 3

3. System Design

Our API is a specialized discipline that involves the design of an interface for part of a system or an entire system. This API can be used by another part of the system or our web service is an independent system to facilitate communication between user and cloud platform. As we know APIs are an integral part of modern software development so we developed both API as well as Web service so anyone will be able to integrate our API or they will be able to communicate via our Web service.

3.1 Methodology

After an immense study on the CSPs‟ services and research on the issues raised by the organizations depending on Cloud Computing, we developed a prototype of an API to resolve the problems by integrating all the CSPs to a common platform. A standard and portable JAX-RS (Java API for RESTful web service) API is designed in order to simplify its development and their clients in Java. The API contains a resource class which is a Java class annotated with JAX-RS annotation to represent a web resource. There is also a root resource class called POJO (plain old Java object) that is annotated with at least one @Path or a request method designator such as @GET, @PUT, @POST, or @DELETE to handle request on the corresponding resource.

Jersey (Sun's open source, production-quality reference implementation for JAX-RS) and Apache CXF (Apache's Java API for RESTful web services) REST frameworks are used for developing RESTful services in Java. Among these two REST frameworks we preferred to develop our Web Service by Jersey. Jersey RESTful Web Services framework is open source, production quality framework for developing RESTful Web Services in Java that provides support for JAXRS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation.

Apache CXF is an open source services framework. CXF helps you build and develop services using frontend programming APIs, like JAX-WS and JAX-RS. These services can speak a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI. [9] Figure 2 shows the use case diagram of the API, where the user creates a secure account and login to the web service. The web service allows the authenticate user to select a Cloud platform and then select a services of that platform. It also allows the user to shift between the services of different Cloud platforms.

3.2 System Architecture

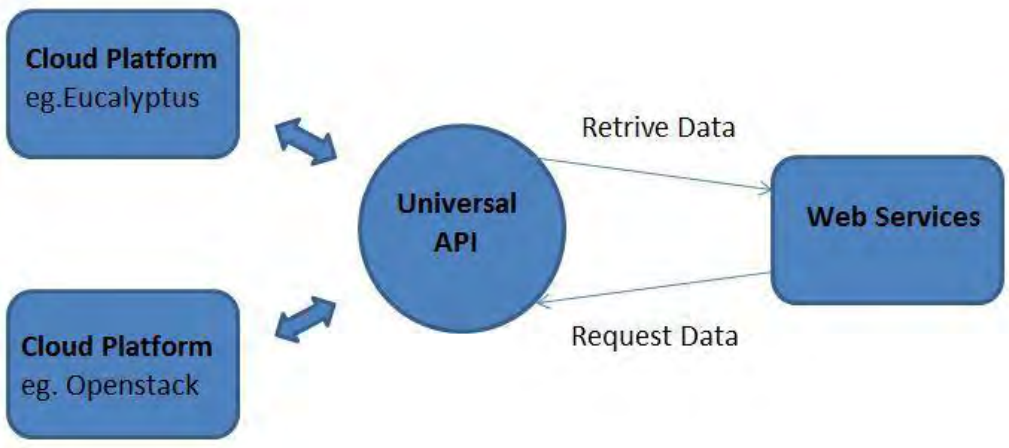


Figure 7: *System Architecture*

3.3 Use case Diagram

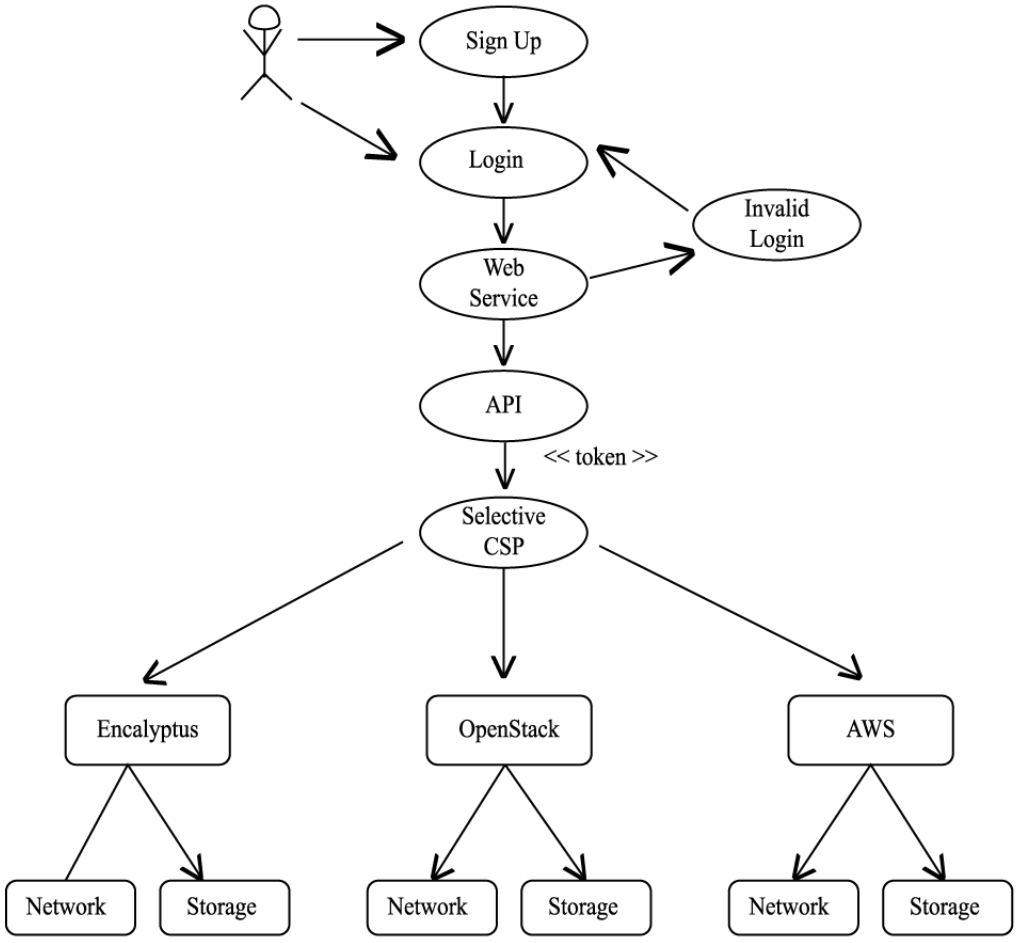
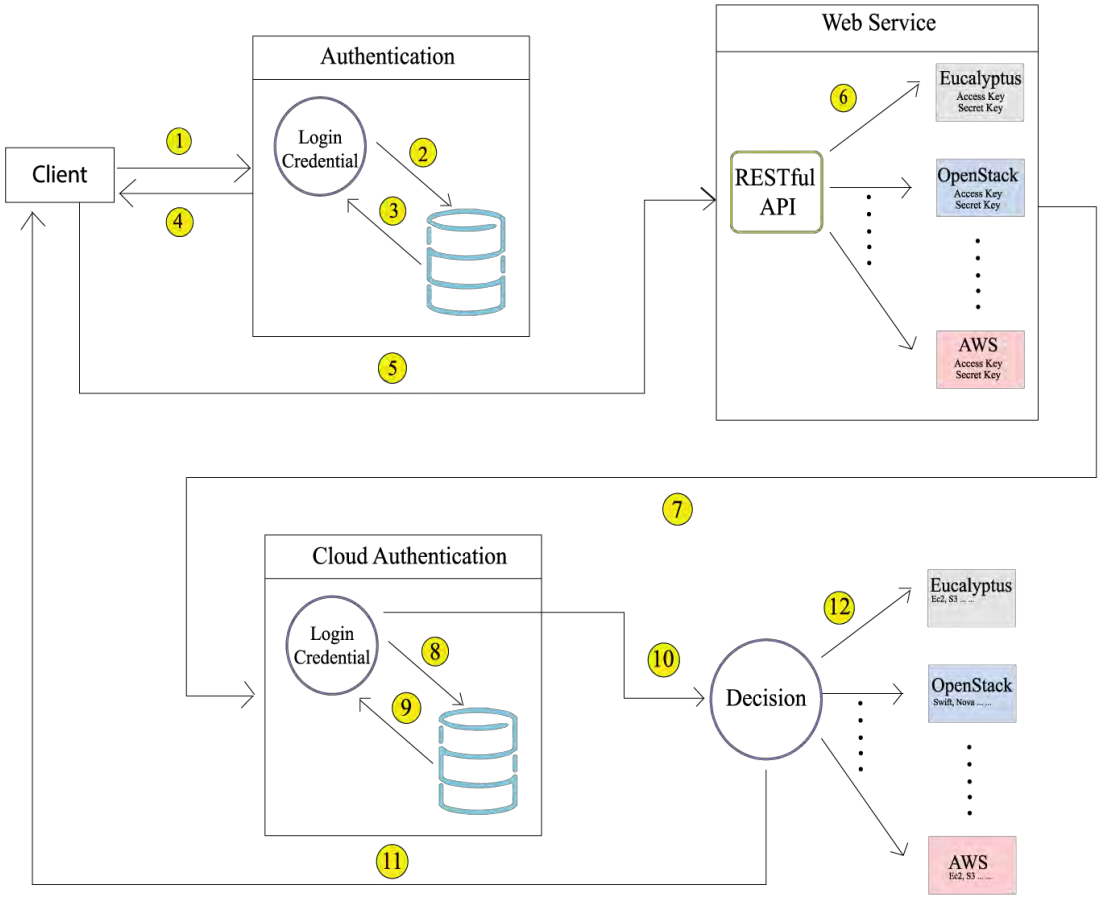


Figure 2: *Use case Diagram of the API*

For the proof of concept of the model, we choose to use two cloud IaaS, AWS for Eucalyptus and Trystack for Openstack, for testing the design. Both AWS and Trystack provide a demo of the corresponding cloud IaaS to the developers without having to commit to a full deployment. AWS is a secure cloud services‟ platform, offering compute power, database storage,

content delivery and other functionality to help businesses scale and grow. Explore how millions of customers are currently leveraging AWS cloud products and solutions to build sophisticated applications with increased flexibility, scalability and reliability [10]. TryStack is a free and easy way for users to try out OpenStack, and set up their own cloud with networking, storage, and computer instances [11]. At first we set up the environment for Jersey framework by importing all the necessary packages to the RESTful Web application. After that by implementing the APIs of the services (such as Nova API, Swift API and Amazon S3) we integrate all the service methods provided by AWS and Trystack. Figure 3 represents the Data flow diagram of the model.

3.4 Data Flow Diagram



01. http @GET Request

02. Credential Check

03. Confirm Credential

04. http @POST Request

05. RESTful Web Service Access

06. Access in Cloud Platform

07. @GET Request (Account Credential)

08. Cloud Credential Check

09. Confirm Credential

10. Decision Parameter

11. Cloud Service Access

12. Decision Processing

Figure: Data Flow Diagram

The whole process begins when the client sent Http @GET request to the web service. After getting the confirmation of the valid credential from the authentication module, the client gets access to the RESTful Web Service. Our RESTful API gives access to the Cloud platforms that are integrated into the Web Service. Client select a Cloud platform from the Web Service and the Web Service then sent a @GET request to the selected cloud platform's account. After a proper authentication check from the cloud account a decision parameter is forwarded to a process that decides on which cloud service to access. The Client than have full access to all the services of the chosen Cloud Platform.

Chapter 4

4.1 Tools Used:

- Eclipse Mars.1
- Apache cxf
- JAX-WS (Jersey)
- AWS
- TryStack
- PHP
- XAMPP
- Brackets(Text Editor for PHP)
- Bootstrap Framework

4.2 Experimental Result:

The demo API is tested by implementing it on a client web service. The client web service sends a HTTP request to our API in the server, this API then sends the Access Key and Secret Key to the cloud API to be authenticated for the other service API in the cloud IaaS. The positive result of the experiment proves that the API is implementable to any client service.

Storage service is an important cloud computing service and file uploading from the client site to the server is a major feature, due to security issue the browser do not allow to show the directory location of the file which brings complexity in integrating the Cloud platform. In this case, a temporary folder is created to transect the file from client service to the server.

This is the Homepage for Web Service where user can select Cloud providers. After that user have to submit account credentials for login purpose.

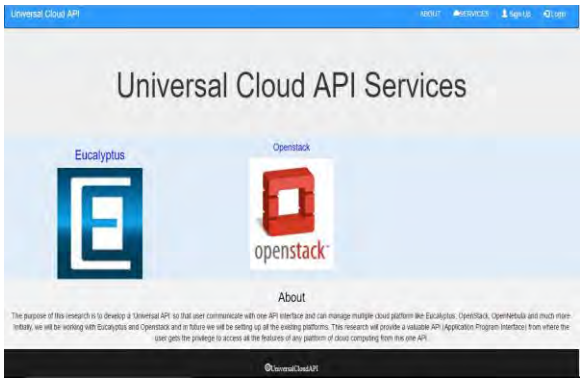


Diagram 1: Home page

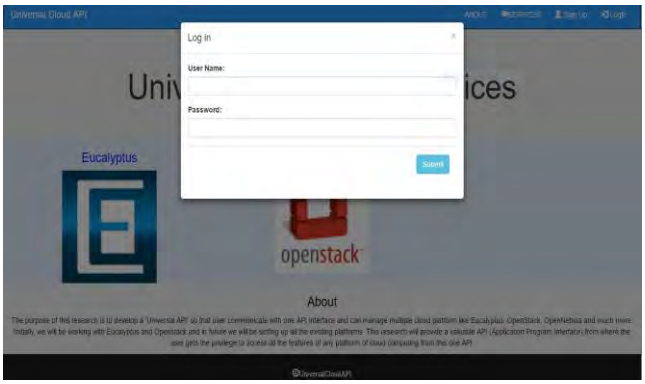


Diagram 2: Login Form

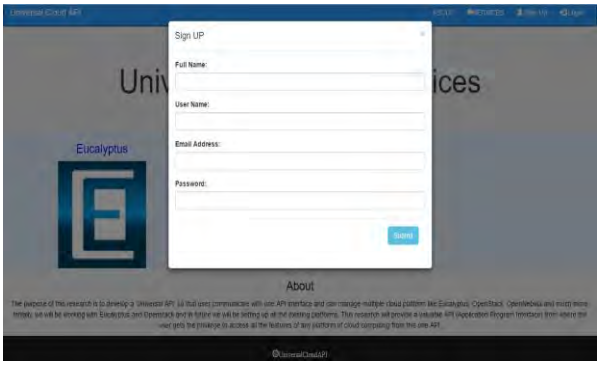


Diagram 3: Sign up Form

If user does not have any account in our web service user has to sign up for it. After registration user have to give account credentials for particular cloud platform. Service Provider's account credentials taken at the early stage of account creation through which user can get access any of the service of particular service provider's.

Diagram 4: User Account Information form

Create container in the AWS Server for the Eucalyptus Storage Service

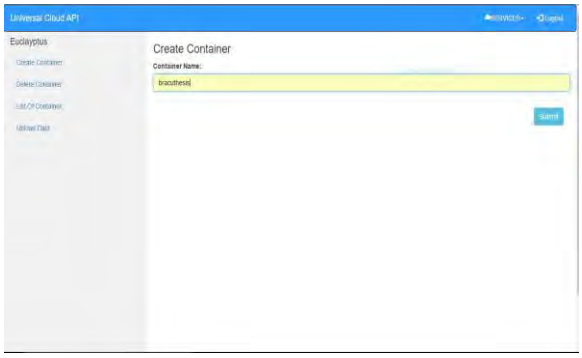


Diagram 5: Client enters container name

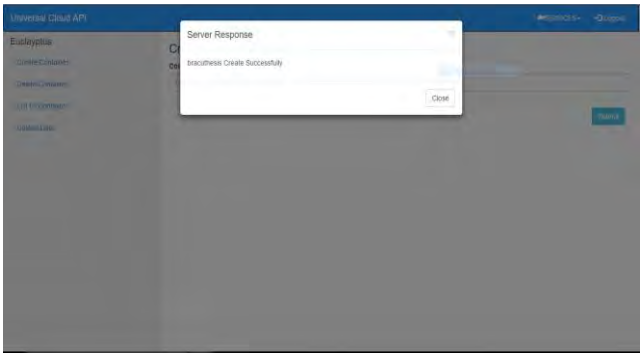


Diagram 6: Server response for creating new container

Diagram 7: Container list in Sever

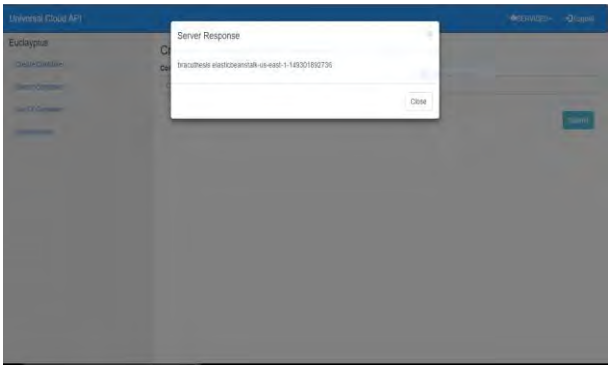


Diagram 8: List of container response from server

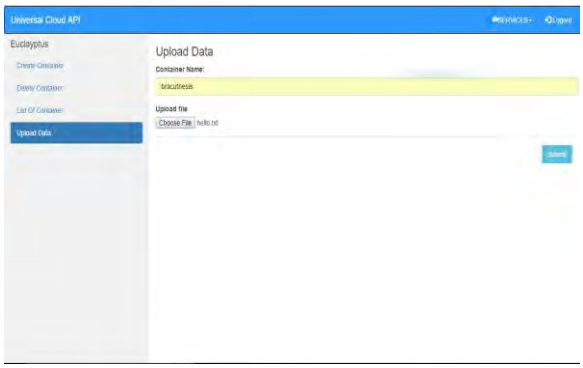


Diagram 9: Client's File Selection

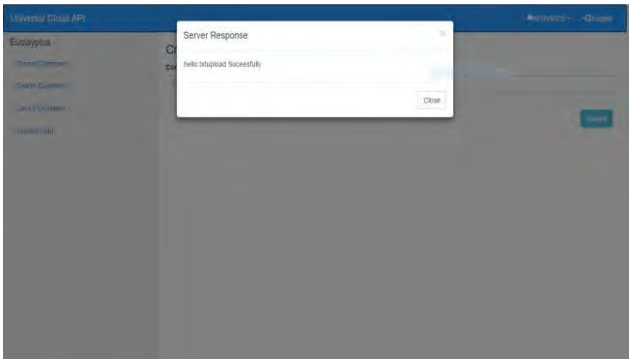


Diagram 10: Response for File Upload

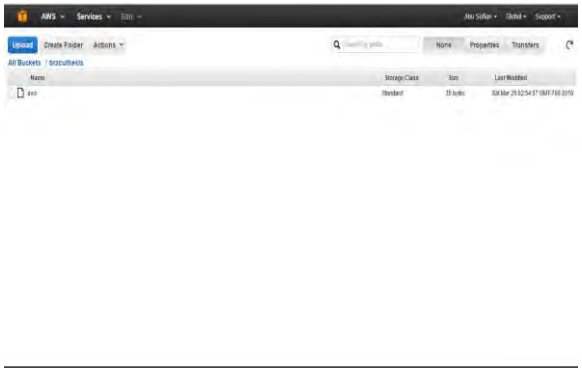


Diagram 11: File Uploaded in AWS Server

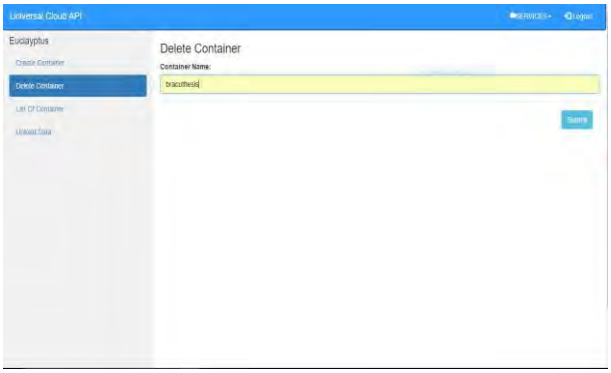


Diagram 12: Delete Container from AWS Server

Create container in the tryStack Server for the OpenStack Storage Service

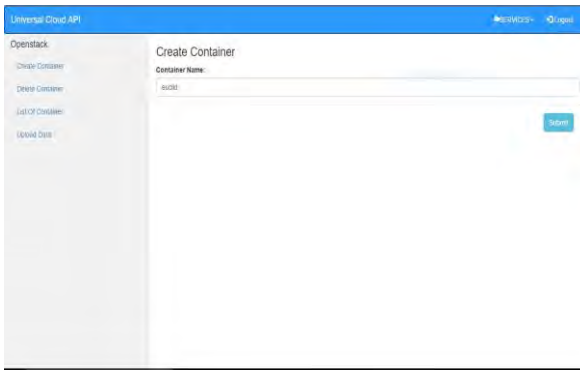


Diagram 13: Create Container in OpenStack

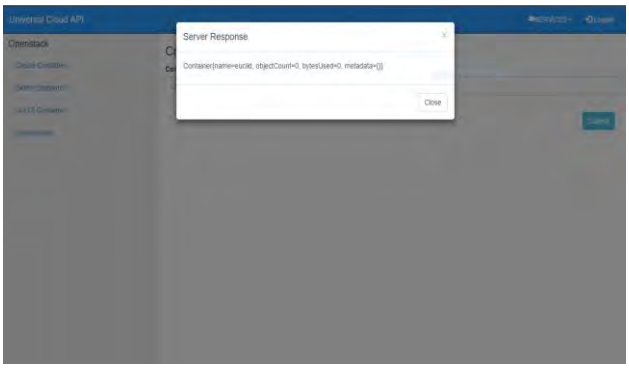


Diagram 16: List of container in Server

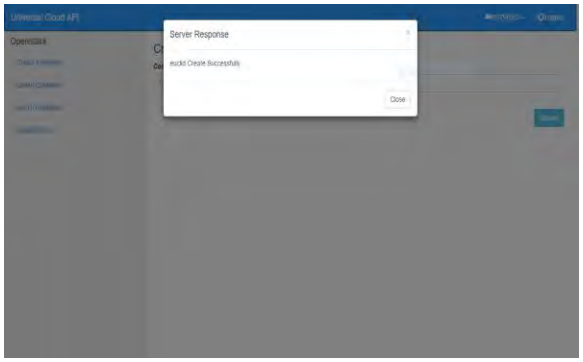


Diagram 15: Server response for OpenStack

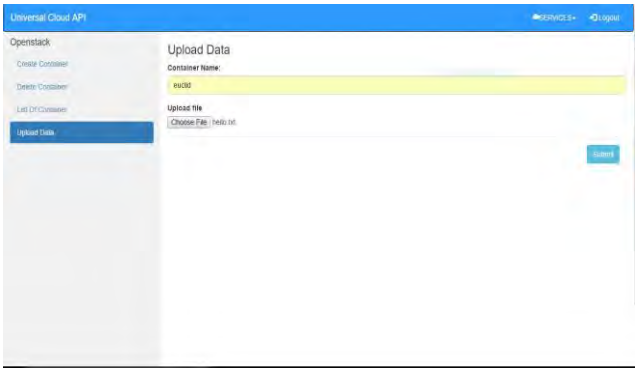


Diagram 17: Upload file to OpenStack

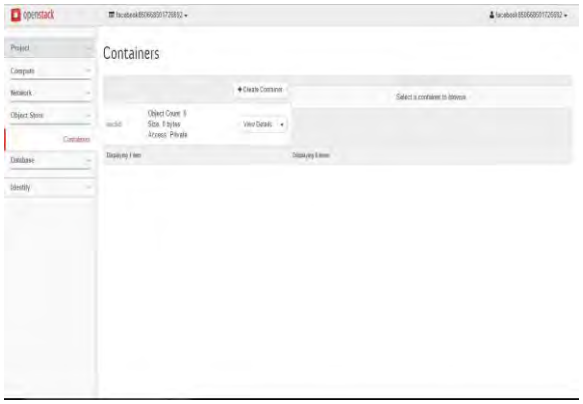


Diagram 14: View Container in Server

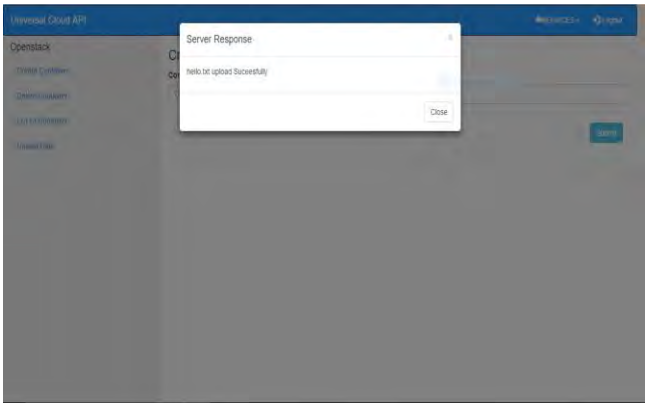


Diagram 19: Server Response

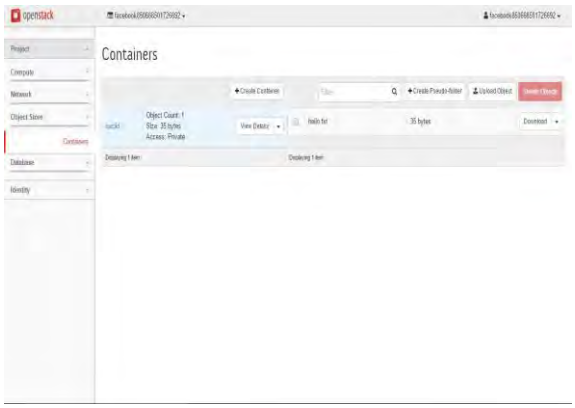


Diagram 21: File Uploaded in OpenStack Server

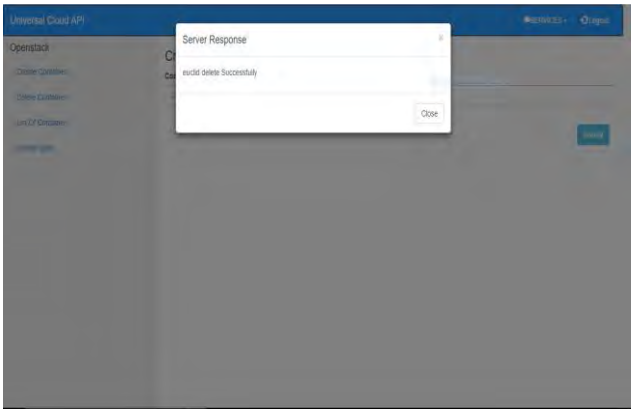


Diagram 20: Server Response

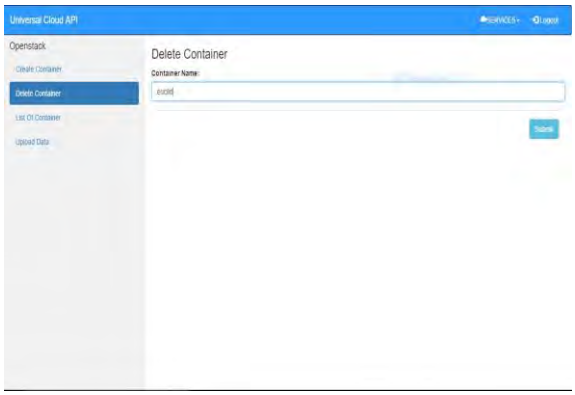


Diagram 18: Enter container name to be deleted

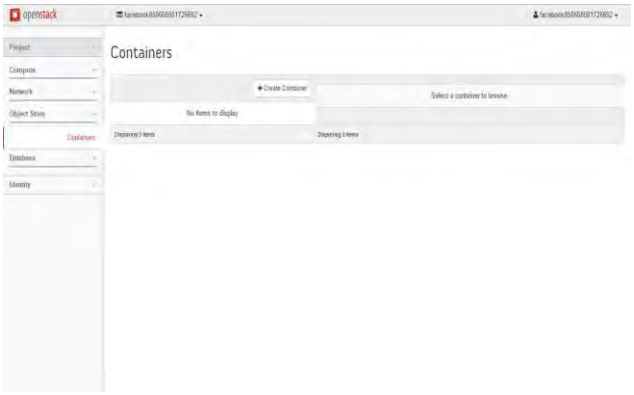


Diagram 22: Container deleted from Server

Chapter 5

5.1 Conclusion & Future Work

This paper introduces a prototype of an API that integrates multiple Cloud platforms. Implementation of the API gives a better solution to the problem faced by the organizations running on Cloud system. The API adds an intermediate connection between the Cloud platforms which makes cloud computing system more versatile. It brings a new scope of research on the relationship between different Cloud platforms. Currently a demo API is developed. Only Eucalyptus and OpenStack are merged but in future we will enhance the API by add more Cloud platforms to the API. We hope to introduce more features that would make this API more functional for the client service.

Reference

[1] Cloud Computing. (n.d.). Retrieved March 22, 2016, from http://www.hcltech.com/technology-qa/what-cloud-computing-what-are-services-offered

[2] News and Views - articles and case studies for better strategies. (n.d.). Retrieved March 22, 2016, from http://www.bitheads.com/what-types-of-cloud-services-are-available/

[3] Shackleford, D. (n.d.). Where The World Talks Security | RSA Conference. Virtualization and Cloud: Orchestration, Automation, and Security Gaps. Retrieved from http://www.rsaconference.com/writable/presentations/file_upload/csv-r02-virtualization-and-cloud-orchestration-automation-and-security_gaps_v2.pdf

[4] S. D. (2016, February). How startups can yield benefits from cloud technology? Retrieved March 22, 2016, from http://www.esds.co.in/blog/how-startups-can-yield-benefits-from-cloud-technology/#sthash.kO0lMd7j.dpbs

[5] Lenk, A. (2015). Cloud Standby Deployment: A Model-Driven Deployment Method for Disaster Recovery in the Cloud. *2015 IEEE 8th International Conference on Cloud Computing*. doi:10.1109/cloud.2015.127

[6] Types of Cloud Computing. (n.d.). Retrieved April 17, 2016, from https://aws.amazon.com/types-of-cloud-computing/

[7] Cloud platform comparison: CloudStack, Eucalyptus, vCloud Director and OpenStack. (2012). Retrieved April 17, 2016, from http://www.networkworld.com/article/2189981/tech-primers/cloud-platform-comparison--cloudstack--eucalyptus--vcloud-director-and-openstack.html

[8] Cloud services for your virtual infrastructure, Part 1: Infrastructure-as-a-Service (IaaS) and Eucalyptus. (n.d.). Retrieved April 17, 2016, from http://www.ibm.com/developerworks/library/os-cloud-virtual1/

[9] Welcome to OpenStack Documentation. (n.d.). Retrieved April 17, 2016, from http://docs.openstack.org/

[10] Product Overview. (n.d.). Retrieved April 17, 2016, from http://www.rightscale.com/products-and-services/products

Appendix

```
import java.io.File;
import java.util.Set;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import org.jclouds.ContextBuilder;
import org.jclouds.io.Payload;
import org.jclouds.io.Payloads;
import org.jclouds.logging.slf4j.config.SLF4JLoggingModule;
import org.jclouds.openstack.swift.v1.SwiftApi;
import org.jclouds.openstack.swift.v1.domain.Container;
import org.jclouds.openstack.swift.v1.domain.ObjectList;
import org.jclouds.openstack.swift.v1.domain.SwiftObject;
import org.jclouds.openstack.swift.v1.features.ContainerApi;
import org.jclouds.openstack.swift.v1.features.ObjectApi;
import org.jclouds.openstack.swift.v1.options.CreateContainerOptions;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.ListObjectsRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.S3ObjectSummary;
import com.google.common.collect.ImmutableMap;
import com.google.common.collect.ImmutableSet;
import com.google.common.io.ByteSource;
import com.google.common.io.Files;
import com.google.inject.Module;


@Path("/main")
public class Main {

  public SwiftApi swiftApi;

  @Path("{a}/{s}")
  @GET
  @Produces("application/xml")
  public String eucacontainerlist(@PathParam("a") String a,@PathParam("s") String s) {
    AWSCredentials credentials = null;
```

```
    credentials = new BasicAWSCredentials(a, s);

    AmazonS3 s3 = new AmazonS3Client(credentials);
    Region usWest2 = Region.getRegion(Regions.US_WEST_2);
    s3.setRegion(usWest2);

    String container="";
    for (Bucket bucket : s3.listBuckets()) {
      container=container +"\n"+ bucket.getName();
    }

    String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
container+"\n\n";
    return "<euca>" + "<create>" + container + "</create>" + "<output>" + result +
"</output>" + "</euca>";
  }

  @Path("{a}/{s}/{b}/{c}")
  @GET
  @Produces("application/xml")
  public String eucacontainercreate(@PathParam("a") String a,@PathParam("s") String
s,@PathParam("b") String b,@PathParam("c") String c){

    if(c.equals("create"))
    {
      AWSCredentials credentials = null;
      credentials = new BasicAWSCredentials(a, s);

      AmazonS3 s3 = new AmazonS3Client(credentials);
      Region usWest2 = Region.getRegion(Regions.US_WEST_2);
      s3.setRegion(usWest2);
      String bucketName = b;

      s3.createBucket(bucketName);

      String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
bucketName+"\n\n";
      return "<euca>" + "<create>" + bucketName +" Create Successfully"+ "</create>" +
"<output>" + result + "</output>" + "</euca>";

    }else if(c.equals("delete")){

      AWSCredentials credentials = null;
      credentials = new BasicAWSCredentials(a, s);

      AmazonS3 s3 = new AmazonS3Client(credentials);
```

```
    Region usWest2 = Region.getRegion(Regions.US_WEST_2);
    s3.setRegion(usWest2);
    String bucketName = b;

    ObjectListing objectListing = s3.listObjects(new
ListObjectsRequest().withBucketName(bucketName).withPrefix("aws"));
    for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
      s3.deleteObject(bucketName, objectSummary.getKey());
    }
    s3.deleteBucket(bucketName);

    String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
bucketName+"\n\n";
    return "<euca>" + "<create>" + bucketName +" Delete Successfully"+ "</create>" +
"<output>" + result + "</output>" + "</euca>";
   }

   return null;
  }


  @Path("{i}/{c}/{b}/{o}/{e : http?://.*}")
  @GET
  @Produces("application/xml")
  public String openstackStrorage(@PathParam("i") String i,@PathParam("c") String
c,@PathParam("b") String b,
                          @PathParam("o") String o,@PathParam("e") String e) {

   if(o.equals("create"))
   {
    String bucketName = b;

    Iterable<Module> modules = ImmutableSet.<Module>of(new
SLF4JLoggingModule());

    String provider = "openstack-swift";
    String identity = i;
    String credential = c;

    swiftApi = ContextBuilder.newBuilder(provider).endpoint(e).credentials(identity,
credential).modules(modules).buildApi(SwiftApi.class);

    ContainerApi containerApi = swiftApi.getContainerApi("RegionOne");
    CreateContainerOptions options =
CreateContainerOptions.Builder.metadata(ImmutableMap.of("key1", "value1","key2",
"value2"));
```

```
    containerApi.create(bucketName, options);

    String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
bucketName+"\n\n";
    return "<euca>" + "<create>" + bucketName +" Create Successfully"+ "</create>" +
"<output>" + result + "</output>" + "</euca>";

   }else if(o.equals("delete")){

    String bucketName = b;

    Iterable<Module> modules = ImmutableSet.<Module>of(new
SLF4JLoggingModule());

    String provider = "openstack-swift";
    String identity = i;
    String credential = c;

    swiftApi = ContextBuilder.newBuilder(provider).endpoint(e).credentials(identity,
credential).modules(modules).buildApi(SwiftApi.class);

    ContainerApi containerApi = swiftApi.getContainerApi("RegionOne");
    Set<Container> containers = containerApi.list().toSet();
    for (Container container : containers) {
     if((container.getName().equals(bucketName))){
      ObjectApi objectApi = swiftApi.getObjectApi("RegionOne",
container.getName());
       ObjectList objects = objectApi.list();
       for (SwiftObject object: objects) {
        objectApi.delete(object.getName());
       }
       swiftApi.getContainerApi("RegionOne").deleteIfEmpty(container.getName());
     }
    }

    String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
bucketName+"\n\n";
    return "<euca>" + "<create>" + bucketName +" delete Successfully"+ "</create>" +
"<output>" + result + "</output>" + "</euca>";
   }
   return null;
  }

  @Path("{i}/{c}/{e : http?://.*}")
  @GET
```

```
  @Produces("application/xml")
  public String openstackcontainerlist(@PathParam("i") String i,@PathParam("c") String
c,@PathParam("e") String e) {

    Iterable<Module> modules = ImmutableSet.<Module>of(new
SLF4JLoggingModule());

    String provider = "openstack-swift";
    String identity = i;
    String credential = c;

    swiftApi = ContextBuilder.newBuilder(provider).endpoint(e).credentials(identity,
credential).modules(modules).buildApi(SwiftApi.class);

    ContainerApi containerApi = swiftApi.getContainerApi("RegionOne");
    Set<Container> containers = containerApi.list().toSet();

    String list="";
    for (Container container : containers) {
      list=list+"  " + container;
    }
    String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
list+"\n\n";
    return "<euca>" + "<create>" + list + "</create>" + "<output>" + result + "</output>"
+ "</euca>";
  }

  @Path("{x}/{y}/{b}/{ob}/{o}/{p : [a-zA-z]?:/.*}/{e : http?://.*}")
  @GET
  @Produces("application/xml")
  public String openstackfileupload(@PathParam("x") String x,@PathParam("y") String
y,@PathParam("b") String b, @PathParam("ob") String ob,@PathParam("o") String
o,@PathParam("p") String p ,@PathParam("e") String e) {

    if(o.equals("openstack"))
    {
      Iterable<Module> modules = ImmutableSet.<Module>of(new
SLF4JLoggingModule());

      String provider = "openstack-swift";
      String identity = x;
      String credential = y;

      swiftApi = ContextBuilder.newBuilder(provider).endpoint(e).credentials(identity,
credential).modules(modules).buildApi(SwiftApi.class);
```

```
      File file = new File(p);
      ByteSource byteSource = Files.asByteSource(file);
      Payload payload = Payloads.newByteSourcePayload(byteSource);
      swiftApi.getObjectApi("RegionOne", b).put(ob, payload);

      String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
ob+"\n\n";
      return "<euca>" + "<create>" +ob+ " upload Suceesfully" + "</create>" + "<output>"
+ result + "</output>" + "</euca>";
    }
    else if(o.equals("eucalyptus")){
      AWSCredentials credentials = null;
      credentials = new BasicAWSCredentials(x, y);

      AmazonS3 s3 = new AmazonS3Client(credentials);
      Region usWest2 = Region.getRegion(Regions.US_WEST_2);
      s3.setRegion(usWest2);

      String bucketName = b;
      String key="aws";
      File file = new File(p);
      s3.putObject(new PutObjectRequest(bucketName, key, file));

      String result = "@Produces(\"application/xml\") Output: \n\nbucketname:" +
ob+"\n\n";
      return "<euca>" + "<create>" +ob+ "upload Suceesfully" + "</create>" + "<output>"
+ result + "</output>" + "</euca>";
    }
    return null;
  }
}
```