ucc

Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

Conor Molloy
UCC Computer Science

A Cloud Visualization Framework

112467548 - 20th April 2017 - John Herbert

Abstract

Cloud Computing Frameworks such as Open Stack have a graphical user display, which shows textual information that requires all users to know precisely all the details involved on how to run a virtual machine. This style of management is tedious and difficult to train new users. This can be avoided by constructing a 2D model of the cloud computing management to provide a better experience for the user. In the model, each entity such as the virtual machines, physical servers and the networks are modelled as a 2D entity that can be manipulated by the user. In doing this a user managing a cloud computing infrastructure no longer requires commands or a web based interfaces. Instead it can be done by the manipulation of 2D entities. In particular, an interactive 2D modelling of a cloud infrastructure can be very useful for monitoring, managing, designing, and planning for an existing and/or new cloud and data centre infrastructure.

Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award.

I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;
- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Signed: . . . . . .

Date: . . . . . .

Acknowledgements

Thank you to Dr. John Herbert for all your help and advice you gave me throughout the project.

Thank you to Dr. Dapeng Dong for all your help with all the technical specific questions you answered and advice you gave to me throughout the project.

Thanks to my family and friends for the countless times you all helped me with this project.

Table of Contents

Contents

Table of Figures

1 Introduction

1.1 Introduction

Since 1979 when Steve Jobs and others at Apple began working on their latest Graphical User Interface (GUI) concepts for the new Macintosh computer, there has been a great demand for systems run by buttons rather than command lines, cloud computing is no exception. In the world of cloud computing, bringing online and offline servers quickly is considered a job for the most advanced technicians due to its incredibly complicated command line required. Existing cloud computing technologies provide graphical user interfaces to interact with users. Many of these systems such as the current OpenStack dashboard and VMware's VSphere use textual data and graphical images. This style of management is rather tedious and requires costly training to be able to operate. In-depth knowledge is required about the particular network, flavour, image and hardware configurations to be able to carry out simple commands such as starting a server or shutting one down.

The aim of this project is to allow users with minimal experience in cloud computing to be able to quickly and effectively launch servers to access resources that are needed. The environment I will be working in is an installation of OpenStack which was installed on the servers inside of University College Cork (UCC). The OpenStack package allows the installation of a base operating system on many different servers/PC's and permits a user to control them in a basic way via the online dashboard. OpenStack makes it possible to be able to run API calls through a programming language to control the OpenStack environment. The next step is how to run these API calls when certain actions that are familiar to the user are done.

1.2 importance of my approach

It can be very difficult for a new user to operate these simple commands that can be vital to the success of a project so it is necessary to be able to give users new and more intuitive ways so that they can operate as an administrator on the cloud network. Instead of displaying textual data and requesting the user to input every variable, the approach I developed is to automatically display information depending on the configuration of the cloud operation system, and to allow users to manipulate 2-D images to do basic commands such as starting, pausing and deleting servers. As

OpenStack does not have a graphical user interface like this, I saw an opportunity to develop something that hasn't been done, that could benefit the average user of OpenStack. (Omar SEFRAOUI, October 2012)

1.3 Objectives of the project

My objectives are to learn how OpenStack works and to gain insight into how a cloud computing architecture is structured. After learning how a cloud implementation of OpenStack works the next objective is to learn how to gather the information needed to run a graphical user interface. I will then work out how to send information to a cloud service that would allow it to be manipulated in a programming language. The next step is to figure out the best way to allow a user to run this programme with the convenience of the user in mind. During this step, a graphical user interface will need to be designed to allow basic commands to be sent to the Cloud infrastructure by manipulating 2D objects in a familiar way. Finally, my last objective is to develop the interface so it can be able to display important information to the user in an intuitive and easily understandable way.

2 Design

2.1 What is Cloud Computing?

Cloud computing is a general term for hosted services available through the internet. Cloud computing is essentially a metaphor for the internet. Cloud computing allows people to use computing resources such as virtual machines, storage or applications as a utility rather than having to set up the hardware and maintain it themselves. A company can use a public cloud service such as Amazon Web services to supply their cloud computing needs, or if they specific requirements preventing them from going to a public service they can set up their own private cloud computing service. There are several benefits to cloud computing.

- Self-service: All end users of cloud computing software can start any computing resources to achieve whatever goal they want to achieve on demand. This eliminates the need for an IT administrator to build and maintain each individual request.

- Elasticity: Companies can scale up or scale down very easily based on the demands put on the infrastructure. This saves the need to have a local server running all the time taking up resources even if it is not needed.

- Some online providers of cloud services like Amazon web services offer customers a pay for use model that allows customers the ability to only pay for the computing resources used.

In the public cloud computing model customers only pay for the resources they consume such as CPU computing power, the amount of storage and the bandwidth of the connection. These are paid on demand either charged by the minute or by the hour. This comes at the cost of the customer having no control over the cloud infrastructure.

With private cloud computing architecture the customer can use their own architecture, or use the hardware from a data centre to set up their cloud computing infrastructure. This gives the customer more access to the management, control and security over the cloud. Companies can also choose to use a hybrid system where sensitive data or important workloads can be operated on in the private cloud but operations that can vary in number that needs to be scaled on demand can be operated by the public cloud infrastructure. [1][2]

2.2 What are the different types?

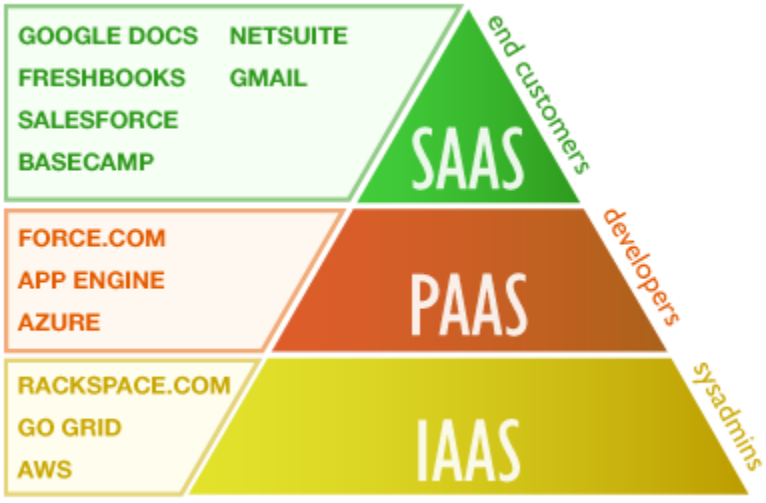


Figure 1 The different cloud computing services

There are generally three types of cloud computing services:

- SaaS (Software as a service)
- IaaS (Infrastructure as a service)
- PaaS (Platform as a service)

SaaS or software as a service is when a provider “rents” particular software to customers that have expensive license fees but for a subscription based payment, or a pay on the go system. This service can be very attractive to customers who want to have access to enterprise software but only pay for what they actually use without having to worry about maintaining, updating or patching the software. Examples of SaaS are Salesforce CRM, email (Gmail), Office 365, customer service and expense manager.

PaaS or platform as a service is similar to SaaS but is marketed at developers. It provides a platform that allows developers to create web applications easily and quickly without having to maintain the infrastructure underneath it. Specifically, if a cloud service provides an operating system, programming language execution environment, database or a web server. Users access these services via API’s, web portals or gateway software. Examples of PaaS include Google app engine, AWS and Elastic beanstalk.

IaaS or Infrastructure as a Service supply virtual server instances and storage as well as API’s to allow users to transfer workloads to the virtual machines. Users have a set amount of storage and can stop, start, pause and resume the virtual machines as desired. IaaS providers normally supply set amounts of computing resources (RAM and CPU cores) and storage that can be started and stopped and that are labelled small medium and large for any image a user might want to launch his or her application. Examples of IaaS include OpenStack and Amazon wen services.

2.3 Why OpenStack?

OpenStack is a free open sourced software platform for cloud computing, that is mainly used as an IaaS. It provides many different features that can control hardware pools, can be made by any manufacturer and can be diverse in nature. This allows the user to take any computing hardware such as older model computers or

new servers and use them for storage processing and networking in a data centre. This diversity of it hardware pool is one of OpenStack's unique selling points in the market for cloud computing. (Juhani Toivonen)

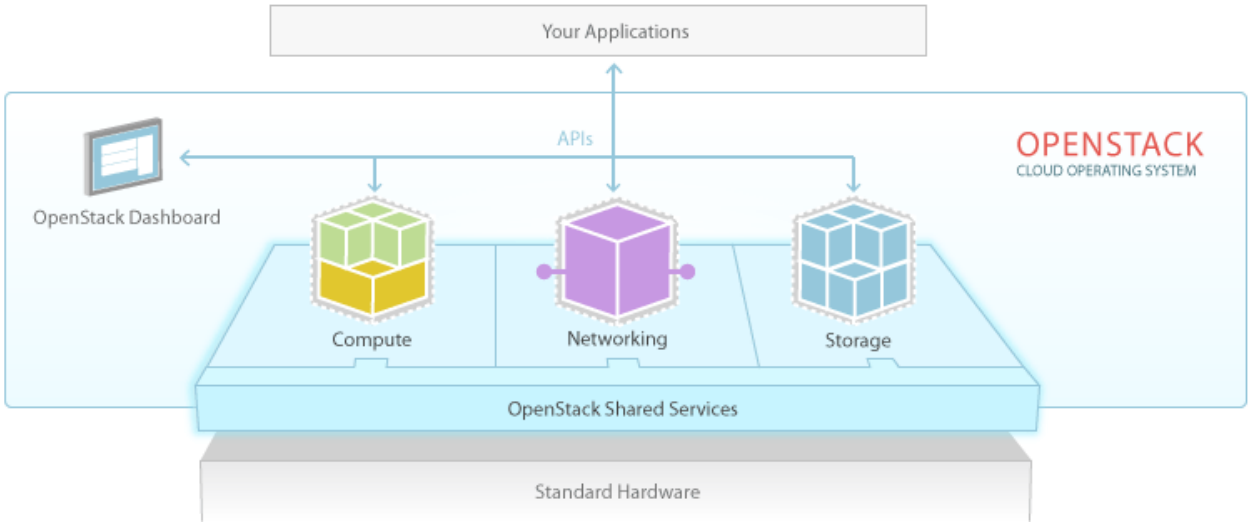


Figure 2 OpenStack diagram

Users can manage the OpenStack Infrastructure either through command line tools, through the optional web dashboard or through RESTful API's.

Figure 3 Example of commands to run OpenStack

Currently more than 500 companies have joined the OpenStack project. OpenStack is very flexible with hardware and can be configure to be any size while still being scalable. It achieves this

by using API's for each service it provides, these services communicate with each other to make it like a modular service.

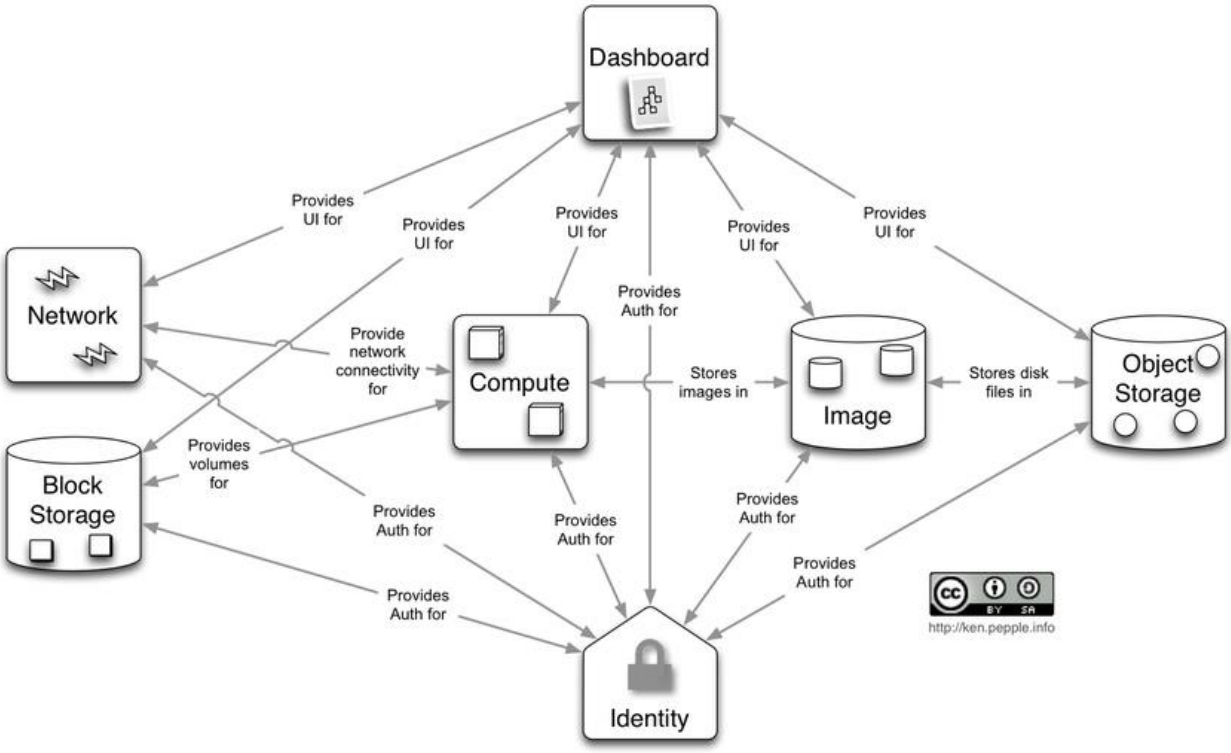


Figure 4Layout on how OpenStack API's work

Each one of these services has a different API name that it refers to itself by, for example the compute service is called "Nova", the network API is called "Neutron" and the imaging catalogue is called "Glance".

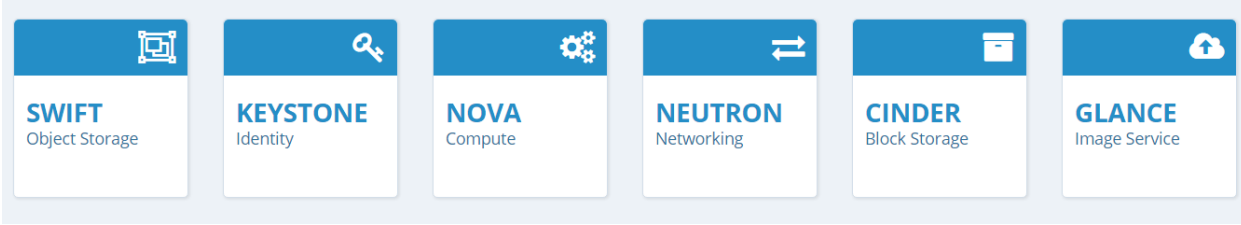


Figure 5 Core OpenStack services

These are all examples of some of the core services that OpenStack provides but OpenStack also has a range of optional services including as telemetry, MapReduce, DNS and Database.

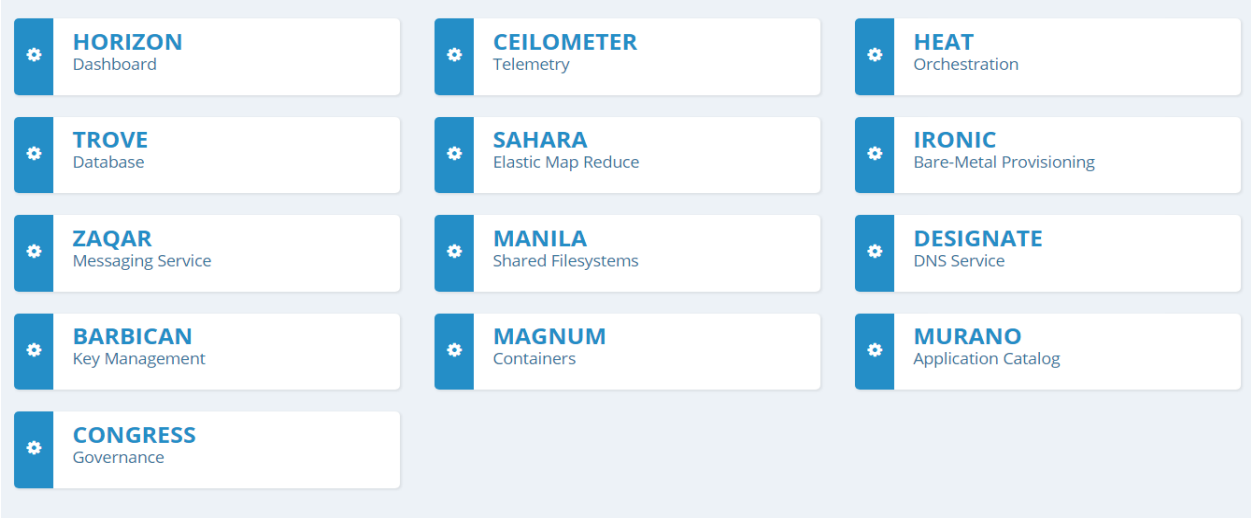


Figure 6 Optional OpenStack services

2.4 Web apps and Servlets

Java Servlet is principally a Java program that can be run on a server. This allows a user the ability to run dynamic web projects written in Java. This is advantageous as the most comprehensive and powerful API' library for OpenStack is written in Java and it is called OpenStack4J. OpenStack4j is an open sourced OpenStack client that has all the major API abstractions; it is straight forward and easy to learn with very detailed error reporting. The main advantages of OpenStack4J are:

- **Expected results**: the data coming back from the server is always what is to be expected.
- **Concrete API**: all the interfaces for the different parts are well-defined which means a user does not have to refer to the implementations.
- **Tested**: The library has been used in several projects and has been tested fully to ensure it is working properly.
- **Exception Handling**: The errors provided by the API library shows in great detail the exact cause of the error in the code. It does not just return an error 404 when an item isn't found.

The use of the Servlet in this project allows the use of a JavaScript front end design that is powerful enough to allow the manipulation

of images and then to output a request to a servlet running OpenStack4J library and receive data in response. No JavaScript front end library was used in this project, but JQuery was used to send http: POST request to the servlet in order to get it to run the commands. (Juhani Toivonen)

3 Analyses

3.1 initial impressions

The project specified *"The main part of the project involves developing a visualization frontend to display the structure layout of a cloud infrastructure and show (in near real-time) the various levels of activity occurring in a cloud environment."* This is quite broad and lead to worry at the start of the project as to which framework would be used to gather information from. The research initially focused deciding what framework would be used to make the front end display. It was clear that a number of objectives would have to be completed for this project:

- A front end to display basic information about the system.
- Have information update in real time or close to real time.
- Show activity going on in the environment.

Incorporate image dragging functionality to be able to change the cloud environment in a more user friendly Way. (Rekimoto, 1997) There are a number of options to choose from when deciding what framework to use.

Snap is an open sourced telemetry framework that allows the collection and processing of data about a telemetry network through one API. While the framework seems very useful and exactly what the project is looking for it doesn't have any features that would be relevant to people from an administrative perspective. So the manipulation of 2-D objects would not be possible using this Framework. Some concerns appeared early about having the network privileges to run administrate commands on the Framework from inside the UCC network. If the framework was to be developed it would be crucial that whichever one that is going to be used allows for users to have permission to run the

services without network admin privileges. This is another reason why it was decided to move away from technologies such as SNAP and ganglia.

Dr. Dapeng Dong suggested that it would be best to use the OpenStack framework as this would provide the ability to complete all of the project objectives. OpenStack had a large number of available API's written in numerous languages with plenty of documentation. An OpenStack implementation could be accessed from within the UCC Network and from outside UCC using a virtual private network. OpenStack does not have a graphical user interface that can be controlled by manipulating 2-D images so if the project is successful there could be a potential market and demand for this kind of application.

I focused on to determine the best framework to use for the visual display of the project.Vis.js and Neo4j contain many useful interactive graphs that can present data from databases in very impressive and simple ways. However, these frameworks did not provide any features that could aid in the drag and drop functionality that was the main objective of the project. After failing to find a relevant JavaScript framework that had drop and drag functionality built in it, it was determined that creating the JavaScript from scratch would be achievable. A template was created using images and the boxes.

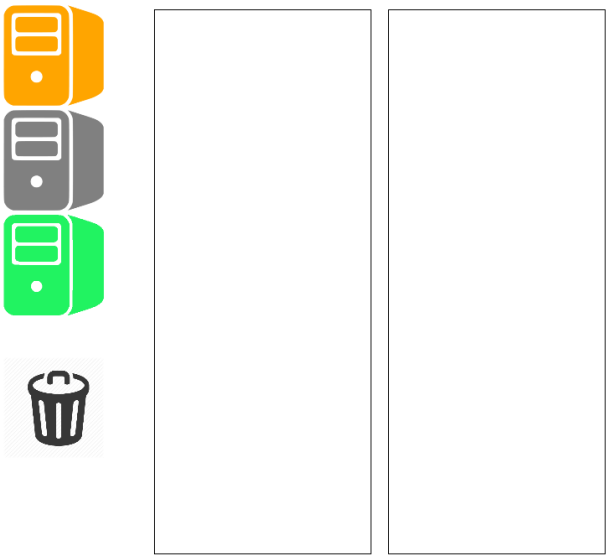


Figure 7 First user interface design

This design presented to the project supervisor who was satisfied with it and felt it was how they imagined the front end would look like. This then became the template for the front-end design of the entire application. From here the server side code can be implemented behind this front end.

Bootstrap is a framework to help make HTML and CSS websites easier and better to use. They have many features that can be incorporated in to the design of the page. The Alert messages were produced using the bootstrap alert system. It makes it very easy and quick to develop HTML features by simply copying and pasting code from the bootstrap website and to apply them to the webpage. Bootstrap also has functionality built in to allow the quick development of a responsive website that can be useful for mobile phone users.

4 Implementation

4.1 Server Side Code

This project was developed inside of eclipse IDE, as it has a number of advantageous features that were used during the development of this project.

- Integration with Tomcat server to allow quick deployment of the project for testing.
- Real time changes that are automatically saved and changes applied to the live server project making it easy to check changes in the code.
- Error handling is comprehensive and is straightforward to spot mistakes and Eclipse even help you fix them. It is suitable for anyone who isn't confident in their coding skills.

As previously discussed the server side code is developed using Servlets running Java. The OpenStack4J library provided the code necessary to run the commands that provides the data of the OpenStack implementation and the commands that were issued to the server to manipulate the implementation.

```
30      //Authentication variables
31      final static String USERNAME = "UserName";
32      final static String PASSWORD = "Password";
33      final static String DOMAIN = "Default";
34      final static String TENANT_NAME = "CloudVisualization";
35      final static String KEYSTONE_ENDPOINT = "http://10.10.10.1:5000/v2.0";
36      
37      /**
38       * @see HttpServlet#HttpServlet()
39       */
40      public MyServlet() {
41          super();
42          // TODO Auto-generated constructor stub
43      }
44
45      /**
46       * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
47       */
48      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
49
50          // Declaring variables that will hold data
51          String flavorID = "";
52          String imageID = "";
53          String networkID = "";
54          String data = "";
55          //The OpenStack4J authentication
56          OSClientV2 os = OSFactory
57          .builderV2()
58          .endpoint(KEYSTONE_ENDPOINT)
59          .credentials(USERNAME, PASSWORD)
60          .tenantName(TENANT_NAME).authenticate();
61
```

Figure 8 Code for how Servlet returned information

This section of the main Servlet gathers the information about the system and returns it in HTML to the JSP page. After importing in the library's that are required in the Servlet, then use the doGet() method that gathers the data and returns it. This servlet is the only servlet that has a doGet() method. Lines 31 to lines 35 is where the logging in information is required. This is where a user types in their username/password and the url for the authentication. Then inside the doGet() method on line 56 to line 60 is the code taken from the OpenStack4J website that is used to authenticate with the OpenStack implementation. The method done in this Servlet for authentication is how the rest of the Servlets authenticate as well.

After the authentication, the work of collecting the data can begin. Using the OpenStack4J documentation we can find the code used to request information from each different part of the implementation.

```
64          // Find all Compute Flavors
65          List<? extends Flavor> flavors = os.compute().flavors().list();
66          data += "<ul>";
67          for (Flavor f : flavors) {
68
69              try{
70              Integer g = Integer.parseInt(f.getId()); }
71
72              catch(Exception e) {
73                  data += ("<li><b>Name: </b>" + f.getName() + "<b> No of CPU's:</b> " + f.getVcpus() + "<b> RAM:</b> " + f.getRam() +"mb</li>");
74                  }
75
76              if (f.getName().equalsIgnoreCase("m1.tiny")) {
77                  flavorID = f.getId();
78              }
79          }
80          // Some HTML feature to present the data
81          data += "</ul>";
```

Figure 9 Servlet code to get Flavour information

This is the method used to get all the flavour information from OpenStack, The information is stored in a list and the relevant information like the names, number of CPU cores, and RAM sizes are outputted to a list that will display on the JSP page. Here is an example of the output that this section of code generates on the main JSP page.

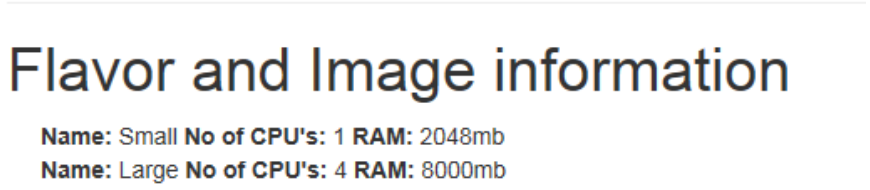


Figure 10 What Figure 9 outputs

The next section works in similar way except using the specific OpenStack4J feature to query that specific service. The Glance service that gathers up all the information regarding what system images can be used to create Virtual machines with. This section of code outputs a dropdown box, with the image ID sored as the list ID so that when a user selects an image that ID is then passed to the JavaScript to be used to call a virtual machine.

```
85          // List all Images (Glance)
86                  List<? extends Image> images = os.images().list();
87                  for (Image img : images) {
88                      data += ("<option value = \""+img.getId()+"\">" + img.getName() + "</option>");
89                      if (img.getName().equalsIgnoreCase("cirros")) {
90                          imageID = img.getId();
91                      }
92
93                  }
```

Figure 11 How Servlet gathers information on images

An example of the output of the code from this section of code is shown here:

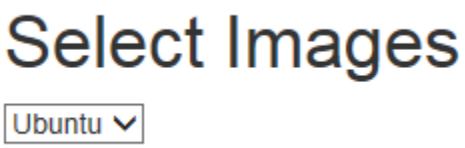


Figure 12 Figure 11 output

After all the information is gathered the Servlet then sends the data to the index.jsp page like in Figure 13 This sends out the information every time it is requested from the Servlet, where the data is then displayed in the <body> tag of the JSP page.

```
116             request.setAttribute("data", data);
117             request.getRequestDispatcher("index.jsp").forward(request, response);
```

Figure 13 How data is sent to index.jsp page

This report will not refer to each method as many of the methods used to output information in this Servlet behave in the same way as above specified method.

4.2 System manipulation

There are four different Servlets in this project and they behave as follows:

- MyServlet.java Gathers information using doGet() and starts virtual machine using its doPost() method
- DeleteServer.java has a doPost() method that deletes virtual machines
- PauseServer.java shuts down virtual machines using doPost()
- UnpauseServer.java reboots shut down virtual machines

The doPost() methods have to be called by the JavaScript inside the index.jsp page to work. The JavaScript outputs the ID's of whatever is needed to complete a task using JQuery to the relevant Servlet and the Servlet then actions the operation.

Take the PauseServer.java Servlet as an example. The code for this section is relatively short and it has the usual authentication that is seen in all of the Servlets and just a few lines of code at the end. This is because to pause a server in the OpenStack4j library all that is needed is the virtual machines ID.

```
51          String VMID = request.getParameter("VMID");
52          PrintWriter reply = response.getWriter();
53
54          if(VMID.length() > 0 || VMID == null) {
55              os.compute().servers().action(VMID, Action.SUSPEND);
56              reply.write("Success");
57          }
```

Figure 14 doPost() method for suspending a server

The method first collects the "VMID" or the virtual machine ID that the user requested to be shut down, then after it makes sure it's not empty the Virtual machine is paused on line 55. This is very similar to how all operations on the system are carried out, similar operations are done on the UnpaseServer.java and the DeleteServer.java servlets.

4.3 Graphical User Interface/ JavaScript

The main page (index.jsp) finished page looks like the below figure when there are no virtual machines instances launched in the

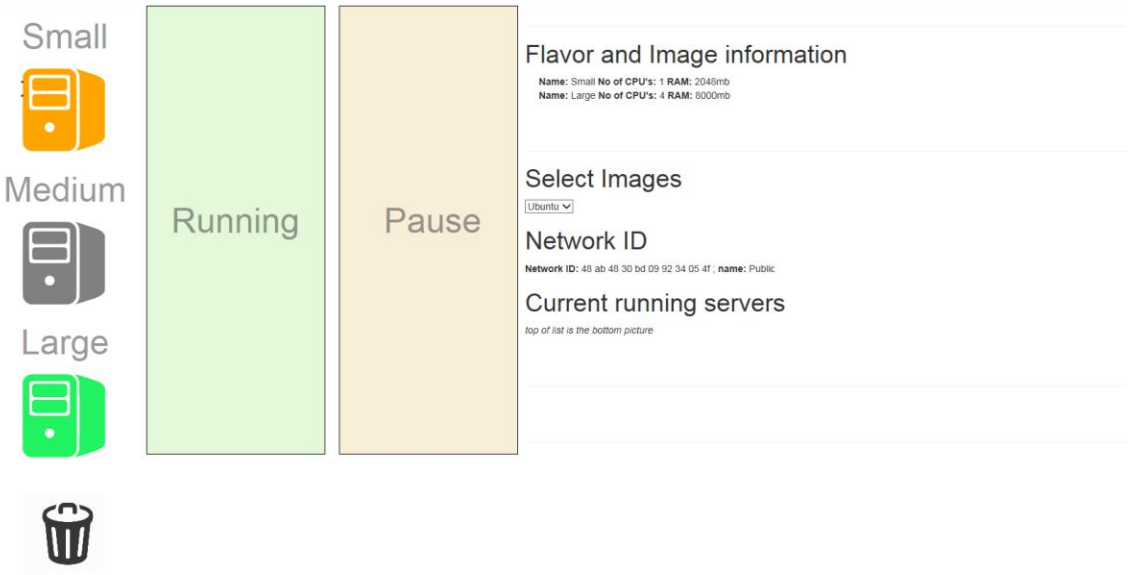


Figure 15 Front page with no servers running

Once an image has been selected from the dropdown box the user then selects which flavour machine they want represented as colours on the left. The user can then drag that machine to the running box which will then launch a virtual machine using the image selected and the flavour details.

The below figure is what a machine looks like after a successful deployment of a server using this application.

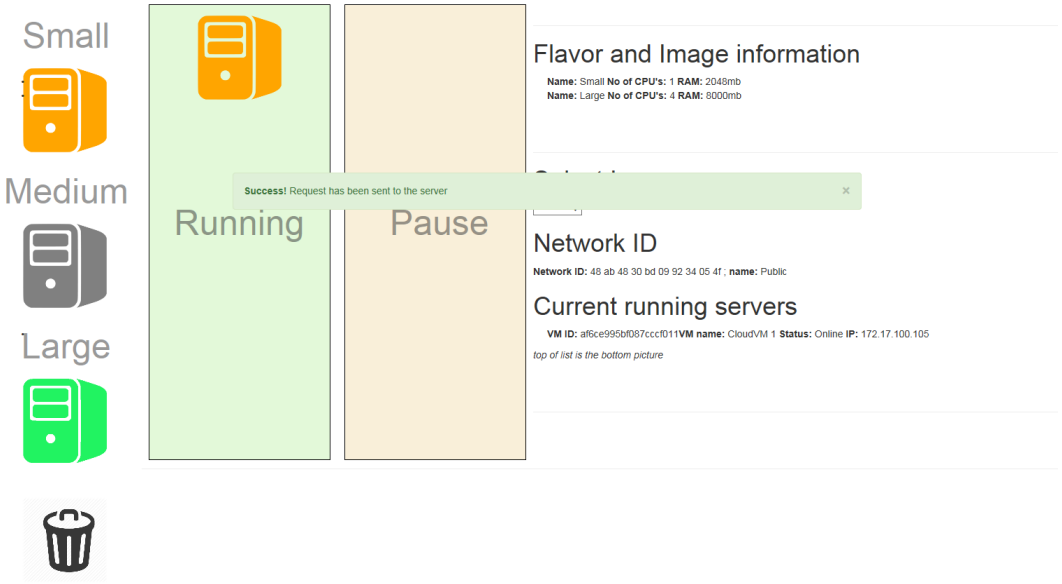
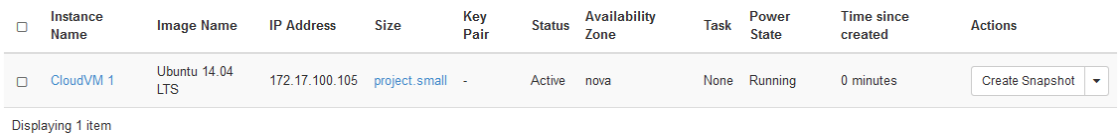


Figure 16 Front page with a successful launch of a VM

As can be seen by the figure above the machine has now been updated and added to the current servers list and a success message is displayed. To prove that the instance is running the OpenStack dashboard can be opened up and the Virtual machine details can be confirmed there.



Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
CloudVM 1	Ubuntu 14.04 LTS	172.17.100.105	project.small	-	Active	nova	None	Running	0 minutes	Create Snapshot

Displaying 1 item

Figure 17 OpenStack Horizon Dashboard showing the Virtual machines state

Here it can be seen that the instance has been made with the same name and IP address, as well as having the same flavour and image selected in this example. When the user wants to pause the virtual machine instance which will shut off the virtual machine all they need to do is drag the image from the running box in to the pause box. This sends the VMID to the PauseServer.java servlet that then shuts down the virtual machine. After successful completion of this operation the screen should look like so:

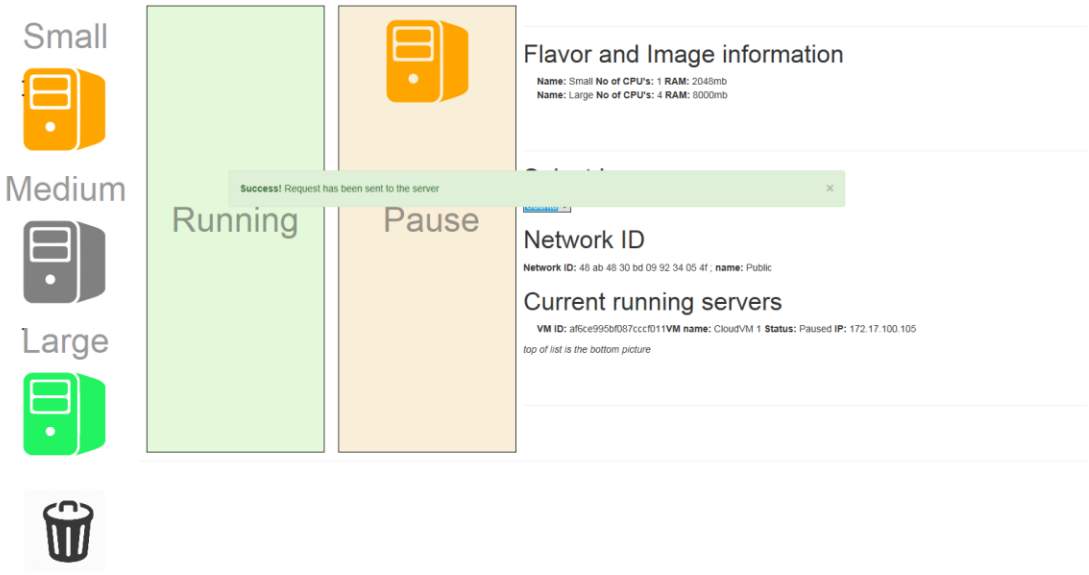
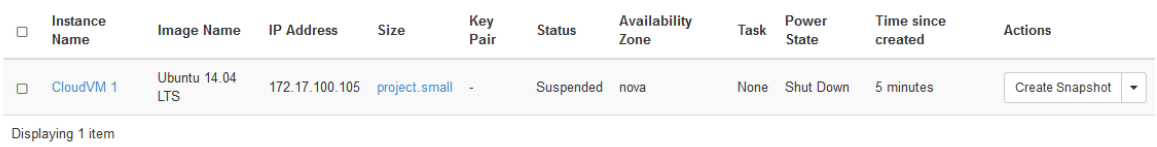


Figure 18 Output when VM is paused

And once again this change can be seen in the OpenStack horizon dashboard:



Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
CloudVM 1	Ubuntu 14.04 LTS	172.17.100.105	project.small	-	Suspended	nova	None	Shut Down	5 minutes	Create Snapshot

Displaying 1 item

Figure 19 Horizon dashboard to show paused server

The server can then be shut off by dragging the image in to the bin image and that will run the Delete server servlet.

The JavaScript that is used to control the events of when images are dropped is done in to the “target” are which is the running box in the application.

```
226 function drop(ev)
227 {
228     ev.preventDefault();
229     var data=ev.dataTransfer.getData("Text")
230     var flavorID = document.getElementById(data).getAttribute('value');
231     var VMID = document.getElementById(data).getAttribute('class');
232 
233     if ( event.target.className == "target" ) {
234         event.target.style.border = "";
235 
236         if (window.imageID == null && data.indexOf("drag") >= 0 ){
237             $("#alertmessage").show();
238             return;
239         }
240 
241         if (VMID != null){
242 
243             var clone = document.getElementById(data).cloneNode(true);
244             if(data == "drag1"){
245                 clone.id = Math.floor((Math.random() * 10) + 1);
246             }
247             else if(data == "drag2"){
248                 clone.id = Math.floor((Math.random() * 20) + 11);
249             }
250             else{
251                 clone.id = Math.floor((Math.random() * 30) + 21);
252             }
253             ev.target.appendChild(clone)
254             unpauseserver(VMID)
255 
256         }
257         else if (VMID == null){
258             var clone = document.getElementById(data).cloneNode(true);
259             if(data == "drag1"){
260                 clone.id = Math.floor((Math.random() * 10) + 1);
261             }
262             else if(data == "drag2"){
263                 clone.id = Math.floor((Math.random() * 20) + 11);
264             }
265             else{
266                 clone.id = Math.floor((Math.random() * 30) + 21);
267             }
268             ev.target.appendChild(clone)
269             createServer(window.imageID, flavorID)
270         }
271 
```

Figure 20 Code for the drop functionality in JavaScript

At the start of this drop function, there is an attempt to collect data from the image that has been dropped, such as if it has a virtual machine ID or if it has a flavour ID located in the value class of the

dropdown box list of images. On line 233 it asks if the location of the image is dropped is on the running box. Inside of that if statement there is a quick check to make sure an image has been selected before the user attempts to start a virtual machine. The error message looks like this:

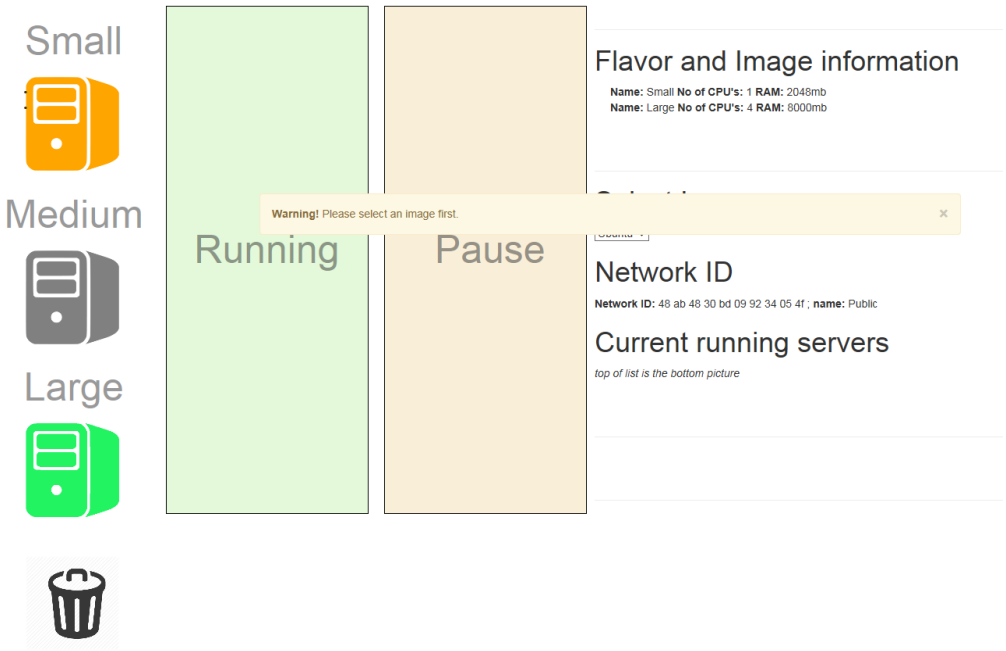


Figure 21 Warning message example from the application

On line 241 and line 257 we check to make sure if the image is already associated with a virtual machine i.e. if it has a VMID a new image is created with ids based on its size and unpauseserver(VMID) method is run. If the image is not associated with a VMID then the application assumes the user is trying to create a new instance. It creates a new image in the box and starts the method createServer(window.imageID, flavourID) which sends the data of the flavour and image selected to the method. The methods createServer() and unpauseserver() all behave the same way with only slight changes to the URL the message is sending to

and the data it is sending.

```
133 function createServer(imageID, flavorID){
134
135
136     jQuery.ajax({
137
138         url:"http://localhost:8080/test2/MyServlet",
139         data:{"flavor":flavorID,"image":imageID},
140         type:'POST',
141         dataType : 'xml',    //use 'jsonp' for cross dom
142         success:function(data, textStatus, jqXHR){
143             // access response data
144             $("#selectCodeNotificationArea").show();
145         },
146         error:function(data, textStatus, jqXHR){
147             $("#selectCodeNotificationArea2").show();
148         }
149     });
150
151 }
```

Figure 22 JQuery code for outputting POST to Servlet

The code above is an example of how data was sent to the server using JQuery. The URL to the server is specified and the data is collected an inserted in to the POST and it is then shipped to the servlet. If it encounters an error it returns an error message and if it returns successfully then it displays a success message. The rest of the methods work the same way except for small changes to the URL and the data.

This is an overview on how the JavaScript behaves by reacting to drop events and running methods to send the relevant information to the Servlets After every drag and drop action is completed the page needs to reload to get the newest system information , so a method has to be implemented to draw the images in the box when the page loads.. A section in JavaScript was made so that when a server was already running, the JavaScript will generate the correct flavour image with its virtual machine ID inside of it, in the box that corresponds to the state of the machine i.e. if it's running it will be in the running box. This is carried out by this section of code in the index.jsp page

```
83      var listLength = $("#mylist li").length;
84      var tmp = listLength -1;
85      var lis = document.getElementById("mylist").getElementsByTagName("li");
86      while(tmp >= 0){
87          if (lis[tmp].id == ("5f35e168-98d3-4457-93a6-1c33a993fa4c")){
88              var clone = document.getElementById("drag1").cloneNode(true);
89
90              clone.className = lis[tmp].className;
91              if (lis[tmp].textContent.indexOf("PAUSED") >= 0 || lis[tmp].textContent.indexOf("SUSPENDED") >= 0){
92                  $("#div2").append(clone);
93                  clone.id = Math.floor((Math.random() * 500) + 101);
94              }
95              else{
96                  $("#div1").append(clone);
97                  clone.id = Math.floor((Math.random() * 10) + 1);
98              }
99              tmp -= 1;
```

Figure 23 Code from JavaScript on how images are drawn in to boxex

The image goes through the list of data about the current running servers, it then checks to see what flavour it is on line 87. If it is that flavour which in this case is the small flavour it will clone the small server image. Then it needs to find out if the server is paused or running so it searches the node to see if it has the word "PAUSED" or "SUSPENDED" in it. If the server is in the paused state then the image will be drawn to the "paused" box (#div2) and if it is in the running state then the server is put in the "running" box. The while loop variable is then reduced and it moves on to the next item in the list. The JavaScript checks against all three image flavours that have similar methods like the one shown above.

4.4 Problems

Coding in Eclipse IDE framework made the project much easier to code. The platform handles all the hosting part of this project, all that is needed is a link to a tomcat server and eclipse will host it for you. There were a few issues in implementing this project the first was the network in UCC. The OpenStack implementation was on another separate network in UCC which made it difficult to gain access to the network. Because of this there is no way to access the UCC OpenStack from outside the UCC network. To fix this problem more RAM was purchased for my computer at home so that it can have the minimum specifications to run OpenStack. This allowed for more time to get experience on how a Cloud environment worked. Another problem faced during the implementation had to do with Eclipse, although the IDE made it simple once the process of setting up this project to work on another machine is quite difficult and made version control very difficult in this project. Normally Git would be used in software

projects like this, but it ended up being impractical in this project due to external jars/tomcat and file permission issues. Instead I used Bazzar in this project it worked much better than Git did.

5 Evaluations

As this project was a software development project, the two ways that can be used to test the project are:

1. Code testing

2. Surveys

It was decided that in order to determine that the code outputs correctly was to check ID the html file that the programme is outputting is valid. It is also crucial to get feedback from users about the usability of the software, and if possible to collect data from users with past experience using Cloud management software.

5.1 HTML validating

The first area in the code testing was to make sure that the JSP file was outputting the correct HTML. The validation of the HTML code was carried out by running the test on https://validator.w3.org/. Although no errors can be seen from the browser that doesn't always mean that the HTML code is correct. After running the HTML validator on the project there were over 41 errors in total just on the main page alone. Most of these errors were simple errors including not writing "type=" after some tags on the page.

Other errors showed big inconsistencies in the writing of the HTML that revealed flaws in the image dragging. This error was creating difficulties for the server images to be dragged in to the boxes and the bin. Fixing these errors in the HTML resulted in the most successful test that could have been done to help usability on the webpage.

Once the page was successfully validating on the main page, the next step was to make sure that all other pages are producing valid html pages. The HTML page test validation was run several times

after turning on/off servers and changing image selections. They all produced valid HTML pages as shown below.

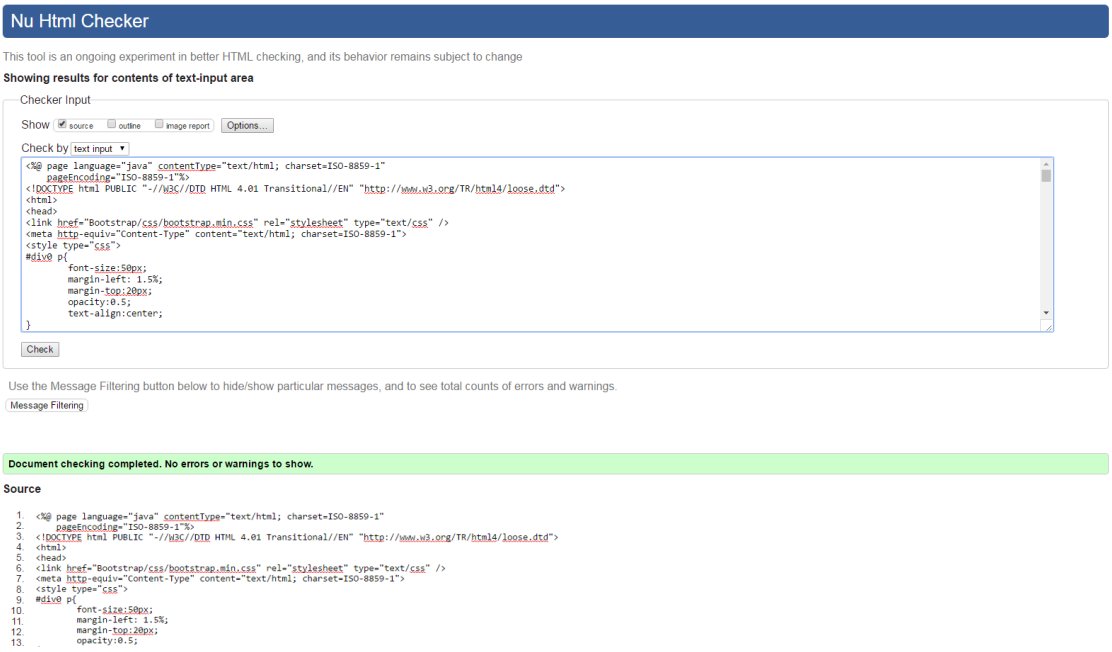


Figure 24 HTML 5 validator output

5.2 CSS Validator

The CSS validator like the HTML validator was crucial to ensure the CSS code was correct and behaving as expected.To do this the W3C school CSS validation service was use at https://jigsaw.w3.org/css-validator/. There are many reason why someone would want to make sure the CSS is valid:

- **Validation as a debugging tool:** Not all programs handle errors gracefully and many will handle errors in the CSS in different ways. Making sure the CSS is valid will ensure it runs the same on all browsers.

- **Validation for future proofing:** Just because a application runs correctly on a number of browsers today does not mean that it will work in the future. By checking that the CSS complies with the web standards will security that the code should work in the future.

- **Easier maintenance:** By making sure that the HTML/CSS is valid and complies with the web standard makes the code

much easier to maintain even if developed or changed by another person.

- **Valid CSS/HTML can lead to a better score:** Google search algorithm checks all pages to make sure that the pages have valid pages. If the pages have errors they will be ranked lower in Googles search queries. If the project was to be released it would be crucial to the success if it was given a high search ranking.

Due to all these reasons the validation of the CSS was crucial to ensure the successful running of the project. When the validator was run on the project only one error came back and it was fixed immediately and now the project is validated in CSS. These test are very important so that the user design of the project doesn't have the same problems that some of the other Cloud computing frameworks. (Goyal, 2010)

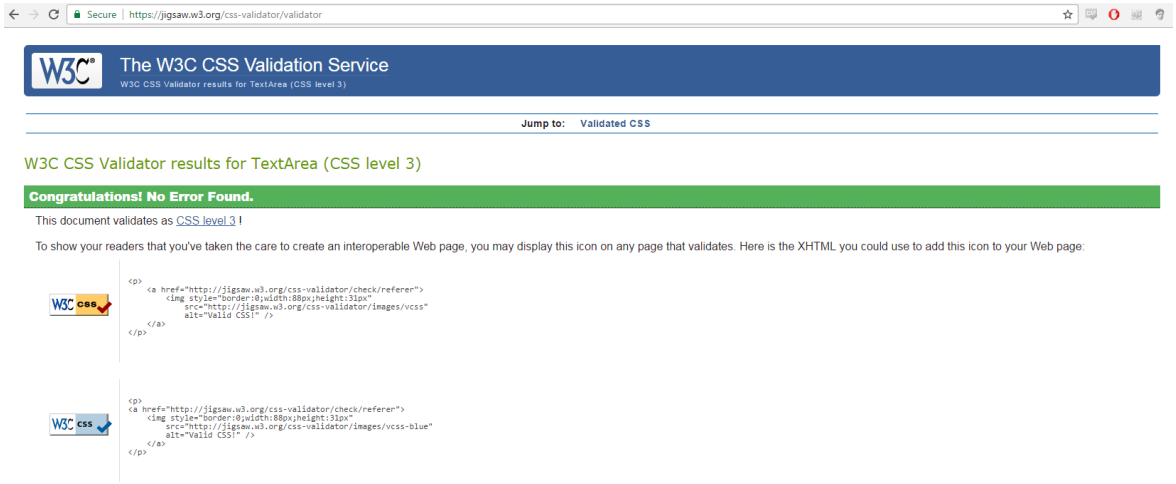


Figure 25 CSS Validation output

Valid CSS information

```
#div0 p {
  font-size : 50px;
  margin-top : 20px;
  opacity : 0.5;
  text-align : center;
}

#div0 img, #div1 img, #div2 img {
  display : block;
  margin-left : auto;
  margin-right : auto;
  margin-top : 5px;
  margin-bottom : 5px;
}

#pausetxt {
  position : absolute;
  font-size : 50px;
  margin-left : 3.2%;
  top : 30%;
  opacity : 0.5;
}

#runningtext {
  position : absolute;
  font-size : 50px;
  margin-left : 1.5%;
  top : 30%;
  opacity : 0.5;
}

.alert {
  width : 900px;
  position : fixed;
  left : 350px;
  top : 250px;
  z-index : 2;
}

ul {
  position : relative;
  left : 20px;
}

#div0 {
  float : left;
  width : 220px;
  left : 5px;
}

#div1 {
  background-color : rgba(201, 243, 180, 0.5);
}

#div2 {
  background-color : rgba(243, 223, 180, 0.5);
}

#div1, #div2 {
  float : left;
  width : 260px;
  height : 650px;
  margin : 10px;
  padding : 10px;
  border : black solid 1px;
}
```

Figure 26 Code from CSS validation output

The application has been tested on and works very well on chrome and Internet explorer. Further development is needed to get the application working in more browsers such as Firefox and Safari. The only problems that occur on the application are generally networking issues, but the OpenStack4J error handling makes it very easy to fix the problem. If this was to be developed on access to a network with admin privileges there wouldn't be any networking problems at all.

5.3 Survey

As this project depends so much on usability the most efficient test that could be done for this project was to take a survey of a group of users after they tried out the project. A survey was made using

Google Forms, where a task was giving to the user taking the survey to create an Ubuntu virtual machine using the programme, then to shut down the sever and to restart it again.

Almost all users were able to complete the task without any assistance at all which indicated high degree of usability that the users were familiar with. The Survey then asked the user if they had any previous experience using cloud management software before, if they had then follow up questions are asked comparing the difference between using whatever cloud management software that the user has used before and using the system supplied in this project. The questions were to determine if the user found using the project system was easier or harder than using it the original way. The survey also asked if the user had any feedback on the project. On the day of the Computer Science 4th year project open day users could choose to fill out the survey on the computer next to the one that hosted the project or the user could scan a QR code linking to the survey to complete on their phone.

Figure 27 QR code to survey shown on open day

In total fourteen users completed the survey with varying degrees of experience with previous cloud computing software.

The first question asked the user to grade how hard the task was that they had to complete where 1 was very easy and 5 was very difficult. The results were as follows:

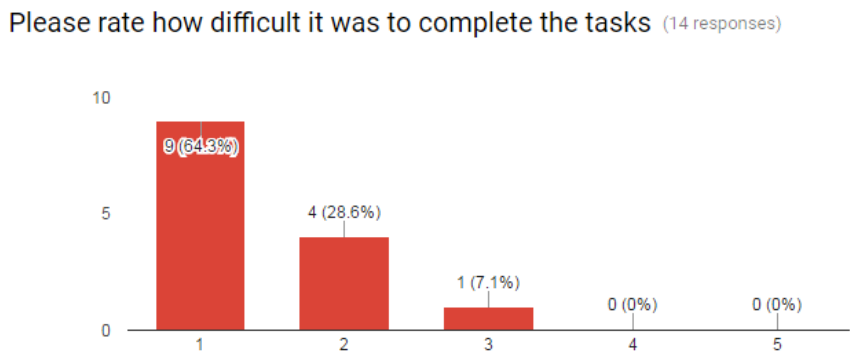


Figure 28 Difficulty to complete task

The results show that most users were able to accomplish the task very easily and the rest (35.7%) were still able to accomplish the task easily or in a normal amount of time. No users found the programme difficult or very difficult to use.

The next question asked users to rate how useful was the information that was provided to them, 1 being very useful and 5 representing not useful.

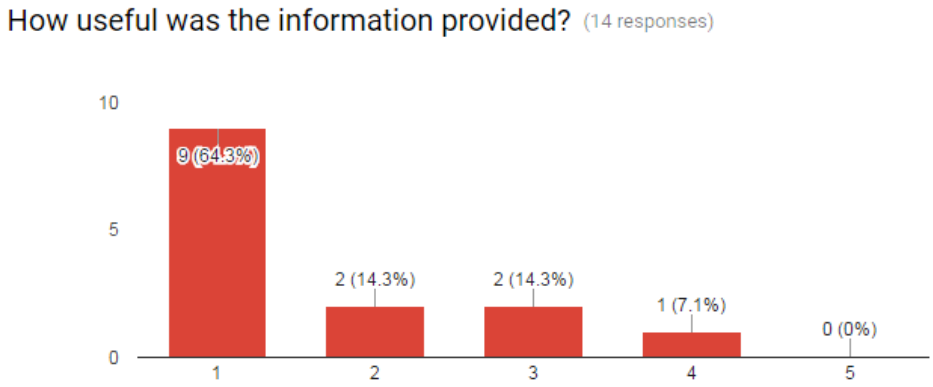


Figure 29 graph for information usefulness

As can be seen by the results above the same percentage of people found the information provided to be very useful. However this time users were more dissatisfied with the information shown to them with one user rating the information as not helpful. This is a clear indication that improvements will be required to improve the information given to the user, by making the messages clearer and to display information in an intuitive way such an interactive graphs.

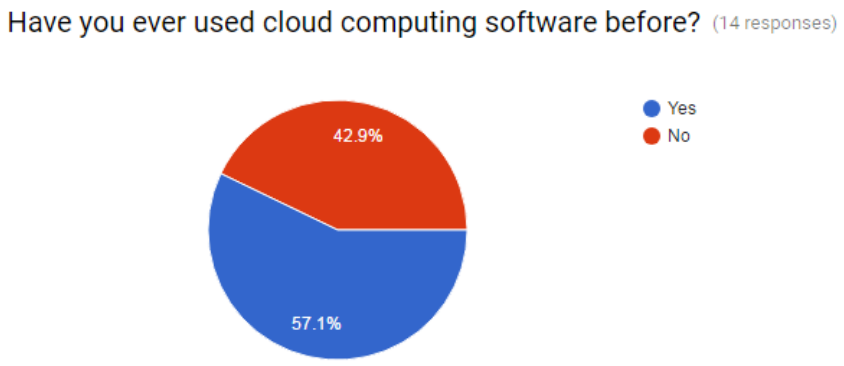


Figure 30 Pie chart for previous experience

We can see from this pie graph that the majority of people who took the survey have previous experience with cloud computing software. These people were judged to provide better feedback than users who have never used cloud computing software before, and follow up questions were only asked to this group of people.

Figure 31 Pie chart comparison of ease

From the results above an overwhelming majority of users that did use previous cloud computing software found the application in this project to be easier than what they are used to. This was the main objective of the project and this indicated that the design used in this project for drag and drop functionality really does provide users an easier and more user friendly ptogramme to accomplish

tasks on the system.

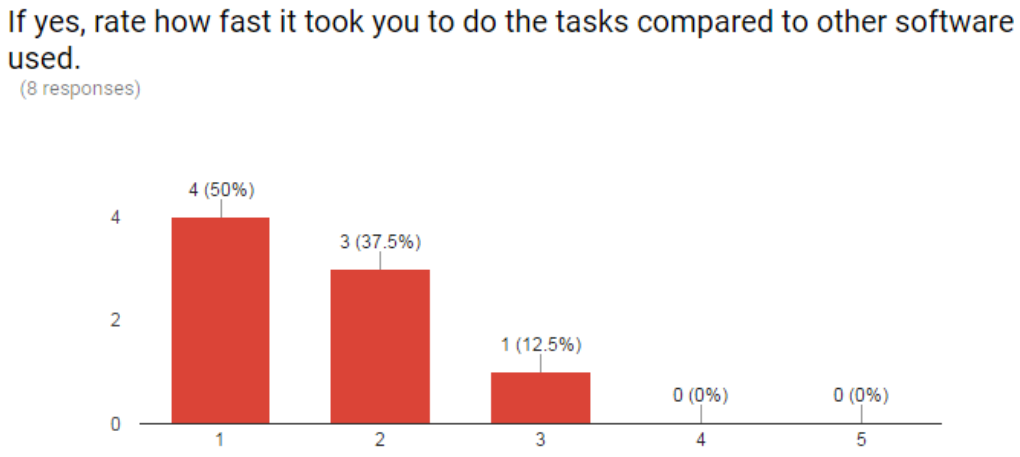


Figure 32 Graph of speed of application

From the graph displayed above it can be seen that over 85% of users found that the application was not only easier to use but also quicker in its operations. This is a great benefit to end users as saving time doing each operation is generally seen as good user interface design.

Some feedback was left on the survey but the feedback received in person was more detailed than the responses on the survey.

Overall from this survey it can be seen that the project was a success with the majority of people finding the new application faster and easier than conventionally used cloud management software. Any follow up surveys would include questions asking users what cloud computing software they have used and to compare this application with other applications that users have tried.

6 Conclusions

6.1 What has been accomplished?

The objective of this project was to make a graphical user interface that can be used to manipulate a cloud computing network by dragging 2-D images. While this objective has been reached further

improvement work will be undertaken to the functionality of the application. The server side code has been developed and now the next step for this application is to add additional features. So far the system displays basic information about the system such as what images are installed, CPU/RAM sizes, IP addresses, and virtual machine names. Feedback suggests that this information isn't enough and it could be done better. This information will be crucial to any future success of this software.

The implementation of this project as a web based application proved to be a success making the application much easier to access and use. Once the application was set up on the network its ease of use was found to be very helpful to users.

During this project I set out to learn about how cloud networks operate and how they function on a deeper level. Being able to manipulate virtual machines on a network was a great learning experience and I have gained a tremendous amount of knowledge from this project. Having set up a cloud implementation of OpenStack on my own computer provided a learning experience that normally wouldn't be experienced otherwise.

6.2 What can be done next?

From the feedback on the survey and from some of the feedback received from others at the project open day, it is clear that the main fall back of the application is the need for more at a glance feedback from system specific information. Information such as the total amount of CPU cores available vs amount of CPU cores used. One very helpful advice received on the day was to used Google Charts, a JavaScript tool to make interactive graphs from system information. An example is to make a pie chart to display the hard drive space of the system so that users can at a glance know how much system storage there is left.

After the project was finished all of the core functions for the OpenStack library have been used in the application. The "Glance" API was used for the image manipulation and "Nova" for its computing features such as turning off and on virtual machines.

The OpenStack library has much more optional functionality than this. Currently the application only publishes virtual machines to one public network; using the "Neutron" features for network we could add more options for the user to create networks and subnets of their own. Using the servlet template we can add functionality for this as long as a tool can be found for displaying this information to the user. This way more features can be added on to the existing framework modularly. It would allow for faster and easier developments especially with the adaptation of interactive graphs. Adding more features would add more incentive to use the software as not only will it be easier to use and faster but it will be as comprehensive as other cloud management systems.

Due to the nature of the user design interface and the drag and drop manoeuvres of the images; it would be very interesting to develop this project as a mobile phone app. Allowing users to easily control their cloud networks from their phones would be an interesting feature. There are different ways this could be achieved. An android app could be developed to interface with the data from the network. This way a user would be more immersed in an application and have access to more usability functionality. The project could also be devolved so that the webpage can responsive. And have mobile phone specific functionality. This would be a very interesting experiment to see if the same functionality display from a desktop screen can fit in to a mobile phone screen. (Andreas Konstantinidis, 2012)

A suggestion from Dr. Dapeng Dong was that additional functionality could be designed allowing a user to right click a specific virtual machine and the programme to provide an option to ssh to that machine should appear. This would be an interesting but difficult development to do as JavaScript will not allow users to open ssh connections from the browser. It may still be possible to do in the future by implementing it with FireSSH, a JavaScript based plugin for browsers. This would make it very easy for users to quickly go in and make individual changes to each machine by quickly connecting to it via ssh

Overall the project main goals were accomplished users can now do basic cloud computing manipulation easier and faster using the application. The application shows basic cloud system data that are

updated in near real time. The knowledge gained from this project is very valuable and the experience was very fun and exciting.

Appendix
Works Cited

Andreas Konstantinidis, C. C.-Y. (2012). Demo: a programming cloud of smartphones. *MobiSys '12 Proceedings of the 10th international conference on Mobile systems, applications, and services*, 465-466.

Goyal, P. (2010). Enterprise Usability of Cloud Computing Environments: Issues and Challenges. *2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*.

Juhani Toivonen, S. H. (n.d.). EASI-CLOUDS - Extended Architecture and Service Infrastructure for Cloud-Aware Software. *D5.10 – Final Report on Cloud Computing*, 90.

Omar SEFRAOUI, M. A. (October 2012). OpenStack: Toward an Open-Source Solution for. *International Journal of Computer Applications (0975 - 8887)*.

Rekimoto, J. (1997). Pick-and-drop: a direct manipulation technique for multiple computer environments. *UIST '97 Proceedings of the 10th annual ACM symposium on User interface software and technology*, 31-39.