UNIVERSITY OF CAPE TOWN

Digital Repositories in Private Clouds

by

Mushashu Mwansa Lumpa

Supervised by: A/Prof Hussein Suleman

Dissertation presented for the degree of Master of Science

in the
Department of Computer Science
University of Cape Town

February 2019

Plagiarism Declaration

I know the meaning of plagiarism and declare that all of the work in this dissertation, save for that which is properly acknowledged, is my own.

Signed: Signed by candidate

Date:

"Journeying through the unknowns to the unknown. We'll connect the dots later."

anon

Abstract

This study explores the use of digital repositories in private cloud environments. Private cloud computing is a cloud computing deployment model where compute and storage infrastructure are hosted on-premise by institutions. Digital repositories are used to manage institutions' generated content. The advancement in cloud computing, the promise of elasticity, and the on-demand resource provisioning features of cloud systems are attractive characteristics that institutions can leverage on in delivering digital content to their audiences. In this study, a cloud computing operating system is deployed, and a means to install, monitor, manage and customise a repository system is developed. The repository system used is DSpace.

Eucalyptus cloud software was used to setup a private cloud environment. A prototype application was developed to manage the installation and customisation of DSpace in the cloud environment. The prototype also included a feature to monitor the status of the running DSpace instances. To evaluate the efficiency, installation and customisation of DSpace in the cloud environment, two types of evaluations were carried out – a performance evaluation and a usability study. The performance evaluation was used to ascertain how long it takes to ingest and view items in DSpace. The experiments were carried out with varying numbers of running virtual machine instances in the cloud. The usability study evaluated the ease of installing and customising DSpace with the developed tool, called Lilu. A total of 22 participants took part in the usability study that was carried out within the premises of the University of Cape Town's Computer Science Department. The participants belonged to 3 groups – experts, intermediate and beginners – based on their technical skill levels.

The results show that private cloud environments can run institutional repositories with negligible performance degradation as the number of virtual machine instances in the cloud are increased. From the usability study, the tool developed was positively perceived. Participants in the study were able to install and customise DSpace.

Institutional repositories can efficiently be installed and used in private cloud environments. Building tools that enable users to create single-click installations of the repositories, and creating user friendly interfaces to customise repositories would potentially increase the adoption and utilisation of private cloud environments by institutions.

Acknowledgements

I am grateful to my supervisor, Prof. Hussein Suleman. He has been supportive even in times when I had completely given up. I cannot see how I would have seen this to its end without his desire and effort to continue providing guidance and supervision for this work. Through his guidance during my studies he's transformed how I logically view and fix challenges.

Thanks to Dr. Roma for the support and the continued mentorship, to Dr. Bolton and Dr. Savory, for the push to make sure I finished my studies, To both Dr. Roma and Dr. Bolton for finding me funds to complete my Masters.

Maletsabisa, Martha, Gina, Ngoni and Lighton - in you I have made friends for life. Grateful for the outdoor events, the support in all forms provided, academically, socially, and emotionally. Mpande M, a very good work colleague and friend for continually being helpful and encouraging.

Russel Philander, thanks for the visits and making me feel less of a stranger in your city. You have been a true friend.

Many thanks to my family for all the support during my studies.

To my mother for the unwavering unconditional love, I will remain grateful eternally.

Contents

List of Figures

List of Tables

To my mother, Mabel Mukatimui Muyangana…

Chapter 1

Introduction

Cloud computing is not necessarily a new technology but rather a new operations model that brings together a set of existing technologies to run in a different way [1]. A number of definitions for cloud computing have been proposed, but in the context of this project, the National Institute of Standards and Technology's (NIST) definition, paraphrased, describes it as a model of enabling on demand network access to a shared pool of computing resources, that can be provisioned and released with minimal management effort [2].

Cloud computing brings with it some characteristics that are desirable to institutions. As Han [3] notes, some advantages that cloud computing include: cost-effectiveness – institutions will only use resources as they need them, and only pay for what they use; flexibility – the availability of compute resources on-demand, deviates from traditional approaches where purchasing of computing resources would be a pre-requisite to prototyping software applications; etc.

Digital repositories are software tools that are used to manage digital content, share it and provide means to provide potential long-term preservation of that content. Digital repository tools will primary be used by libraries and generally institutions like universities that continually generate content through scholarly publications and teaching materials.

If institutions can deploy digital repositories in cloud environments, they can leverage the benefits that cloud computing provides. Institutions would then devote more time to managing their digital content than their compute infrastructure and the software make-up of the digital repositories. And on the infrastructure end, administrative functions would benefit a lot from the self managing features of a cloud environment coupled with the efficient usage of their computing resources. Institutions' digital repository content, or generally its content, is always growing and, as such, infrastructures that scale and provide inherent elastic features are ideal to contain this content growth. Also, digital repositories are required to contribute to the preservation of their content, and running them in cloud infrastructures have a benefit of having their data potentially replicated in the variety of storage options that cloud environments provide.

For institutions to take advantage of the benefits of the cloud, it is important that there exist tools that simplify the deployment, management and monitoring of the digital repositories.

This study explores the use of private clouds for hosting digital repositories. The following sections give the research questions to be investigated, a summary of the methodology to be used and the overall scope and limitations of this study.

1.1 Project Objectives

In general, the objective is to develop tools that are compatible with cloud computing technologies and standards. In doing so, different tools commonly used in cloud environments will be explored. In a nutshell, the following are the objectives:

1. Create a one-click installation of digital repository tools.
2. Develop a way for novice users to monitor digital repository activity in cloud environments.
3. Develop a way for users to customise repositories even when such users are lacking in Web development technology skills.

1.2 Motivation

There is a growing need for institutions to have their own institutional repositories. This may be attributed to technology advancements, and the mode in which users are increasingly accessing information using the Web. Institutions also need to continually implement ways to increase the potential preservation of their content. Digital Repositories may help to meet some of these institutional objectives. Additionally, cloud computing presents desirable features, that given user friendly tools that help interact with cloud environments there would be, arguably, an increase in the adoption of institutional repositories.

Therefore, this study considers ways for hosting digital repositories in private cloud environments. The proposed solution abstracts the complexity of the underlying cloud infrastructure, and thus enabling users to focus more on content they wish to publish.

1.3 Research Questions

This project attempts to answer the following research questions

1. Is it possible to host digital repositories in Private Cloud environments efficiently.
2. How can common repository tools like DSpace be adapted for the cloud for easy management and deployment?

1.4 Methodology

This study involves hosting a digital repository in a private cloud computing environment. The process involves simplifying the installation process of DSpace and automating its installation in a virtual machine instance. The virtual machines are run in a cloud environment setup using Euaclyptus.

Eucalyptus is a cloud computing operating system that enables the provisioning of Infrastructure as a Service service.

A browser based application is developed to aid the installation process, customisation of DSpace and virtual machine management for ordinary repository end users. Installation is performed via a few clicks and provides the necessary information required to identify and log on to the DSpace repository installed. The application also enables end users to perform customisation of the DSpace repository, branding it with colors and logos according to an institution's branding policies.

Two types of evaluations are carried out: 1) performance experiment, 2) usability study of the developed browser based application for installation and customisation of DSpace.

The performance experiment involves viewing and ingesting items in DSpace. The experiment is run with varying number of instances in the cloud. The tests were carried out in a private cloud that supports a maximum of 12 instances. The results are plotted to check for correlation between the performance of viewing/ingesting items and the number of instances running in a cloud. In addition, the evaluation looks at the ingestion order of items to see if there is any noticeable effect as items are added to a repository one after the other.

The usability study involves asking participants to carry out an installation and also customise their installed instance. The System Usability Scale (SUS) is utilised to assess the overall usability of the developed tool – in effect assessing the efficiency of installing DSpace in a cloud environment. In addition, the After-Scenario Questionnaire (ASQ) is used to assess individual tasks of installation and customisation using the developed application.

1.5 Scope and Limitations

This work does not explore the importance of public vs private cloud computing environments. It specifically explores the deployment of an institutional repository in a private cloud computing environment and how it can allow ordinary non-skilled users to operate it.

The implementation looks at taking advantage of the cloud computing's ephemeral nature of compute instances and the cloud's block storage that provides persistent storage for the running instances.

The study does not evaluate the existing cloud computing systems and the available configuration and systems management tools. It rather builds on the core principles of cloud computing to provide the suggested solution. It is expected that the proposed solution's underlying principles can be applied in other private cloud computing environments that provide IaaS features for compute and storage.

1.6 Dissertation Organisation

Chapter 2, Related Work, gives a background to the technologies used in this study. Related work in the area of hosting digital repositories in private cloud environments is discussed.

Chapter 3, Design and Implementation, gives an overview of the rationale to the solution provided, the choice of the technologies and how they function. The process to install DSpace is discussed. The Web application to manage installations and customisation of DSpace is described.

Chapter 4, Evaluation and Results, describes the experiment and usability study that was carried out to evaluate the proposed solution. Results of the experiment and analysis of those results is given.

Chapter 5, Conclusion, summarises the work done in this study, and whether the objectives were met. It ends by suggesting potential future work for this study.

Chapter 2

Related work

This chapter introduces the concepts used in the rest of this report, and goes on to discuss works in the area of hosting repositories in Private Clouds. Digital Repositories are described, detailing their common architectures and the different technologies used in their implementation. An explanation of cloud computing follows, laying the foundation for later sections that discuss possible tools used to automate, manage and monitor applications in clouds.

2.1 Digital Repositories

Digital Repositories are used by institutions to manage digital content, share it and provide means that aid in the long-term preservation of that content. Institutions continually generate content through scholarly publications, teaching materials, theses, and research outputs. Therefore, digital repositories serve an important function as a tool to aid institutions' challenge to manage their ever growing content. With the advent of the Open Access[1] movement, there is increased motivation for institutions to generate more digital content and publish it to a wider audience on the Internet. Digital repositories are used to manage different data formats, some of which include images, videos, and free text files.

There are different types of digital repository software products available, and some of the commonly [4] used ones include DSpace [5, 6], EPrints [7] and Fedora [8]. These repository tools share similar features, and to some extent are developed using the same technologies. A number of studies [4, 9, 10, 11] have been done to compare the different features that each of these provide. Of interest in this study is the technological make up of the digital repository tools, and the modes in which they are installed and customised.

[1]What is Open Access?, `http://www.digital-scholarship.org/cwb/WhatIsOA.htm`

DSpace is an open source product, which is widely used and has an active developer community. It is developed in the Java programming language and runs off a PostgresSQL or Oracle database backend. Execution of Java requires the use of a Java Container and the commonly used one is Tomcat. DSpace installation [12] requires running commands that may require users to have technical skills [13]. Installation of DSpace is not trivial and often requires multiples tries and a sizeable amount of time for to get it right.

Installation, configuration and customisation of EPrints and Fedora are not any less complicated compared to DSpace. They do not come with a guided installation graphical user interface [14, 15]. They both require one to have some technical skills to run the commands that are shared in their installation documentation.

Generally, installation and configuration of digital repositories is not a trivial task. Korber et al [13] established in their study of repository tools that more attention is paid to improving end-user usability of repository tools, whilst giving in little work into the improvement of administrative tasks. To cement their argument, they point out that in the DSpace mailing list, installation and configuration questions are among the most common questions end-users ask. This is an important consideration in this study as one of the research questions tries to establish how efficient it would be to install and configure repositories in cloud environments.

Like installation and consiguration, open source repository tools require someone with some degree of familiarity with software development to be able to customise them. Familiarity with HTML and XLST are some of the skills required for a successful basic customisation of DSpace, for instance. Customisation of repositories is desirable as it allows institutions to brand their repositories to adhere to their branding policies, while also improving the asthetics of the repository to enhance its appeal. Verno [16] documented Boston Biomedical Consultants, Inc's experience in implementing a DSpace instance. As much as the it was successful, the difficulties with the installation, configuration and customisation of DSpace were highlighted.

In summary, digital repository products come with the necessary repository features that institutions require out of the box. Installation, configuration and customisation may not be a trivial task to repository owners and administrators. The technological software stack of digital repository tools require technical users to provide reliable technical support. Therefore, it is desirable to have systems or mechanisms in place that allow repository users and managers to concentrate more on the management and curation of digital content, and less on the overall low-level management of digital repositories.

An appreciation of the addressed challenges in installing, configuring and customising repository tools will aid in understanding the design and implementation decisions that are discussed in the next chapter.

2.2 Cloud Computing

Cloud computing is a computing paradigm that allows for remotely located computing resources (e.g., applications, bandwidth, networks, servers, storage) to be provided as a service to consumers. The National Institute of Standards and Technology (NIST), defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [2]. Cloud computing's underlying resources are abstracted from end users (cloud consumers) and provisioned to them as services.

There are 3 service models commonly used to describe the nature of services provided in cloud environments namely, *Software as a Service (SaaS)*, *Platform as a Service (PaaS)* and *Infrastructure as a Service (IaaS)*. **Software as a Service** is a suite of applications that cloud consumers access using the Intranet or the Internet. This service model does not provide the cloud consumers with the capability to manage low levels resource like the amount of RAM, or the number of CPUs that their cloud based applications are using. **Platform as a Service** enables cloud consumers the capability to deploy custom made applications using cloud based libraries or application frameworks. Similar to SaaS, consumers of this service model have no access to the underlying low level cloud infrastructure. **Infrastructure as a Service** provides access to the basic computing resources to cloud consumers. Cloud consumers have control of the storage, computing power and other low level resources that are assigned to them. Refer to Figure 2.1 for an illustration of services that cloud consumers access by service model type [17].

There are 4 main modes of cloud computing deployments, which describe where the cloud infrastructure is hosted, who manages it and who are the cloud consumers. The 4 deployment modes are: Private Cloud, Public Cloud, Hybrid Cloud and Community Cloud. **Private Clouds** are hosted within an organisation's enterprise infrastructure or maybe hosted by a cloud provider solely on behalf of and for a given organisation's use and management. **Public Clouds** are managed and hosted for use by various organisations and individuals not necessarily affiliated with the cloud provider. This eliminates the need for upfront investments costs in resources by cloud consumers as that would be readily provided by the cloud providers. Recipients of the services provided in a Public Cloud are oblivious to the infrastructural make up of the cloud. **Community Clouds**

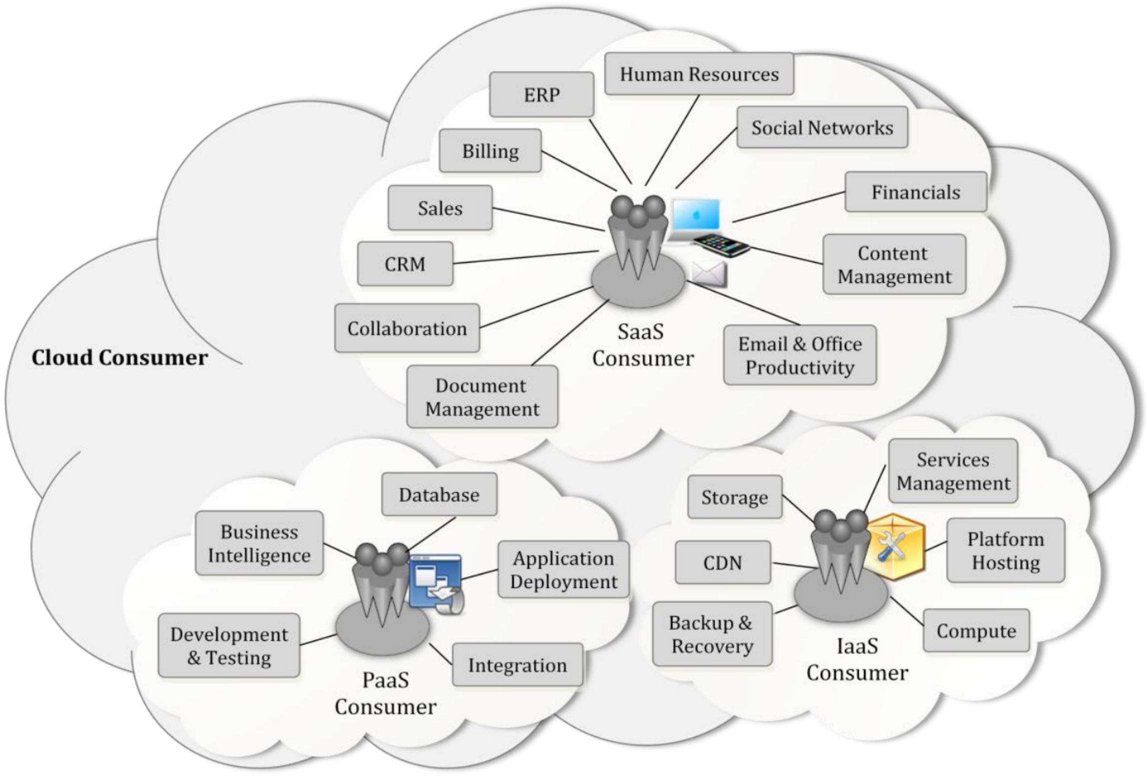


FIGURE 2.1: Example of Cloud Computing models and service provided [17]

are run and hosted for one or more organisations sharing similar goals or interests. The organisations are responsible for purchasing, maintenance and setup of the cloud environment. **Hybrid Clouds** are deployments that utilise both Public and Private clouds. These are useful for organisations that would like to build redundancy around their computing resources.

The National Institute of Standards and Technology provides detailed definitions [2] and a reference architecture [17] for Cloud computing.

2.3 Infrastructure as a Service (IaaS)

Infrastructure as a Service forms the basic layer of service delivery in cloud computing. Consumers/end-users are aware of their interactions with the lower level features of the computing infrastructure, they can decide which operating system to use, what and how applications can be installed, request to use more RAM, persistent storage, etc. All these features are provided to end users in a manner very different from traditional computing – end users have no need to directly interact with the physical machines, and can quickly change their preferences and effect them in a matter of seconds or minutes. However, to provide this abstraction, elasticity, machine orchestration, and on-demand features that cloud computing promises/provides, cloud computing software is used. They could be thought of as cloud computing operating systems. These software can be used to run either public clouds or private clouds. Examples of some of these cloud computing software platforms include: Amazon AWS [18], OpenStack [19], Eucalyptus [20], and OpenNebula [21].

Sharing of compute resources on servers to provide IaaS is achieved using virtualisation technology – virtual machines are created on the servers with user determined specifications. Cloud systems manage and monitor the creation of virtual machines using hypervisors [22, 23]. Hypervisors are software programs that enable operating systems to host one or more virtual machines. Examples of hypervisors include Xen [24], KVM [25], Nitro [26], and vSphere [27]. See Figure 2.2 for an illustration of how virtual machines share resources of a single server using a hypervisor.

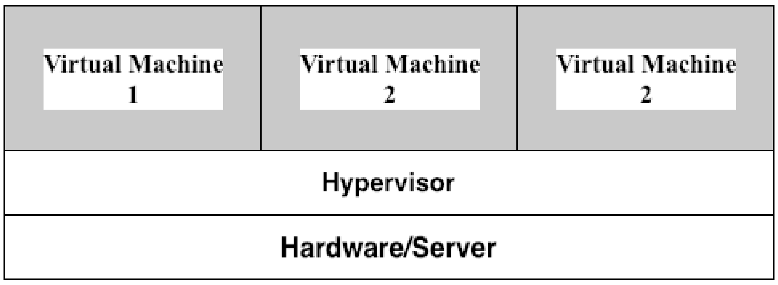


FIGURE 2.2: Virtual machines running off a single server using a hypervisor

Important to cloud systems is persistent data storage and internetworking of compute systems. Virtual machines as earlier discussed are the primary means to share resources for IaaS purposes. However, virtual machines by default have ephemeral storage devices – that is, they will lose their data when they are shutdown. Cloud Systems provide various modes to support persistent data storage as will be discussed in detail in the next sections.

In summary, IaaS is provided by Cloud systems that have at the very minimum 3 components that manage: 1) Compute resources 2) Data Storage and 3) Internetworking of virtual compute resources

To answer this study's research questions, Eucalyptus was used as the IaaS platform and the details are discussed in Chapter 3. In the following section, a description of Amazon AWS, Eucalyptus, and OpenStack cloud systems are given.

2.3.1 Amazon AWS

Amazon's AWS [18] is one of the world's popular, and one of the first prominent public cloud computing platforms. It is closed source, and as such not an option for developing private cloud implementations. However, it provides a set of services that other cloud computing tools aspire to incorporate in their distributions. It has a rich set of APIs to manage the services it provides and has a variety of software applications for end-user's varied needs either in machine management or general application management [28]. Its superior features and popularity has seen other cloud operating systems build APIs that enable interoperability with it [19, 20, 21].

The main services provided with Amazon's AWS are: Elastic Compute Cloud (EC2) [29], Elastic Block Storage (EBS) [30], and Simple Storage Service (S3) [31]. EC2 "is a Web service that provides secure, resizable compute capacity in the cloud" [29]. EC2 is responsible for the provision of IaaS. It runs its virtual machine instances off Xen or Nitro hypervisors. At creation, the virtual machine instances make use of the default ephemeral storage. To launch an instance, an Amazon Machine Image (AMI) is used. A Machine Image contains the basic software configuration to get the virtual machine into a usable state. The Machine Image can be created with an Operating System to the preference of the user. Amazon Web Services has pre-created AMIs that end users can choose from. To persist the data in the instance, EBS is used. EBS provides the choice between Solid State Drive volumes or Hard Disk Drives. End users can also determine the sizes of the volumes to attach to their instances. S3 provides simple storage of files and APIs are provided to deposit and retrieve data from it. All these features are accessed via a subscription service and can be brought up and running in a matter of minutes.

2.3.2 Eucalyptus

Eucalyptus is an open source cloud computing operating system that can be utilised for building Infrastructure as a Service environments both for private and public cloud deployments [20]. One of the reasons for Eucalyptus' development was to have a system open for experimentation [20]. Thus, its open-source nature made it one of the reasons why it was adopted by this study to be used as the IaaS platform. Eucalyptus was developed to have interfaces that use similar commands as Amazon's EC2.

Eucalyptus comprises 5 components that deliver a complete implementation of a cloud computing system. The components are: Cloud Controller (CLC), Cluster Controller (CC), Walrus Storage Controller (WS3), Storage Controller (SC) and Node Controllers

(NC) [32]. As alluded to earlier, IaaS delivery is based on virtualisation. The **Node Controller** is responsible for hosting and managing the lifecyle of virtual machines. Eucalyptus supports both KVM and Xen hypervisors. To launch a virtual machine instance, Eucalyptus uses a Eucalyptus Machine Image (EMI), which is analogus to Amazon's AMI discussed in the previous subsection. Users have the option of building their custom images or utilising existing ones provided on the Eucalyptus Image Store. As with Amazon's EC2, virtual machine instances will by default have an ephemeral storage device, thus once the virtual instance is terminated, all its data will be lost. This is where the **Storage Controller** comes in. It is responsible for providing block storage volumes to instances for data persistence beyond the lifecycle of a virtual machine. The Storage Controller is analogous to Amazons's EBS. Users can decide what sizes they want and attach the volumes at runtime. The **Cluster Controller** is responsible for managing one or more Node Controllers. The Cluster Controller decides which Node Controller should run instances and also monitors the state of the Node Controller, which it relays to the **Cloud Controller**. The Cloud Controller is the front-facing interface that end users interact with, either through command-line instructions or UI tools. The Cloud Controller provides overall administration of the cloud environment. All requests for virtual machines go through this component, which then decides based on the information it gathers about the environment where to deploy an instance. It is also responsible for management of the other components of Eucalyptus. **Walrus Storage Controller** provides a persistent simple storage service whose interface is compatible with Amazon's S3. This is responsible for storing Eucalyptus Machine Images, files and virtual machine instance snapshots.

In summary, Eucalyptus comes with a set of APIs that can be interfaced with Amazon's Web Service. This allows for creation of services that can be run on both Eucalyptus and Amazon's Web Service, and in turn enabling the creation of hybrid cloud environments. Eucalyptus was created as a research study within an academic institution but has since been acquired by HP.

2.3.3 OpenStack

OpenStack [19, 33] is an open source cloud computing software toolset that allows for the creation of private or public clouds. Openstack is defined as a system capable of managing a large pool of compute and storage resources while providing interfaces for users and administrators to control and provision those resources [34].

OpenStack has arguably one of the most complete and complex modular architectural designs of existing Open Source cloud computing operating systems, a characteristic that can be attributed to its large and active developer community [35]. Following in similar fashion to provide compute and storage services for IaaS delivery, OpenStack is comprised of at least 4 major components: OpenStack Compute Service (nova), OpenStack Block Storage Service (cinder), OpenStack Object Storage Service (swift), and OpenStack Dashboard (horizon).

OpenStack Compute Service is the equivalent of Amazon's EC2 and Eucalyptus' Node and Cluster Controllers. It is the primary OpenStack service that delivers compute services for IaaS delivery. It generally manages the lifecycle of virtual machine instances. The default hypervisor it supports is KVM. Other hypervisors supported by OpenStack include LXC, QEMU, Xen, etc [36]. The default storage device that the compute resource uses is ephemeral. The **OpenStack Block Storage Service** provides the service for data persistence for virtual machine instances. It is responsible for creating volumes and attaching them to instances. This is the equivalent of Amazon's EBS and Eucalyptus' Storage Controller. Similar to Amazon and Eucalyptus, OpenStack also provides a simple file storage system service called **OpenStack Object Storage Service**. This service provides an API that applications can use to store and read static data like images and music files. In addition to block storage and object store, OpenStack provides another form of data persistence called **Shared File System Storage**. Both the block storage and file-system storage can be mounted as drives in a given instance, however, the file system storage is created and managed only by a system administrator. Unlike the block storage and shared file-system storage, Object Store can be accessed via a Web service and access is not restricted to only within an instance. Management of cloud resources by administrators and other cloud users is done using multiple interfaces with the **OpenStack Dashboard** being the main such interface. OpenStack provides other management interfaces, which include OpenStack API, Secure shell (SSH), nova-manage, glance-manage, etc. [37].

The modular nature of OpenStack's architecture has enabled it to build a lot of services that enhance its overall functionality and make it one of the most feature rich cloud operating systems. Some of its other services not discussed here include Identify Service (keystone), Image Service (glance), and Data Processing Service (sahara).

As is the case with Eucalyptus, OpenStack has APIs that are compatible with some of Amazon's Web Services and as such can be used to build hybrid cloud environments with EC2.

2.3.4 Other Cloud Computing Platforms

There are a number of other cloud operating systems that are in use but were not considered for use for this study. Some of those cloud operating systems reviewed but not considered for use include: OpenNebula [21, 38], Nimbus [39, 40], and Apache CloudStack [41, 42, 43]. Each of these cloud systems possesses, to some extent, a similar architectural skeleton to the previously discussed platforms. They all utilise virtualisation technology, provide some form of persistent storage to virtual machines, provide APIs to interact with the compute and storage resources and provide abstractions to low level details of the provided infrastructure.

2.4 Configuration Management in Cloud Environments

The core objective of this study is to host digital repositories in private cloud environments efficiently. To realise that, installation and configuration of the digital repository tools have to be done in a systematic and standard manner. This can be achieved through use of configuration and management tools. These tools have been used in traditional computing environments to manage installations of similar software over several computers. This is useful as it would reduce significant man-hours spent in carrying out the work only with minor alterations for each iteration and virtual machine instance. In the same way, these tools can be used in cloud computing environments where administrators have remote access to their computing resources – depending on the size of the cloud environment, system administrator can potentially be overseeing a cloud computing environment that is capable of hosting dozens of compute and storage resources. The advent of cloud computing and the popularity of using these tools have actually resulted in new terminology to describe teams and methods of systems configuration and system deployment. Those terms are DevOps [44] and Infrastructure as Code (IaC) [45].

There are two main architectural designs for the available tools. Some operate with a server machine, and execute all commands from centrally located machines. Others adopt a client-server architecture, which requires the installation of an agent on the target nodes/machines. Other options take the route of bundling virtual machines with all the necessary software and required configurations, end users are only required to boot the right images and they will have all their needed applications running, with only minor configuration performed by the end users.

Below, a short description of some of the tools used for configuration and infrastructure management is given.

2.4.1 Wrangler

Wrangler is a software tool that automatically provisions and configures virtual clusters in cloud environments [46]. A virtual cluster in this case is a collection of virtual machine instances. Wrangler is used to configure the desired compute characteristics of those virtual machine instances. Wrangler defines the preferred final state/configurations of virtual resources in the cloud using XML. Wrangler is comprised of 3 components: Client, Coordinator and Agent. The Client is a command line interface that sends the details of how to install and configure the virtual resources. The Coordinator is responsible for retrieving information from virtual resources and also directing configuration details to actual virtual machines. It essentially serves an information broker. Agents execute the necessary configurations on the specific virtual machines. The Agent retrieves the configurations from the Coordinator from which it ascertains what instructions are to be executed on that virtual machine.

2.4.2 Ansible

Ansible is an open source automation, configuration management, cloud provisioning and system administration software tool [47]

Ansible works by pushing *small programs*, called Ansible Modules, to target remote nodes that are to be configured. Ansible Playbooks utilise Ansible Modules to execute advanced configurations and orchestration of remote resources. Ansible Playbook is a language that defines the desired state on target remote machine(s) and is expressed using a human and machine readable language, which is a variation of the YAML language.

Ansible does not require any dependency software or agents pre-installed (an exception here is Python, which in most cases will come pre-bundled with Operating Systems) on the target machines to carry out configurations or deployments.

2.4.3 Puppet

Puppet [48] is another well known, and one of the earliest configuration management tools. It has both open source and enterprise distributions. Puppet functions in a client/server style, with the server managing the configurations on the nodes, and the client carrying out the actual configurations on the target nodes. The server runs what is called the Puppet master, and the client runs the Puppet client. The desired state of a target remote machine is done using what are called Puppet Manifest files, which make use of Puppet's declarative language. The declarative languages requires some programming experience to use. The enterprise version comes with a GUI that gives a status of each of the target nodes being configured or deployed.

Puppet is arguably the most feature rich configuration tool with a wider enterprise and open source community.

2.4.4 Chef

Like Puppet, Chef [49] is another older, well known configuration management and automation tool. It has both a client/server capability, and also what it calls chef-solo, which can function without having an agent running on the target nodes. Users define the state of the target machine using what is called a *Recipe*, which can be combined together into what is called a *Cookbook*. The Chef Server is a repository of cookbooks and information of the target machines/nodes. The Chef Client runs on the target machines, and retrives the latest configuration instructions, which it executes on the target machines to realise the prescribed and desired state.

Like Puppet, Chef can be used to manage and deploy machines at enterprise scale.

2.4.5 Fabric

Fabric is a Python library and command-line tool for application deployment, configuration management and executing systems administration tasks [50]. Fabric commands are run from a central location to execute commands on remote machines. There is no requirement to install Fabric or Python on the target machine where configurations or system administration tasks are to be executed. Fabric can be used with other configuration management tools to carry out far more complex deployments. It is an open source product and thus free to use. To carry out configurations, a Python script, called fabfile, is written that has definitions of what and how to carry out the required configurations, or management tasks.

2.5 IaaS and Containers

In the previous sections, cloud computing has been discussed from a virtual machine stand-point. This is in part due to the basic setup of cloud computing platforms for delivering IaaS. The related provisioning and orchestration of these virtual machine were also discussed. However, recent trends in cloud computing have seen a rise in the adoption of container-based virtualisation which is generally called as Containerisation[2] [51].

There is, as of this writing, no standard definition of containerisation or containers. The setting up of the Open Container Initiative (OCI)[3] established in 2015 may lead to the overall standardisation of terminologies within the containerisation technology ecosystem. Various vendors have different working definitions of containers, with the underlying theme being that containers are a form of virtualisation that abstract resources at Operating System level [52, 53, 54, 55]. Containers are run within a host operating system and share the same resources as that host's Operating System. This is in contrast to the use of virtual machines that require running a guest Operating System on top of a host Operating System. This makes containers lightweight in nature. Note, however, that containers can be run within virtual machines and that some cloud providers now offer a combination of both container based virtualisation and hypevisor virtualisation [56, 57, 58].

The recent rise in the adoption of using containers can arguably be attributed to the advent of Docker Engine [59, 60], a containerisation platform. Docker Engine is comprised of 3 components namely Server, REST API and Command Line Interface (Docker CLI) client. The Docker Server is responsible for running docker containers and managing other docker objects that include images, storage volumes and networks. The Docker CLI client is the interface that docker users utilise to interact with the Docker Server. This communication between between the Docker Client and the Server is via the Docker REST API. To manage containers across servers and compute clusters, container orchestration systems are utilised, some of which include Kubernetes [61] and Docker Swarm [62]. Besides Docker, other containerisation platforms include LXC [63], rtk [64] and OpenVz [65].

In the context of this study, the relevance of containerisation technology is that repository tools could be pre-packaged in containers with all their dependencies and be portable across various cloud platforms. For instance, a container could be packaged with a DSpace repository tool including all its dependencies. As containers are very comparable to how virtual machines would be managed in cloud environments, the methods to simplify the installation and management of repository tools using containers would still follow the same approach as using virtual machines.

For this study, container technology for either packaging a repository tool, or provisioning computing instances was not considered. The focus is on the primary delivery of IaaS

[2]DataDog, `https://www.datadoghq.com/docker-adoption/`

[3]Open Container Initiative, `opencontainers.org/about`

through virtual machines and the simplification of installing and managing repository tools in private cloud environments.

2.6 Repositories in Clouds

This section gives an account of some of the related work on hosting institutional repositories in cloud environments. Note that to our knowledge, there is little published work in the area of hosting institutional repositories in private cloud computing environments. The following accounts are the closest, and relevant related work.

Wu et al described their work in migrating a digital repository called CiteSeerX into a private cloud environment [66]. Long terms costs, compared to migrating to a public cloud, were one of the reasons for their motivation to setup a private cloud infrastructure to host their digital repository. They labelled CiteSeerX as a medium sized digital library in comparison to digital libraries like Google Scholar and Microsoft Academic Search. Their setup utilises proprietary software – VMware ESXi – for the hypervisor and VMware VSphere for instance provisioning and general cloud orchestration and monitoring. They give a detailed account of the life cycle of a digital library, challenges faced by their growing content and mechanisms for handling fault tolerance presented in a cloud infrastructure. However, as much as they give a good account for their work, their work is very specific to migrating CiteSeerX to a private cloud environment. They do not provide a systematic and automated solution to effectively and efficiently host digital libraries in clouds. Their process requires high-level expertise, which may not be easily replicated when hosting institutional digital repositories, like DSpace and EPrints, in private cloud environments. Aljenaa et al have explored the possibility of hosting elearning systems in cloud based environments [67]. Their evaluation leads them to recommend hosting their systems in a private cloud environment. They discuss the on-demand and elasticity features of cloud computing environments as a major reason to recommend the use of cloud computing. They provide potential principles to be adopted when running applications in cloud environments and have no implementation to evaluate their suggested solutions.

The Texas Digital Library described their efforts in first replicating their in-house computing infrastructure onto EC2, then completely migrating all their digital library services to Amazon's EC2 [68]. In the cloud, their services are provided on 48 virtual machine instances. Overall, they describe their move to the public cloud environment as a positive one. Their work demonstrates the successful implementation of a repository on virtualised infrastructure, which does not necessarily speak directly to hosting and migrating repositories to private clouds. However, due to the similarities in the infrastructure setup of EC2 and Eucalyptus, it gives an insight into potential challenges in setting up repositories in private clouds. Thakar et al [69] described their experience migrating the Sloan Digital Sky Survey science archive, which has a database of 5 TB. They describe their experience as frustrating based on two reasons: degraded performance of the queries run off the database compared to their inhouse infrastructure; and their inability to transfer the entire 5 TB to the cloud. The discussed works looked at public clouds and also the performance of cloud systems. Others have looked into content preservation across cloud environments. DuraCloud [70] for instance, utilises different public cloud storage options to replicate content, providing the needed redundancy and potential data preservation. Digital repository tools can store content to DuraSpace through the different interfaces that DuraCloud provides. Kindura [71] is another project that is encouraged by the possibilities of content preservation using cloud storage systems. Both DuraCloud and Kindura do not explore the use of private cloud infrastructure and simplifying the process of installing and managing digital repository tools. Both are driven by long term content preservation needs, which repository tools can plug into to push and retrieve data.

Doelitzscher et al describe their work in setting up a private cloud managed by their inhouse developed Cloud Infrastructure and Application (CloudIA) cloud system [72]. They describe the different components built into it that support IaaS, PaaS and SaaS. They go into great detail describing the functionality of CloudIA and how to access its supported e-learning applications. However, this work does not address the work of hosting institutional repository tools. However, provides an extensive description of the automation tasks in cloud environments.

None of the discussed related work proposes a method to run and manage institutional repositories in private cloud environments. They do, however, give a good account of working in cloud computing environments. The features for on-demand computing power provisioning and elasticity of resources are indeed attractive. This study builds on these features and proposes a solution that allows for a single point for system administrators to install, manage and monitor institutional repositories in private cloud environments.

2.7 Summary

A brief background was provided on digital repositories and cloud computing. A thorough discussion on providing IaaS was given, listing the components that make up cloud computing operating systems. Tools necessary for automating and managing repository tools in cloud environments were also discussed. The information provided was to illustrate the details and underlying architecture of cloud environments and digital repositories. Previous work in running repository tools, and generally electronic software systems, in cloud environments were discussed. From the discussed literature, there has been more effort in hosting repositories in Public cloud environments than in Private ones. The work in Private cloud environments did not utilise existing cloud operating systems, but rather made use of virtualisation software to build a semblance of an out-of-the-box cloud system. This dissertation's objective is to build a tool to enable installation of an institutional repository tool in an open source cloud operating system efficiently. Therefore, the objective of this dissertation is not covered in previous works. The next chapter describes the motivation and architectural design of the proposed solution. The chapter gives a detailed account of Eucalyptus and the design of the proposed tool, Lilu.

Chapter 3

Design and Implementation

This chapter describes the steps taken to host a digital repository in a private cloud environment, and a Web application called *Lilu*, that was developed to allow repository system administrators and users to install and manage a repository.

The following section gives the overall objective for the solution to be developed followed by a brief description of the implementation. What follows is a brief background for each technology used in the proposed solution, and the rationale for the choice of the named technology. A high level overview and workflow of the proposed solution is then described. The chapter ends with a discussion on the challenges encountered in developing the proposed solution.

3.1 Implementation Objectives

There are two major objectives for this project:

1. Host a digital repository in a cloud environment.
2. Develop a tool to install and manage digital repositories in clouds and their associated compute resources.

Achieving these objectives helps to answer the research questions that were framed in Chapter 1.

3.2 Implementation Overview

A private cloud computing environment was setup using Eucalyptus, an Infrastructure as a Service (IaaS) software tool. Eucalyptus came bundled with the Ubuntu operating system and went by the alias of Ubuntu Enterprise Cloud (UEC). Version 10.10 of UEC was used for this study. The institutional digital repository used was DSpace using version 1.8. Installation and configuration of DSpace was achieved through use of an orchestration and configuration management Python API called Fabric. *Lilu* was developed using the Django Web framework [73]. Django Celery [74] formed a core part of the overall solution: it enabled execution of tasks (e.g. installation and customisation of repositories) in an asynchronous manner.

3.3 Design Rationale and Implementation Approach

The soluton to be developed builds on the principles/characteristic of cloud computing. In fact, the solution leverages the aspects of cloud computing that make it very desirable – **elasticity and on demand resource/service provisioning**.

With that in mind, an Infrastructure as a Service platform is adopted to manage all cloud computing resources. The management of many computing resources is transparent to the overall system administrator. This will allow system administrators to add and remove resources seamlessly without having to concern themselves with the low-level details of managing and assigning virtual machines when requested for. Use of the Infrastructure as a Service tool helps to add the elasticity and on-demand provisioning of resources features to the overall proposed solution.

Automation and resource orchestration are central to cloud computing. For the proposed solution, Fabric, a Python implemented API is used to orchestrate and automate some system functions.

Ordinary systems users need to be abstracted from all low level components that comprise the solution proposed. For these users, the tasks of installing and configuring repository tools are achieved through mere clicks of a button.

3.3.1 Infrastructure as a Service

Eucalyptus was the choice for this project. Its ease of installation, as it was bundled with some distributions of Ubuntu, made this choice easy. Its active community and also detailed readily available documentation were other reasons for going with Eucalyptus.

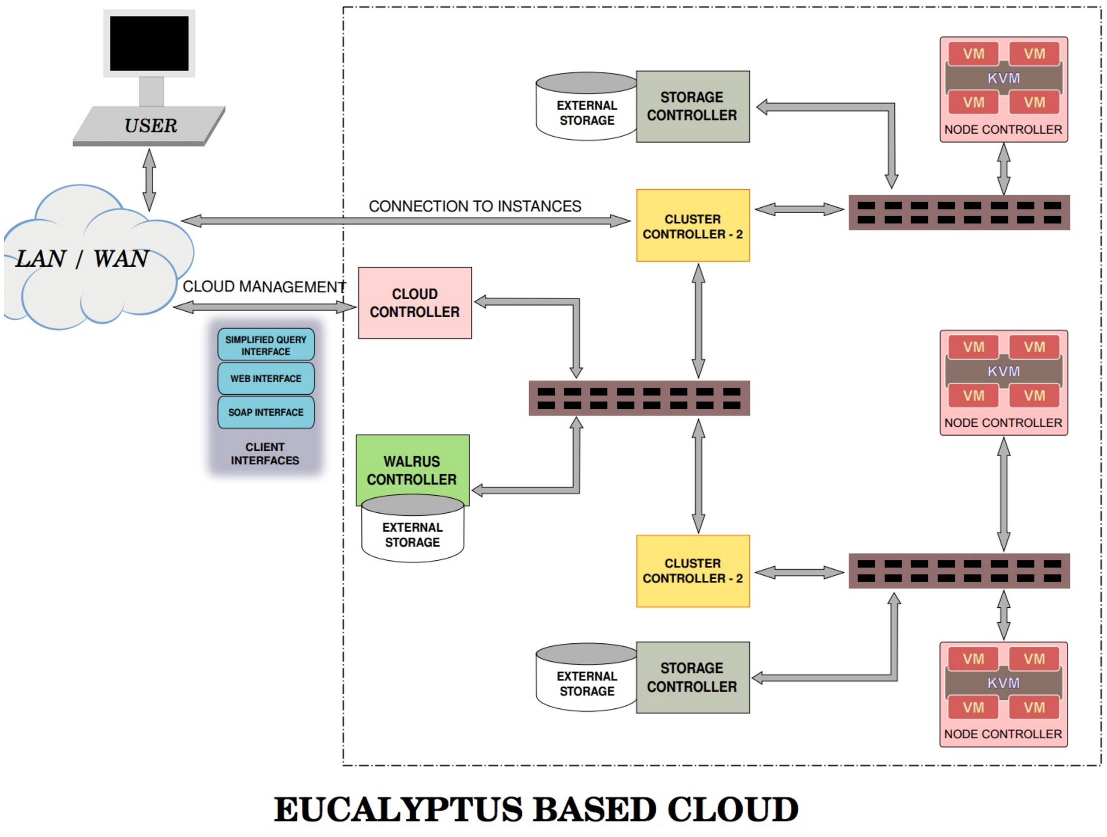


FIGURE 3.1: Eucalyptus cloud components [1]

To fully grasp the solution proposed, it is important to understand the different components of the cloud infrastructure, and how they relate to each other. The different components were discussed at length in Chapter 2. Figure 3.1 shows the components in a Eucalyptus Cloud.

Based on what has been discussed so far, the following are important characteristics of the cloud environment that have to be taken into account in the solution developed:

1. Virtual machines have ephemeral storage – the life cycle of a running instance ends once the virtual machine has been shutdown. Any data saved in the instance will be lost, unless it was being saved on persistent storage managed by the Storage Controller. When booting, the instance will have an ephemeral storage disk that is predefined upfront.

2. Persistent Storage – persistent storage is provided by the Storage Controller. Varying sizes of storage volumes can be created, which can then be attached to running instances. When an instance has been shutdown, any data saved on the volume will not be lost.

[1]Source, `https://cssoss.files.wordpress.com/2010/12/eucabookv2-0.pdf`

3. Backing up of Persistent Storage – copies of the persistent storage can be made and then be saved by the Walrus Storage Controller (WS3).

Any solution that would be developed would have to take advantage of the ephemeral nature of the virtual machines in the cloud and that persistence of data is through the use of the volumes provided by the Storage Controller.

3.3.2 DSpace Digital Repository Toolkit

DSpace is an open source digital repository toolkit that is used by a lot of institutions across the world[2] – its wide usage is the reason why it was chosen for prototyping for the proposed solution. It is developed in the Java programming language, and uses a PostgreSQL database.

Installation of DSpace is not a trivial task [13], which can take a significant amount of time to successfully complete [16]. An efficient installation approach would have to be adopted that abstracts a lot of the steps that users would take to complete a DSpace installation.

In summary, the steps to install DSpace are as follows:

1. Install PostgreSQL server
2. Install Tomcat Java container
3. Install Maven
4. Configure PostgreSQL server; create database for DSpace
5. Configure Tomcat
6. Download, compile and build DSpace files

In this study, all these steps will be transparent to the user of the system when installations are carried out.

3.3.3 Configuration Management and Automation

For this study, Fabric was used because of its simplicity and the programming language it supports – Python. Eucalyptus has Python administrative tools and APIs [32].

[2]DSpace, `http://registry.duraspace.org/registry/dspace`

This means that a seamless solution can be provided using Fabric and Eucalyptus for provisioning of cloud based services.

Fabric allows for remote execution of tasks such installation of software and also configuration of that software.

3.3.4 UI Front-end

The Django Web framework was used to develop the user interface frontend. The functions to be provided to a user on the frontend are: DSpace installation, starting and shutting down of instances, and the customisation of DSpace. These tasks take slightly over a minute to complete. Django Celery is used in the backend to allow for the execution of these functions in an asynchronous manner. That way, a user can perform other tasks provided to them.

The Django Web framework and Django Celery both support Python and thus help simplify their integration with Eucalyptus.

3.4 System Description and Data Flow

Figure 3.2 is a high level depiction of the steps that follow once a request to install DSpace has been made.

For the purposes of the discussion that follows, a DSpace instance is a virtual machine instance in the cloud environment with a completed installation of DSpace. The DSpace instance may also be referred to as a repository. Virtual machine instance and instance are used interchangeably.

3.4.1 User Front End

The process flow depicted in Figure 3.2 begins with a user logging onto the prototype application that was developed called *Lilu*, using a username and password. Figure 3.3 shows what is presented to a user or system administrator once they have logged in. Figure 3.4 shows the functions that can be carried out on existing installations. At this point, the user or system administrator are able to carry out a DSpace installation or manage already existing repositories.

Installation of a new DSpace instance starts with providing information that should be associated with a given repository – this is repository-identifying information and system

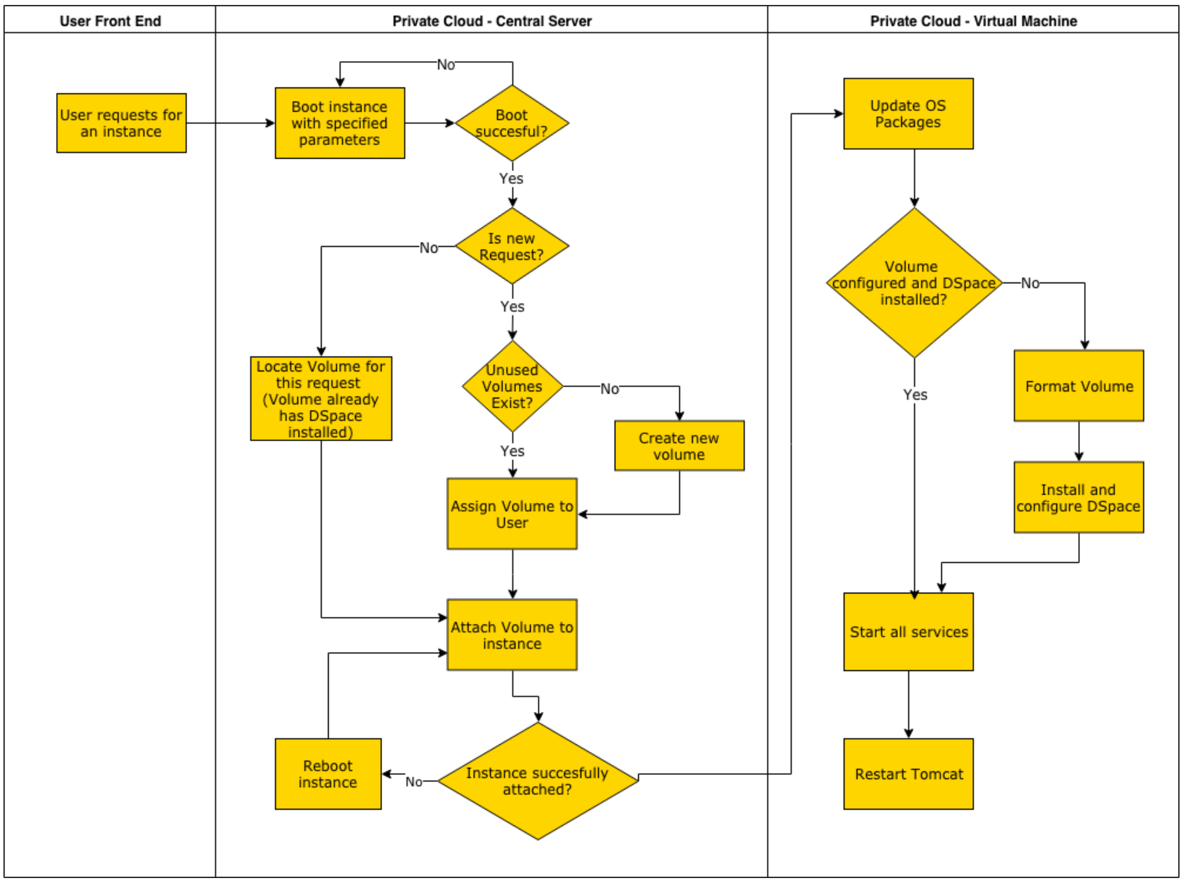
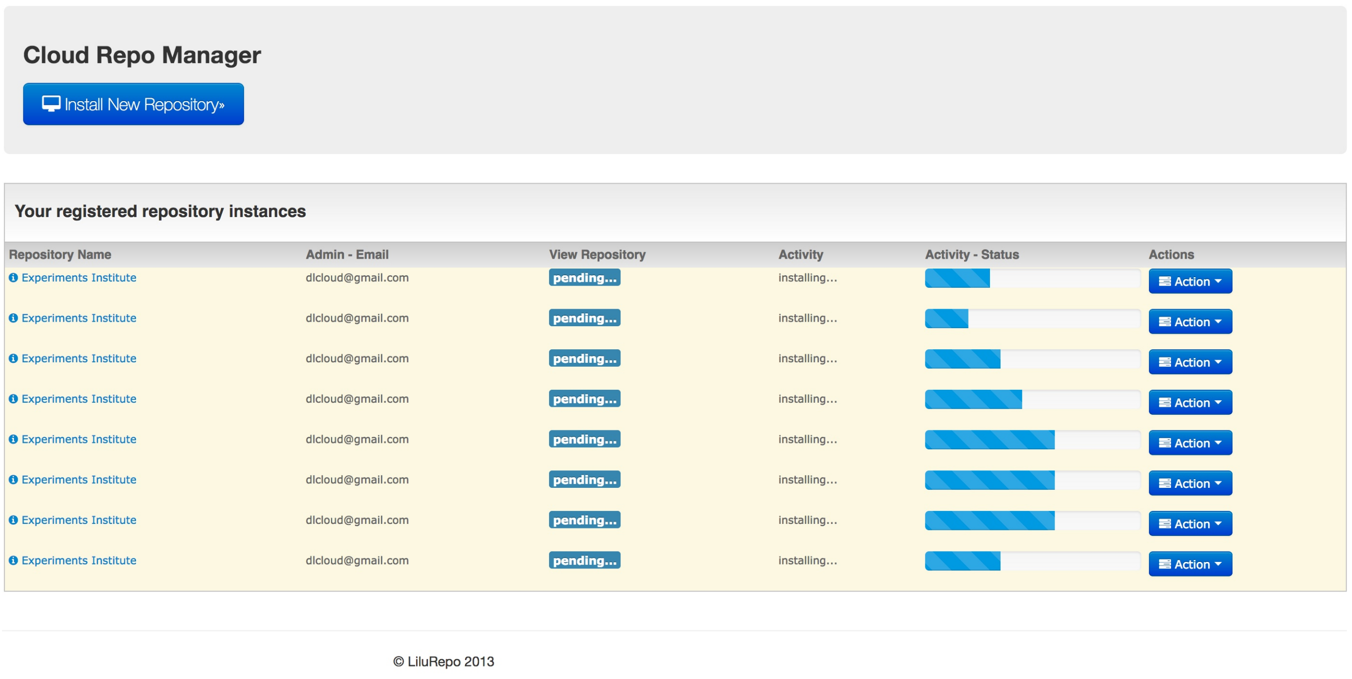

User Front End
Private Cloud - Central Server
Private Cloud - Virtual Machine
User requests for an instance
Boot instance with specified parameters
Boot succesful?
No
Yes
Is new Request?
No
Yes
Locate Volume for this request (Volume already has DSpace installed)
Unused Volumes Exist?
No
Yes
Create new volume
Assign Volume to User
Attach Volume to instance
Reboot instance
No
Instance succesfully attached?
Update OS Packages
Volume configured and DSpace installed?
No
Yes
Format Volume
Install and configure DSpace
Start all services
Restart Tomcat

FIGURE 3.2: System implementation overview



Cloud Repo Manager

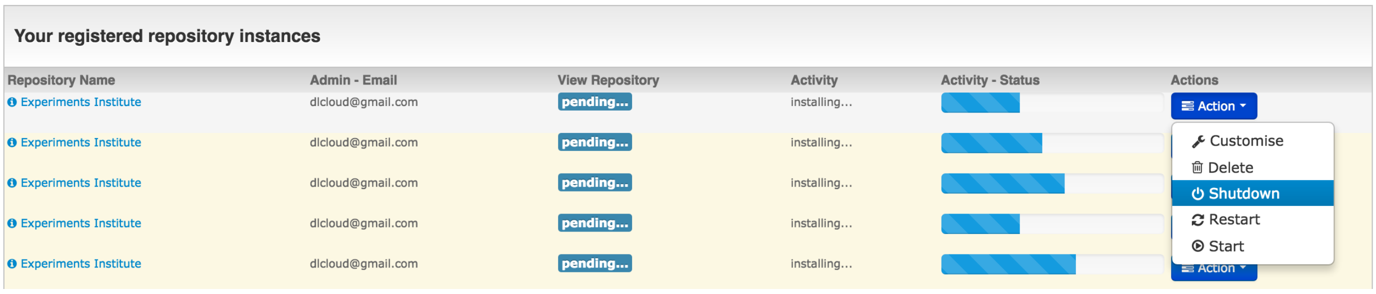
Install New Repository»

Your registered repository instances

Repository Name	Admin - Email	View Repository	Activity	Activity - Status	Actions
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action

© LiluRepo 2013

FIGURE 3.3: Lilu initial landing page

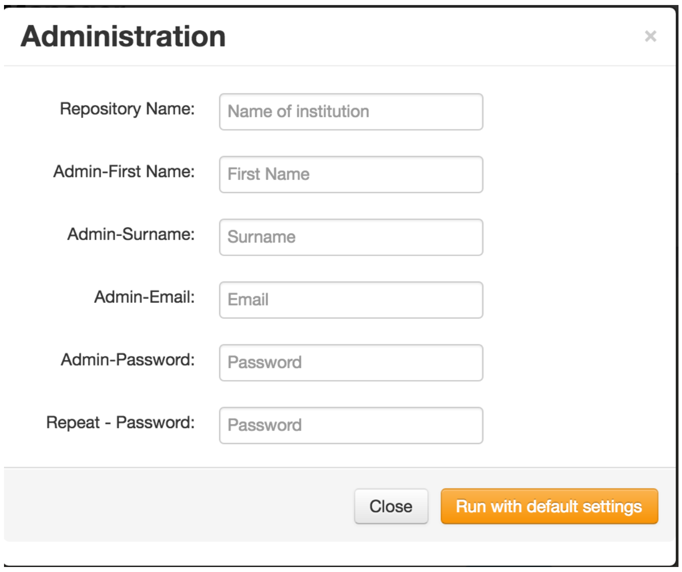


Your registered repository instances

Repository Name	Admin - Email	View Repository	Activity	Activity - Status	Actions
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		
Experiments Institute	dlcloud@gmail.com	pending...	installing...		
Experiments Institute	dlcloud@gmail.com	pending...	installing...		
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action

FIGURE 3.4: Features on a repository once it is installing or it has fully installed

administrator credentials. The needed information is provided via the interface depicted in Figure 3.5. Installation of the repository takes over 10 minutes to complete. However, the browser does not block until the installation has completed. Control is returned to the user, who can continue to perform other tasks provided by the prototype. Progress status of the installation is given – a successfully completed installation will have a green tick and its progress status information is updated accordingly.



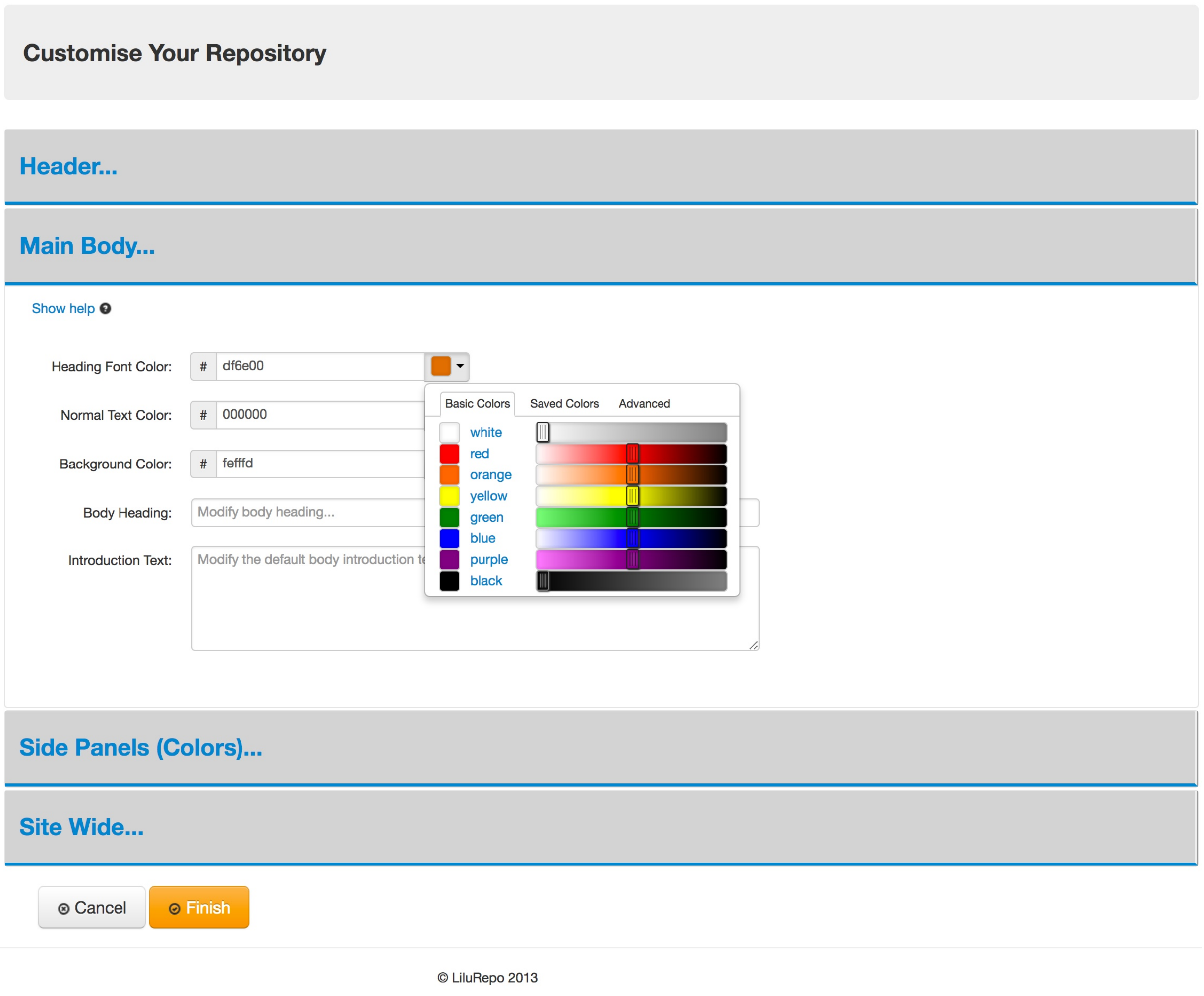
Administration

Repository Name:	Name of institution
Admin-First Name:	First Name
Admin-Surname:	Surname
Admin-Email:	Email
Admin-Password:	Password
Repeat - Password:	Password

FIGURE 3.5: Information required to create a new repository

While the installation is progressing or has completed successfully, basic customisation can be performed on the given repository or any repository associated with the currently logged in user. Figure 3.4 shows how to access the customisation function. The customisation feature allows for making minor modifications to the repository's overall look and feel. Positioning of the repository's logo can be switched between left and right, custom images for logos can be used, colors on the site can be changed, and the name of the repository and text on the body of the repository can be adjusted. Figure 3.6 shows the customisation page, showing the different parts of the repository that can be customised.

Other features available include shutting down the repository, restarting the repository and completely deleting the repository.


Customise Your Repository

Header...

Main Body...

Show help

Heading Font Color:	# df6e00
Normal Text Color:	# 000000
Background Color:	# fefffd
Body Heading:	Modify body heading...
Introduction Text:	Modify the default body introduction t

FIGURE 3.6: Repository customisation page

As earlier pointed out, the frontend's functionality is made possible through use of Django Web framework. Bootstrap[3], an HTML and CSS framework, is used to develop rich Web controls. Django Celery was used to enable asynchronous execution of tasks, which is the reason control is returned back to the user after requesting to install or customise a repository.

3.4.2 Backend

The application on the backend manages the workflows that are needed to complete the requests made from the frontend. It does so by interacting with Eucalyptus cloud's APIs via *euca2ools*[4] [32]. euca2ools provides an abstraction and a set of programmable

[3]Bootstrap, `http://getbootstrap.com/`
[4]euca2ools, `https://wiki.debian.org/euca2ools`

functions that enable administrative management of Eucalyptus cloud services. Some of the euca2ool functions utilised in this proposed solution are described below:

1. **Start (run) virtual machine instances** – this is achieved by executing the *euca-run-instance* from the command line or *run_instance* python function of the euca2ools API. This function essentially boots the instance, bringing to life the virtual machine instance. Depending on the image used to run the instance, the virtual machine may or may not have an operating system in it. For this study, the image that was used had the Ubuntu operating system. Each instance that is run will have a system generated ID by Eucalyptus.

2. **Terminate virtual machine instances** – the command line function associated with this function is *euca-terminate-instances*, with the equivalent python API call being *terminate_instances*. It is the equivalent of shutting down and powering off a computer. All computing resources i.e. RAM, Volumes, CPU etc., will be freed up and ready for use by another virtual machine instance.

3. **Reboot virtual machine instances** – rebooting a virtual machine mantains all the instance's information and its connected peripheral devices. The command for this function is *euca-reboot-instances*. This call is used when the virtual instance needs to maintain its state of its connected peripheral devices and all other data saved on its ephemeral volume.

4. **Create and delete block storage volumes** – block storage volumes are what are used to persist data for virtual machine instances. The call to create volumes is *euca_create_volume* while the one used to delete the volume is *euca_delete_volume*. A given volume can only be deleted when it is not attached to a running virtual machine instance.

5. **Attach block storage volumes to instances** – this will attach the created block storage volume to a running instance. The euca2ool function called to attach volume(s) is *euca-attach-volume*. This function only ends at attaching the volume to the running instance. For the volume to be usable, it would need to be mounted by the operating system of the virtual machine and also formatted – formatting of the volume is only done once. Subsequent attachment of the same volume to instances requires no formatting unless it is the user's preference to do so.

6. **Check the status of virtual machines and volumes** – *euca_describe_volumes* and *euca_describe_instances* will check the status of virtual machine instances and available volumes in the cloud, respectively. Virtual machines will be in either of 2 states: running or terminating. Once terminated, the virtual machine ceases to

exist. Volumes too will be in one of 3 states: deleting, available, or attached. The available state indicates that the volume is ready for use by an instance.

7. **Monitoring of the clouds' resource utilisation** – *euca-describe-availability-zones* is one of the important functions that helps ascertain the amount of resources that are in use against the capacity of the cloud environment. The function can be used to determine whether the cloud still has enough resources to run another virtual machine instance.

The application builds on these functions to provide the features available on the user frontend. The backend application will receive the requests, namely 1) *install DSpace*, 2) *customise DSpace*, 3) *shutdown DSpace*, and 4) *start DSpace*.

Install DSpace (install new repository) is the primary function in the developed application prototype. When the request to install DSpace is received, the backend application goes through the following phases (this is as depicted in Figure 3.2):

1. Via the cloud's API, boot a new virtual machine instance.
2. If the instance booted successfully, the application will check if there is any available free block storage volume. If not, a new volume will be created. The successfully booted instance will have a private IP address.
3. The volume in step 2 will then be attached to the booted instance. The application will generate a unique identifier for this volume. This identifier will also be associated with the user requesting this installation.
4. The attached volume will be formatted and prepared for use in the virtual machine instance.
5. Using Fabric, DSpace and its dependency libraries will be installed on the virtual machine instance.
6. Configuration will be done for the DSpace PostgreSQL database and the location where the DSpace bitstreams should be saved. All data that should be persisted will be saved on the attached volume.
7. Restart Tomcat server on the virtual machine instance.
8. Assign this instance a URL and register it on the primary front-facing Apache Webserver. This URL will be used to access the DSpace instance in the cloud. The URL will be mapped to the instance's private IP address in the cloud environment. Note that the DSpace instance in the cloud is using Apache Tomcat as

its Webserver and Java Container. The re-routing of external calls via the front-facing Apache server to the Tomcat server is achieved by using an Apache module called *mod_alias*[5] – this is installed on the front-facing Apache server, where the publicly accessible URL is mapped to the internal private IP address for the virtual machine instance.

Customise DSpase. When the backend receives this request to customise a given repository, it keeps a copy of the customisations to be made before publishing them to the DSpace instance. The customisation process entails overwriting files on the target DSpace instance. Overwriting of files is achieved through rysnc[6] calls. Using Django Celery, the user can perform the customisation while the DSpace instance installation is ongoing.

Shutdown DSpace. When this request is received on the server, the application will detach the volume from the instance before terminating the virtual machine instance. Detaching of the volume is only carried out once the Postgres database and Webservers' (i.e. Tomcat and Apache) services have been stopped successfully. Termination of the virtual machine instance is via calls to the Eucalyptus API *terminate-instance.* This API call results in the equivalent process of shutting down the machine's operating system and powering off the virtual machine. Termination of an instance frees up the compute resources that can be used for other purposes. Note that this is transparent to the frontend user, who will have the perception of a traditional computer shutdown.

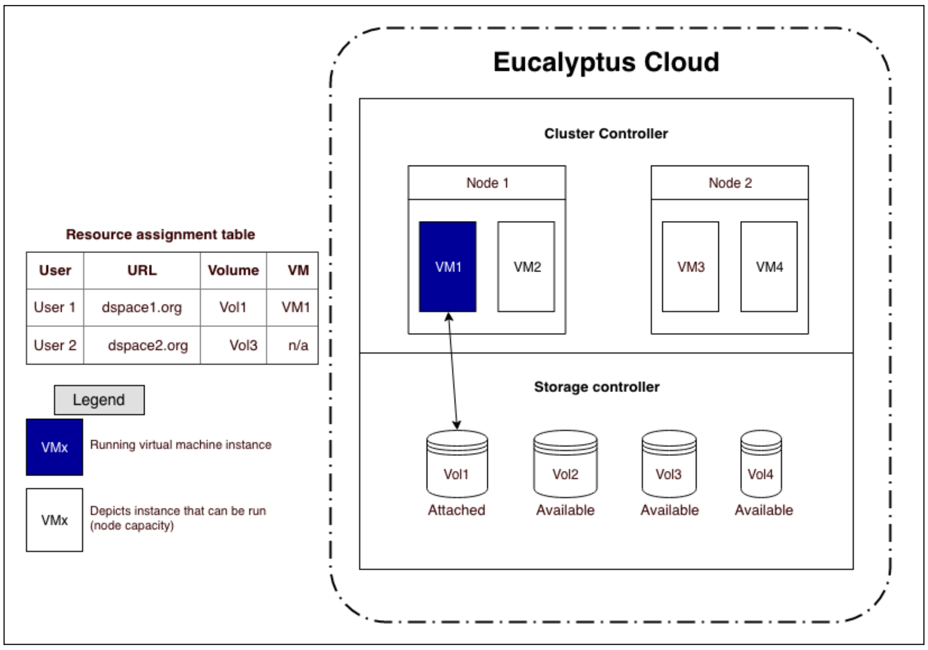
Start DSpace (or restart DSpace). This action can only be executed in the event that the DSpace instance was previously shutdown. Starting a DSpace instance follows the same process as installing a new DSpace instance. The difference is that there is no DSpace installation and configuration required. A start DSpace task reattaches the volume that was detached during the shutdown. This volume will still have all the information about the DSpace instance before it was shutdown.

3.5 Implementation notes

Users of the system own volumes and not virtual machines. The application will store all the identifying information of each virtual machine instance together with its associated DSpace installation. Instances can be created and terminated at will, but the DSpace instance's data and all access information will remain intact. The importance of this feature is that compute resources are freed when the instance has been shutdown and

[5] mod_alias, `https://httpd.apache.org/docs/2.4/mod/mod_alias.html`
[6] rsync, `https://linux.die.net/man/1/rsync`

thus available for use by other cloud users, or for other purposes. For an example, refer to Figure 3.7 and Figure 3.8. Both figures show a cloud environment that has a capacity to host a total of 4 virtual machines, and has 2 registered users. In Figure 3.7, one instance – VM1 – is running belonging to User 1. User 2 has a volume assigned to them but has their DSpace instance shutdown. Figure 3.8 shows that User 1 had shutdown their instance, and on restarting they were assigned a different virtual machine – VM3. When User 2 decided to run their instance, they were assigned virtual machine VM1, which had previously been assigned to User 1. The assignment of the virtual machines is transparent to the users.

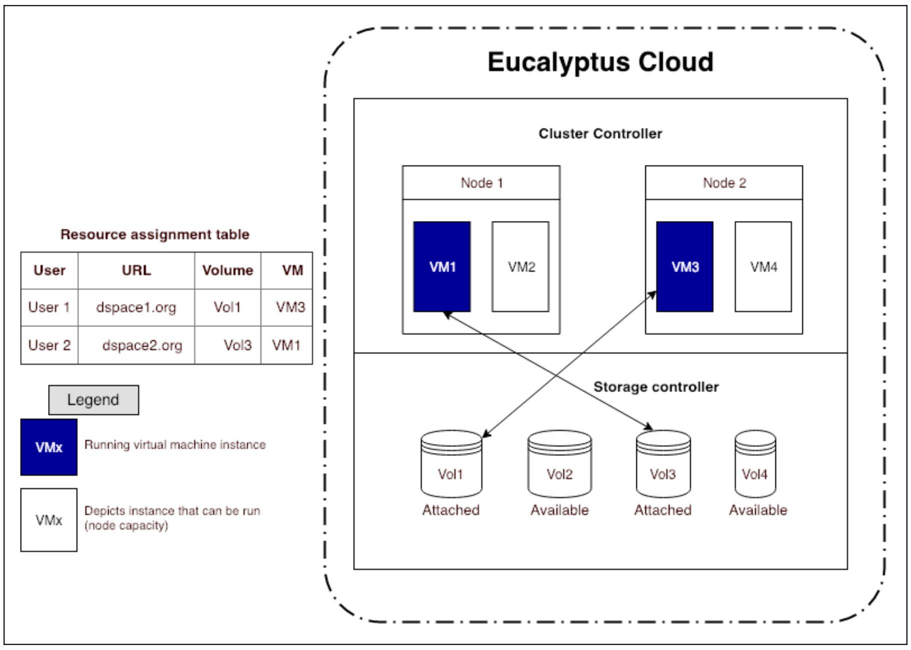


User	URL	Volume	VM
User 1	dspace1.org	Vol1	VM1
User 2	dspace2.org	Vol3	n/a

FIGURE 3.7: Eucalyptus cloud state with only one virtual machine running

The whole process of booting a virtual machine instance in the Eucalyptus cloud setup during this implementation to completing the installation of DSpace takes, on average, about 12 minutes. This is a relatively long time from a user's perspective. There are options that can be explored to reduce this time significantly. One way would be to pre-create instances and also pre-create volumes so that, at the time they are requested, only configurations of DSpace will be performed.

This suggested solution removes the burden on the system administrator to manage the hardware for each instance that is running. The system administrator can instead concentrate on only pre-creating configurations for the repositories – an activity that needs to only be carried out once. However, the running of unused resources, those that were pre-created, would be a waste of resources - that would also result in unnecessary performance hits on the resources that are actually in use.



Resource assignment table

User	URL	Volume	VM
User 1	dspace1.org	Vol1	VM3
User 2	dspace2.org	Vol3	VM1

FIGURE 3.8: Eucalyptus cloud state with two virtual machines running

3.6 Summary

This chapter provided a description of the solution proposed to answer the research questions on the possibility of hosting digital repositories in private cloud environments.

A private cloud environment was setup using Eucalyptus. A Web based tool was then developed to automate the installation of DSpace in the private cloud environment. The Web based tool did not only provide a feature to install DSpace, but to customise, shutdown and restart DSpace instances.

The following chapter will evaluate this proposed solution in order to answer the study's research questions.

Chapter 4

Evaluation and Results

Eucalyptus was used as an Infrastructure as a Service platform to setup a private cloud environment. Deployment of instances and installation of DSpace in them was done. In addition, a prototype management application tool was developed. This chapter discusses the steps taken to evaluate the use of DSpace in a private cloud environment and also the tool developed to install and manage DSpace. The evaluation aims to answer the following research questions:

1. Is it possible to host digital repositories in Private Cloud environments efficiently?
2. How can common repository tools like DSpace be adapted for the cloud for easy management and deployment?

Two different types of evaluations are carried out: a performance experiment to ascertain the efficiency of the deployed DSpace instance in the cloud; and a usability study to answer the question on the management of repositories in a private cloud environment.

4.1 Performance Experiment

This experiment will answer the research question on whether it is possible to host repositories in private clouds efficiently and effectively. The tests measure the response times for ingesting and viewing repository items. Response time is the length of time it takes to complete a task, tasks here being: 1) item ingestion; and 2) viewing and downloading an item.

Another metric that was considered for measurement is instance install time. This metric was to ascertain how long it takes from requesting an instance for DSpace installation until it is ready for use. However, this is covered in the usability study where wait time contributes to the participants' satisfaction with the installation task. The performance experiment did not measure instance install time.

4.1.1 Experiment Setup

To accomplish this test, a cloud environment was setup. Starting and stopping of computing virtual machines and installation of DSpace were automated. The cloud environment could hold a total of 12 virtual machines. Figure 4.1 depicts the overall infrastructure of the cloud environment with all its components.

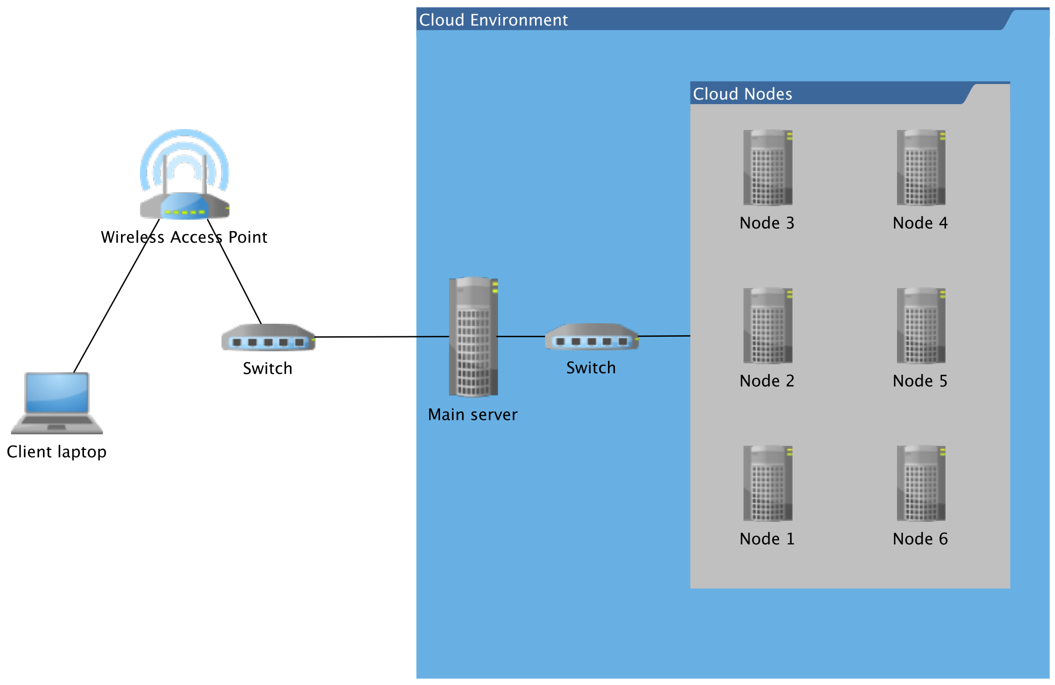


FIGURE 4.1: Diagrammatic representation of the private cloud environment for running experiments

Each of the components in Figure 4.1 plays a specific role. Below is a brief description of each component:

- **Main Server** – This hosts the core components of Eucalyptus. The four components running are *Cloud Controller*, *Cluster Controller*, *Walrus* and the *Storage Controller*. These components were described in detail in Chapter 2 and Chapter 3. In addition to managing the overall cloud infrastructure, the main server serves as the entry point to the virtual machines that are hosted on the node servers.
- **Nodes** – Each of the nodes runs Eucalyptus' *Node Controller*. It is the nodes that run the virtual machines that are later provisioned to users of the cloud in which

Table 4.1: Hardware specifications used in performance experiment

	Main Server	Nodes	Virtual machines	Client laptop
CPU	3.2 GHz Intel Core i5 (4 cores)		3.2 GHz QEMU Virtual CPU (2 cores)	2.5 GHz Intel Core i7 (4 cores)
RAM	8 G		1 GB	16 GB
Hard Disk	300 GB		5 GB	500 GB

Table 4.2: Software specifications used in performance experiment

Software	**Version**	**Description**
Eucalyptus	2.0.2	Infrastructure as a Service platform
DSpace	1.8.2	Digital Repository software
JMeter	2.13	Application performance measurement tool

DSpace is installed. In the experimental setup, a total of 6 nodes are used, with each node running a total of two virtual machines.

- **Client laptop** – The client laptop is external to the cloud environment. It was used to access the services provided by the cloud environment.

The hardware and software specifications used for this test are listed in Table 4.1 and Table 4.2, respectively.

4.1.2 Methodology

The overall objective of the test is to measure ingestion and viewing of items in DSpace, and ascertain if there is any effect on performance when the number of instances in the cloud are increased or when concurrent requests are made to different instances. It would serve no meaningful purpose if only one instance is evaluated as that would be as good as evaluating a single standalone server machine. The performance experiment answers the research questions by measuring how long common repository tasks are completed in when executed by repository users. In addition, by comparing the performance of a single server installation in the private cloud with multiple cloud server machine instances in the cloud, the performance experiment measures the scalability of private cloud environments and therefore the efficiency and effectiveness part of the research question. Scalability is an important feature of cloud platforms. Note, however, that the experiments are run on commodity desktops machines, machines that would be for very low resourced institutions but provide a good proxy for performance in environments with server machines with medium to high end computer specifications.

To carry out the experiment, Apache JMeter[1] was used to simulate repository user actions and also take response time measurements. JMeter was installed on the client laptop. All requests to ingest and view repository items were carried out from this instance of JMeter on the client laptop. Simulation of repository actions entailed reproducing the two common actions on repository tools, item ingestion and item viewing (downloading). Ingestion of items goes through a series of Web pages where information and metadata of the item being ingested is entered by a repository user. Viewing an item requires knowing a given items' URL endpoint, then navigating to it to load or view it. If the given repository item has an associated resource, for instance a PDF document, the action of viewing of the item proceeds to download that associated resource. For this study, the item used can be found at the URL, `https://doi.org/10.1007/978-3-642-24469-8_57`. It is a PDF document which is 60KB in size. For the experiment, all its metadata was associated with it during the ingestion. The choice for this item was driven by the need to use a real world document, similar to what would be uploaded in institutional repositories.

The experiment began by running a number of virtual machine instances in the cloud environment. Once the virtual machine instances had fully booted, DSpace would be installed in each one of the instances, following the steps outlined in Chapter 3. Using JMeter, 15 items would be ingested into DSpace installed in each of the running virtual machine instances. **Item ingestions in a single DSpace instance were carried out sequentially, while ingestions in all the other running instances in the cloud were run in parallel**. The length of time to ingest an item was measured, which in this study is called ingestion time. Once the 15 items in each DSpace instance were successfully ingested, JMeter was used to view (load) the item. This was to mimic the process of viewing and downloading items with their associated attachments/documents. This step was used to measure the performance of accessing items in a DSpace instance running in a cloud environment. The process of booting virtual machine instances, installing DSpace in the instances, ingesting and viewing all 15 items in each of the running DSpace instances constituted a single run in the experiment. Each run consisted of one or more instances running in parallel. A total of 5 runs were carried out. Each run had a different number of instances running in the cloud. The predefined number of instances at each run were as follows: 1, 2, 5, 8 and 11. That is, in the first run, 1 instance was used; in the second run, 2 instances were used; in the third run, 5 instances were used; then 8 and 11 instances in the fourth and fifth runs, respectively. Note that, each run was also repeated 5 times. The choice for the order 1, 2, 5, 8 and 11 instances was deemed representative enough as was observed during the pilot phase of the evaluation exercise. The exception to that was for not running 12 instances, which was the full capacity of the cloud, due to the experimental environment becoming unstable and thus

[1]JMeter, `https://jmeter.apache.org/`

requiring resetting the entire cloud every time. This has been noted as one of the issues to take note of when running private clouds.

A summary of the process is as follows:

```
begin
  instances := $[1, 2, 5, 8, 11]$
  for instances-to-boot $in$ instances :
    for $i := 1$ to $5$ step $1$ do
      $reset\ cloud.\ no\ instances\ running$
      $boot$ instances-to-boot $at\ once$
      $using$ JMeter, $ingest$ 15 $items\ in\ each\ of\ booted$ instances-to-boot
      $using$ JMeter, $view + download$ 15 $items\ that\ were\ ingested\ in\ each\ of$ instances-to-boot
    end
  end
end
```

From the code above, the experiment has a pre-set number of instances to run at each cycle (loop). In the first cycle, 1 instance is run, then the following five cycles have 2, 5, 8 and finally 11 instances running, respectively. Before each cycle begins, there should be no running instances in the cloud. In addition, each cycle has to complete before the next cycle is started. For the cycle where 2 or more instances are run, calls to boot each instance are made at the same time and thus the instances are run concurrently. Once instances are up and running, each of the running instances will have 15 items ingested in them. The action of ingesting items across the running instances is carried out concurrently while the ingestion of each of the 15 items in each running instance is carried out sequentially. For a given instance, once the ingestion step has completed, the 15 ingested items will be viewed and their associated file downloaded in sequential order.

4.1.3 Results – Ingestion Time

The average times to ingest an item into DSpace are shown in Figure 4.2. Figure 4.3 shows the average ingestion time by order of ingestion of the DSpace items and, finally, Figure 4.4 has a detailed breakdown of the average ingestion times by instance and also ingestion order of the item.

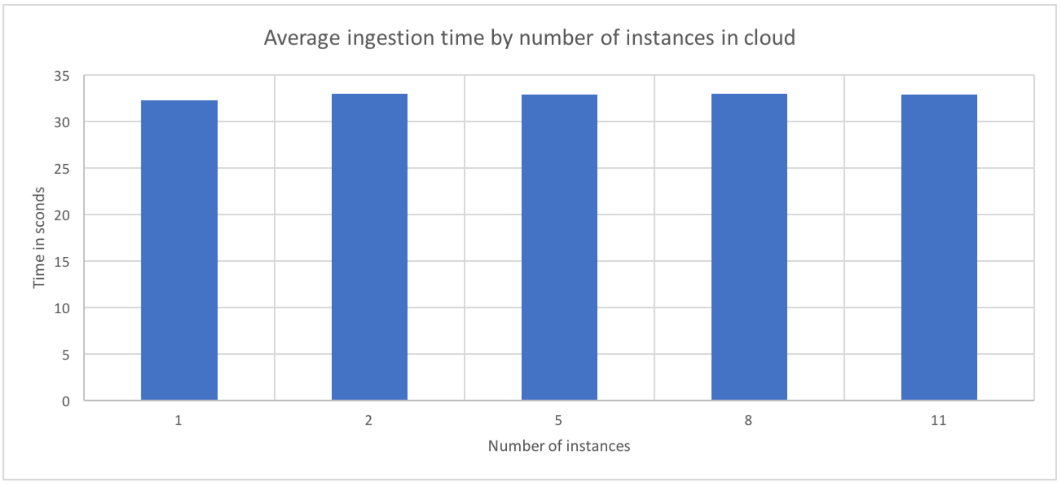


FIGURE 4.2: Graph depicting the average time to ingest a DSpace item by the number of running instances in the cloud

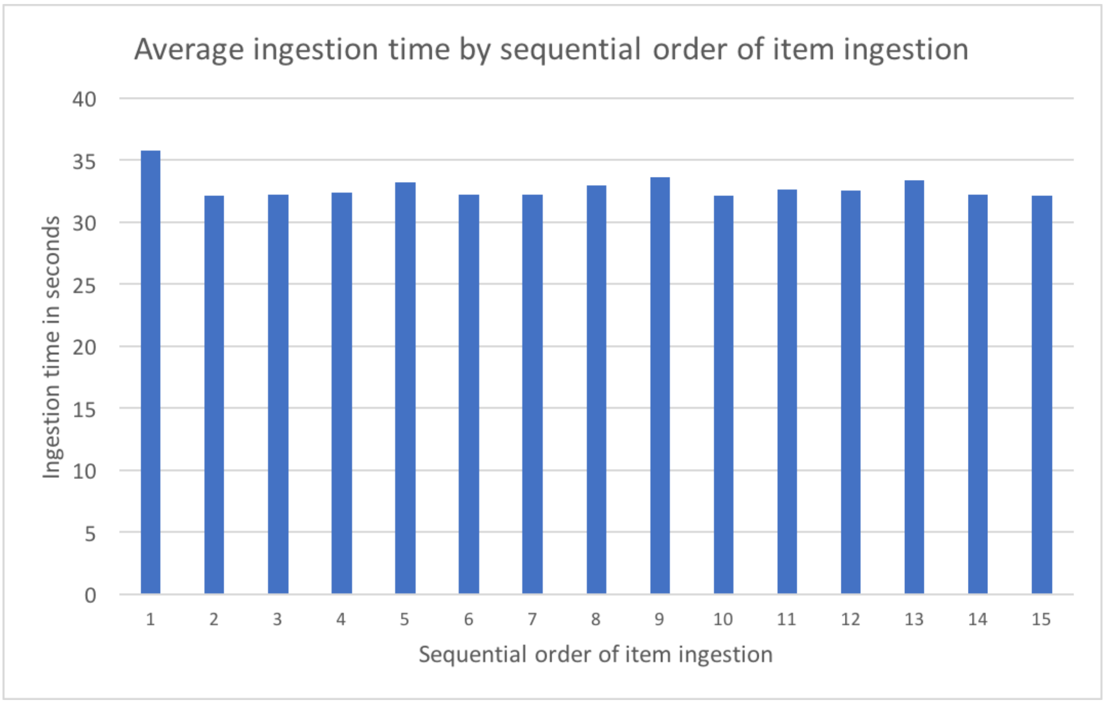


FIGURE 4.3: Overall average ingestion time by order of ingestion and number of instances

4.1.4 Discussion – Ingestion Time

At the onset of this project, it was not certain what performance hits would be incurred when running a DSpace instance in a cloud environment, especially when the cloud has other virtual machines running. It can be deduced from Figure 4.2 that an increase in the number of instances does not have a considerable effect on the performance of the ingestion of an item in DSpace. The overall average ingestion time remains between 30

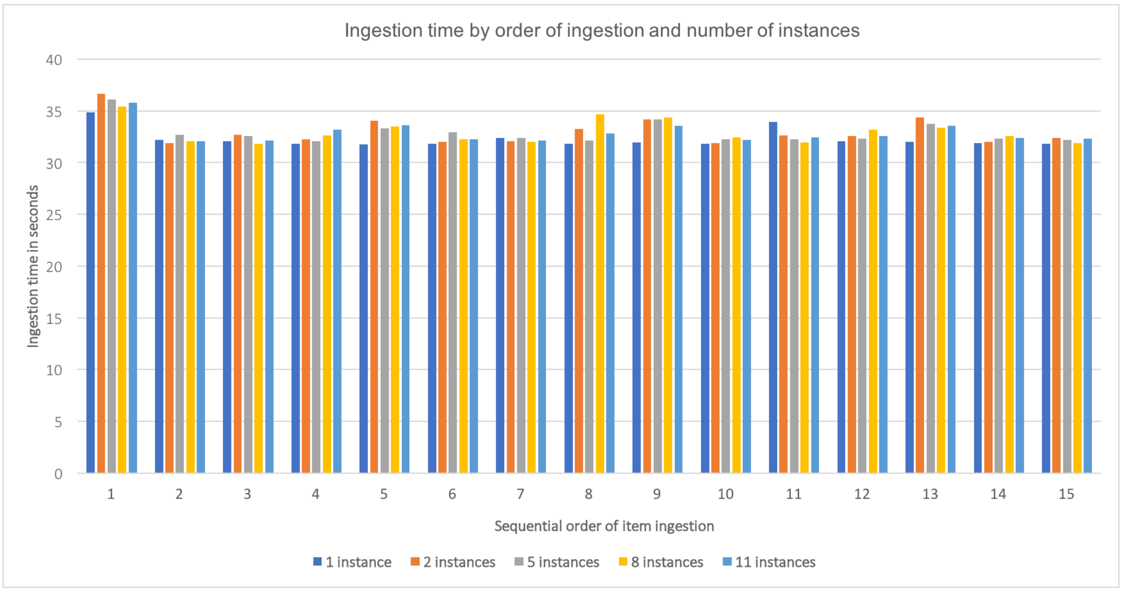


FIGURE 4.4: Overall average ingestion time by order of ingestion and number of instances

and 35 seconds. The average ingestion time by 1, 2, 5, 8 and 11 instances are 32.27, 32.98, 32.86, 32.93 and 32.86 seconds respectively.

Figure 4.3 and 4.4 show that the order in which the items are ingested into DSpace also remains stable: there is no spike in the response time as more items are ingested into DSpace. However, the initial ingestion time into DSpace is noticeably higher than the rest of the ingestions. This can be attributed to Apache server on the central cloud server making its initial connection with the Tomcat server on the DSpace instance in the cloud. In addition, at first run, Tomcat on the DSpace instance will be loading the necessary resources, which may contribute to this slowness. Once connections have been established and other configurations cached by Apache on the main server and the Tomcat server has been loaded fully, subsequent requests are noticeably faster.

Since in a cloud environment the cluster of machines are managed from a single point (the cloud server) and resources are shared between virtual machines (on node servers), it would be expected that the more instances that are running, the slower the tasks' execution. This experiment demonstrates that, for the ingestion task, the performance degradation is noticeably small. As can be seen from Figure 4.4, even an increase in the number of DSpace instances, which also resulted in more requests being sent to the server concurrently, there is no noticeable increase in the wait time. Because the increase is in milliseconds, it would not be noticed on the part of the user of the system.

In order to further ascertain the magnitude of the difference in the ingestion times when run in an environment with varied numbers of instances, a one-way ANOVA test is carried out on the 5 different instances used. Using SciPy's f_oneway function, an F-statistic of 3.19 and p-value of 0.012 is obtained. At a significance level of 95%, it can be concluded that there is a significant difference in the ingestion times, that is, rejecting the null hypothesis that there is no difference between the average item ingestion times into dspace when multiple instances are run at the same time. However, when the differences are compared between each number of instances, ingestion times when there is a single instance in the cloud are lower than when multiple instances are running. When the ingestion time is compared between multiple running instances, Table 4.3 shows that there is no significant difference in the ingestion times between them. It can be concluded that there is indeed a significant difference when one instance is used compared to any number of multiple instances. But, because the differences are in milliseconds, it can be argued that there is no practical implication on any number of instances running in the private cloud.

TABLE 4.3: Ingestion time, p-values for paired number of instances in the cloud

Number of instances	**t-statistic**	**p-value**
1 vs 2	-3.83	0.0001
1 vs 5	-4.07	0.00007
1 vs 8	-4.53	0.00001
1 vs 11	-4.39	0.00002
2 vs 5	0.58	0.55
2 vs 8	0.30	0.75
2 vs 11	0.79	0.42
5 vs 8	-0.41	0.67
5 vs 11	0.21	0.83
8 vs 11	0.75	0.44

4.1.5 Results – Item View and Download Time

Item view time (item view response time) is the time taken to load an item with its metadata in the browser after a user has requested for it, and the mode in which this was measured is discussed in the methodology section of this experiment. The average response times are shown in Figure 4.5. The download time is the time taken to download a file associated with a given DSpace item. The download average time is also shown in Figure 4.5. Figure 4.6 shows the average item view time by number of instances and in sequential order that the file was requested for. Figure 4.7 is a summary of the average download times by number of instances and in sequential order of item download. Note that as much as there is a noticeable difference between running 1 instance and 11 instances, all response times are under a second, with the maximum response time about one eighth of a second.

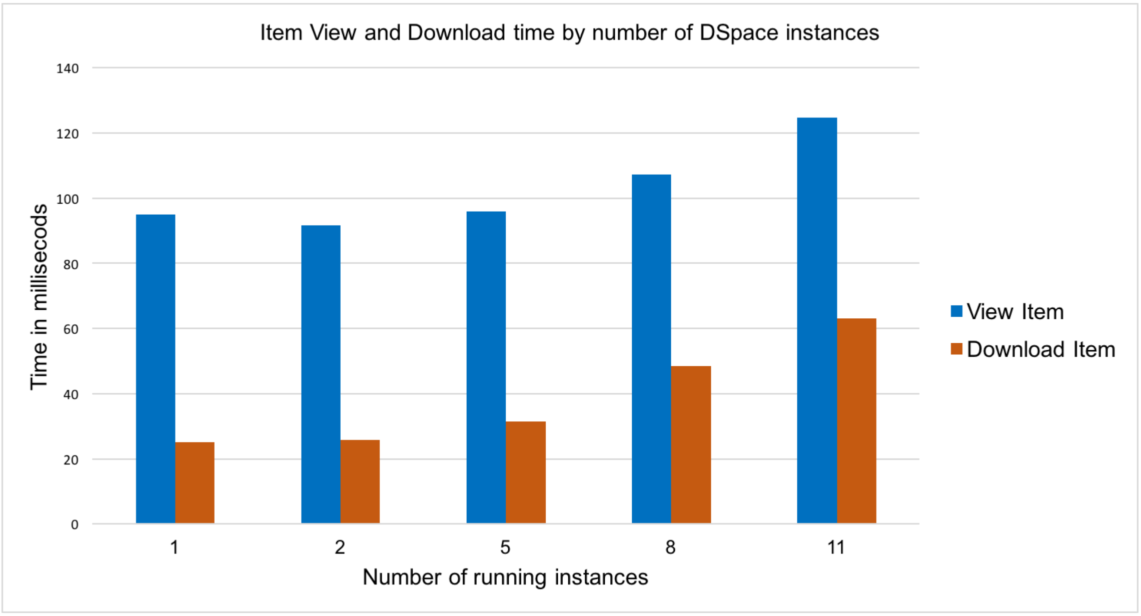


FIGURE 4.5: Overall average item view and download time by number of instances

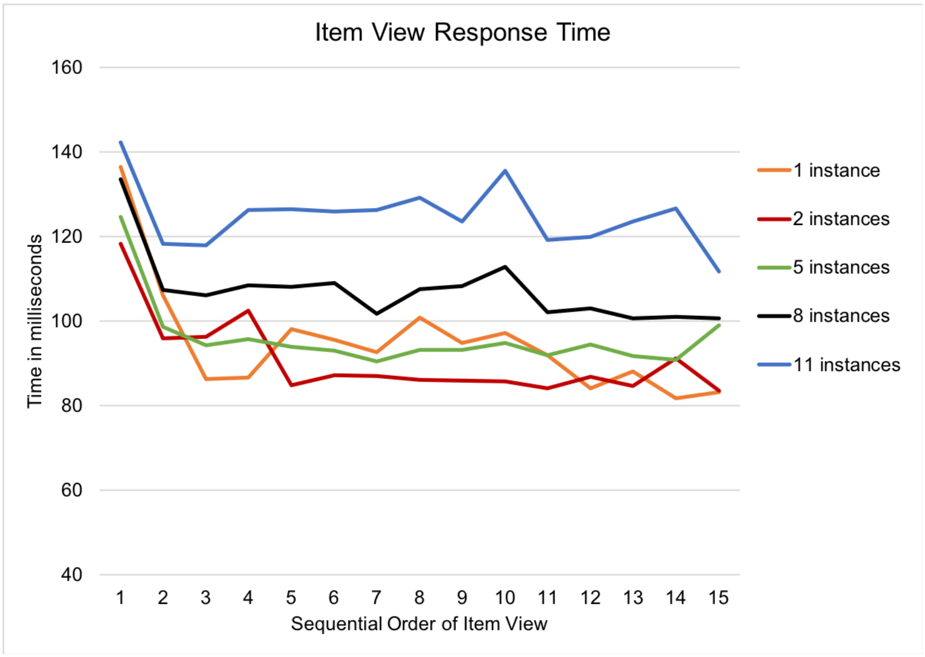


FIGURE 4.6: Average item view response times by number of instances and order of item view

4.1.6 Discussion – Item View and Download Times

Overall, the average response times for the two tasks – item view and item download – are all under one eighth of a second. However, there is a noticeable linear decrease in performance as the number of instances running in the cloud are incrementally increased to full capacity. This difference would be hardly felt by users of the system. Using the ANOVA test to compare the differences between the view times of 1, 2, 5, 8 and 11

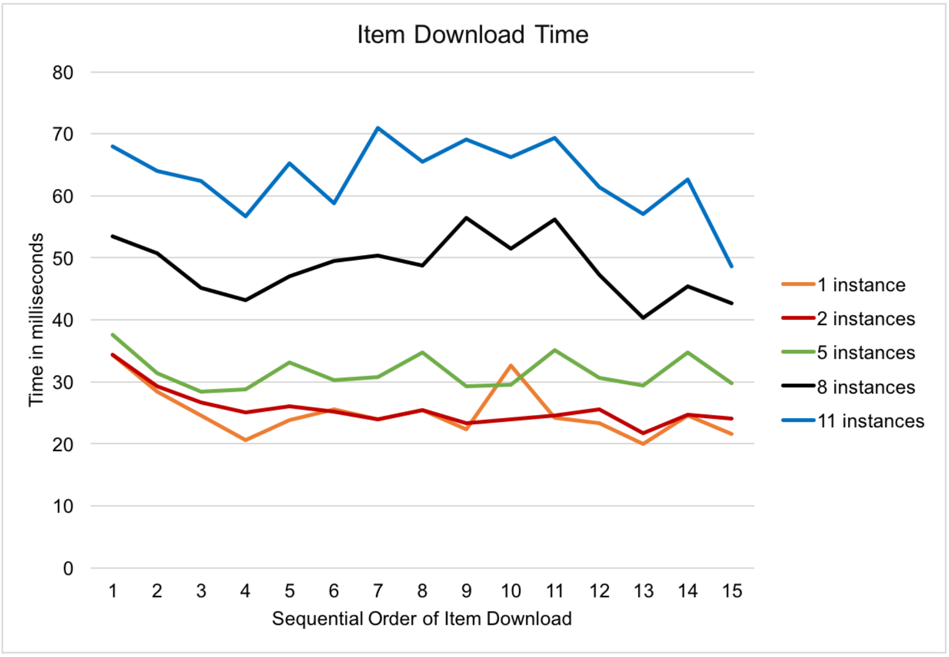


FIGURE 4.7: Average item download response times by number of instances and order of item download

instances, an F-statistic of 72.57 and p-value of $2.23e-55$ is obtained. Since p<0.05, it can be said that there is a significant difference in the view times experienced.

Therefore, overall performance is affected as the number of instances in the cloud are increased. However, due to the increases being in milliseconds, users will barely notice the performance degradation. With these results, it can be said that for private clouds of similar sizes as used in this experiment, slight performance degradation can be experienced when clouds are run at full capacity.

4.1.7 Performance Experiment – Summary

From the experiments, it can be summarised that while the ingestion time when multiple instances are running in the cloud is significantly different, the difference would not be discerned by system users. Because the difference the repository user would experience when one machine is running in a cloud environment compared to when multiple instances are being run would not practically be discerned by users, it can be said that the users of repositories would complete their tasks effectively and efficiently.

The experiment on item view and item download showed that there is a significant difference between a single server setup and multiple server setups in a cloud environment. However, due to size of the increase in the view time, it can be argued that there is no practical significance between the view times. It is also important to note that these actions of viewing and downloading items are what will be subjected on the repository more often by users than the act of ingestion of items.

The results from the two performance experiments show that from a practical standpoint, it is possible to host digital repository in Private Cloud environments efficiently and effectively.

4.2 Usability Evaluation

Cloud systems come with unique challenges and, unlike in traditional systems where the computing systems running applications are in physically close proximities, in cloud systems all interactions have to be carried out remotely. This presents a challenge that non-expert users would contend with. Effectiveness, hence in this context, is determined to mean the ease with which carrying out installation and management tasks on digital repositories can be achieved. Commonly, digital repository installations are carried out from a command line interface. In this study, a browser based software application interface is provided to carry out installations and management of instances and, as such, effectiveness and efficiency evaluations are based on the usability of the management interface developed. The objective of this evaluation is centered around the following questions:

1. Is the developed tool easy to use?
2. Are the features in the prototype sufficiently complete. This will help identify what additions need to be made to the application's features to make it more useful and usable.
3. Do results from completed tasks meet user expectations? Here we establish if, for instance, the customisation function produces results that meet the users' expectations.

4.2.1 Questionnaire Rationale

To answer the questions posed in the previous section, usability studies were conducted. Usability, as defined by ISO 9241-11 [75], comprises aspects of efficiency, effectiveness and satisfaction [76]. These components form what is at the core of the objectives to be investigated in this study. The System Usability Scale (SUS) [77] is adopted for this study. Of the other potential options [2,3], SUS not only meets the needs of this study, because of its applicable set of questions, but it is also widely used[4], has respectable reliability [78] and is available for free use without a licence. For the purposes of evaluating the developed prototype, the questions in SUS have been slightly adapted, with appearances of *system* replaced with *application* or *tool*. SUS has 10 questions only, therefore making it easy for participants to use and also for scoring purposes.

As SUS will only give a usability score for the whole system, a post-task subjective psychometric questionnaire is used. The After-Scenario Questionnaire (ASQ) [79] is a three-item after scenario rating, that measures user satisfaction immediately after a task has been completed. Though simple, it has been shown that it fares well with other competing evaluation methods [80, 81]. For the purposes of this study, the question *Overall, I am satisfied with the support information (on-line help, messages, documentation) when completing the tasks?* was dropped from the list of questions used in this study. However, a free text field was added to allow participants to express their feelings that might not have been captured by the provided ratings. The free text field was also added in the overall system usability evaluation using SUS.

4.2.2 Study Methodology

This study was conducted from within the University of Cape Town's intranet. Participants were recruited from a pool of postgraduate students. The minimum requirements for participants were that they should be familiar with Web technologies and be everyday users of the Internet. They were not expected to be expert technology users. Before the participants engaged in the study, they were asked to sign a consent form. All participants consented to take part in the study.

The evaluation exercise involves completion of 2 main tasks that were the core features of the developed prototype: installation and customisation of a digital repository. After completion of each of those two tasks, users were asked to answer the task's associated post-task question. When the two tasks have been completed, participants then proceeded to answer the SUS questionnaire that measures the overall system usability.

Refer to Appendix A: Survey Questionnaire, for the detailed task descriptions, instructions and survey questionnaire.

[2] User Interface Usability Evaluation, `http://hcibib.org/perlman/question.html`

[3] A Comparison of Questionnaires, `http://www.upassoc.org/usability_{}resources/conference/2004/UPA-2004-TullisStetson.pdf`

[4] Measuring Usability with the SUS, `https://www.measuringusability.com/sus.php`

4.2.3 Participants and Technical Ability Level

Twenty two participants took part in the evaluation study. From the outset, participants were asked to list any software applications that they had installed and /or configured before. This was a test to gauge roughly how familiar they were with carrying out installations that could be time-consuming and ascertain their technical ability levels. Based on the results, participants were identified to fall into three (3) categories:

- **Non-expert**: Users who have never installed nor configured an application whose characteristics are similar to any DSpace dependent application.
- **Intermediate**: These users have installed an application before that shares some similarities with DSpace's dependent software. The application could also be DSpace's dependent application, e.g., a user who lists Apache as a single system they have installed or configured is considered an intermediate user.
- **Expert**: These users should have installed either DSpace (or any Repository application), at least any two of DSpace's dependents components, or at least any two such applications that share similarities with any of DSpace's dependent applications. For instance, a given user who has installed MySQL and GlassFish would be classed as an expert.

The assumption behind these classes is that non-experts are not expected to understand the intrinsic details of configuring Web applications, intermediates would have been exposed to running Web applications and may even have some understanding of basic HTML. Experts (the word expert is used very loosely), have a good understanding of running Web applications and they are very familiar with installation and configuration of developer environments required to setup a Web site. Note however that a high level of technical ability was not a prerequisite to take part in the study. This classification of participants will be used in the other sections to help identify whether users' technical abilities affect their perception of the application's usability.

Table 4.4: Breakdown of participants by category

Level Of Familiarity	**Number of Participants**
Expert	10
Intermediate	6
Non-expert	6

Table 4.4 shows the breakdown of each category and its associated number of participants. For an understanding of how these categories were arrived at, refer to Table 1 of Appendix B: Survey Questionnaire Results.

4.2.4 Installation Task – After Scenario Questionnaire Results and Discussion

Using the developed prototype for installing and managing repositories, participants were asked to carry out an installation of DSpace and immediately were asked to rate their experience based on the following statements: 1) *Overall, I am satisfied with the ease of completing the tasks in this scenario*, and 2) *Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.* A five point Likert scale was used for the responses, with 5 being strongly agree and 1, strongly disagree. All questionnaires were adequately filled in.

Results

The complete raw results can be found in Appendix B: Survey Questionnaire Results. Figure 4.8 shows the average scores for each category of the participants and the overall scores for all participants. The overall average score for the perception of the duration of time to install DSpace is 3.4 while the median score is 3.5, both of which are just above average. The score for the ease of installation is more positive compared to the installation duration perception. The average score for the ease of installation is 4.3, while the median score is 4.5. The boxplot in Figure 4.9 gives a distribution summary of the results

When asked to comment on what they liked about the installation process, the study participants highlighted the simplicity of the user interface and also the few steps required to complete the installation. The comments included, 1) *"very few details needed; clean install; progress bar to show install progress";* 2) *"very simple and required no experience; very clean and simple UI";* 3) *"one click installation, very simple"*, etc. When the participants were asked what they did not like about the installation process, their responses showed that the duration taken to complete the installation was a problem. Some of the comments included the following, 1) *"installation took too long"*; 2) *"not sure how long it normally takes but installation seemed too slow"*; 3) *"it took too long, it was too slow"*; etc. These comments help explain why the duration of the installation had a median score of 3.5 and why the median score for the ease of use was 4.5.

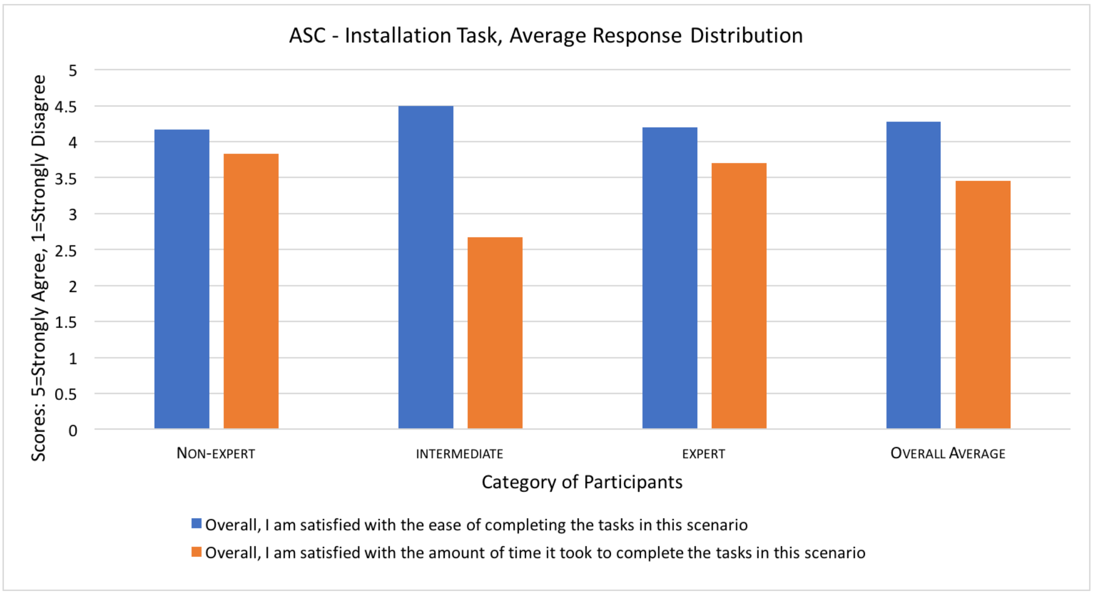


FIGURE 4.8: Average scores for the installation task by category of the study participant

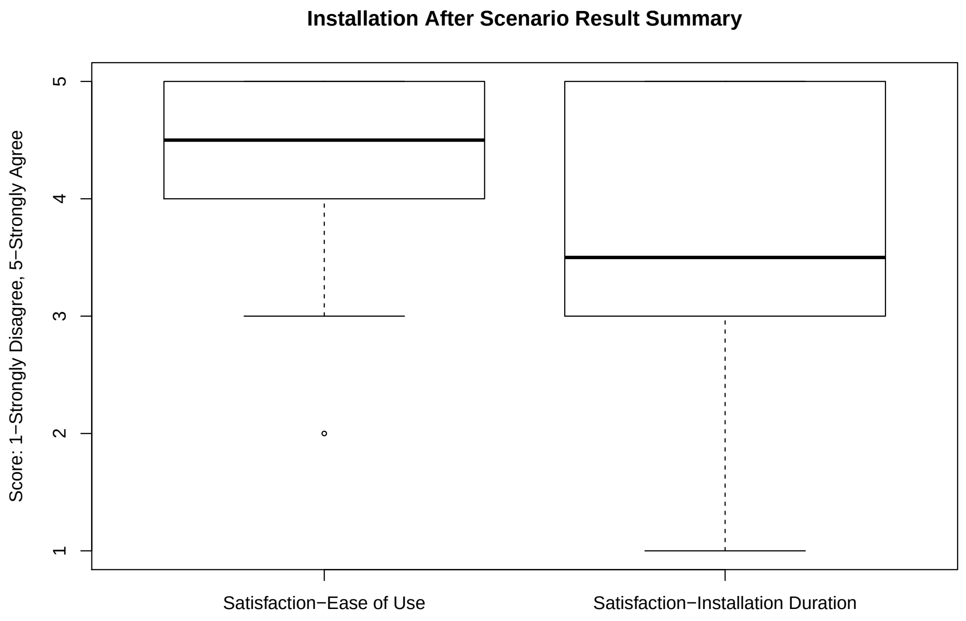


FIGURE 4.9: Boxplot showing a summary of the results for the installation task.

Discussion

The main result from this part of the usability study is that the time taken to complete the installation is long for the average user. It is important that for any improvements to the installation function, installation duration time would have to be a priority. However, even with the mixed responses on the duration time, the overall usability perception of the installation task was a positive one. The accompanying comments were of a satisfactory tone, with all the participants, bar one, completing the installation process with very little help offered.

The time taken to complete the installation is a constraint of the cloud environment's response times. A number of techniques can be applied to ensure that users do not have to wait for over 8 minutes to start administering their repositories. One way would be to create instances in advance – once they are allocated to users, their only task would be to configure the system to their preferences. This is an aspect that would only be useful in a high traffic environment. For organisations where requests would only be started once in a while, the wait times would suffice in comparison to the official DSpace installation steps.

4.2.5 Customisation Task – After Scenario Questionnaire Results and Discussion

The complete raw results for the subjective multiple choice questions asked immediately after the participants carried out the customisatiom task are in Table 3 of Appendix B: Survey Questionnaire Results. For the customisation task, users had to make changes to the look and feel of the DSpace installation, making modifications following their instinctive preferences. Figure 4.10 shows the average scores by each category of participants. The overall scores are above average. This means that the users found the task easy to complete and they perceived the time taken to complete the customisation task positively.

For the question, *Overall, I am satisfied with the ease of completing the tasks in this scenario*, the median and the mean score both are tied at 4 – the Likert scale used had 1 as strongly disagree and 5 as strongly agree. Figure 4.11 shows the distribution of the results, indicating that most participants scored the ease of use for the customisation task with 4. Of the 22 participants, 2 indicated that they had an unsatisfactory perception of the ease of usability of the customisation feature of the application by scoring the usability with a 2. The general picture that arises when the results are presented using a boxplot as shown in Figure 4.12 is that the majority of the participants found the feature easy to use as indicated by a median of 4 and all participants in the 4th quartile scoring it with 5 – strongly agree.

The time it took to complete the customisation in itself had high scores, with no participant scoring it below 3. This is understandable as the user interface for the customisation

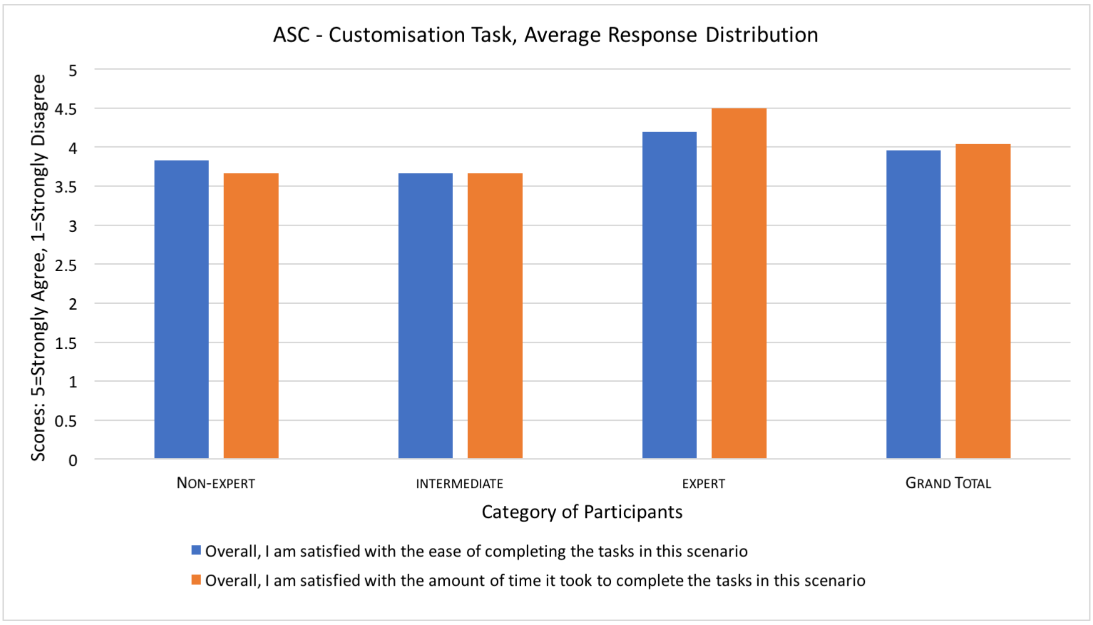


FIGURE 4.10: Average customisation task scores by participant category

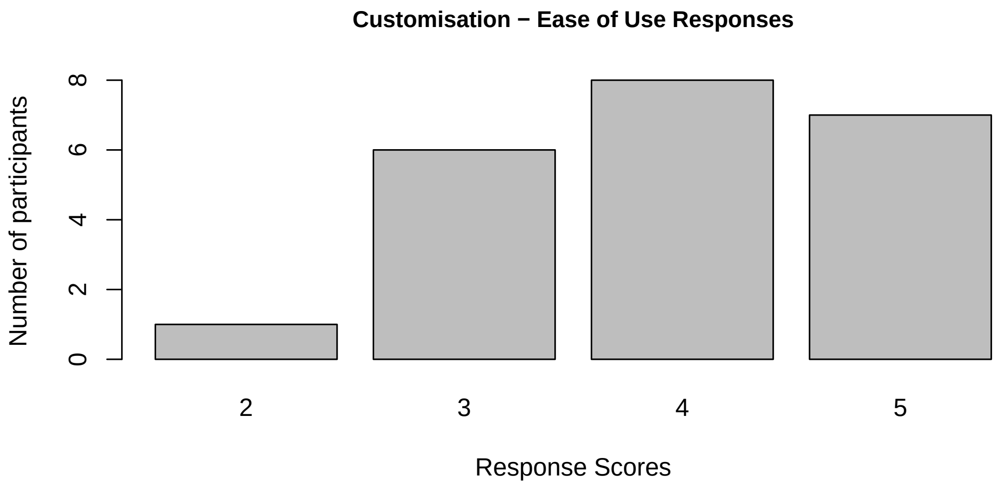


FIGURE 4.11: Distribution of responses to the ease of use question after completing the customisation task.

tasks provided relatively few fields to make changes to, and once the participant completed filling in their preferences for the different sections, there was little to no waiting at all.

It can be deduced from the given results that the overall perception of the customisation process is positive. However, as Figure 4.12 shows, some improvements would still improve the user's usability perception.

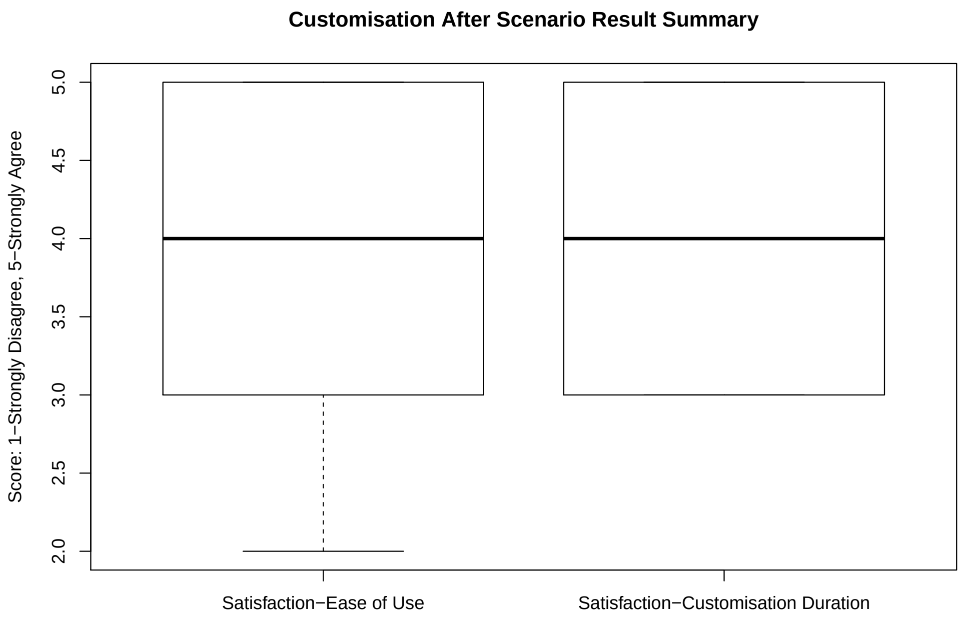


FIGURE 4.12: Boxplot showing a summary of the results for the customisation task.

4.2.6 Overall Application Usability – Results and Discussion

Results

The overall average SUS score from the evaluation was 74. SUS is scored out of 100. There was an observed difference in the average scores by each of the categories that were devised: experts scored an average of 81, intermediates scored 65 and non-experts scoring 70. Table 4.5 has the results broken down by category and their associated standard deviations. Figure 4.13 is a graphical representation of the results.

TABLE 4.5: SUS Scores by User Category

Category	**Average SUS**	**No. Participants**	**Standard Deviation**
Non-expert	70.8	6	19.14
Intermediate	65.0	6	14.91
Expert	81.97	10	24.30
Overall SUS	73.97	22	21.08

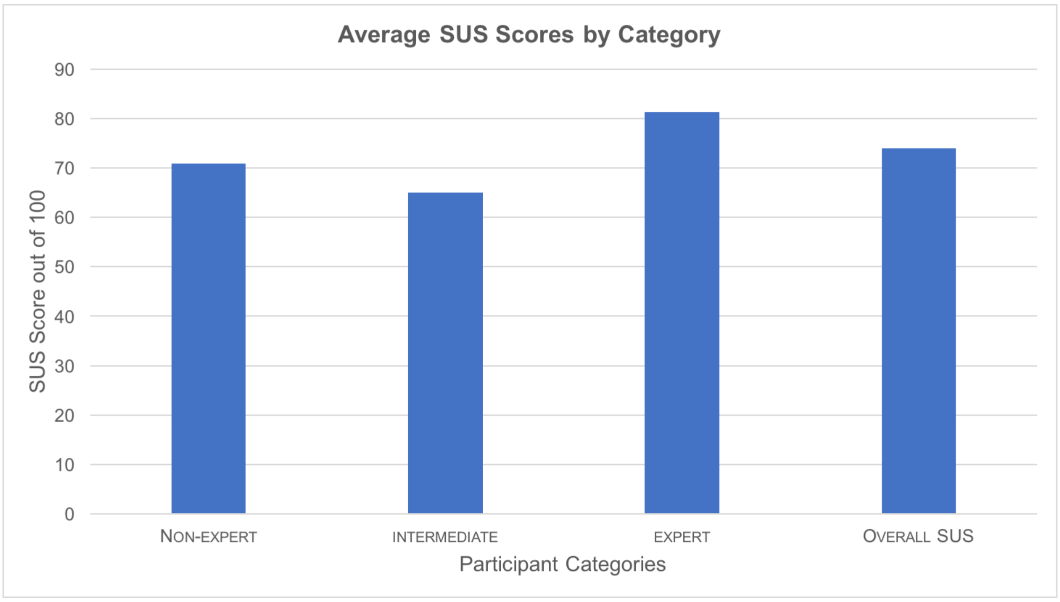


FIGURE 4.13: Overall SUS scores by participant category

Responses to the question, *What did you like about the application?* and *What did you NOT like about the application?* were transformed into categories for easy grouping and analysis. Table 4.6 shows an example of how the translations were done. Essentially, each response was interpreted and transformed into a specific positive or negative area of the application. The response was then assigned a specific short phrase/code. When a response included multiple positives or negatives, the response was assigned multiple codes equivalent to the nunber of positives or negatives in the response. For instance, one participant's response to the question that solicited a negative aspect of the application was, *"Tricky to understand; slow to install; not enough user feedback; finish button bug"*, and was translated to 3 codes namely, *"Not-responsive-enough"*, *"Difficulty-to-understand"*, and *"Installation-takes-long."* From the translations, more positive comments were given compared to the negative ones. There was a total of 39 unique positive comments and 27 unique negative comments. Figures 4.14 and 4.15 show a summary of what each group of participants said as what they perceived as negative and positive. From the figures, it can be deduced that ease of use was the most positive perception and installation taking long as the most negative perception of the system.

TABLE 4.6: Sample translations and grouping of negative responses

What did you NOT like about the application?	**Response translation**
Use of personal information; settings not retained	Personal-information-not-reusable
Takes some time installing	Installation-takes-long

Some minor bugs: about the link not working; cursor treats customisation labels as links and acts inconsistently when clicking these labels	Minor-bugs
The long wait time on creation, having to clear cache to view upate	Need-to-clear-cache
The long wait time on creation, having to clear cache to view upate	Installation-takes-long
Installationspeed; effects of customisation should be seen during the customisation, not after	Installation-takes-long
Installationspeed; effects of customisation should be seen during the customisation, not after	Not-responsive-enough
Limited options – information about what is going on	Limited-options
Tricky to understand; slow to install; not enough user feedback; finish button bug	Installation-takes-long
Tricky to understand; slow to install; not enough user feedback; finish button bug	Not-responsive-enough
Tricky to understand; slow to install; not enough user feedback; finish button bug	Difficulty-to-understand
Still buggy around the edges – some refreshing was needed; needs more details when performing tasks (updates, installs etc)	Minor-bugs

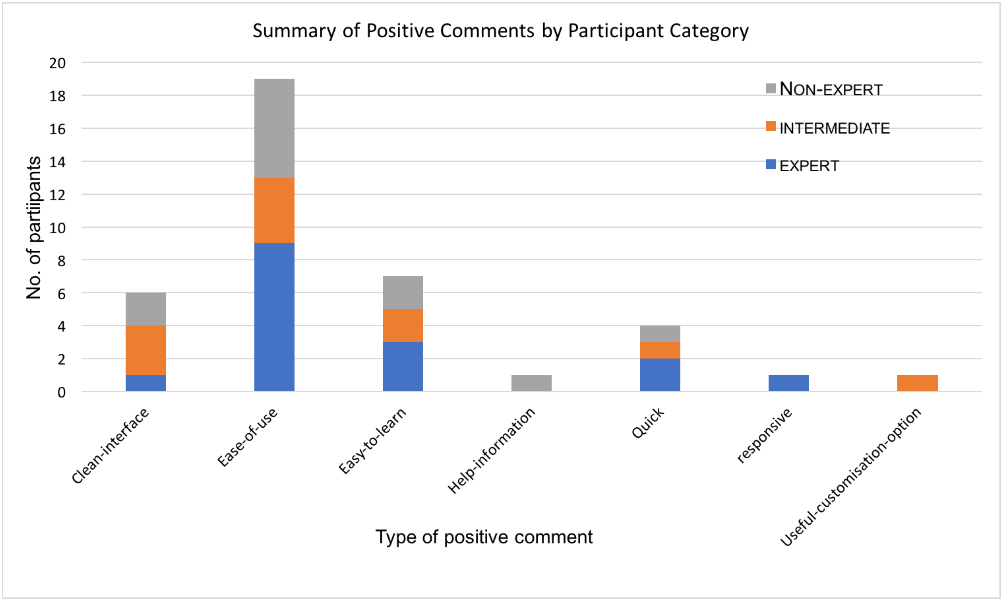


FIGURE 4.14: A summary of positive comments given after use of the management tool

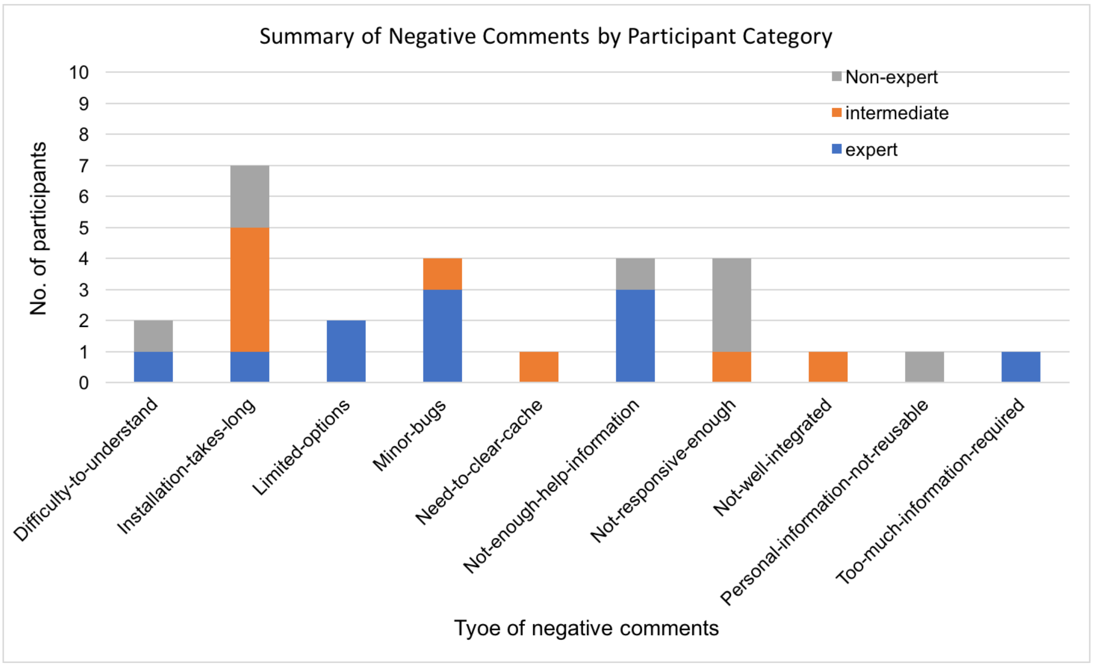


FIGURE 4.15: A summary of negative comments given after use of the management tool

Discussion

From the overall SUS score, it can be concluded that the system's usability of 74 is above average. Of note, Intermediate study participants scored their perception of the developed application less positively compared to both the Expert and Non-expert participants. From the open-ended responses given, it can be deduced that the length of time to install the repository was important to the Intermediate participants as it affected how they perceived the usability of the application. It can be speculated that their previous experience of installing dependent software of a digital repository made them expect the installation of the digital repository to take about the same time as its dependent software. This can be contrasted with the Expert participants (Expert participants had experience installing digital repositories) who only registered one negative response that has to do with the duration of the installation.

For Non-expert and Expert participants, it can be deduced from the comments that the ease of use of the application is one of the reasons they scored the system relatively more positively compared to Intermediate participants. Non-expert participants' relatively lower score compared to the Expert participants' can be attributed to how they perceived the responsiveness of the application – a majority of the Non-experts' negative comments were about how the application was not very responsive.

A comparison of the overall positive and negative comments attributed to the system provides more support that the system is generally usable as there are more positive comments provided. Apart from the installation time, which was queried by 7 participants, the other negative perceptions were expressed by 4 participants.

The results from this evaluation indicate that users are able to complete tasks of installation and customisation of the system, and there is an above average satisfaction in interacting with repositories in the cloud through use of the developed system. This answers the question on the efficiency of hosting digital repositories in private cloud environments.

4.3 Adapting DSpace for Cloud Deployment

This study's second research question is about how DSpace, or other commonly used Institutional Repository tools, can be adapted for easy management and deployment. The two experiments carried out dwelt on evaluating the performance of DSpace in a cloud that was deployed using the application tool developed to manage repositories in cloud environments – which was the second evaluation done. Therefore, the success from the evaluation help answer the research question on how repository tools should be adapted for cloud environment.

DSpace was not adapted in any way. However, what was tweaked was the installation process of DSpace, which contributed to building an interface allowing for easy management and installation of DSpace. In summary, the architectural makeup of DSpace was not adapted, but its installation steps were.

The developed tool included a feature to customise DSpace's look and feel. This too, did not require a change to the overall design of DSpace, but instead were simple edits to DSpace's front-end technology – HTML and XSLT – that enabled automating and simplifying the process of how customising the look and feel is carried out by developers. The edits were a creation of placeholders for changing the colors, fonts and placements of certain user interface componets. Using user supplied details, these placeholders would be updated during runtime with changes made reflecting user defined preferences. This was described in detail in Chapter 2 of this report.

It can therefore be concluded that deploying DSpace in the cloud requires no architechural changes to the repository application. Building tools to manage repositories and customise their look and feel may be necessary to widely have low cost private cloud environments built and supported by non-techincal users. In addition, simplifying the process of installation is also beneficial to all users as was discussed in the usability evaluation of this study.

4.4 Summary

This chapter attempted to answer the research questions that were set out for this study. Two different evaluations were performed: a performance experiment and a usability study.

The performance experiment revealed that private cloud environments can indeed host digital repositories. Much as there was a significant difference in the view and download times as the number of instances increased, the degradation in time would be barely noticed by users as the time difference is in milliseconds.

The usability study looked at the prototype developed to help manage digital repositories. Different types of participants were recruited for the study. The outcome of the study showed that the tool developed was positively perceived. Most participants were able to install DSpace and, at the same time, carry out simple customisations of DSpace.

Chapter 5

Conclusion and Future Work

It was set out at the onset to host digital repositories in private cloud environments. Eucalyptus was identified as the Cloud System to provide IaaS. A prototype was developed to aid the installation and management of a repository in a cloud environment. DSpace was used for this project. A performance experiment of DSpace running in the cloud was done, and a usability study for the installation and customisation of DSpace was done.

This work was to realise the objectives set out, which were as follows:

1. Create a one-click installation of a digital repository tool
2. Develop a way for users to monitor repository activity in cloud environments
3. Develop a way for users to customise repositories even when such users are lacking in Web development technology skills

The listed objectives were to help answer the following research questions:

1. Is it possible to host digital repositories in private cloud computing environments efficiently?
2. How can repository tools like DSpace be adapted for the cloud for easy management and deployment?

5.1 Summary of Findings

Is it possible to host digital repositories in private cloud computing environments efficiently?

The performance experiment for ingesting and viewing items in DSpace showed that there is noticeably little effect on the performance of a repository even as the number of instances were increased. The increases were linear in nature and as such, when the hardware resources' specifications are increased, traffic into the cloud would be gracefully handled.

The usability study focussed on the experiences of end users completing installation tasks and customising an installed repository. The time to complete an installation was found to be an issue that can be improved on to improve the overall user perceptions of the task. However, even with that drawback, the overall score for the installation was above average.

It can be concluded that DSpace was installed successfully in a cloud environment with ease.

How can repository tools like DSpace be adapted for the cloud for easy management and deployment?

The proposed solution involved developing a prototype to aid installation and management of DSpace. For the prototype to achieve the single-click install of DSpace, dependent software and libraries had to be pre-compiled. An automation and configuration management tool was used that interfaced with a Web frontend to manage requests from users and execute them on the target machine instances. This required that files that needed to be changed, like configuration files, were written as templates that would be updated with parameters (which are user defined details) passed from the front-end at configuration time, all transparent to the user. The tool had a feature to make minor customisations to the look and feel of DSpace. This feature is not a trivial one as without the prototype developed, technical skills are required to achieve the desired look-and-feel.

A running DSpace instance in the cloud required no architectural changes. Therefore, for easy management and deployment in cloud, DSpace did not require any adaptation. However, developing the prototype to manage installations and configurations was important in answering the second research question. The usability study conducted, provided the evidence that the prototype made it easy to manage, customise and deploy repository tools in cloud and thus answering the research question.

5.2 Study Contributions

This study has made two contributions to the body of knowledge on hosting repositories in private cloud environments:

1. Development of a DSpace installation and management tool for private cloud environments. This would enable users or system administrators perform installation of repositories with a lot of ease.
2. It has shown, through its step-by-step documentation, how to efficiently install a repository in a private cloud environment. This work can serve as a template to guide setting up of private cloud environments that run DSpace.

5.3 Future Work

This work solely focussed on hosting a new repository in a private cloud environment. On this account, the objectives were met. However, there should be an exploration of migrating an existing repository into a private cloud environment.

Eucalyptus was used as the cloud system to enable the serving of an Infrastructure as a Service service. Another study should look into using other existing cloud systems like OpenStack.

The setup of the prototype was off commodity computers. Some future works should consider using enterprise grade computers for the private cloud. Small to medium sized institutions could leverage their data centers to setup and run private cloud environments.

5.4 Lessons Learnt and Reflection

The features of the cloud systems that provide on-demand resources, and make them elastic come at a cost. It is not always that instances will be successfully booted the first time. Applications developed for cloud systems, or managing applications in cloud systems should factor in failure of compute resources.

It is important to keep a close eye on the underlying infrastructure of the private clouds. Obvious as it may seem, most of the times, failures to terminate a virtual machine, or start one, was because of problems on the commodity machine on which the virtual instance was to be provisioned.

It was observed that when running the cloud at full scale, some virtual instances would abort without any useful information provided. This resulted in the performance experiments being run with a maximum of 11 virtual instances instead of the cloud capacity, which is 12 virtual machine instances.

Interfacing Django Webframework and euca2ools was particularly easy due to both applications using Python. However, this project having been using the early releases of euca2ools and generally Eucalyptus, low-level calls from Django would have to be made to euca2ools instead of using the provided API.

For this study, 4 components of the cloud system were installed on one machine. The 4 components were Cloud Controller, Cluster Controller, Walrus Controller and Storage Controller. This setup should never be used in a production environment as it creates a single point of failure.

References

[1] Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Andrew Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, and Ion Stoica. Above the Clouds: A Berkeley View of Cloud Computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13), 2009.

[2] Mell Peter and Grance Timothy. The NIST Definition of Cloud Computing, 2011. URL `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`.

[3] Yan Han. On the Clouds: A New Way of Computing. *Information Technology and Libraries*, 29(2):87–92, 2010. URL `https://search.proquest.com/docview/325033464/fulltextPDF/A52FC66FB8DE4D9BPQ/1?accountid=14500`.

[4] Bankier Jean and Gleason Kenneth. Institutional Repository Software Comparison. pages 1–16, 2014. URL `https://unesdoc.unesco.org/ark:/48223/pf0000227115`.

[5] DSpace. DSpace – A Turnkey Instutitional Repository Application, 2018. URL `https://duraspace.org/dspace/`.

[6] MacKenzie Smith, Mary Barton, Margret Branschofsky, Greg McClellan, Julie Harford Walker, Mick Bass, Dave Stuve, and Robert Tansley. DSpace – An Open Source Dynamic Digital Repository. *D-Lib Magazine*, 9(1), Jan 2003. doi: 10.1045/january2003-smith. URL `http://www.dlib.org/dlib/january03/smith/01smith.html`.

[7] EPrints. EPrints Services, 2018. URL `http://www.eprints.org/uk/`.

[8] Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. Fedora: An Architecture for Complex Objects and their Relationships. *International Journal on Digital Libraries*, 6(2):124–138, Apr 2006. ISSN 1432-5012. doi: 10.1007/s00799-005-0130-3. URL `http://link.springer.com/10.1007/s00799-005-0130-3`.

[9] Michal Kökörčený and Agáta Bodnárová. Comparison of Digital Libraries Systems. In *Proceedings of the 9th WSEAS International Conference on Data Networks, Communications, Computers*, DNCOCO'10, pages 97–100, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-960-474-245-5. URL `http://dl.acm.org/citation.cfm?id=1948805.1948823`.

[10] Adewumi OA and Omoregbe NA. Institutional Repositories: Features, Architecture, Design and Implementation Technologies. *Journal of Computing*, 2(8), 2011.

[11] Shahkar Tramboo, Humma, S M Shafi, and Sumeer Gul. A Study on the Open Source Digital Library Software's: Special Reference to DSpace, EPrints and Greenstone. *CoRR*, abs/1212.4, 2012. URL `http://arxiv.org/abs/1212.4935`.

[12] DSpace. Installing DSpace - DSpace 6.x Documentation - DuraSpace Wiki, 2018. URL `https://wiki.duraspace.org/display/DSDOC6x/Installing+DSpace`.

[13] Nils Körber and Hussein Suleman. Usability of Digital Repository Software: A Study of DSpace Installation and Configuration. In *Digital Libraries: Universal and Ubiquitous Access to Information: 11th International Conference on Asian Digital Libraries, ICADL 2008, Bali, Indonesia, Dec 2-5, 2008. Proceedings*, pages 31–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-89533-6. doi: 10.1007/978-3-540-89533-6_4. URL `https://doi.org/10.1007/978-3-540-89533-6_4`.

[14] EPrints. Installing EPrints on Debian/Ubuntu – EPrints Documentation, 2018. URL `https://wiki.eprints.org/w/Installing_EPrints_on_Debian/Ubuntu`.

[15] Fedora. Installation and Configuration - Fedora 3.8 Documentation - DuraSpace Wiki, 2016. URL `https://wiki.duraspace.org/display/FEDORA38/Installation+and+Configuration`.

[16] Alicia Verno. IVDB ... for Free! Implementing an Open-Source Digital Repository in a Corporate Library. *Journal of Electronic Resources Librarianship*, 25(2):89–99, 2013. doi: 10.1080/1941126X.2013.785286. URL `http://dx.doi.org/10.1080/1941126X.2013.785286`.

[17] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. NIST Cloud Computing Reference Architecture. *NIST Special Publication*, 500(2011):1–28, 2011.

[18] Amazon. Overview of Amazon Web Services – Overview of Amazon Web Services, 2018. URL `https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/introduction.html`.

[19] Omar Sefraoui, Mohammed Aissaoui, and Mohsine Eleuldj. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*, 55(3):38–42, 2012.

[20] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The Eucalyptus Open-Source Cloud Computing System. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 124–131. IEEE, 2009. URL `http://ieeexplore.ieee.org/document/5071863/`.

[21] Borja Sotomayor, Rubén S. Montero, Ignacio M. Llorente, and Ian Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. *IEEE Internet Computing*, 13(5):14–22, Sep 2009. ISSN 1089-7801. doi: 10.1109/MIC.2009.119. URL `http://ieeexplore.ieee.org/document/5233608/`.

[22] VMware. What is a Hypervisor?, 2018. URL `https://www.vmware.com/topics/glossary/content/hypervisor`.

[23] David Freet, Rajeev Agrawal, Jessie J Walker, and Youakim Badr. Open Source Cloud Management Platforms and Hypervisor Technologies: A Review and Comparison. In *SoutheastCon 2016*, pages 1–8. IEEE, Mar 2016. ISBN 978-1-5090-2246-5. doi: 10.1109/SECON.2016.7506698. URL `http://ieeexplore.ieee.org/document/7506698/`.

[24] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 164–177. ACM, 2003.

[25] Fabrice Bellard. QEMU, A Fast and Portable Dynamic Translator. In *USENIX Annual Technical Conference, FREENIX Track*, volume 41, page 46, 2005.

[26] Amazon Web Services. Amazon EC2 FAQs - Nitro Hypervisor, 2018. URL `https://aws.amazon.com/ec2/faqs/#compute-optimized`.

[27] VMware. vSphere Hypervisor, 2019. URL `https://www.vmware.com/products/vsphere-hypervisor.html`.

[28] Amazon. AWS Management Console, 2019. URL `https://aws.amazon.com/console/`.

[29] Amazon Web Services. Amazon EC2, 2018. URL `https://aws.amazon.com/ec2/`.

[30] Amazon Web Services. Amazon Elastic Block Store (EBS) - Amazon Web Services, 2019. URL `https://aws.amazon.com/ebs/`.

[31] Amazon Web Services. Amazon Simple Storage Service, 2019. URL `https://aws.amazon.com/s3/`.

[32] Johnson D, Murari Kiran, Raju Murthy, Suseendran RB, and Girikumar Yogesh. Eucalyptus Beginner's Guide – UEC Edition, 2010. URL `http://cssoss.files.wordpress.com/2010/12/eucabookv2-0.pdf`.

[33] Gregor Von Laszewski, Javier Diaz, Fugang Wang, and Geoffrey C Fox. Comparison of Multiple Cloud Frameworks. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 734–741. IEEE, 2012.

[34] OpenStack. What is OpenStack?, 2019. URL `https://www.openstack.org/software/`.

[35] Stanley Lima, Álvaro Rocha, and Licinio Roque. An Overview of OpenStack Architecture: A Message Queuing Services Node. *Cluster Computing*, Jul 2017. ISSN 1573-7543. doi: 10.1007/s10586-017-1034-x. URL `https://doi.org/10.1007/s10586-017-1034-x`.

[36] OpenStack. OpenStack Docs: Hypervisors, 2018. URL `https://docs.openstack.org/ocata/config-reference/compute/hypervisors.html`.

[37] OpenStack. OpenStack Docs: Management Interfaces, 2019. URL `https://docs.openstack.org/security-guide/management/management-interfaces.html`.

[38] OpenNebula. Home - OpenNebula, 2019. URL `https://opennebula.org/`.

[39] Nimbus Project. About Nimbus – Nimbus, 2019. URL `http://www.nimbusproject.org/about/`.

[40] Peter Sempolinski and Douglas Thain. A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 417–426. IEEE, Nov 2010. ISBN 978-1-4244-9405-7. doi: 10.1109/CloudCom.2010.42. URL `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5708480`.

[41] Apache Software Foundation. Apache CloudStack: Open Source Cloud Computing, 2017. URL `https://cloudstack.apache.org/about.html`.

[42] Sonia Shahzadi, Muddesar Iqbal, Zia Ul Qayyum, and Tasos Dagiuklas. Infrastructure as a Service (IaaS): A Comparative Performance Analysis of Open-Source Cloud Platforms. In *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2017.

[43] Salman A. Baset. Open Source Cloud Technologies. In *Proceedings of the Third ACM Symposium on Cloud Computing - SoCC '12*, pages 1–2, New York, New York, USA, Oct 2012. ACM Press. ISBN 9781450317610. doi: 10.1145/2391229.2391257. URL `http://dl.acm.org/citation.cfm?id=2391229.2391257`.

[44] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. DevOps. *IEEE Software*, 33(3):94–100, 2016.

[45] Amazon Web Services. What is DevOps? - Amazon Web Services (AWS), 2019. URL `https://aws.amazon.com/devops/what-is-devops/`.

[46] Gideon Juve and Ewa Deelman. Wrangler: Virtual Cluster Provisioning for the Cloud. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 277–278, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0552-5. doi: 10.1145/1996130.1996173. URL `http://doi.acm.org/10.1145/1996130.1996173`.

[47] Ansible. How Ansible Works, 2019. URL `https://www.ansible.com/overview/how-ansible-works`.

[48] Puppet. Puppet 6 Documentation - Puppet (PE and Open Source) 6.1, 2019. URL `https://puppet.com/docs/puppet/6.1/puppet{_}index.html`.

[49] Chef. Automate IT Infrastructure, 2019. URL `https://www.chef.io/chef/`.

[50] Fabric. Fabric Documentation, 2018. URL `http://www.fabfile.org/`.

[51] Stephen Soltesz, Herbert Pötzl, Marc E Fiuczynski, Andy Bavier, and Larry Peterson. Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 275–287. ACM, 2007.

[52] Kubernetes. Kubernetes Reference - Standard Glossary, 2019. URL `https://kubernetes.io/docs/reference/glossary/?fundamental=true#term-container`.

[53] OpenVZ. OpenVZ - Container, 2019. URL `https://wiki.openvz.org/Container`.

[54] IBM. IBM - Containerization, 2019. URL `https://www.ibm.com/cloud/learn/containerization#toc-what-is-co-r25Smlqq`.

[55] Docker. What is a Container?, 2019. URL `https://www.docker.com/resources/what-container`.

[56] OpenStack. OpenStack and Containers, 2019. URL `https://www.openstack.org/containers/`.

[57] Google Cloud. Containers at Google, 2019. URL `https://cloud.google.com/containers/`.

[58] Amazon AWS. Containers on AWS, 2019. URL `aws.amazon.com/containers/`.

[59] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal*, 2014(239):2, 2014.

[60] Docker. Docker Overview, 2019. URL `https://docs.docker.com/engine/docker-overview/`.

[61] Kubernetes. Kubernetes, 2019. URL `https://kubernetes.io/`.

[62] Docker. Swarm Mode Overview, 2019. URL `https://docs.docker.com/engine/swarm/`.

[63] Linux Containers. What is LXC?, 2019. URL `https://linuxcontainers.org/lxc/introduction/`.

[64] CoreOS. rkt Overview, 2019. URL `https://coreos.com/rkt/`.

[65] OpenVZ. OpenVZ, 2019. URL `https://wiki.openvz.org/Main_Page`.

[66] Wu J, Teregowda P, Williams K, Khabsa M, Jordan D, Treece E, Wu Z, and Giles CL. Migrating a Digital Library to a Private Cloud. In *2014 IEEE International Conference on Cloud Engineering*, pages 97–106, Mar 2014. doi: 10.1109/IC2E.2014.77. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6903462`.

[67] Aljenaa E, Al-Anzi FS, and Alshayeji M. Towards an Efficient e-Learning System Based on Cloud Computing. In *Proceedings of the Second Kuwait Conference on e-Services and e-Systems*, KCESS '11, pages 13:1–13:7, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0793-2. doi: 10.1145/2107556.2107569. URL `http://doi.acm.org/10.1145/2107556.2107569`.

[68] Peter Nuernberg, John Leggett, and Mark McFarland. Cloud as Infrastructure at the Texas Digital Library. *Journal of Digital Information*, 13(1), 2012. ISSN 1368-7506. URL `http://journals.tdl.org/jodi/index.php/jodi/article/view/5881`.

[69] Ani Thakar and Alex Szalay. Migrating a (Large) Science Database to the Cloud. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, page 430, New York, New York, USA, Jun 2010. ACM Press. ISBN 9781605589428. doi: 10.1145/1851476.1851539. URL `http://dl.acm.org/citation.cfm?id=1851476.1851539`.

[70] DuraSpace. DuraCloud Guide, 2019. URL `https://wiki.duraspace.org/display/DURACLOUDDOC/DuraCloud+Guide#DuraCloudGuide-WhatisDuraCloud?`

[71] Simon Waddington, Jun Zhang, Gareth Knight, Mark Hedges, Jens Jensen, and Roger Downing. Kindura: Repository Services for Researchers Based on Hybrid Clouds. *Journal of Digital Information*, 13(1), 2012.

[72] Frank Doelitzscher, Anthony Sulistio, Christoph Reich, Hendrik Kuijs, and David Wolf. Private Cloud for Collaboration and e-Learning Services: From IaaS to SaaS. *Computing*, 91(1):23–42, Jan 2011. ISSN 1436-5057. doi: 10.1007/s00607-010-0106-z. URL `https://doi.org/10.1007/s00607-010-0106-z`.

[73] Django Foundation. Django Web Framework, 2019. URL `https://www.djangoproject.com/start/overview/`.

[74] Celery Project. Celery - Distributed Task Queue, 2019. URL `http://docs.celeryproject.org/en/latest/index.html`.

[75] International Organization for Standardization. ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability, 1998.

[76] Aaron Rich and Mick McGee. Expected Usability Magnitude Estimation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(5):912–916, 2004.

[77] John Brooke. SUS – A Quick and Dirty Usability Scale. *Usability Evaluation in Industry*, 189:194, 1996.

[78] Aaron Bangor, Philip T Kortum, and James T Miller. An Empirical Evaluation of the System Usability Scale. *Intl. Journal of Human–Computer Interaction*, 24(6): 574–594, 2008.

[79] James R Lewis. Psychometric Evaluation of an After-Scenario Questionnaire for Computer Usability Studies: The ASQ. *ACM SIGCHI Bulletin*, 23(1):78–81, 1991.

[80] Jeff Sauro and Joseph S Dumas. Comparison of Three One-Question, Post-Task Usability Questionnaires. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1599–1608. ACM, 2009.

[81] Donna Tedesco and Tom Tullis. A Comparison of Methods for Eliciting Post-Task Subjective Ratings in Usability Testing. *Usability Professionals Association (UPA)*, 2006:1–9, 2006.

Appendix A: Survey Questionnaire

Digital Libraries in Private Clouds - Usability Evaluation

Mushashu M. Lumpa
Digital Libraries Laboratory
Computer Science Department
University of Cape Town

Introduction

Thank you for accepting to take part in this evaluation exercise. This forms part of the evaluation for the research work in creating tools to manage Digital Repositories in Private Cloud environments.
Cloud Computing is a paradigm of computing that allows users to provision and acquire remote computing resources on demand, whose location is transparent to the users. Currently, the major public provider is Amazon AWS. Digital Repositories on the other hand, are applications that enable sharing, accessing and preservation of digital content. Installation of application in Cloud environments is similar to that in traditional environments, with the major difference being the physical machine's proximity to the user carrying out the installation.
Using the provided application, you will be asked to complete tasks that will result in the installation of a digital repository, which you will later be able to modify.

Prerequisite:

1. Use the credentials that were provided to you to access the Repository Manager at: sarabi.cs.uct.ac.za:8000/home
2. A default installation of DSpace looks like this: http://sarabi2.cs.uct.ac.za/xmlui/
3. An example of a customised installation is here: http://sarabi1.cs.uct.ac.za/xmlui/.
4. Please, ask any questions when you are not clear: emailto. mushashu@gmail.com/cell: 07 1568 9069.
5. IMPORTANT: This exercise is not an ASSESSMENT of your skills. Instead, you are ASSESSING the application. If something breaks, it is definitely not your fault.
6. Access the questionnaire from: http://goo.gl/pqngYa
7. When comfortable, please proceed to the instructions and attempt the listed tasks.

Instructions

Please follow closely the instructions below:

1. Before you begin with the tasks, fill in questions (1) through to (4).
2. **Task 1:** Install an instance of DSpace.
3. When installation is complete, view the instance of DSpace you just installed.
4. Fill in question (5), (6) , (7)and (8) in the provided questionnaire.
5. **Task 2:** Modify the DSpace installation you made in (2). **[Note: some logos you may wish to use find here: http://goo.gl/aMSmXl]. Note also that you should not strive for perfection with the modifications you will be making. The goal is to ascertain the usability of the customisation function.**
6. View your changes. [**On some browsers, reloading the page may be enough to notice the changes but in others you may have to clear your cache to notice them**].
7. Fill in question (9), (10), (11) and (12) in the provided questionnaire.
8. You are done with the tasks, you may proceed to answer the rest of the questions in the questionnaire.

Digital Libraries in Private Clouds - Survey

Dear Respondent,

Thank you for taking the time to participate in this study. This survey forms part of the evaluation for the research work in creating tools to manage Digital Repositories in Private Cloud environments. The overall objective is to measure the user experience of the developed Cloud Repository Management tool.

* Required

Participant Consent

This study has received approval from the Ethics in Research Committee of the Science Faculty at the University of Cape Town. Be advised that it is confidential and no identifying information will be kept along side your responses.

1. **1. I agree with the terms and hereby consent to participate in this study.** *

   Mark only one oval.

   ◯ Yes

   ◯ No

Background Information

2. **2. Have you ever installed and/or configured any of the following before?**
   Check all responses that apply.
   Check all that apply.

   - ☐ DSpace
   - ☐ EPrints
   - ☐ Drupal
   - ☐ Fedora
   - ☐ Tomcat
   - ☐ Postgres
   - ☐ Apache
   - ☐ MySQL
   - ☐ Other web based software
   - ☐ Other: ....

3. **3. Are you familiar with installing and managing of applications in cloud environments like Amazon Web Services?**
   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. **4. Name any applications you have installed and configured before?**
   Name as many as you can recall.

   ....

   ....

   ....

   ....

   ....

Installation - Experience

You must perform an installation of a repository as described in the instructions before proceeding to respond to questions in this section.

5. **5. Overall, I am satisfied with the ease of completing the tasks in this scenario**
   Mark only one oval.

	1	2	3	4	5	
Strong disagree	◯	◯	◯	◯	◯	Strongly agree

6. **6. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario**
   Mark only one oval.

	1	2	3	4	5	
Strong disagre	◯	◯	◯	◯	◯	Strong agree

7. **7. Please list what you liked about the installation process**

8. **8. Please list what you did NOT like about the installation process**

Customisation - Experience

You must first customise the repository you installed as described in the instructions given before proceeding to answer questions in this section.

9. **9. Overall, I am satisfied with the ease of completing the tasks in this scenario**
   Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strong agree

10. **10. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario**
    Mark only one oval.

	1	2	3	4	5	
Strong disagree	◯	◯	◯	◯	◯	Strong agree

11. **11. Please list what you liked about the customisation process**

12. **12. Please list what you did NOT like about the customisation process**

Overall Application Usability

This section is based on the System Usability Scale. You will be asked to respond to questions which you will rate between 1 and 5, with 1 being strongly disagree and 5 strongly agree.

13. **13. Would you use this application frequently?**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strong agree

14. **14. I found this system unnecessarily complex**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strong agree

15. **15. I thought the system was easy to use**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strong agree

16. **16. I think I would need support to be able to use this application**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

17. **17. I found the various functions in this application well integrated**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

18. **18. I thought there was too much inconsistency in this application**
Mark only one oval.

	1	2	3	4	5	
Strong disagree	⬭	⬭	⬭	⬭	⬭	Strong agree

19. **19. I would imagine that most people would learn to use this application very quickly**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

20. **20. I found the application very cumbersome to use**
Mark only one oval.

	1	2	3	4	5	
Strong disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

21. **21. I felt very confident using the application**
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

22. **22. I needed to learn a lot of things before I could get going with this application**
Mark only one oval.

	1	2	3	4	5	
Strong disagree	⬭	⬭	⬭	⬭	⬭	Strong agree

23. **23. What did you like about the application?**

....
....
....
....
....

24. **24. What did you NOT like about the application?**

....
....
....
....
....

Appendix B: Survey Questionnaire Results

1 Participants and their Categories

Table 1: Participants and Their Assigned Categories

ID	Have you ever installed and/or configured any of the following before?	Level of Familiarity (Participant Category)
22	n/a	Non-expert
1	linux mint;	Non-expert
2	mysql	intermediate
3	mysql	intermediate
4	wordpress	intermediate
5	fedora;mysql	expert
6	n/a	Non-expert
7	fedora; tomcat; apache; mysql; php; node.js	expert
8	ubuntu os	intermediate
9	ms windows xp; ms office; vlc media player; advanced system care	intermediate
10	apache; mysql	expert
11	n/a	Non-expert
12	apache	intermediate
13	apache; mysql	expert
14	tomcat; mysql; wordpress	expert
15	eprints	expert
16	apache; mysql	expert
17	Apache, MySQL, Joomla	expert
18	n/a	Non-expert

19	DSpace, Tomcat, Postgres, MySQL	expert
20	Tomcat, Apache, MySQL, Other web based software	expert
21	n/a	Non-expert

In the tables that follow, the rows represent the question numbers while the columns represent individual participants.

2 Results – After Scenario Questionnaire

Table 2: After Scenario Questionnaire - Installation

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22
Q 1	5	4	5	5	4	5	4	5	5	4	2	3	4	3	3	5	4	5	5	5	5	4
Q 2	5	5	3	3	2	4	3	4	4	1	1	2	3	2	4	3	4	5	5	5	5	3

p(number): participant

Q(number): question

Table 3: After Scenario Questionnaire - Customisation

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22
Q 1	4	5	5	3	2	5	3	4	3	5	4	4	4	5	3	5	4	3	4	5	4	3
Q 2	4	5	3	5	3	5	3	5	3	4	4	3	4	5	4	5	4	3	4	5	5	3

p(number): participant

Q(number): question

Results – System Usability Scale

Table 4: System Usability Scale Questionnaire Results - Positive Questions

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22
Q 1	3	3	2	2	3	3	3	3	3	3	4	4	3	1	3	5	3	3	3	4	5	4
Q 3	5	5	5	4	4	5	4	5	4	2	5	2	4	2	4	5	5	5	4	5	4	4
Q 5	4	5	4	3	2	4	4	3	3	4	4	3	3	2	4	4	5	3	4	4	4	4
Q 7	5	5	5	4	4	5	4	5	5	1	5	3	2	1	4	5	5	5	5	5	5	4
Q 9	4	5	4	3	4	5	2	4	3	1	3	2	4	1	4	5	4	5	4	5	4	4

p(number): participant

Q(number): question

Table 5: System Usability Scale Questionnaire Results - Negative Questions

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22
Q 2	1	1	1	2	2	1	2	1	1	2	1	3	2	3	2	1	2	1	1	1	1	2
Q 4	2	1	2	2	2	1	4	1	1	4	1	4	2	5	2	1	3	1	3	1	1	2
Q 6	2	1	2	2	2	1	3	1	3	5	2	3	2	3	3	1	1	1	2	1	1	2
Q 8	1	1	1	2	2	1	3	1	2	2	1	4	3	5	2	1	1	1	2	1	1	2
Q 10	1	1	2	2	1	1	4	1	1	3	2	2	2	5	1	1	1	1	4	1	1	1

p(number): participant

Q(number): question

Table 6: After Scenario Questionnaire - Installation Task General Comments

	Please list what you liked about the installation process	**Please list what you did NOT like about the installation process**
p1	It was easy to do and quick.	I was not quite sure about what the information I provided (name, email, etc.) would be used for.
p2	it was easy to create an instance of dpsace, however, it was initially unclear as to what each field required	it was unclear as to what each field required
p3	user is informed of current progress; simple clean interface	
p4	one button creates instance; creating repository is simple and clean	time taken to create instance
p5	creating a new instance is pretty straight forward and directions are clear	the installation tool had to be run twice; installation seemed to take a rather long time and the feedback was not entirely specific (percent figures) of how long was left or if it was progressing or hanging (stuck / frozen)
p6	easy and quick(no need to fill in lots of information); good interface	maybe the possibility to view in a bit more detail what is going on would be useful
p7		confusing to someone who has not worked with these sorts of things before
p8	very few details needed; clean install; progress bar to show install progress	no detail given on what the installation is busy with; not ETA on install time
p9	very simple and required no experience; very clean and simple UI	once installed, it wasn't obvious how to run the application
p10	simple interface	took too long to install; no instructions on the screen to tell me what to do next
p11	one click installation was very simple	installation took a very long time(over half an hour)
p12	the drop list menu to go to an action; the ease to create a instance	the time it took to install an instance; the amount of space available in the cloud
p13	simple ui; few actions required to install	labels and progress could be a little clearer; not sure how long it normally takes but installation seemed slow

p14	it was actually pleasantly simple; the form that we had to fill in was so long	it took too long, it was too slow
p15	its simplicity	
p16	its simplicity	I like everything
p17	simple; clear; direct	
p18	It's quick, and straight forward	
p19	Very easy!	
p20	There was minimal required information from the user to install a Dspace instance. The automation of the process makes it simpler for an end-user. The instructions were adequate to complete installation without requiring help	
p21	There big visible button to start the installation made it very easy to know where to start	It was rather long (this was expected, of course), but a nicer thing would have been to warn me that the process may take a few minutes..to some peope, when a process takes long, they take it to mean the thing is broken/non-responsive.. On the + side, the progress bar is useful, but a message saying the installation will begin and may take a few minutes could be useful
p22	It was simple and user-friendly.	It took a little longer than I expected.