Investigating the feasibility of digital repositories in private clouds

Mushashu Lumpa and Hussein Suleman[0000−0002−4196−1444]

University of Cape Town, South Africa
{mushashu@gmail.com,hussein@cs.uct.ac.za}
http://dl.cs.uct.ac.za/

Abstract. Installing and configuring a digital repository toolkit for an organisation is a non-trivial task for which many organisations now seek external third-party service providers. Some of these service providers offer a cloud-hosted environment. However, universities increasingly have such cloud infrastructure in-house to support internal systems, and to maintain control and custody of data and systems. This study investigated the feasibility of using a private cloud internal to an organisation for the management of digital repositories. 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. A usability study of the prototype tool received positive feedback. Participants in the study were able to install and customise their own DSpace repositories.

Keywords: private clouds · digital repositories · institutional repositories.

1 Introduction

Cloud computing can be described it as a model of enabling on-demand network access to a shared pool of computing resources, which can be provisioned and released with minimal management effort [26]. Cloud computing has promised cost-effectiveness, flexibility and scalability for software and hardware requirements of organisations [19].

Digital repositories are software tools that are used to manage digital content, share it and provide the means for potential long-term preservation of that content. Digital repository tools are used extensively by libraries and institutions like universities that continually generate original 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 could 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 internal management features of a cloud environment coupled with the efficient usage of their computing resources. Institutions' digital repository 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. One option is to adopt a service such as that provided by Bepress or Duraspace, where third party and public cloud infrastructure is used as the basis for digital repositories. Private clouds, based on equipment hosted completely within an organisation, are an alternative as institutions increasingly adopt this technology for local infrastructure; arguably, this could serve as the platform for the institution's own digital repositories as well.

This study therefore explores the feasibility of using private clouds for hosting of digital repositories. The key questions asked in this study focus on: the efficiency of the current generation of private cloud tools coupled with the current digital repository tools; and the feasibility of managing repository management at a high level on top of low-level private cloud tools.

2 Literature Review

2.1 Digital Repository Toolkits

There are different types of digital repository software toolkits available, and some of the commonly [6] used ones include DSpace [12][28], EPrints [15] and Fedora [23]. These repository tools share similar features, and to some extent are developed using the same technologies. A number of studies [6][21][1][31] have been done to compare the different features that each of these provide.

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 [13] requires running commands that may require users to have technical skills [22]. Installation of DSpace is not trivial and often requires multiples tries and some amount of time 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 [16][17]. They both require one to have some technical skills to run the commands that are shared in their installation documentation.

Like installation and configuration, 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 aesthetics of the repository to enhance its appeal. Verno [32] documented Boston Biomedical Consultants, Inc's experience in implementing a DSpace instance. As much as it was successful, the difficulties with the installation, configuration and customisation of DSpace were highlighted. These complexities need to be addressed in any cloud implementation.

2.2 IaaS and Eucalyptus

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 [3], OpenStack [27], Eucalyptus [25], and OpenNebula [29].

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 [33][18]. Hypervisors are software programs that enable operating systems to host one or more virtual machines. Examples of hypervisors include Xen [7], KVM [8], Nitro [4], and vSphere [34].

For this study, Eucalyptus was used as the IaaS platform. 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 [25]. Eucalyptus was developed to have interfaces that use similar commands as Amazon's very popular EC2 system. Eucalyptus comprises of 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) [20].

2.3 Repositories in Clouds

Wu, et al. described their work in migrating CiteSeerX into a private cloud environment [36]. Long terms costs, compared to migrating to a public cloud, were one of the reasons for their motivation to set up a private cloud infrastructure to host their digital repository. Their custom setup utilises proprietary software

- VMware ESXi - for the hypervisor and VMware VSphere for instance provisioning and general cloud orchestration and monitoring. Aljenaa, et al. have explored the possibility of hosting e-learning systems in cloud based environments [2]. 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.

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 [24]. 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, albeit public, infrastructure. Thakar, et al. [30] described their experience migrating the Sloan Digital Sky Survey science archive, which has a 5TB database. They describe their experience as frustrating based on two reasons: degraded performance of the queries run off the database compared to their in-house infrastructure; and their inability to transfer the entire 5TB to the cloud. Their experience suggests that performance is a factor in assessing feasibility.

Others have looked into content preservation across cloud environments. DuraCloud [14], 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 [35] is another project that is encouraged by the possibilities of content preservation using cloud storage systems. Both DuraCloud and Kindura are driven by long term content preservation needs and public cloud infrastructure, but suggest that repository management within a cloud can be feasible.

3 Methodology

3.1 System Development and Deployment

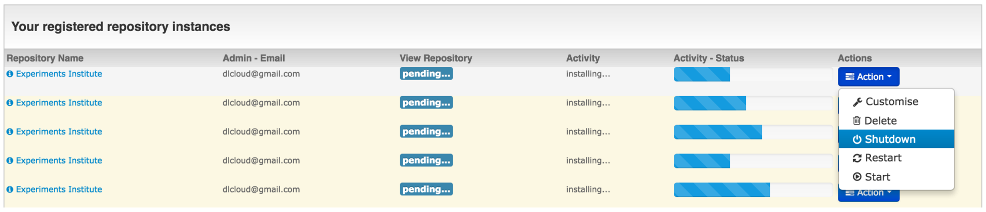
This study involved hosting digital repositories in a private cloud computing environment. The process involved simplifying the installation/configuration process of DSpace and automating its installation in a virtual machine instance.

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

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 v1.8. Installation and configuration of DSpace was achieved through use of an orchestration and configuration management Python API called Fabric. The repository management tool was developed using the Django Web framework [11]. Django Celery [10] formed a core part of the overall solution: it enabled execution of tasks (e.g. installation and customisation of repositories) in an asynchronous manner.

3.2 User Front End

The process flow begins with a user logging onto the prototype application that was developed called *Lilu*, using a username and password. Figure 1 shows what is presented to a user or system administrator once they have logged in, as well as the functions that can be carried out on existing installations. At this point, the user/system administrator are able to carry out a DSpace installation or manage already existing repositories.



Your registered repository instances

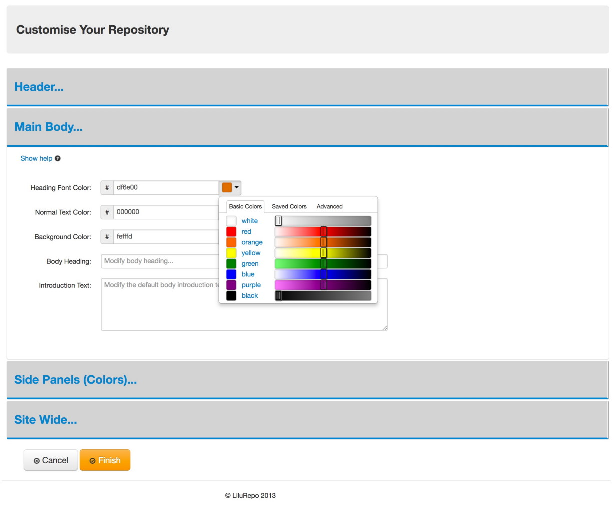
Repository Name	Admin - Email	View Repository	Activity	Activity - Status	Actions
Experiments Institute	dlcloud@gmail.com	pending...	installing...		Action
Experiments Institute	dlcloud@gmail.com	pending...	installing...		
Experiments Institute	dlcloud@gmail.com	pending...	installing...		
Experiments Institute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

Fig. 1. Features on a repository once it is installing or it has fully installed

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 administrator credentials. The needed information is provided via a Web form. 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.

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 1 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 2 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.



Heading Font Color: # df6e00

Normal Text Color: # 000000

Background Color: # fefffd

Body Heading: Modify body heading...

Introduction Text: Modify the default body introduction te

Fig. 2. Repository customisation page

3.3 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*[1] [20]. euca2ools provides an abstraction and a set of programmable functions that enable administrative management of Eucalyptus cloud services. Some of the euca2ools functions utilised in this experimental solution are described below:

1. **Start (run) virtual machine instances -** this is achieved by executing the *euca-run-instance* from the commandline 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 commandline function associated with this function is *euca-terminate-instances*, with the equivalent

[1] euca2ools, https://wiki.debian.org/euca2ools

python API call being *terminate_instances*. This call destructs the running virtual machine that is being terminated. 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 maintains 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 euca2ools 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:

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 Web server. 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 Web server and Java Container. The rerouting of external calls via the front-facing Apache server to the Tomcat server is achieved by using an Apache module called *mod_alias*[2] - 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 DSpace. 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[3] 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 Web servers (i.e. Tomcat and Apache) 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 front-end 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.

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

3.4 Evaluation

Two types of evaluations were carried out: 1) performance measurement to assess impact of virtual environment on typical activities; and 2) usability study of the developed browser-based application for installation and customisation of DSpace.

Performance Experiment The objective of the performance experiment was to measure ingestion and viewing/downloading 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.

The hardware and specifications used for this test are listed in Table 1.

Table 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

Apache JMeter[4] was used to simulate repository user actions and also take response time measurements.

The experiment began by running a number of virtual machine instances in the cloud environment. Once the virtual machine instances had fully booted, DSpace was installed in each one of the instances, following the steps outlined previously. Using JMeter, 15 items were 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.

Each run had a different number of instances running in the cloud - 1, 2, 5, 8 and 11 - and every run had 5 replications to establish a stable average measurement.

Figure 3 has a detailed breakdown of the average ingestion times by instance and also ingestion order of the item.

[4] https://jmeter.apache.org/

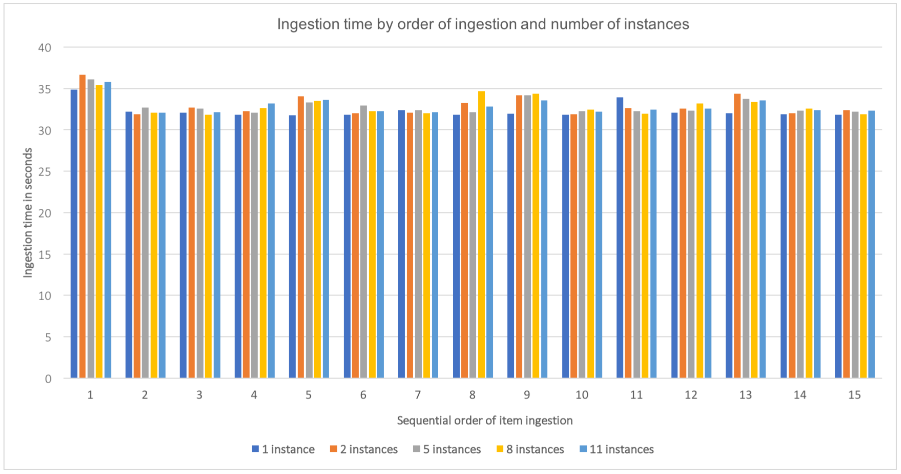


Fig. 3. Overall average ingestion time by order of ingestion and number of instances

It can be deduced from the data 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 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.

The initial ingestion time into DSpace is marginally higher than the rest of the ingestions. This can be attributed to Apache server on the central cloud server making its first 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.

The average view and download response times are shown in Figure 4.

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 instances, an F-statistic of 72.57 and p-value of 2.23 is obtained. Since $p>0.05$, it cannot be said that there is a significant difference in the view times experienced.

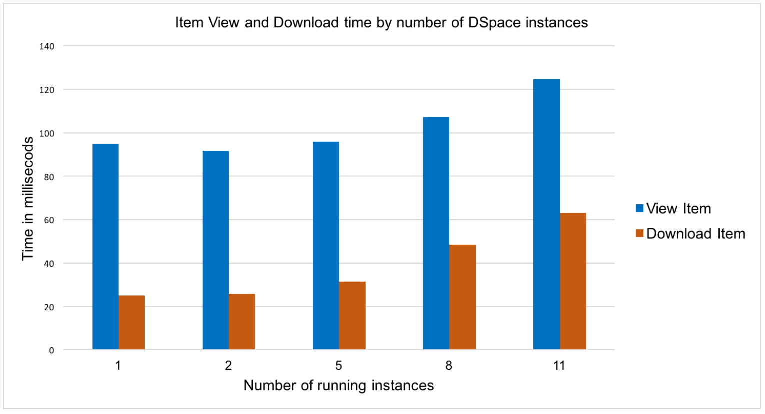


Fig. 4. Overall average item view and download time by number of instances

Usability Evaluation The System Usability Scale (SUS) [9] was adopted for this study. SUS is widely used[5], has respectable reliability [5] and is available for free use without a licence. Users were also asked to comment on satisfaction and provide any additional comments.

The evaluation exercise involved completion of 2 main tasks that were the core features of the developed prototype: installation and customisation of a digital repository.

Twenty two (22) postgraduate student participants were recruited for this study. The minimum requirements for participants were that they should be familiar with Web technologies and be everyday users of the Internet. Based on prior experience with installation of complex software applications, participants were identified to fall into 3 categories: non-expert, intermediate and expert.

The overall average SUS score from the evaluation was 74. SUS is scored out of 100, and the global average from the SUS project is stated as 68. 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 scored 70.

Responses to the question, *What did you like about the application?* and *What did you NOT like about the application?* were grouped and analysed by common themes. There was a total of 39 unique positive comments and 27 unique negative comments. "Ease of use", "ease of learning" and "a clean interface" were the most mentioned positive comments. "Installation time", "responsiveness", "system help" and "minor bugs" were the most mentioned negative comments.

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

[5] Measuring Usability with the System Usability Scale,https://www.measuringusability.com/sus.php

repositories. One option 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.

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.

4 Conclusions and Future Work

This study set out to investigate the feasibility of hosting a digital repository system in a managed way in a private cloud environment. The system that was developed was functional, usable and had an acceptable level of system performance. Thus, a private cloud is indeed a feasible option for such software systems. Current interest in containerisation of software tools, through mechanisms such as Docker, are highly compatible with cloud management systems, as they each address a different aspect of the ease of management equation.

During the course of this research, no substantive changes were made to the DSpace repository toolkit. Thus, the move to a managed repository ecosystem, whether in a private cloud or elsewhere, can likely treat existing tools as black boxes to be encapsulated and managed at a higher level.

This form of turnkey management will ease the burden on administrators, many of whom are repository managers rather than systems administrators, and enable the adoption of repositories more easily than is the case at present.

Future work can investigate hybrid models that include replication and scalability, as well as managed containerisation of future back-end systems.

Acknowledgements

This research was partially funded by the National Research Foundation of South Africa (Grant numbers: 85470 and 105862) and University of Cape Town. The authors acknowledge that opinions, findings and conclusions or recommendations expressed in this publication are that of the authors, and that the NRF accepts no liability whatsoever in this regard.

References

1. Adewumi, A.O., Omoregbe, N.A.: Institutional repositories: Features, architecture, design and implementation technologies. Journal of Computing **2**(8) (2011)
2. Aljenaa, E., Al-Anzi, F.S., 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. pp. 13:1—-13:7. KCESS '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/2107556.2107569, http://doi.acm.org/10.1145/2107556.2107569

3. Amazon: Overview of Amazon Web Services - Overview of Amazon Web Services (2018), https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/introduction.html
4. Amazon Web Services: Amazon EC2 FAQs - Nitro Hypervisor (2018), https://aws.amazon.com/ec2/faqs/#compute-optimized
5. Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the system usability scale. Intl. Journal of Human–Computer Interaction **24**(6), 574–594 (2008)
6. Bankier, J., Gleason, G.: Institutional Repository software comparison, vol. 33. United Nations Educational, Scientific and Cultural Organization (2014), http://works.bepress.com/jean_gabriel_bankier/22/
7. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: ACM SIGOPS operating systems review. vol. 37, pp. 164–177. ACM (2003)
8. Bellard, F.: Qemu, a fast and portable dynamic translator. In: USENIX Annual Technical Conference, FREENIX Track. vol. 41, p. 46 (2005)
9. Brooke, J.: SUS-A quick and dirty usability scale. Usability evaluation in industry **189**, 194 (1996)
10. Celery Project: Celery - Distributed Task Queue (2019), http://docs.celeryproject.org/en/latest/index.html
11. Django Foundation: Django Web Framework (2019), https://www.djangoproject.com/start/overview/
12. DSpace: DSpace - A Turnkey Instutitional Repository Application (2018), https://duraspace.org/dspace/
13. DSpace: Installing DSpace - DSpace 6.x Documentation - DuraSpace Wiki (2018), https://wiki.duraspace.org/display/DSDOC6x/Installing+DSpace
14. DuraSpace: DuraCloud Guide (2019), https://wiki.duraspace.org/display/DURACLOUDDOC/DuraCloud+Guide#DuraCloudGuide-WhatisDuraCloud?
15. EPrints: EPrints Services (2018), http://www.eprints.org/uk/
16. EPrints: Installing EPrints on Debian/Ubuntu - EPrints Documentation (2018), https://wiki.eprints.org/w/Installing_EPrints_on_Debian/Ubuntu
17. Fedora: Installation and Configuration - Fedora 3.8 Documentation - DuraSpace Wiki (2016), https://wiki.duraspace.org/display/FEDORA38/Installation+and+Configuration
18. Freet, D., Agrawal, R., Walker, J.J., Badr, Y.: Open source cloud management platforms and hypervisor technologies: A review and comparison. In: SoutheastCon 2016. pp. 1–8. IEEE (mar 2016). https://doi.org/10.1109/SECON.2016.7506698, http://ieeexplore.ieee.org/document/7506698/
19. Han, Y.: On the clouds: A new way of computing. Information Technology and Libraries **29**(2), 87–92 (2010), https://search.proquest.com/docview/325033464/fulltextPDF/A52FC66FB8DE4D9BPQ/1?accountid=14500
20. Johnson, D., Kiran, M., Murthy, R., Suseendran, R., Yogesh, G.: Eucalyptus Beginner ' s Guide UEC Edition Eucalyptus Beginner ' s Guide - UEC Edition. Eucalyptus, v2.0 edn. (2010), http://cssoss.files.wordpress.com/2010/12/eucabookv2-0.pdf
21. Kökörčený, M., Bodnárová, A.: Comparison of Digital Libraries Systems. In: Proceedings of the 9th WSEAS International Conference on Data Networks, Communications, Computers. pp. 97–100. DNCOCO'10, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2010), http://dl.acm.org/citation.cfm?id=1948805.1948823

22. Körber, N., Suleman, H.: Usability of Digital Repository Software: A Study of DSpace Installation and Configuration. In: Buchanan, G., Masoodian, M., Cunningham, S.J. (eds.) Digital Libraries: Universal and Ubiquitous Access to Information: 11th International Conference on Asian Digital Libraries, ICADL 2008, Bali, Indonesia, December 2-5, 2008. Proceedings, pp. 31–40. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89533-6_4, https://doi.org/10.1007/978-3-540-89533-6_4
23. Lagoze, C., Payette, S., Shin, E., Wilper, C.: Fedora: an architecture for complex objects and their relationships. International Journal on Digital Libraries **6**(2), 124–138 (apr 2006). https://doi.org/10.1007/s00799-005-0130-3, http://link.springer.com/10.1007/s00799-005-0130-3
24. Nuernberg, P., Leggett, J., McFarland, M.: Cloud as Infrastructure at the Texas Digital Library. Journal of Digital Information **13**(1) (2012), http://journals.tdl.org/jodi/index.php/jodi/article/view/5881
25. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on. pp. 124–131. IEEE (2009), http://ieeexplore.ieee.org/document/5071863/
26. Peter, M., Timothy, G.: The NIST Definition of Cloud Computing (2011), http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
27. Sefraoui, O., Aissaoui, M., Eleuldj, M.: Openstack: Toward an open-source solution for cloud computing. International Journal of Computer Applications **55**(3), 38–42 (2012)
28. Smith, M., Barton, M., Branschofsky, M., McClellan, G., Walker, J.H., Bass, M., Stuve, D., Tansley, R.: DSpace - An Open Source Dynamic Digital Repository. D-Lib Magazine **9**(1) (jan 2003). https://doi.org/10.1045/january2003-smith, http://www.dlib.org/dlib/january03/smith/01smith.html
29. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Virtual Infrastructure Management in Private and Hybrid Clouds. IEEE Internet Computing **13**(5), 14–22 (sep 2009). https://doi.org/10.1109/MIC.2009.119, http://ieeexplore.ieee.org/document/5233608/
30. Thakar, A., Szalay, A.: Migrating a (large) science database to the cloud. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10. p. 430. ACM Press, New York, New York, USA (jun 2010). https://doi.org/10.1145/1851476.1851539, http://dl.acm.org/citation.cfm?id=1851476.1851539
31. Tramboo, S., Humma, Shafi, S.M., Gul, S.: A Study on the Open Source Digital Library Software's: Special Reference to DSpace, EPrints and Greenstone. CoRR **abs/1212.4** (2012), http://arxiv.org/abs/1212.4935
32. Verno, A.: IVDB . . . for Free! Implementing an Open-Source Digital Repository in a Corporate Library. Journal of Electronic Resources Librarianship **25**(2), 89–99 (2013). https://doi.org/10.1080/1941126X.2013.785286, http://dx.doi.org/10.1080/1941126X.2013.785286
33. Vmware: What is a hypervisor? (2018), https://www.vmware.com/topics/glossary/content/hypervisor
34. VMware: vSphere Hypervisor (2019), https://www.vmware.com/products/vsphere-hypervisor.html
35. Waddington, S., Zhang, J., Knight, G., Hedges, M., Jensen, J., Downing, R.: Kindura: Repository services for researchers based on hybrid clouds. Journal of digital Information **13**(1) (2012)

36. Wu, J., Teregowda, P., Williams, K., Khabsa, M., Jordan, D., Treece, E., Wu, Z., Giles, C.L.: Migrating a Digital Library to a Private Cloud. In: 2014 IEEE International Conference on Cloud Engineering. pp. 97–106 (mar 2014). https://doi.org/10.1109/IC2E.2014.77, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6903462