

JISC



JISC Final Report

Project Information			
Project Acronym	Fedorazon = Fedora Commons + Amazon Web Services		
Project Title	Fedorazon - Cloud Repository		
Start Date	October 2007	End Date	October 2008
Lead Institution	Birkbeck College		
Project Director	Philip Payne		
Project Manager & contact details	David F. Flanders twitter: twitter.com/dfflanders blog: dfflanders.wordpress.com email: d.flanders@bloomsbury.ac.uk mobile: +44 (0)7852581975 skype: david.flanders		
Partner Institutions	The Bloomsbury Colleges		
Project Web URL	http://www.ukoln.ac.uk/repositories/digirep/index/Fedorazon		
Programme Name	Start-Up and Enhancements		
Programme Manager	Andrew McGreggor		
Document Name			
Document Title	Fedorazon Final Report		
Author(s) & project role	David F. Flanders, Project Manager		
Date	31 October 2008	Filename	FinalReport_Fedorazon_JISC
URL	http://docs.google.com/Doc?id=dfccf5n6_140c9xkjmht		

Document History		
Version	Date	Comments
1.0	2008-10-31	licensed as a Creative Commons Attribution-Sharealike work

Table of Contents

1. [Glossary](#)
2. [Executive Summary](#)
3. [Background](#)
4. [Aims and Objectives](#)
5. [Methodology](#)
6. [Implementation](#)
7. [Outputs and Results](#)
8. [Outcomes](#)
9. [Conclusions](#)
10. [Implications](#)
11. [Recommendations](#)
12. [References](#)

Acknowledgements

Fedorazon is a [JISC](#) funded project (as part of the [Start-Up and Enhancements](#)) Capital Funding.

'Shout-outs' go to Mark Hedges and Andreas Aschenbrenner who have been watching the gathering Cloud coming for quite some time as part of the Grid community. Ben O'Steen who is the guru of repository storage nirvana. My host institution (Birkbeck College) and line managers Philip Payne and Tim Fletcher for letting me pursue innovation in the library. Last but not least, Andrew McGregor for being there throughout as a supportive programme manager.

Glossary and Explanation of the Cloud

Please take the time to quickly go over the various terms being used for "the Cloud" and its various parts:

- Please read the Fedorazon post on "[a glossary of terms for 'the Cloud'](#)" <-- essential for this report.
- Terms specific to the use of Amazon Web Services are explained on the Fedorazon website, [here](#).
-
- Please note: all other terms used in this report should be findable on the web within [Wikipedia for 2008-11-04](#).

Executive Summary

The Fedorazon project is first and foremost the experiences of a small HE/FE team running and maintaining a Repository in the Cloud for one year. Being early adopters

we provide both technical, fiscal and practical advice for both our successes and failures in this endeavour. We hope this report provides insight for other institutions wishing to utilise the Cloud for their Repository instance which we wholeheartedly recommend given they read this report first and prepare accordingly.

The Fedorazon project has discovered that a 'Repository in the Cloud' is easy to get up and running (both figuratively and literally); after that, all the complexity of hardware management, political costings and human resource allocation are still right where you left them. None the less we think there are significant cost savings in the Cloud that will only increase over time. We also believe that utilising the 'network effect' of the Cloud institutions can relieve the burden of having a local hardware expert to manage the repository instance. Finally, we believe that Cloud will lead to a significant change in the way we view repository architectures, especially in regards to how a 'preservation architecture' is achieved.

We would also encourage those of you who prefer having a play to discover things out for themselves to proceed right to the [website](#) where they can launch their own version of a Cloud Repository in far less time than it will take you to read this report.

Background

The Fedorazon project was an early adopter of cloud computing for repository software i.e. rather than hosting our own instance of a repository on a local server (e.g. in one of the library closets) we installed our repository on a "Cloud" server i.e. we used the rentable servers from Amazon to host the computing stack (EC2) and the storage components (S3).

The projects roots began back in 2004, with the realisation that to enhance repository development we needed a quick and easy way to launch a repository so testing could be done with minimal effort, i.e. spend as little amount of money and time as possible to install and run a clean repository instance. After we were finished testing our new developments we wanted to get rid of the instance and start anew. To do this kind of development with multiple repositories was not easily accomplished if we were to buy our own servers and utilise them accordingly. First the overhead cost of buying the servers was not conducive to this temporary developmental testing, and second it was not to our advantage to buy several servers when we had no idea how many we would need or to what effect. Also the time it takes to select a server and have it delivered is far too long to wait if a development idea is ready to go. There were other options around for this kind of testing environment (i.e. the National Grid), however to do development on Grid servers required an application process as well as configuration of certain environment variables that were not always conducive to the repository software we wanted to install. The cloud provided us with servers that we could turn on immediately (literally with the click of a button, thanks to Elastifox), and since our

installation image could be copy and pasted to any other cloud server we thought it a good idea to make this available to the wider world so they too could launch a repository with a button click (or two) all for themselves.

Amazon in launching their Web Services company branch (AWS) in 2002 enabled anyone with a credit card to sign up so they could rent Amazon servers. This was possible by Amazon because of what is known as 'virtualisation' (that and they have a lot of extra servers sitting around just in case they need more computing capabilities, e.g. when everyone at Christmas wants to buy the latest Harry Potter book). Coincidentally AWS uses OpenSource virtualisation software known as Xen which was originally created by the University of Cambridge's Computer Laboratory. AWS currently has thirteen different services you can purchase from them for various computation processes, however the primary one that enabled Fedorazon to share its blank copy of repository software with everyone else was the service known as the Elastic Computing Cloud (EC2). EC2 is essentially a blank server that can be rented and used however you like, you are charged per hour that you use EC2 along with number of times that information is accessed via your rented server instance. In this way, many people compare this kind of computer rental to the current model we have for electricity, where we use the amount we need and pay for what we use at the end of the month; this includes high demand times when more than one computer processor may be required for processing large amounts of data (something we as a small development team could have never afforded on our own server budget). However for Fedorazon the primary significance of having a centralised "computing-power plant" is that certain standards can be declared by the central company -in this case Amazon- on how servers can be commonly configured for use. For example, in configuring the Fedora repository software for use on EC2 we declared a specific "stack" (see ['Glossary'](#) above) for the environment set-up variables. Once we had set-up this stack (and all of the customisation variables) we were then able to publish the public version of the stack that anyone else could immediately install on their own instance of EC2; this 'public stack' is known as an 'Amazon Machine Image' (AMI) and is available to anyone using AWS.

The Fedorazon hope (and project aim) of having this public stack or AMI, was that any institution wishing to have a repository could start one up with no more effort than turning the key of a car (and of course setting up a credit card to use it). While the project accomplished this ease of 'starting a repository' definitively, it did not take into account the concerns that the repository community would have for cloud computing at the time of the project. In short, the repository community was primarily concerned with the physical location of their data; and rightfully so as some Gov't funded projects require storage of data on European soil. This concern was partially met when AWS released regional data centres that would guarantee storage of data in "the European region". Yet, as repositories are primarily based in the library sector it was still the

digital bytes and their physical location that concerned repository managers. Recently, Amazon has offered up location based Cloud computing, which means an automatic copy of the data can be made and placed within separate servers on opposite sides of the planet which provides a kind of disaster recovery mechanism that a single library could never accomplish on its own.

Yet despite multiple cloud computing services and business models assuring various service levels, the discussion of 'the Cloud' as an architecture of preservation continues to be worked on by several other projects (see ['Other Projects'](#) section below). For Fedorazon, the crux of the argument comes down to the concern by the library sector for a business corporation like Amazon as a plausible entity for assuring preservation (especially during economic instability). Because of this concern Fedorazon (and other cloud services) remain primarily a tool for testing what the cloud has to offer, however there is good reason to believe that the early adoption by Fedorazon of the Cloud will lead to further business models, new preservation strategies and cost calculations that will justify the Cloud as the foremost method for hosting and administering a Repository (see ['Outcomes'](#) section below). Perhaps Fedorazon's most substantial knock on effect is its support of a small-medium company that will offer hosting and support of Fedora based on the Fedorazon AMI (PaaS). It is hoped that this will be the first of many companies/institutions that will be able to utilise the Open Source (BSD) code to launch their own consultancy service (SaaS) around the repository community thereby making a more robust market and service industry.

In summary, being an early adopter of Cloud 'technology' required a great deal more 'non-technological' advice to the sector in what the Cloud has to offer the repository community; furthermore, we believe the sector will require a great deal more advice as additional repository specific cloud technologies are made available.

Aims and Objectives

The original Aim of the Fedorazon project was to enable institutions to "launch their repository service in the same day they decide to have one, and without hiring a 'hardware' expert". And true to our word we achieved this aim. Unfortunately, 'turning a repository on' is really just turning the key to start the engine: you still need to know how to drive a repository (both technically and politically)! Otherwise, all of the original project objectives have been met; see links below for payload of each objective and see the ['Outputs and Results'](#) section for explanation for how these payloads were achieved.

Objectives:

i.) Provide OS code and "how-to" documentation for using Fedora with EC2, S3 and other services

- http://www.ukoln.ac.uk/repositories/digirep/index/Fedorazon_How_to_Guides

ii.) Encourage and disseminate the use of Fedora on AWS to JISC projects and communities

- <http://www.dlib.org/dlib/november08/aschenbrenner/11aschenbrenner.html>

iii.) Provide training on how to deploy Fedora as a component of other services

- http://www.ukoln.ac.uk/repositories/digirep/index/Fedorazon_Training

iv.) Support any and all project wishing to deploy repository services on EC2 and S3

- <http://mediashelf.co.uk/hosting>

Methodology

The Fedorazon project itself was naturally incorporated into the [JISC-SOURCE](#) project which preceded Cloud developments. As originally stated in the [Fedorazon project plan](#) an [adapted agile prototyping methodology](#) was used in both of these projects.

However, the past two years experience has come to embrace a specific kind of Agile Development known as [SCRUM](#). Also of significance for this project is the use of out-sourced consultant developers to accomplish the majority of technical work. In fact, Fedorazon has written up a best practice guide in regards to how best to work with consultants contracts to assure specifications and usability features are achieved for the allocated amount. For further specifics on the methodology (including standards used and scalability of Fedorazon) please see the [project website](#).

Implementation

Overall the 'Implementation Story' of Fedorazon is one about collaboration with other like minded projects and individuals around the globe. The meta-theme that quickly emerges for our project is that the solution for 'how repositories will utilise the Cloud' is one that will be solved by participating and adding our own 'repository story' to the mix with all other sectors trying to solve these commons problems of the Web Architecture.

What follows in this section is the 'Fedorazon story' and the major themes or leitmotifs experienced throughout the project.

In the beginning there was the Grid and ASP: the project began its work with close ties to the Grid community, and have continued to collaborate with Grid projects such as: AHDS's efforts with the [National Grid](#) and the EU's [TextGrid](#). We've also had discussion with various ASP providers (due to NDA shall remain nameless). An indeed there seems to be a continuum between the Grid and ASP that the Cloud bridges. In the case of the former, it is the wide open access of the Grid that allows anyone to put any kind of software they want upon it and run it too capacity, however the emphasis for the Grid is all upon computational cycles and very little support for long term storage. Because of this there is a high threshold for understanding and administering

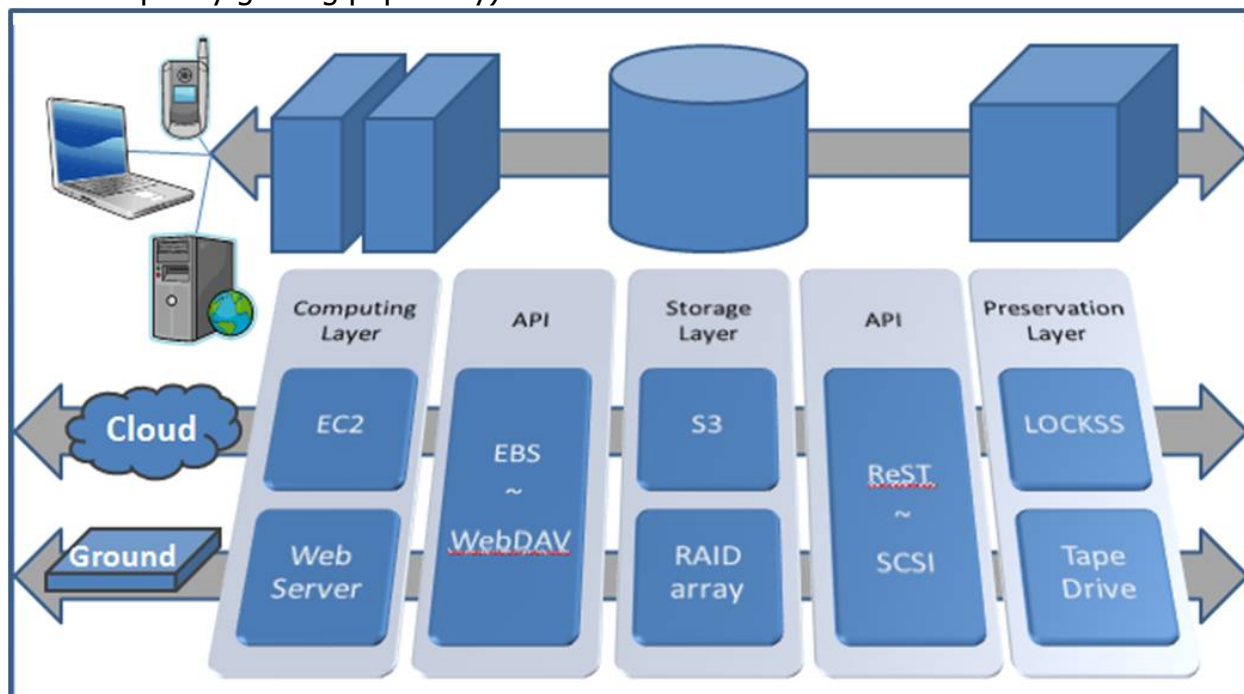
the Grid which requires in-house expertise. On the other hand ASP provides robust storage and scalable computing but only services a specific application for a large user base (e.g. five of the Bloomsbury Colleges utilise a shared VLE ASP provider in Amsterdam). Because of this the cost of ASP is only affordable when multiple licenses are purchased. All the Cloud really does is provides yet another form of compute cycles and storage facility to multiple smaller projects thereby saving cost by utilising the network effect of all potential small/medium institutions. Because of the multiple users out in the Web the Cloud results in lower overhead for administration and user base to justify the costs.



Yet, Fedorazon would not be so hasty as to suggest that the Grid, ASP or The Cloud are better than the other. Rather each have their applied business model for targeted customers and will continue to change to meet market demand. In actuality, the Cloud has not done anything new that couldn't be done with the Grid or ASP; however, what it has provided is more market diversity. Essentially, the Cloud opened up its provision beyond large institutions or large software vendors and made large data centres more affordable to smaller budget projects: both the Internet Archive and New York Times provide exemplar use cases for this diversification. In short, with options such as the Grid, ASP, the Cloud and local servers the market has matured into a competitive one that repositories can pick and choose from based upon their needs for a repository.

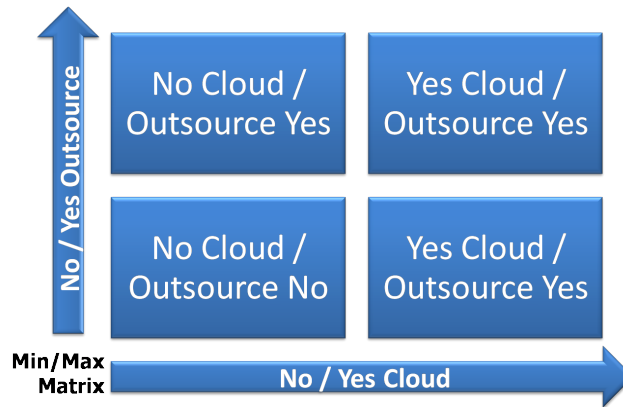
Storage on the ground or in the cloud?: in July of 2008, the Fedorazon project along with the [Preserv project](#) met with representatives from the DSpace Foundation and Fedora Commons (at the Brickskellar's Bar in Washington DC) where the plausibility of a shared low-level software storage layer for DSpace and Fedora was initially discussed and eventually manifested into the Fedora / DSpace [Storage Delegation Layer](#) and [Akubra project](#) (currently in progress). The primary focus of this conversations was the numerous types of storage becoming available and how best to utilise each so as take advantage of different cost models. Repositories have traditionally used one or two local server boxes (or the equivalent of) to run their repository. But "times are a changin'" and data is getting bigger and more complex (this includes both digitally-born data and [paper-based digitised content](#)). Yet with the diversification of storage options several large repositories have begun to ask the question of "what kinds of storage is most cost effective for which kinds of data". For example, large video data could benefit from being placed in Cloud storage so that when multiple video resources are accessed they can be better served to the user by

automatically utilising more computational cycles for delivery. On the other hand, PDFs and text documents that have very few accesses per month/year could be placed on a local box that runs on a low energy compute cycle and only spins up when a 'long-tail' user comes along to access the text resource (this also supports 'Green Computing' which is quickly gaining popularity).



The obvious choices that need to be made when comparing "on the ground" versus "in the Cloud" usually centres around which layer in the stack best supports the needs of the repository. For example an institution could easily take advantage of both, based on how the repository is going to be used. Without debating the various uses (or rather 'services') that a repository offers, it could be said that utilising a repository as a 'preservation service' might best create an architecture where the speed of the cloud is used for serving data (EC2+S3), but then preserved for the longterm via a service on the ground such as a 'Tape Drive'. In the end, it is once again the importance of knowing what your end user wants from the repository as a service that should determine the overall hardware architecture and advantages for "on the ground" and/or "in the Cloud".

More important than the placement of 1's and zero's is the human who will look after them: The final issue builds atop the latter issue of 'where the physical location of my digital bytes will exist over time'; however bytes don't exist on their own they need looking after and for the time being that looking after needs to be by a human. The question in regards to which human to be used could be analogised to a choice of should I "use a person on the ground (technician in a local IT department)" versus "use a person in the cloud (an outsourced company)"? To help make this decision we used a minimum / maximum decision matrix to present the issues involved:



The results of this minimum/maximum matrix are summarised below including the advantages / disadvantages of the three basic questions that any repository architect needs to ask when adopting either the Cloud or Outsourcing (where, who and how much):

1.) No use of the Cloud (use of 'on the ground' servers) and no use of outsourced support (use local IT support):

- 1.1.) Where
 - Advantages: local server means greatest amount of control over your data and where it physically resides.
 - Disadvantages: disaster recovery and back-up becomes more difficult and requires more time from local staff.
- 1.2.) Who
 - Advantages: having an on-site professional who knows the ins and outs of the system could result in faster response.
 - Disadvantages: loss of hardware expert staff could result in slower response rates by local IT department.
- 1.3.) How Much
 - Advantages: no significant cost advantages.
 - Disadvantages: must pay for both the hosting of the server (electricity, estates, maintenance, etc), and the team member(s) on staff to look after the servers. All of which are variable costs and do not conform to allocated set budgets.

2.) No use of the Cloud (use of 'on the ground' servers) and use of outsourced help (either consultancy support for improvements like upgrades or on-call support for when a local technician needs help solving a problem):

- 2.1.) Where
 - see no. 1.1 above

- 2.2.) Who
 - Advantages: able to specify level of service via SLA as well as regular payments for allocated budget
 - Disadvantages: trust of a company outside of your remit.
- 2.3.) How Much
 - Advantages: cost savings are debatable on level of service agreed and response rate; though able to switch providers in a robust marketplace. budgets can be allocated for costs.
 - Disadvantages: may result in higher costs than sourcing system/help locally, but again this is dependent upon provider.

3.) Yes use of the Cloud to host data and no use of outsourced help to support the use of the Cloud instance:

- 3.1.) Where
 - Advantages: data is dispersed around the planet in multiple locations
 - Disadvantages: never sure where data actually resides
- 3.2.) Who
 - see no.1.2. above
- 3.3.) How Much
 - Advantages: costs are based on what you use (or rather what you users use)
 - Disadvantages: costs can scale quickly if system is highly used (but is still cheaper than if you were to scale local IT yourself)

4.) Yes use of the Cloud and yes to having outsourced help to either host or support the use of the Cloud instance:

- 4.1.) Where
 - see 3.1. above
- 4.2.) Who
 - see 2.2. above
- 4.3.) How Much
 - see 3.3. above

While these are our opinions over the past year of using a repository in the cloud we would recommend that the above min/max matrix be used by your team in trying to decide if a Cloud Repository is good for their use. As can be seen by the above much is dependent upon the contract and SLA provided by the Cloud or Outsourced provider.

Outputs and Results

There are two pragmatic outputs the project has accomplished, i.e. if any of this is interesting to you these are the things that will allow you to start experimenting with a 'cloud repository' the fastest:

1.) The first is a publicly accessible version of Fedora 3.0 on Amazon Web Services (EC2+EBS+S3): by going to the [help documentation on the website](#) anyone can follow along with our screencast and step-by-step documentation to launch your own fully robust and scalable instance of Fedora 3.0 (there is no hardware expertise required to launch this publicly available instance of Fedorazon). Though do keep in mind you will need a credit card to create an Amazon account (if you don't have one for amazon.com already). While the actual server boxes you are renting from Amazon cost money (a lot less than buying your own box), all of the Fedorazon software is Open Source (BSD). If you know how much content (size) you are going to put into Fedorazon along with the estimated number of times your users access that data you can calculate your monthly costs via the [AWS monthly pricing calculator](#).

2.) The second significant output, is the [cost analysis report of the project](#) which attempts to provide pragmatic advice for the actual costs in running any repository within the cloud. It does this by adopting some of the better known computation laws (Moore's Law) and Economic Principles (Pareto principle) to describe expected long term costs of using a repository in the cloud. This report also goes so far as to try and predict some realistic cost models for utilising both computing and storage space within the cloud. The report is intended for anyone who has a basic knowledge of repository architectures and is looking for policy advice on utilising cloud computing and storage.

Outcomes and Sustainability

Because of the brief nature of the Fedorazon project the outcomes of the project are primarily aimed at enabling other institutions to pick up the work and build on top of it themselves (never an easy task). While several Academic sector institutions have utilised Fedorazon for development and testing ([see the list of attendees at the Fedorazon training event](#)), there has yet to be a definitive institution to use it as their primary repository instance. The foremost reason for this, is the lack of support for the repository hardware/software once it has been launched. Fedorazon does not compensate for the periodic hardware maintenance (periodic updates, restarts and back-up procedures) required for any active system. However, we do feel that Fedorazon could become more than just a testing and development environment. Currently, Fedorazon could be called a "Platform as a Service" (PaaS), we make this distinction because Fedorazon is just a blank digital shelf for content, it does not include the other value-added services layers of a human providing maintenance or in addition a human providing on-call (email/telephone/forum) support. Because the Fedorazon code was

written with a BSD license any other institution or company can pick up Fedorazon and use it as a base platform (PaaS) to provide value added services (SaaS). And indeed we are glad to announce the use of Fedorazon (PaaS) as part of a hosted (SaaS) service being provided by an SME company called MediaShelf. As of October 2008 [MediaShelf will be providing hosting services to the UK](#) that can take advantage of all the various features of Fedorazon (robust back-up and versioning, regional specific placement of data, software updates, monitoring, analytics and much more).

We would hope that other companies (or rather HE/FE institutions) begin to consider the plausibility of building their own SaaS provisions from Fedorazon's PaaS. The exemplar of other Open Source communities that have moved to the Cloud model (SaaS provision) is extremely encouraging because of the competitive marketplace that exists around the software. For example, SugarCRM has several companies who have built up their business around both hosting the software for organisations as well as providing development consultancy for new tools atop the system (see JISC's CRM4UNI project).

Looking to the future, one can't help but wonder if the age of open source SaaS isn't on the horizon; one can see a band of institutions supporting and contributing to a single Cloud instance making sure it is up to date for their shared use, in addition these same institutions could fulfil the SaaS provision by having round-the-clock telephone/email support via a global network of institutions utilising the same cloud instance? Perhaps this idea is a castle in the cloud a bit too far too see just yet, but then again no institution has yet tried to organise such an effort? Further speculation in regards to more pragmatic ways that we feel the architecture of the Cloud will impact the repository sector can be found in the eFramework SUM on 'Disposable Cloud Computing' (see [Implications](#) below).

Implications (The Future is Nigh)

The Cloud once again reminds us of an architecture that encourage both the fragility and the disposability of the Web. We all know the basis of computer networking as a military technology developed to transition the information and communication required for running a military operation in the case of a nuclear war (DARPA). This singular user case by which computational networking is based upon suggests one primary ethos that all computational technology has been built upon since: that nodes in the network can (and most likely) will be destroyed over a given period of time. Perhaps this is a bit of a Dickensian view of the world, but none-the-less a successful one that has birthed the ability by systems to transform and change themselves over time.

Accordingly, the most discussed service a repository is supposed to provide is one of "preservation"; yet what is true digital preservation? Or rather, more importantly can any repository really be expected to preserve it's resources over time on the scale of paper, papyrus or stone tablets?

The Cloud has contributed to our understanding of what an "Architecture of Preservation" might look like by reminding us that repositories can easily be launched (within minutes) and destroyed (within seconds), in short no system is able to assure that it will exist as a server of data to the web. If we can agree upon this basic assumption then we as repository advocates cannot provide preservation in the long term we will be a step closer to realising what a "preservation architecture" might look like. Based upon this theoretical assumption the Cloud quickly becomes a key component for all repositories if they truly intend to provide the service of preservation.

Recommendations

- Further JISC support for project exploring repositories in the Cloud, especially more in depth cost-analysis studies.
- Support by JISC for standardised AMIs between Cloud computing companies to enable swappable Cloud components. An example of a company attempting to do this is [Aptana](#).
- Funding by JISC to encourage more institutions to provide business models for how institutional (especially consortium) expertise can be developed to deal with Cloud computing.
- A closer working relationship by JISC with repository development teams (EPrints, DSpace, Fedora) in helping provide Cloud computing instances and supporting communities (both proprietary marketplace and open source communities).