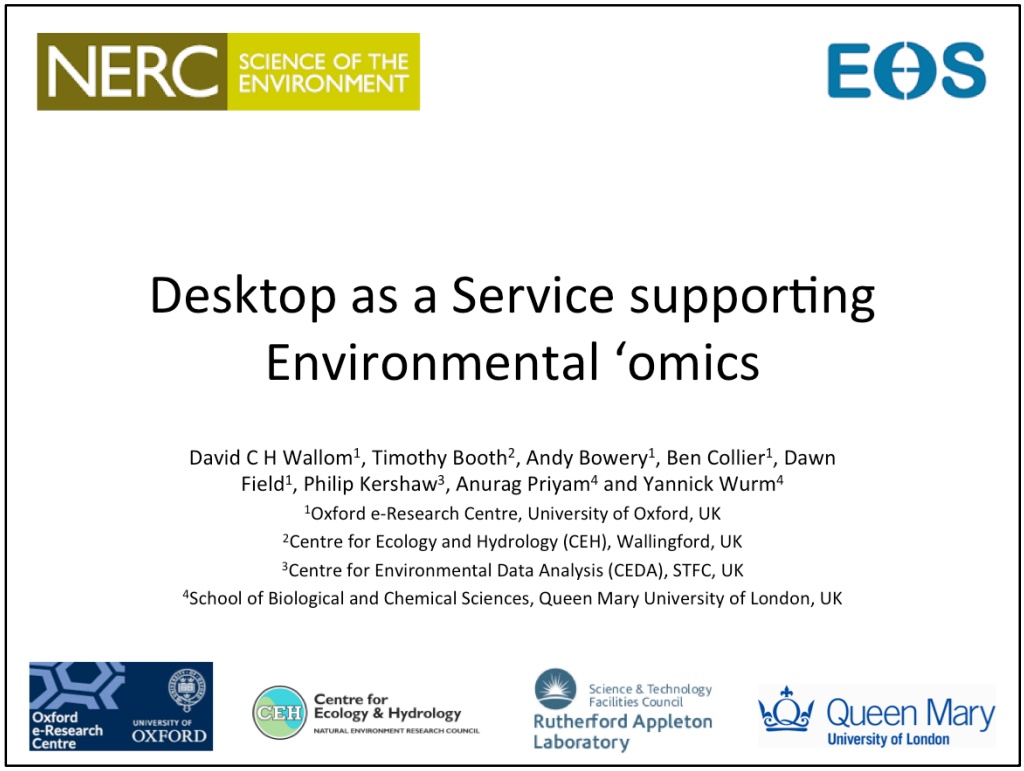
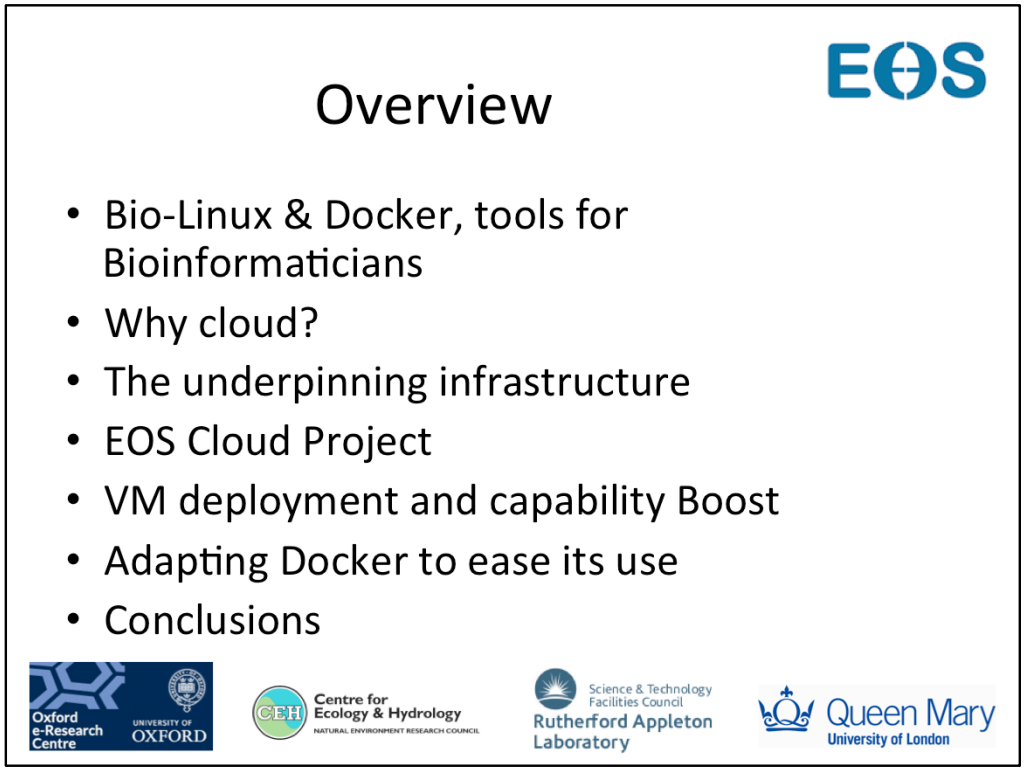
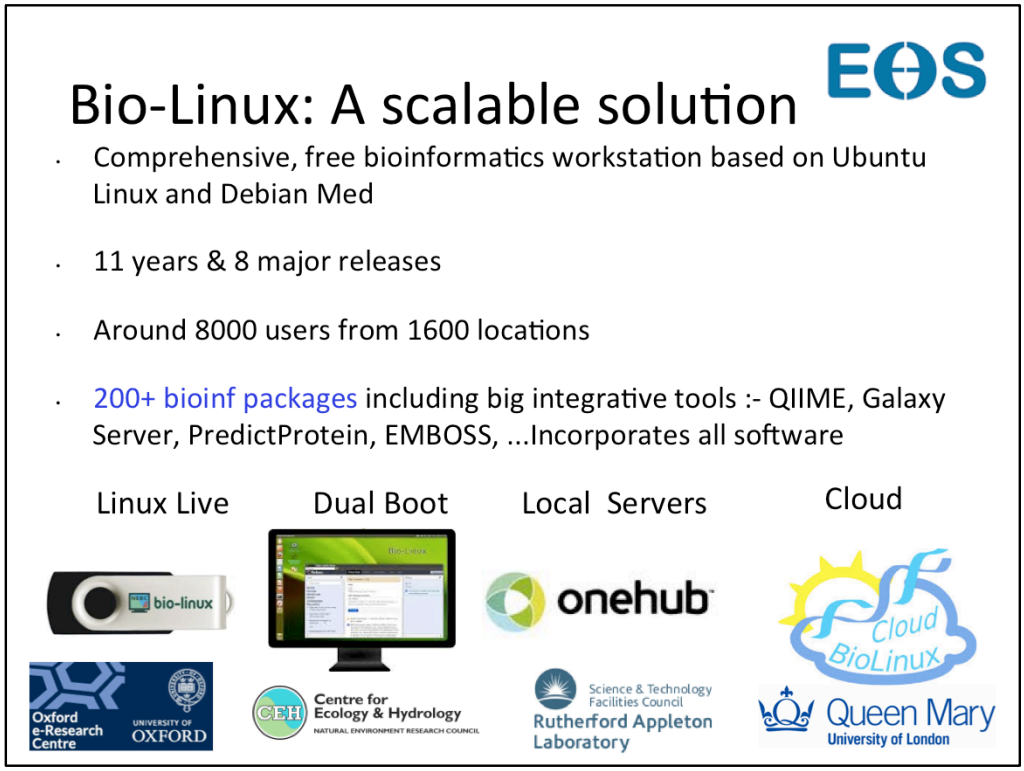


This experience paper describes the development of a new Desktop as a Service (Desktop-aaS) system to support the deployment and utilisation of tools for researchers who are working on Environmental 'omics in the first instance but can be extended to Bioinformatics more generally.

Desktop-aaS is a paradigm where specific user environments, in our case personal Linux virtual machine instances, are operated as Software as a Service solutions. The initial instance is pre-configured but subsequently the end-user has total administrator-level control over it. Each instance is normally left running so that a user may disconnect and reconnect to return to the same working environment whenever needed.



Within this paper we first outline the need for software tools which will simplify the user experience, and describe particularly suitable tools and approaches. Following this we describe our service that presents these as community specific remote desktop solutions. We talk about the management interfaces to isolate the user from complex cloud system management and then conclude by describing two user communities we are already working with.



Bio-Linux[1] is a major success with widespread adoption across multiple communities. It provides a method by which personal bioinformatics workstations can be set up and maintained by novice users, removing the need for individuals to compile and install packages themselves. The project has an active user and developer community and engages with other open-source community efforts like Debian Med[2], Galaxy[3], Open Bioinformatics Foundation[4], etc.

Workstations running Bio-Linux connect to a package repository where hundreds of tools have been built into Debian (.deb) packages and are tested and continually updated by the community of maintainers. Installation or removal of even complex tools is trivial for the end user, and updates are automatic.

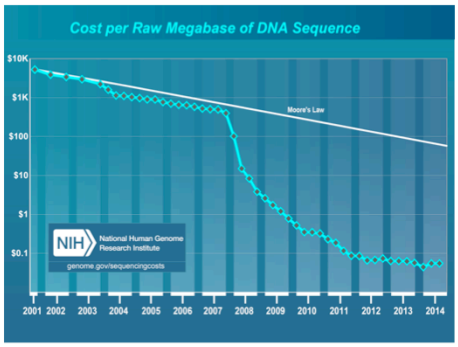
Bio-Linux has traditionally been installed onto bare-metal, possibly on a dual-boot workstation, or run directly from a USB stick for training and demo purposes. In recent times, use on local Virtual Machines(VM) and Cloud systems has increased, with a community CloudBioLinux[5] effort building and releasing machine images on the Amazon EC2[6] public cloud. Also, partly through this current project, support for local VM installations (on VirtualBox[7], Parallels[8], VMWare[9]) has been made a core part of each Bio-Linux release. Users can obtain an Open Virtualization Archive (OVA)[10] file which loads directly into these products to instantly provide a functional desktop system. It is this same pre-packaged virtualised workstation that users now access on the EOS Cloud.

Docker[11] allows a uniform platform-independent application to be built and encapsulated in a format know as a container. As the research community, and people in general, make use of a large number of different operating systems the complexity of installing applications on new systems and can be high. Sometimes there may be incompatibilities between the libraries and helper tools needed by an application and those on the target platform.

The Docker approach aims to get round this by providing single application hosting platform irrespective of hosting system/operating system etc. We have recently seen within the bioinformatics community an increase in sharing of applications and tools in this manner.

One problem faced when using the Docker hosting environment is that the applications that are installed and running within it look and feel like remotely hosted applications, they do not behave as if they have been installed on the users machine. Therefore a tool that bridges this usability gap would greatly increase the acceptability of tools and services that are installed and distributed using Docker.

Bioinformatics software challenges



changes everything for biology

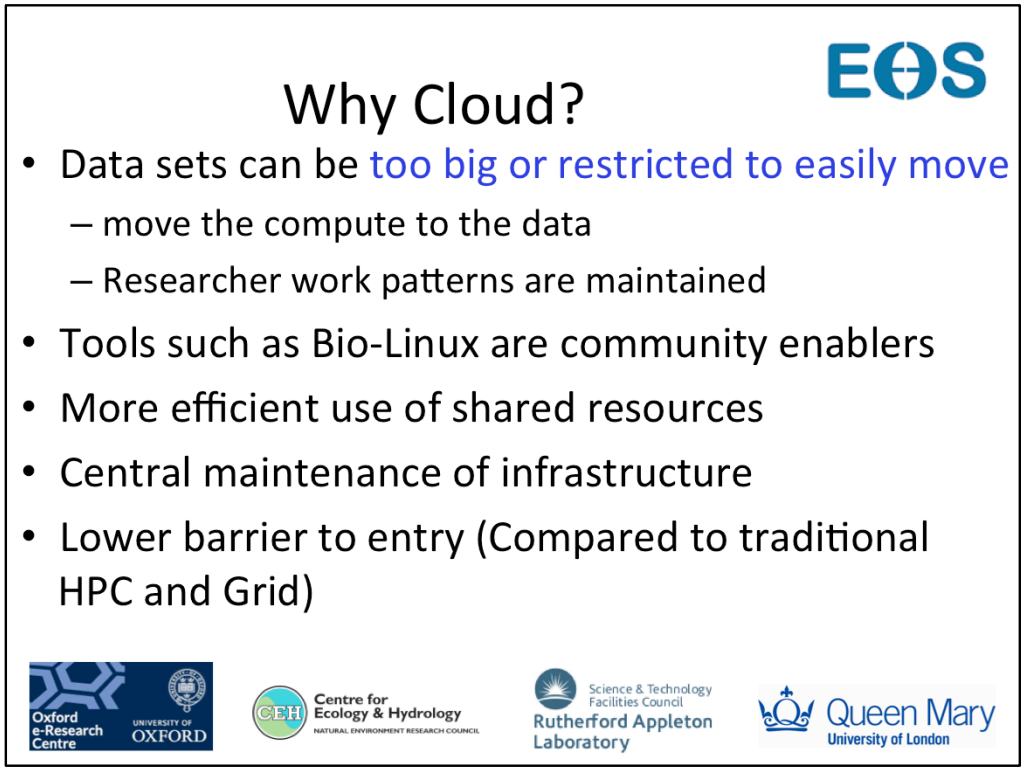
- This brings a onslaught of new challenges for bioinformatics:
  - projects that used to require teams of 500 are now accessible to small teams
  - but biology curricula (i.e. biologists) still lack computational skills.
  - thus biologists are overwhelmed by large amounts of data
  - furthermore data types are young - so software is young, thus
    - software may be badly built (by biologists with no formal software dev training/xp).
    - software needs to be frequently updated (bugfixes, algorithmic improvements (sensitivity/specificity), new data type support).

Oxford e-Research Centre · University of Oxford · Centre for Ecology & Hydrology · Natural Environment Research Council · Science & Technology Facilities Council · Rutherford Appleton Laboratory · Queen Mary University of London

Bioinformatics is a field with multiple drivers of complexity [12] including the following three. Firstly software and pipelines, which are typically open source, are often developed only for a single platform, which may not be what the other end users are running. The choice of this platform may not be something the researcher has any control over.

Second, most researchers that now have to rely on bioinformatics obtained only little computational training because huge datasets are new to biology. Bioinformaticians are thus in general less technically experienced than physicists, astronomers or computer scientists. Simplifications to installation or instantiation approaches are thus generally very well received.

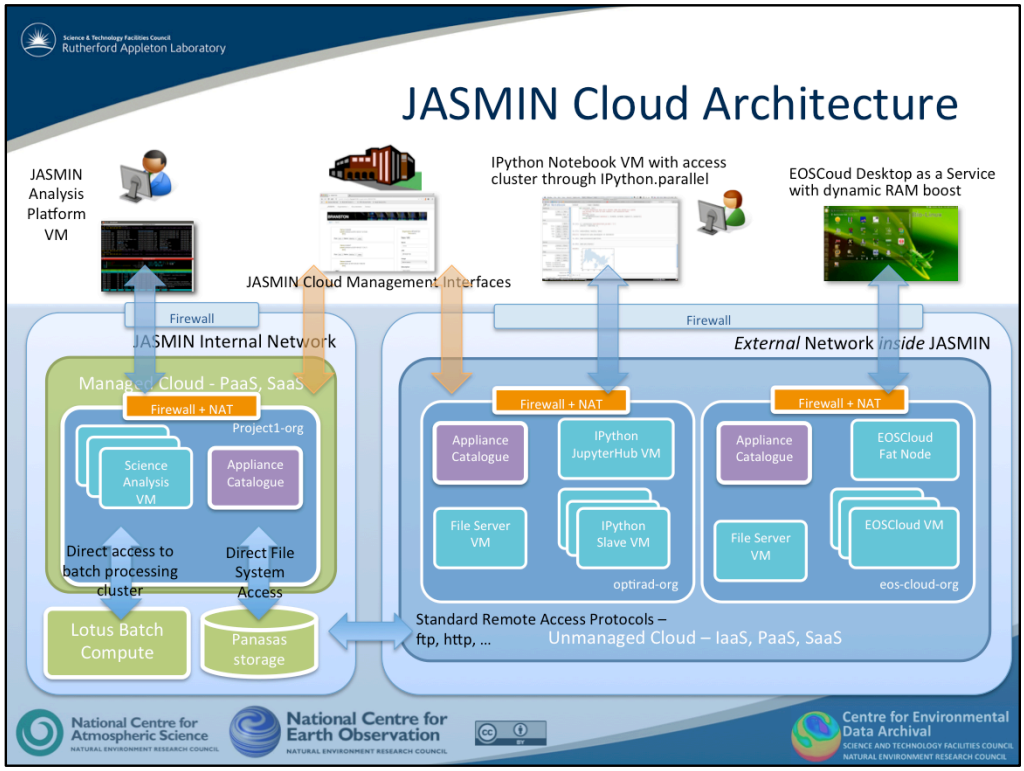
Finally, as the size of data problems increases, there is an increasing need for the use of larger shared systems. This means that the installations are no longer managed by a bioinformatics team but become a central research computing support function. As collaborations also cross institutional boundaries there is no guarantee that the services, software versions and data sets will be the same across collaborators.



When moving into the big data regime it is becoming much more efficient to move the compute to the data. This both reduces time spent transferring and synchronizing data across networks and also simplifies management and access policies where copyright or privacy constraints apply. With Desktop-aaS as the access methodology we maintain the work pattern of the user whilst increasing the available resources to them.

Building on Bio-Linux means that many members of the user community are already used to this type of tool and in the cloud it is just sited on top of more complete and capable resources.

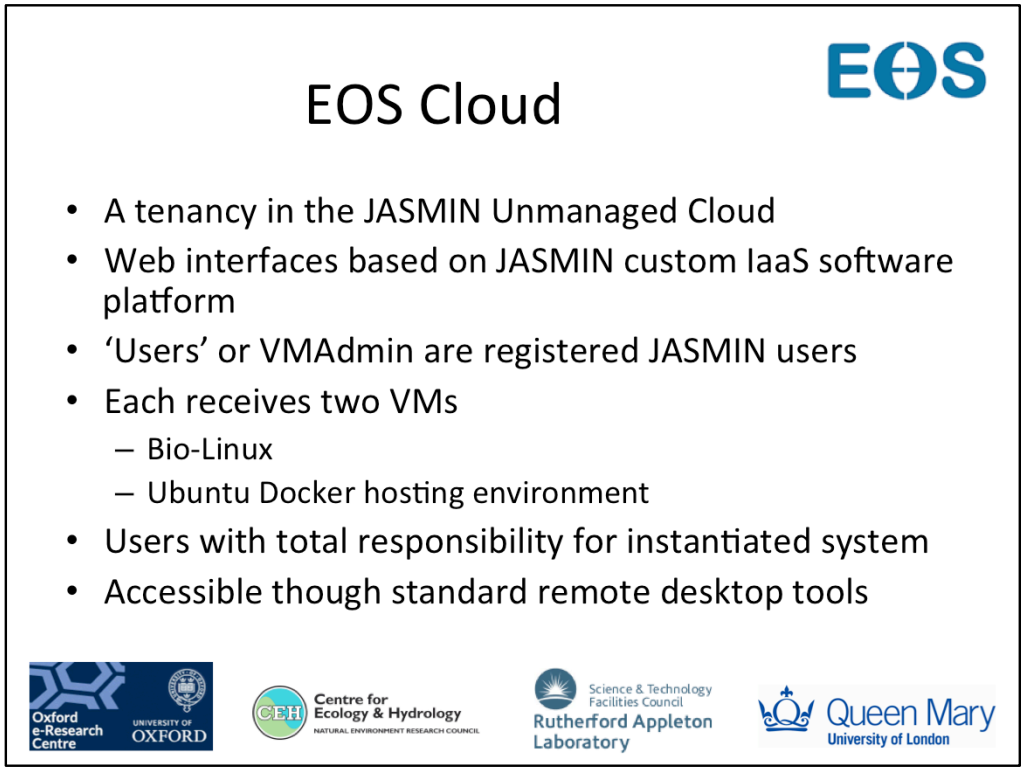
Many central research computing teams are now recognising the importance of the cloud paradigm as a vehicle to simplify their own workload, thus cloud services and capabilities are increasingly provided within institutions. Indeed, the cloud approach is considered an overall simpler and easier method to provide flexible access to resources, compared to traditional shared servers. This is in part due to handing over control to the user rather than the system owner. Thus cloud architectures provide a way of maximising the value of infrastructure investments for a broad range of end-users.



Hosted through the Centre for Environmental Data Analysis(CEDA) at STFC the JASMIN service[13] is intended to be a multi user and multi use-case facility that is able to support a wide range of user communities. The system is built in a modular fashion with a hardware layer that is deliberately designed to support multiple different distributed computing paradigms, from HPC type workloads to cloud hosting. Utilising commercial off-the-shelf private cloud software, which supports API level access, the cloud service operated in two modes: firstly a highly controlled Platform as a Service system in the managed cloud, and secondly a more flexible community or user controlled Infrastructure as a Service system, the unmanaged cloud.

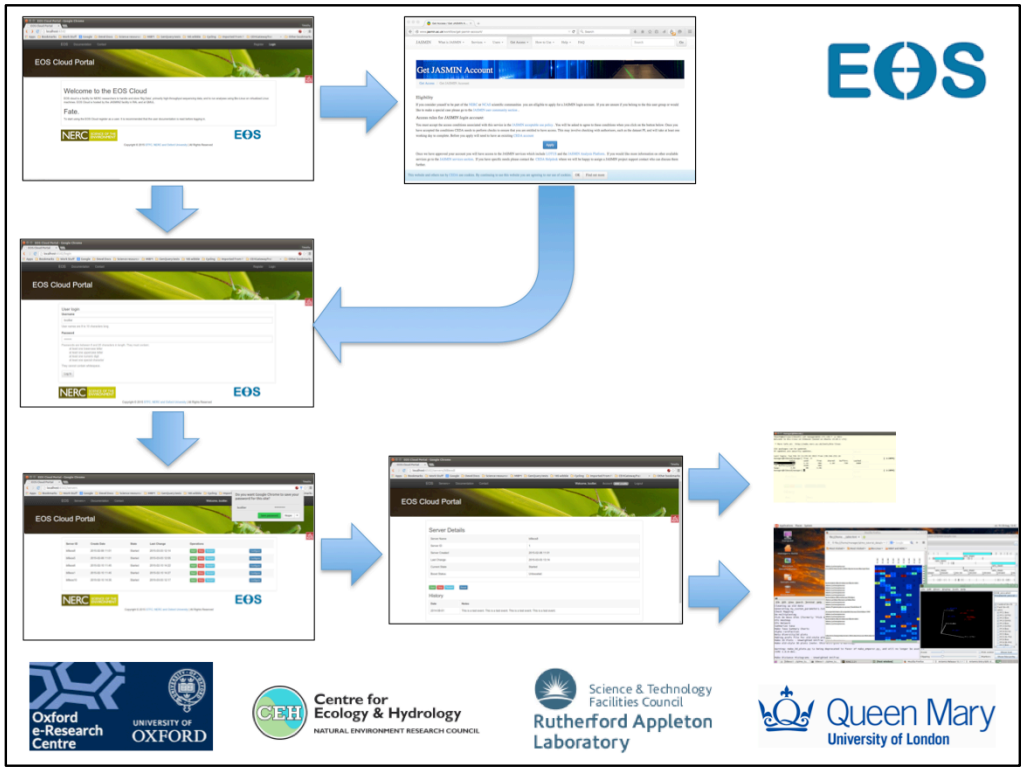
Users may opt to run in the managed mode, where VMs are standardised to JASMIN approved templates and configured by Puppet[14] scripts that apply centralised access controls, or in the unmanaged mode which we use for the EOS Cloud. Here each user can have full root access rights on their VM once it is deployed and handed over to them, and we as clients of JASMIN can make use of the full vCloud Director API[15] to assign resources to VMs according to our own policy.

A further advantage of working on JASMIN is that, as a community cloud, the service already hosts, and synchronises data from relevant providers. These include large reference datasets which our target user community finds useful. Users can rely on this resource and only need to concern themselves with the transfer of their own data into and out of the cloud.



Supported through the 2014 NERC Big Data call EOS Cloud is a user community of the JASMIN service. As part of the user agreement every EOS Cloud user is also a JASMIN registered user. Once registered they gain access and control over two VM images. The first is a full Bio-Linux image that is supported and maintained by Centre for Ecology and Hydrology (CEH) and the second is a cut down minimal system intended just to support Docker container hosting.

An important feature of our service model is that once instantiated and ownership claimed by the user, the user takes full ownership of the VM. These systems are set up with default firewalls and configured to auto-apply security patches via the standard Ubuntu unattended-updates mechanism but other than that the user is responsible for all machine customisation, management, and maintenance.



Workflow to get to the point where a user is able to access resources within the EOS Cloud.

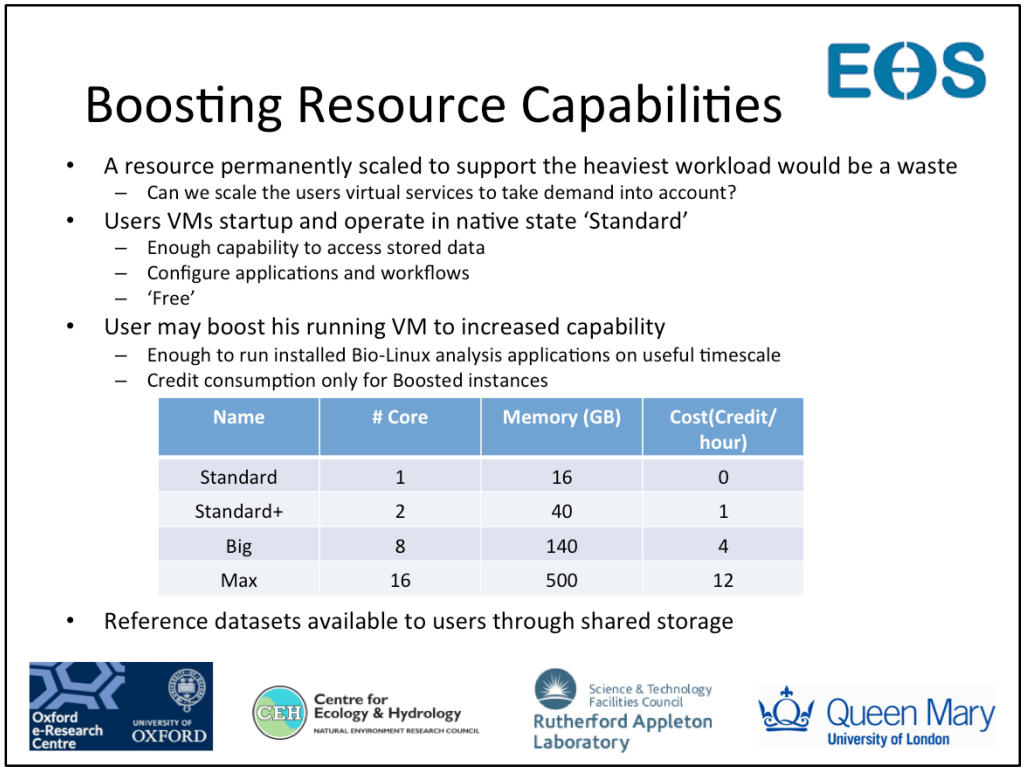
(1) From the welcome portal they must first request a standard JASMIN system user account,

(2) Once registered they are able to log into the system.

(3) The resource status page shows for each user the VM instances they have access and control over

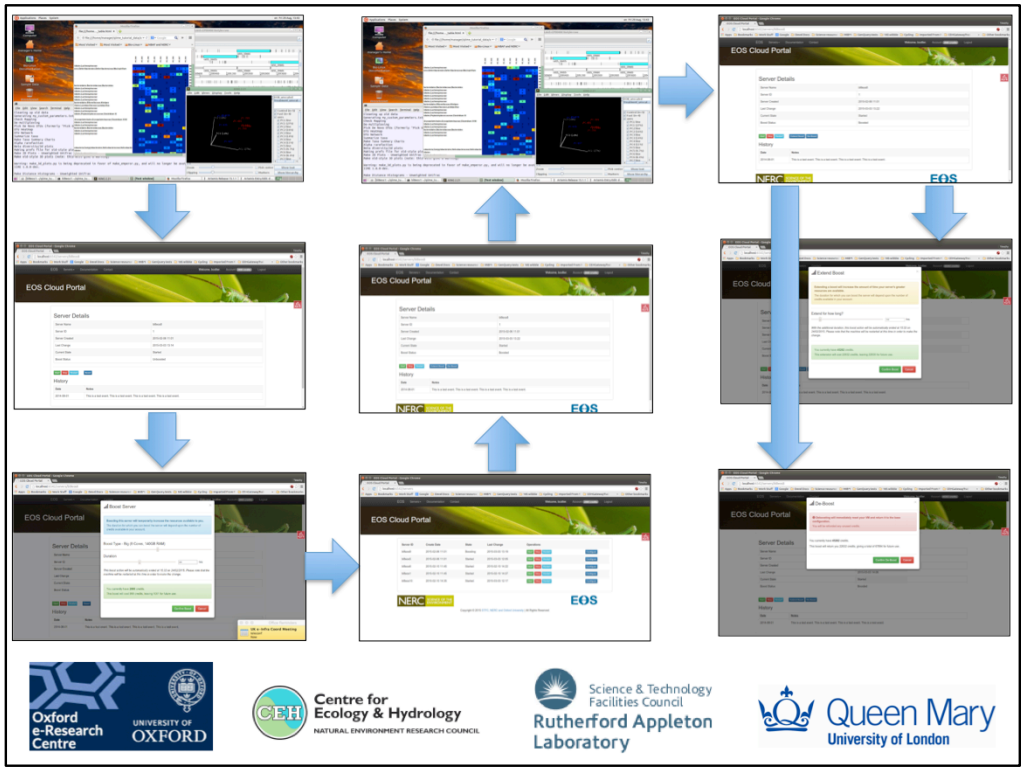
(4) Highlighting a single resource showing the information on each instance available including core resources and status

(5) Users gain access to their allocated resources through either SSH or standard remote desktop tools (we support x2go[16]), both using the public SSH key they deposited on account creation



The key new idea within the EOS Cloud, beyond a standard Desktop-aaS, is that we allow users to expand and contract the resources allocated to their VM depending on anticipated workload. This way the physical resource can be shared more easily and fairly amongst a larger number of people whilst still guaranteeing more performance than would have been the case with simple overprovisioning. In standard mode the user is able to transfer data, configure software, or examine the outputs from previous computational runs. With 16 gigabytes of memory available to it this standard state is still able to start and test out most of the complex tools. To move beyond this basic system configuration users can access the Boost functionality, accessed via our custom web interface. This novel functionality is analogous to adding physical upgrades to a hardware system; Boost allows this in a virtual space. Once selected the user observes their VM halting and then following reconfiguration it is restarted and ready to process large biomolecular datasets as a feature-full large CPU and memory system. The top capacity available is beyond what people would expect on a highly capable desktop workstation. To push for responsible utilisation of the boost feature and to promote the fair sharing of resources between users we have implemented a credit system by which credit is spent to place the VM in a boosted state but in the baseline standard mode it is not.

We also provide disk capacity for on-cloud data storage, and reference datasets from institutions such as EBI are available within the EOS Cloud to be mounted on the VMs with a couple of simple commands or mouse clicks.



Once the user has set up their computational task they need to ensure they have the computational capacity on their VM.

(1) within the portal they highlight the virtual machine they are using and then select ‘Boost’

(2) The boost pane allows the user to select from a set of three predetermined configurations the resources their VM will receive, they also select how long this boost will be occurring for.

(3) The user is shown, to illustrate the Boosting progress, the different states the system transits through during reconfiguration.

(4) Once complete (in about one minute) the summary page for the VM shows that it is in a boosted state.

(5) The user may now use the VM which will appear exactly the same except with more computational capacity available to it.

(6) Once they have completed their particular computational task then they return the system back to its standard state:

(7) This occurs automatically if the boost period expires, though in the event of a computational task taking longer than expected the user is able to extend the boost period by spending further credits.

(8) Alternatively, if the computation takes a shorter length of time they are able to end the boost immediately and receive an automatic refund of credits for the unused time.

The oSwitch application enables seamless switching from one operating system to another - providing access to diverse ranges of tools, whilst ensuring that during their utilisation the environment provided is not too alien to the user community.

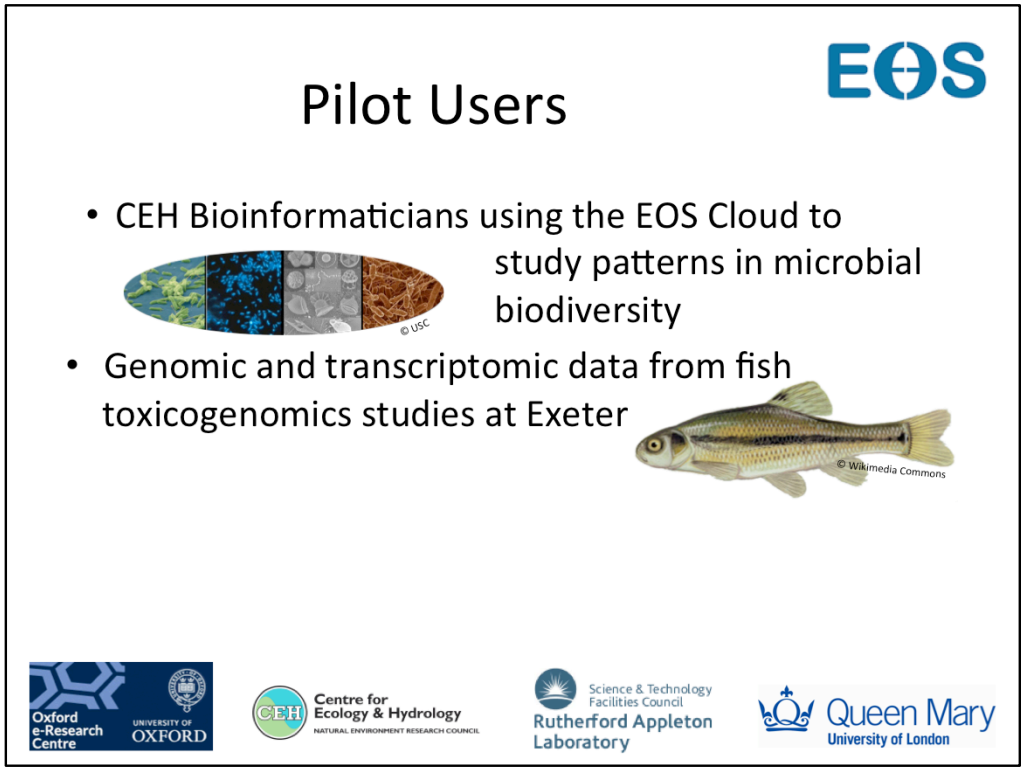
oSwitch is a wrapper facilitating access to Docker images. Importantly, when switching operating systems inside a shell, most things remain unchanged:

- Current working directory is maintained
- User name, uid and gid are maintained
- Login shell (bash/zsh/fish) is maintained
- Home directory is maintained (with all .dotfiles and config files).
- Read/Write permissions are maintained
- Paths are maintained whenever possible. Thus volumes (external drives, Network Attached Storage, USB) mounted on the host are available in the container at the same paths.

The service is designed to sit within the operating system to allow a user to manage their own applications whilst ensuring that the way they interact with their own system is not changed by the running of containerised applications.

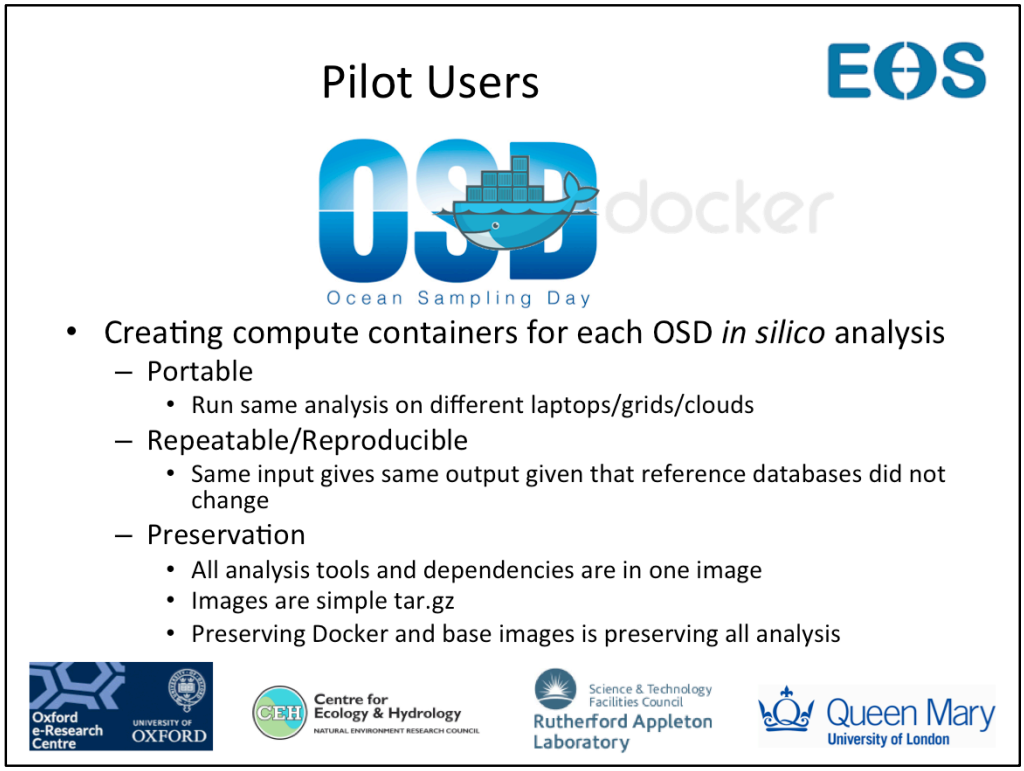
This also overcomes a major hurdle for systems administrators because it makes it possible for even tools with complex dependencies to be installed entirely by users in user-space, and for bioinformaticians because it dramatically increases the agility with which they can test out and use different tools as part of their analysis processes.

We will not have to perform any packaging of any specific tools because docker-packaging is already widely occurring for existing tools (e.g. http://bioboxes.org and see http://hub.docker.com). oSwitch enables access to easily and quickly installed applications with enabled utilisation as if they are installed natively.



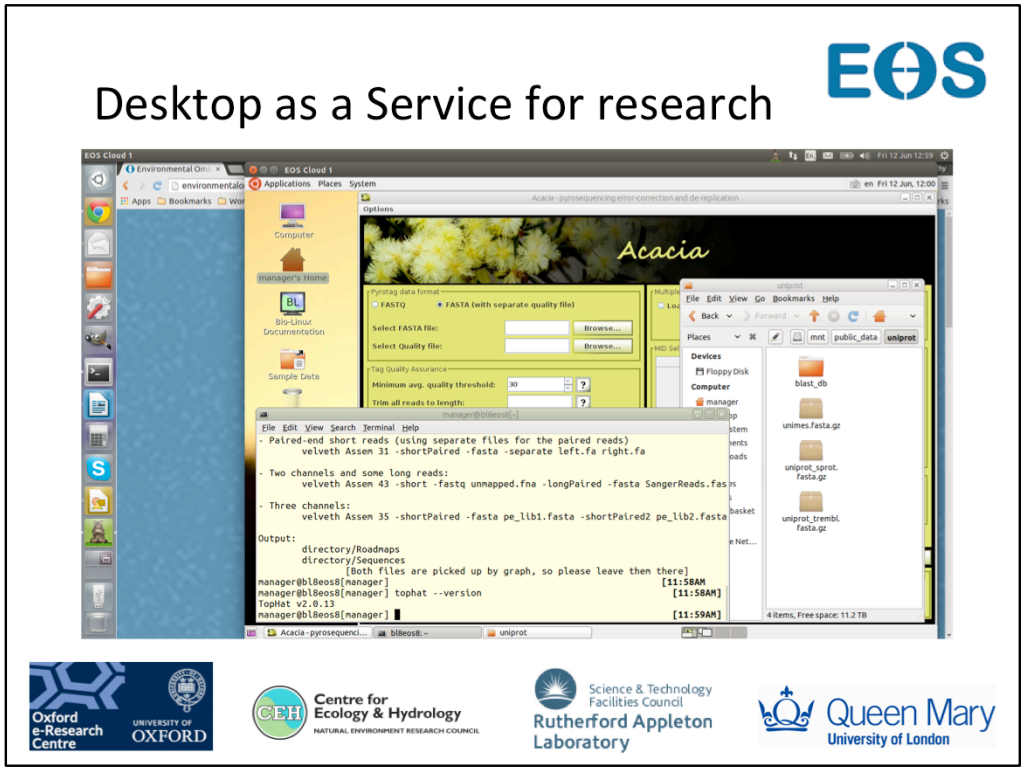
Hyun Soon Gweon and Katja Lehmann, two bioinformaticians as CEH, are familiar with using Bio-Linux on the departmental server and personal VirtualBox VMs. They are trialling the system as part of their regular work and to provide a shared computing environment with project collaborators in microbial community profiling.

Ranny van Aerle at University of Exeter is unable to run a Bio-Linux workstation at his home institute due to local IT policies and budget constraints. He needs a basic workstation plus occasionally some extra computing power. While he is not at present being charged for use of the pilot system, he represents exactly the type of user for whom a paid subscription to the service would make economic sense. Since being given access to a Bio-Linux instance on EOS Cloud in October 2014 he has used it to assemble bacterial genomes, annotate large metagenomics datasets using Blast and Diamond, visualise Blast/Diamond results in MEGAN and generally to try out new software and scripts.



Ocean Sampling Day is a simultaneous global mega-sequencing campaign with the ambitious aim to generate the biggest standardized microbial data set of a single day. As part of this a key component is the uniform protocol that is used for both the physical data capture elements but also the digital components in the workflow. By its very nature repeatability of all steps is key.

To support this over short timescales well documented installations of software applications make this possible. Over the longer timescales that this project requires this can become a serious impediment to experimental repeatability. Therefore Docker is interesting for Ocean Sampling Day due to is ability to disconnect the application from the underlying platform and its current and growing multiplatform support capabilities. Packaging all OSD analysis in Docker containers/images allows better preservation and multi platform options. As each container has all dependencies needed for a compute step, it only requires the continued support of the Docker platform as the run-time platform to allow re-running of the Docker container image at any-time.

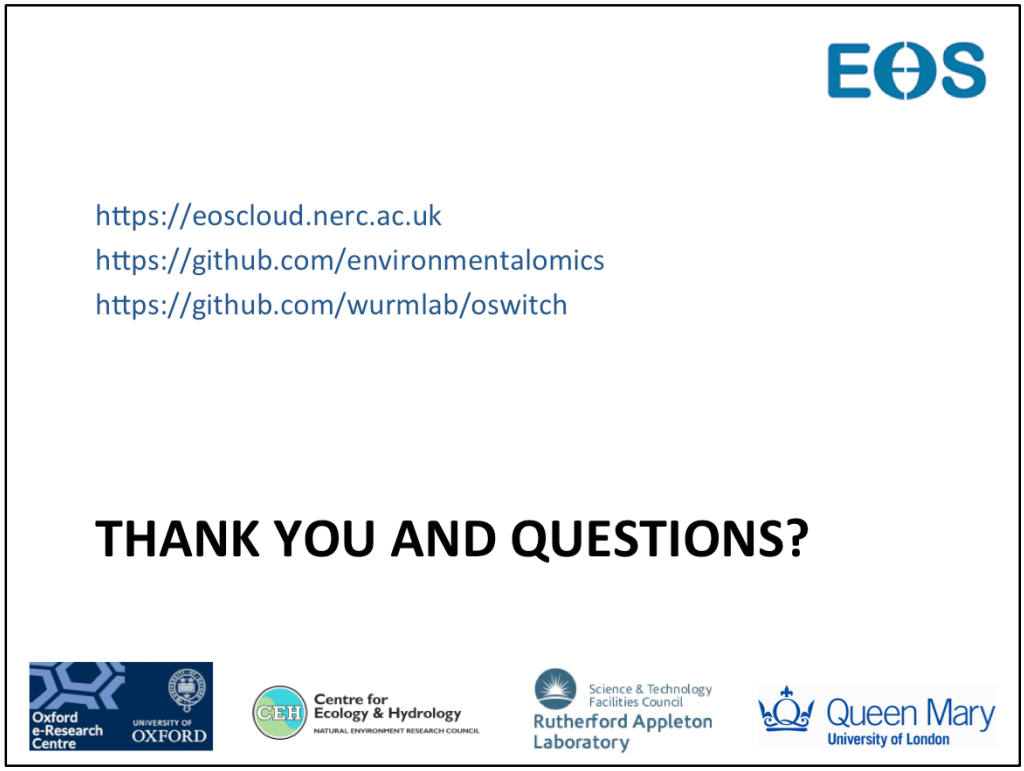


We are creating a new facility for NERC research communities. Users work in a familiar computing environment that they are confident in, while the underlying infrastructure is rapidly reconfigured on-demand to support their big data processing needs.

Our system would operate on any IaaS cloud, subject to adding a small driver layer to the Python code. It will thus represent lower maintenance burden for local systems administrators than maintaining diverse software packages and operating systems on a pool of servers into which users log in directly. Our current implementation is on VMWare VCloud but uses an abstraction layer to facilitate portability; it uses no proprietary capabilities of the platform.

Up until now we have been running closed testing with internal users and the Ocean Sampling Day community. The initial development and testing is in the final stages and from April 2015 we will start taking on further pilot user communities.

As well as providing an attractive platform to existing bioinformaticians we see teaching and on-line learning as key usage models for this new platform, and seek to integrate our technology with wider efforts.



References

1. Field, D., Tiwari, B., Booth, T., Houten, S., Swan, D., Bertrand, N., & Thurston, M. (2006). Open software for biologists: from famine to feast. *Nature biotechnology*, *24*(7), 801-803.
2. Möller, S., Krabbenhöft, H. N., Tille, A., Paleino, D., Williams, A., Wolstencroft, K., ... & Plessy, C. (2010). Community-driven computational biology with Debian Linux. *BMC bioinformatics*, *11*(Suppl 12), S5.
3. Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., ... & Nekrutenko, A. (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, *15*(10), 1451-1455.
4. Open Bioinformatics Foundation (OBF, www.open-bio.org)
5. Krampis, K., Booth, T., Chapman, B., Tiwari, B., Bicak, M., Field, D., & Nelson, K. E. (2012). Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community. *BMC bioinformatics*, *13*(1), 42.
6. Amazon, E. C. (2010). Amazon elastic compute cloud (Amazon EC2). *Amazon Elastic Compute Cloud (Amazon EC2)*.
7. Watson, J. (2008). Virtualbox: bits and bytes masquerading as machines. *Linux Journal*, *2008*(166), 1.
8. I. Parallels, "An introduction to os virtualization and parallels virtuozzo containers," Parallels, Inc, Tech. Rep., 2010. [Online]. Available: http://www.parallels.com/r/pdf/wp/pvc/Parallels Virtuozzo-Containers- WP-an-introduction-to-os-EN.pdf
9. Rosenblum, M. (1999, August). VMware's virtual platform™. In *Proceedings of hot chips* (Vol. 1999, pp. 185-196).
10. Distributed Management Task Force, Inc., Open Virtualization Format Specification (version 1.0.0), DSP0243, 2009
11. Docker container hosting environment; http://docker.com
12. Wurm, Y. Avoid having to retract your genomics analysis (2015). The Winnower. https://thewinnower.com/papers/avoid-having-to-retract-your-genomics-analysis
13. Lawrence, B. N., V. L. Bennett, J. Churchill, M. Juckes, Philip Kershaw, Stephen Pascoe, Sam Pepler, M. Pritchard, and Adrian Stephens. "Storing and manipulating environmental big data with JASMIN." In *Big Data, 2013 IEEE International Conference on*, pp. 68-75. IEEE, 2013.
14. Puppet, Puppet Labs; https://puppetlabs.com/
15. VMware vCloud, A. P. I. Programming Guide.