Open Source Cloud Infrastructure for Encoding and Distribution Documentation

Release 1.0

David Fischer

May 08, 2013

The Author

author David Fischer

address Chemin de Saule 57, 1233 Bernex

contact david.fischer.ch@gmail.com

motto It's not a bug - it's an undocumented feature.

The Thesis

title MPEG-DASH Distribution Platform

school hepia : Haute école du Paysage d'Ingénierie et d'Architecture

organization HES-SO Master of Science in Engineering

domain Multimedia & Informations Technologies

responsible Prof. Andrés Revuelta

date 8th February 2012

The Project

name Open-Source Cloud Infrastructure for Encoding and Distribution

organization European Broadcasting Union (EBU/UER)

project leader Bram Tullemans

main developer David Fischer

license GNU GPLv3

release Basic V1.0

revision 4a6b7d9cae9e49db3670b011c889bc6f65038d6c

I would like to gracefully thanks Mr. Bram Tullemans for his guidance and support during this work.

I would also thanks the EBU Technical staff and my colleagues at hepia for their interest and support.

Thanks to my brother Michaël, who put is experience in web technologies to help me developing the nice Web User Interface.

Thanks to Dimitri Racordon, an invaluable colleague, experienced in cloud technologies for his help with Open Stack.

Thanks to my PA follower, prof. Andrés Revuelta, for his good mood and for all the professional and personal experience we shared in common with our team on the telecommunications laboratory of hepia.

Thanks to my family, especially to my wife, Claire, for the personal support they provided to my during this work and also during the whole time I followed the Master courses.

... And sorry if I forgot anyone !

Kind Regards,

David Fischer

French

Here is the original executive summary.

Titre de Thèse	Plateforme de distribution MPEG-DASH
Projet à l'UER	Infrastructure Cloud Open-Source pour la Distribution et l'Encodage
Responsable	Revuelta Andrés
MRU	TIC / hepia
Orientation	TIC
Axes technologiques concernées	TIC / Systèmes d'information et multimédia
Entreprise	EBU / UER

Résumé

Le Cloud computing est un élément clé permettant de rendre une application capable de s'adapter (*élastique*) à la charge en allouant à la volée de nouvelles ressources informatique. Pour cela, il est nécessaire de disposer d'une plateforme IaaS au sein de l'entreprise. Le projet libre OpenStack propose ce genre de service. Le projet, développé en collaboration avec l'EBU consiste en la création d'un outil Open-Source dédié aux tâches multimédia (transcodage, publication, ...) permettant non seulement d'ajuster l'utilisation des ressources internes à l'entreprise mais aussi de pouvoir monter en charge en utilisant les offres d'IaaS telles qu' Amazon AWS. L'avantage principal de ce genre d'approche et de rendre l'application *élastique* tout en optimisant les coûts d'utilisation de resources informatiques louées (offre IaaS) en temps-réel (coût actuel de l'offre, charge des serveurs privés, situation géographique des clients finaux ...).

D'autres extensions sont envisagées comme la possibilité de générer du contenu au format de streaming adaptatif MPEG-DASH.

Cahier des charges

See also:

Veuillez lire la rubrique *Project Specifications* pour de plus amples informations.

- Réaliser un cahier des charges détaillé
- Installer la plateforme de cloud privé OpenStack sur 4 serveurs Dell (à commander)
- Définir l'architecture de la plateforme Open-Source de démonstration
- Implémenter la version de base de la plateforme Open-Source de démonstration

Connaissances préalables

- Très bonnes connaissances du monde GNU/Linux et capacité à administrer un système Ubuntu
- Capacité à définir l'architecture logicielle d'un système distribué
- Éléments de transmission et codage numérique multimédia
- Problématique de la transmission en temps réel sur IP

English

Thesis Title	MPEG-DASH Distribution Platform
EBU Project	Open-Source Cloud Infrastructure for Encoding and Distribution
Responsible	Revuelta Andrés
MRU	TIC / hepia
Orientation	TIC
Technological domains	TIC / Multimedia & informations technologies
Company	EBU / UER

Resume

Cloud computing is one of the key to create scalable applications or services able to scale-up scale-down on demand. In such model, the computing resources are abstracted and the application will consume them as such. In one of the most known scenarios the computing resources are provided by a cloud provider (e.g. Amazon AWS, HP Cloud, ...) as a service and typically delivered over a network such as Internet. The end-user will consume the resources on a remote fashion. One of the most promising Open-Source project called OpenStack is another key for any enterprise to convert they internal IT infrastructure to a private cloud in the form of an Infrastructure as a Service (IaaS). Thus enable the possibility to uncouple the application of the computing resources and at the same time uses the internal resources of the enterprise. Any application designed to run on top of a cloud should be able to virtually run everywhere a cloud is available. If such application is split into components, they can potentially run on multiple clouds in parallel. The main advantage of such approach is that the service can scale-up scale-down based on the real-time conditions and business rules (actual pricing, load of the private servers, geographical location of end-users, ...).

This project, developed in collaboration with the European Broadcasting Union (EBU/UER) will consist of an Open-Source *demonstrator* in the form of :

1. A minimum setup of 4 machines running OpenStack to provide a private IaaS to the *application*
2. A scalable *application* able to run on top of the private cloud and able to scale-up to the public clouds (Amazon AWS, HP Cloud, Rackspace)
3. A set of nice to have *future extensions*, one of them is adding MPEG-DASH encoding capabilities to the platform

MPEG-DASH ('Dynamic Adaptive Streaming over HTTP') is the upcoming standard for online video deliverance to multiple devices. In this upcoming standard different profiles are described and it seems that the live profile has the biggest potential to be picked up in the market. The goal of the corresponding extension is the addition of MPEG-DASH encoding capabilities to the *demonstrator* in order to popularize MPEG-DASH.

Specifications

See also:

Please see *Project Specifications* for further details.

- Project specifications refinement
- Setup the private cloud with OpenStack
- Design & implement the Open-Source demonstrator

Prerequisites

- Strong knowledges of GNU/Linux & Ubuntu system administration
- Excellent software architecture skills are required
- Good knowledges of the multimedia transmission standards & codecs
- Understand the constraints of the real-time streaming over IP

Keywords

TIC IPTV; TIC Multimedia; TIC Video Streaming; TIC Digital TV

Multimedia

Adaptive streaming over HTTP HTTP-based adaptive bit-rate streaming technologies. Such technologies are specifically designed in order to provide to client a way to handle network conditions variations [1] by continuously selecting an optimized bit-rate representation of the multimedia content. The delivery server must provides multimedia content encoded on multiple representations with specific bitrate and resolution.

Broadband streaming Multimedia content delivery through a broadband network, most (if not all) of the streaming technologies are based on the IP stack of the Internet protocols.

Broadcast streaming Multimedia content delivery through a broadcast network, such networks are designed to provide an unidirectional way to transmit productions from an unique source to the mass. Classically, multimedia content is encapsulated in MPEG-2 TS packets delivered by Digital Video Broadcasting systems.

Linear multimedia content Multimedia content intended to be viewed in real-time, the audience will consume this type of content linearly from the beginning to the end. Typical applications : TV channels, books.

Non-linear multimedia content Multimedia content that can be consumed on a non-linear way. The client can seek freely on such type of content. Typical applications : Video on demand, video games, ...

Media transcoding Transcoding is the direct digital-to-digital data conversion of one encoding to another.

Representation A representation is a specific content encoded on a specific parameter set (quality, geometry, bitrate, ...).

Operational & Cloud

DevOps DevOps is a new term describing what has also been called “agile system administration” or “agile operations” joined together with the values of agile collaboration between development and operations staff.

Hypervisor In computing, a hypervisor or virtual machine manager (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines.

Elasticity ” Elasticity applied to computing can be thought as the amount of strain an application or infrastructure can withstand while either expanding or contracting to meet the demands place on it. ” [2]

Web Services Web services are typically application programming interfaces (API) or Web APIs that are accessed via Hypertext Transfer Protocol (HTTP).

[1] Others metrics can be user at client side : Screen resolution, computational power ...
[2] Defining Elastic Computing - http://www.elasticvapor.com/2009/09/defining-elastic-computing.html

JuJu from Canonical Ltd.

Bootstrap To bootstrap an environment means initializing it so that Services may be deployed on it.

Endpoint The combination of a service name and a relation name.

Environment An Environment is a configured location where Services can be deployed onto.

Charm A Charm provides the definition of the service, including its metadata, dependencies to other services, packages necessary, as well as the logic for management of the application.

Repository A location where multiple charms are stored. Repositories may be as simple as a directory structure on a local disk, or as complex as a rich smart server supporting remote searching and so on.

Relation Relations are the way in which juju enables Services to communicate to each other, and the way in which the topology of Services is assembled. The Charm defines which Relations a given Service may establish, and what kind of interface these Relations require.

Service juju operates in terms of services. A service is any application (or set of applications) that is integrated into the framework as an individual component which should generally be joined with other components to perform a more complex goal.

Service Unit A running instance of a given juju Service. Simple Services may be deployed with a single Service Unit, but it is possible for an individual Service to have multiple Service Units running in independent machines. All Service Units for a given Service will share the same Charm, the same relations, and the same user-provided configuration.

Service Configuration There are many different settings in a juju deployment, but the term Service Configuration refers to the settings which a user can define to customize the behavior of a Service. The behavior of a Service when its Service Configuration changes is entirely defined by its Charm.

Provisioning Agent Software responsible for automatically allocating and terminating machines in an Environment, as necessary for the requested configuration.

Machine Agent Software which runs inside each machine that is part of an Environment, and is able to handle the needs of deploying and managing Service Units in this machine.

Service Unit Agent Software which manages all the life-cycle of a single Service Unit.

CHAPTER

ONE

INTRODUCTION

1.1 Context

The Internet is growing in importance for broadcasters as delivery system for video and video related services to their audience. It allows broadcasters to deliver content directly to end-users and interact with them via interfaces. A downside to this story is the fact that distributing content over the Internet is very expensive for broadcasters.

One of the challenges they face is the transcoding of their on demand content libraries to new file formats that are optimized for multi-screen consumption. Fragmentation of media devices in technical capability (which codec settings they can display) or screen size (aspect ratio and pixel size of the video). Normally they need to encode to at least 8 file representations of the same video. This process is a constant factor when the daily production of content is contributed but also knows peaks in cases when libraries need to be transcoded.

Another challenge is the scaling of distribution servers. This process is dominated by changing demands because traffic peaks during the day, is at a minimum at night and sometimes when a video becomes a hit it will peak also.

Both these challenges are met by an encoding (or transcoding) and distribution environment that can up- or down-scale capacity easily. Ideally a distribution environment is downscaled at night and only upscaled at peak events. A transcoding environment needs to be upscaled when more transcoding jobs are waiting.

The OSCIED (Open-Source Cloud Infrastructure for Encoding and Distribution) project that is described in this paper addresses exactly these use cases.

I build a cloud-aware platform that can up-or down-scale transcoding or distribution nodes in a private (local servers) or in a public cloud (like Amazon AWS). Made possible because platform's functionalities are split into components and therefore can be deployed on multiple clouds in parallel and even more !

This environment will allow all kinds of other interesting functionalities for content providers as broadcasters. The encoding/transcoding can easily make new codecs available, you can optimize costs by up scaling in the night when the cloud computing resources are cheaper, encoding of live-content can be added etc. etc. From the distribution side the media gateways of different cloud providers can be used to cache content closer to the end user and with that optimize the data flows, or add different types of streaming, or define edges in different CDN's and use OSCIED as a CDN overlay. The system as a whole can grow into a full fledged publication platform with professional management layers (which I started with already) that can publish, revoke but perhaps in the future also use cloud computing resources to deliver personalized transformations of the content deep into the network close to the end user.

Open-Source software has been chosen because anyone have access to the level of the source code and it allows other developers to build upon the work done. From the broadcast community and beyond already interest is shown to invest in my approach. Furthermore if the main components of the virtualized services for transcoding (FFmpeg) of distribution (Apache 2) have a new LibDASH or version that is made available by their specific Open-Source development community it can be integrated easily in the system.

1.2 Use Cases and Needs

Here will be explained the use cases this project focused-on such as the transcoding and online delivery of medias.

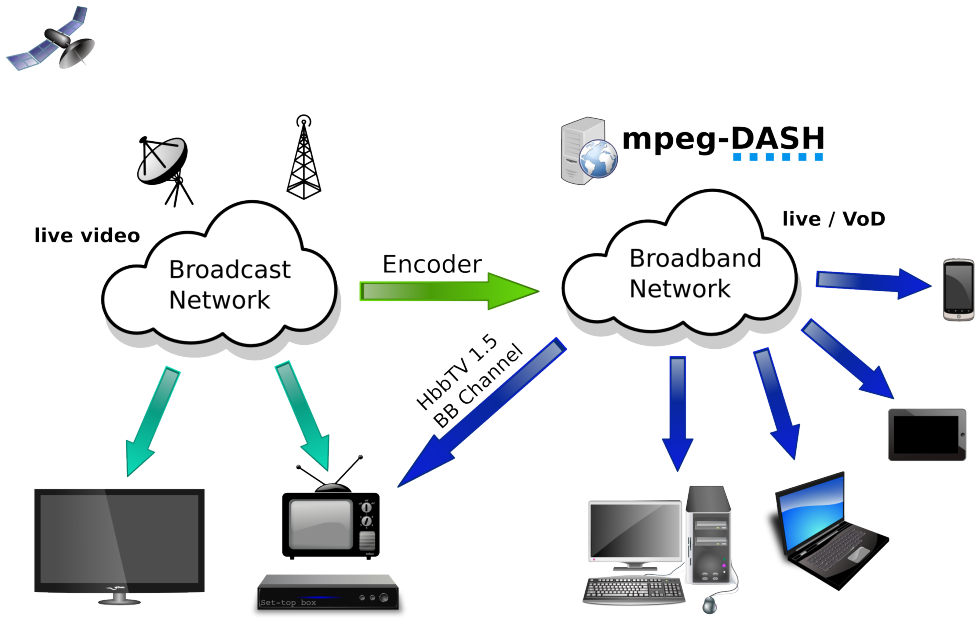


Figure 1.1: Multimedia content delivery through broadcast and broadband networks

1.2.1 Transcoding

> ” ... Transcoding is the direct digital-to-digital data conversion of one encoding to another, such as for movie data files or audio files. This is usually done in cases where a target device (or workflow) does not support the format or has limited storage capacity that mandates a reduced file size, or to convert incompatible or obsolete data to a better-supported or modern format. ... ” source Wikipedia (Transcoding)

A problematic use case of transcoding is when you need to transform a large collection of medias from a format to another. As this process is heavily demanding in computing resources and this requirement will increases with the every increasing quality of the content produced by cameras.

Transcoding is a never ending process as the media formats and AV codecs are evolving rapidly. Luckily the storage needs are partially balanced by the increasing compression efficiency of newer codecs. [1] For example, first version of High Efficiency Video Coding has just been released this January 2013. [2]

Another typical use case is the encoding (aka. digitalization) of archival materials to file based formats. This application needs specific equipments that are out of the preliminary demonstrator’s specifications.

Note: At European Broadcasting Union (EBU/UER) it is actually a dedicated digitalization project called Transition to File where broadcasters are helped to migrate from tape based to file based environments.

1.2.2 Online Delivery

> ” ... Digital distribution ... describes the delivery of media content such as audio, video, software and video games, without the use of physical media usually over online delivery mediums, such as the Internet. ... With the advancement of network bandwidth capabilities, digital distribution became

[1] This sentence is partially true as the older versions may need to be keep in storage.

[2] Article H.265 standard finalized - http://www.extremetech.com/extreme/147000-h-265-standard-finalized-could-finally-replace-mpeg-2-and-usher-in-uhdtv

> prominent in the 2000s. Content distributed online may be streamed or downloaded. Streaming involves downloading and using content "on-demand" as it is needed. ... Specialist networks known as content delivery networks help distribute digital content over the Internet by ensuring both high availability and high performance. ... " source Wikipedia (Distribution)

A problematic use case of online delivery is the publication of media related to high-audience events such as online TV journal with online *breaking* news. The other problematic use case is the every-growing demand of VoD content, as the computing resources can't be as easy scheduled as in any classical linear program.

Next generation connected televisions pushes the usage of broadband content delivery and new services related to that are emerging. Specifically the added possibility for HbbTV users to consume new programs using their Internet connection in parallel to the main broadcast.

TV to Internet : During latest Olympic Games the broadcasted *main* program showed the competitions split in small parts of few minutes. The audience could choose to continue watching one specific discipline by connecting the the Official Website and clicking on the media to play it.

HbbTV BC to BB : When a program (eg. Tennis) is longer than expected, an advertisement is overlayed to explain that the broadcasting of this program will be stopped and next scheduled TV program will be launched soon. The user can choose to continue the *standard* program or he can continue watching the end of the competition (e.g. Tennis) by clicking on the red button. This later choice means that the TV will display content delivered through the Internet.

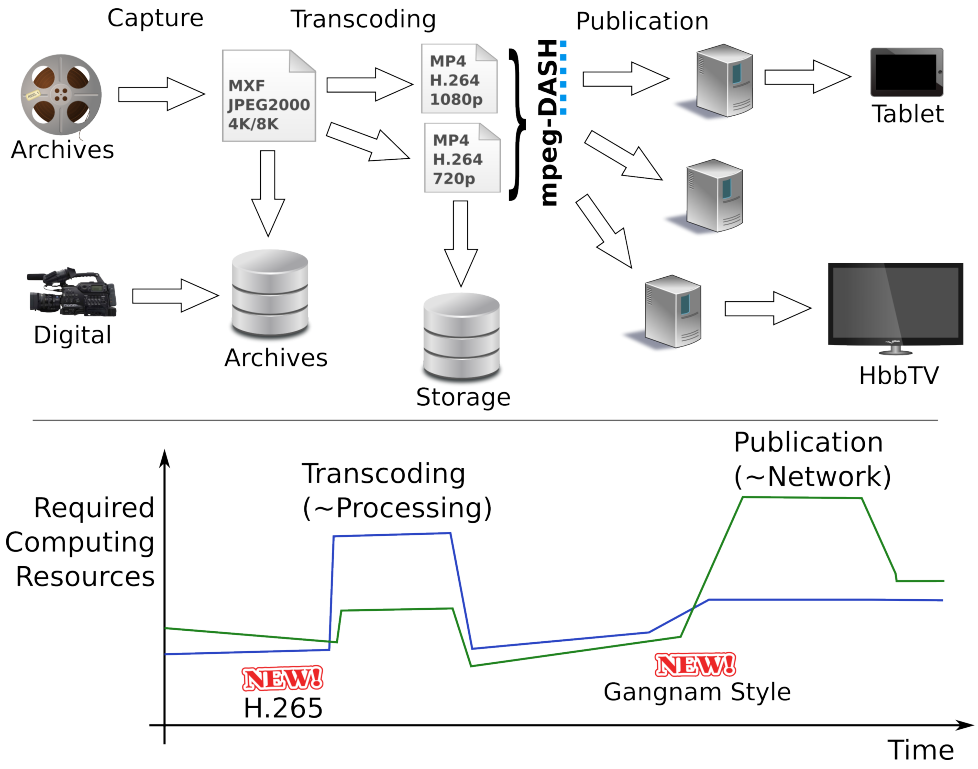


Figure 1.2: Simplified overview of the workflow the project focused-on.

1.2.3 Needs

These use cases are large libraries that needs to be encoded and transcoded all the time (due to new file formats and services). Therefore an adaptive elastic environment is needed that can migrate files to new file formats at certain moments. Virtualization makes it possible to add computing resources when needed and change codecs when needed.

For example when broadcasters want to migrate their library to MPEG-DASH they normally will buy new encoders that support the new file format. However it is much more efficient to rent virtual servers in the cloud during night blocks for the period of the job.

For distribution a flexible environment is great to upscale Apache 2 servers when a lot of traffic is expected instead of having to support a full park that is scaled for a peak event.

... Typically, the computing scalability issues can be solved :

1. By upgrading enterprise's IT infrastructure (the servers room) :
   - Adding new servers to racks ;
   - Replacing servers with more powerful models ;
2. By using appropriate third party services :
   - Content delivery networks (CDN) e.g. Akamai ;
   - Cloud video transcoding platform (SaaS) e.g. Zencoder ;
3. By using cloud provider's resources as a service (IaaS) :
   - Consuming computing resources, e.g. Amazon Elastic Clod Compute (EC2) ;
   - Consuming delivery resources, e.g. Amazon Cloud Front ;
   - Consuming storage resources, e.g. Amazon Simple Storage Service (S3) ;

We can separate any varying workload in two main parts, the *constant* and the *varying* parts.

The *constant* part is related to the solution 1. The most common approach used in any enterprise requiring computing resources. This approach is interesting if the enterprise's workload is quite stable.

The *varying* part is related to any of the remaining solutions, 2 or 3. One typical approach adopted by Broadcasters is to uses services offered by CDN to improve online delivery capacity. The content will be cached by CDN servers closer to the end-user, decreasing networks load and increasing audience QoE.

The third approach is quite promising but not as used as the others (at least by Broadcasters).

This is more difficult to consume these type of services as they are designed to be as generic as possible [3]. The resources are provided through programmatic APIs (Web Services) and the computing resources needs to be *configured* before any *real* usage.

1.3 Proposed Solution

1.3.1 Foreword

The 20-21 November 2012 I attended to a workshop organized by the European Broadcasting Union (EBU/UER) called EBU Cloud Workshop 2012.

From this two days interactive workshop we gather the needs of the broadcasters regarding the cloud. To sum-up what we learned, here is a modified extract of the *Broadcasters Wishlist* for the cloud(s), thanks to Félix Poulin from EBU.

1. Easy to bring back & leave (right to be forgotten)
2. High availability (sometimes 100%)
3. QoS guarantees (depending on the needs)
4. Cost effective (as cheap as in-house)
5. Measurable performances (SLAs)
6. Fast uploads & downloads
7. Customizable & extensible
8. Access on any device
9. Media file aware

[3] I speak about IaaS where computing resources are provided in the form of virtual machines instances.

10. Open Standards

Here are the wishes, regrouped by topic :

1. 1-5 are related to the usage of public-cloud only multimedia platforms
2. 6 is related to the performance of the networks involved for transfer of files
3. 7-10 are related to the multimedia platform itself (the software)

Proposed solution can potentially solve :

- Topic **1** by using private and public clouds at the same time (called hybrid cloud)
- Topic **2** by using in-house private-cloud and scaling to public-cloud when necessary
- Topic **3** by developing an application based on the needs of the broadcasters

1.3.2 The Idea

" Open-Source Cloud Infrastructure for Encoding and Distribution où le Cloud Maîtrisé ! "

The following project, based on cloud-era Open-Source technologies, can potentially fix this scalability issue by providing a rather simple but yet powerful way to consume already existing enterprise's IT resources mixed with necessary amount of public cloud resources !

In proposed solution, a set of the enterprise servers are dedicated to run the Open-Source IaaS called OpenStack. The proposed Open-Source application is then deployed on this setup running the enterprise's private cloud. The application can be scaled-up to any compatible cloud provider in case of workload increasing [4]. Of course the scale-down of the application can be achieved as easily as the scale-up of it !

The main advantage of this solution :

- Is (itself) and is based on Open-Source technologies, no vendor lock-in, community driven developments ;
- Is fast and easy to scale, necessary setup/configuration is handled by the application [5] ;
- Is divided in scalable and (potentially) interchangeable components [6] ;
- Is not only compatible with cloud layers, components can run on standalone hardware [7] ;
- The private cloud can run enterprise's services in parallel to proposed application ;
- Future extensions are already imagined, it is just matter of time and users requirements [8] ;

1.4 Structure of the Report

The report is organized as follow :

- **State of the Art** this chapter gives a short overview of :
  - Cloud-based solutions from dedicated multimedia platforms (SaaS) to more general cloud-based services (IaaS).
  - Most interesting Open-Source technologies that makes the building of proposed solution a reality.
- **Project Specifications** this chapter details the specifications of the project. This chapter begins by exposing the goal of the project and the work to do during the thesis. It also describes the organization of the project in high-level, long-term developments cycles. Then more details are given about the first cycle, this thesis, in the form of development phases describing the high-level tasks to complete during the cycle. Finally, this chapter ends with details about the planning and the management tools used during this thesis.

[4] This scenario describes one possible workflow, please see *Deployment Scenarios*.
[5] Thanks to Canonical's JuJu, please see *Clouds Orchestration Layer* for further details.
[6] Thanks to application's design, please see *Application Layer* for further details.
[7] Thanks to application's charms setup hooks and dev scripted tricks.
[8] Please see *Future Extensions* for an overview of future extensions.

- **OSCIED Demonstrator** this chapter introduce you with the demonstrator in the form of logical layers, from physical servers to developed application. Eeach of those layers is the described in dedicated sections of the chapter. The section about the application shows and details the main components of developed application. A FAQ about the demonstrator is also available in this chapter. Finally, the chapter ends with few example deployment scenarios (including example configuration files) and the tests and results section.
- **Conclusion** this chapter summarize features of OSCIED that are already available for broadcasters and what would be the most interesting features to integrate to developed platform.

CHAPTER

TWO

STATE OF THE ART

2.1 Contents

This chapter gives a short overview of :

- Cloud-based solutions from dedicated multimedia platforms (SaaS) to more general cloud-based services (IaaS).
- Most interesting Open-Source technologies that makes the building of proposed solution a reality.

2.2 Overview of Cloud Based Multimedia Platforms

2.2.1 Quantel QTube

Official Description

> ” QTube is game-changing software that enables content creators, administrators and managers to interact with their content wherever they are and wherever their content is located. QTube is already in daily use transforming content creation in the same way, globalizing workflows for media organizations of all sizes around the world. ” [1]

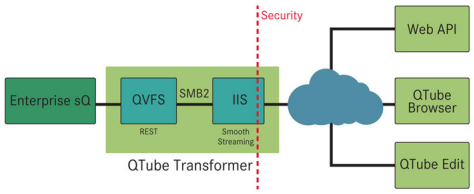


Figure 2.1: QTube workflow over IP, Copyright : Quantel

2.2.2 Zencoder

Official Description

> ” Zencoder is cloud-based video and audio encoding software as a service (SaaS). We have a wide range of customers, from individuals to Fortune 500 enterprise corporations, who all need to automate the encoding process through our encoding API. Because we’re based in the cloud, it means you have access to unlimited video encoding power, without having to pay for, manage, and scale expensive hardware and/or software. If you have any more questions feel free to contact us. ” [2]

[1] Quantel About QTube – http://uk1.quantel.co.uk/page.php?u=fc919cb52f36247c19a4fdade742b8ce
[2] Zencoder Home Page – http://zencoder.com/en/

Figure 2.2: Zencoder live transcoding (beta), Copyright : Zencoder Inc.

2.3 Overview of Compatible Cloud Providers

2.3.1 Amazon Web Services

This is the public cloud provider I used during this project.

Official Description

> ” Amazon Web Services offers a complete set of infrastructure and application services that enable you to run virtually everything in the cloud: from enterprise applications and big data projects to social games and mobile apps. One of the key benefits of cloud computing is the opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with your business. ” [3]

Figure 2.3: Amazon Web Services Logo, Copyright : Amazon

2.3.2 HP Cloud

Official Description

> ” HP Cloud Services is committed to delivering leading edge public cloud infrastructure, platform services, and cloud solutions for developers, ISVs, partners, service providers, and enterprises. HP Cloud Compute and HP Cloud Object Storage are built on HP’s world class hardware and software, with key elements of HP Converged Infrastructure and a developer-friendly integration of OpenStack technology. HP’s use of OpenStack technology and participation in its Open-Source project means that you get innovative, Open-Source-based cloud technology. This also means HP will be at the forefront of public cloud development and advancement as the OpenStack project evolves to deliver massively scalable applications. ” [4]

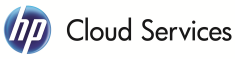


Figure 2.4: HP Cloud Services Logo, Copyright : HP

2.3.3 Rackspace Cloud

Official Description

> ” When you sign up for the Rackspace Cloud, you get access to all the tools you need to make your website or application a reality. Plus, enjoy convenient monthly pricing, and only pay for what you use

[3] Amazon Web Services Home Page – http://aws.amazon.com/
[4] HP Cloud Home Page – https://www.hpcloud.com/

(plus a monthly fee for managed cloud accounts). For example, Cloud Servers provides compute for your sites and apps. You get the persistence of a traditional server, plus the on-demand flexibility of the cloud. Because it delivers the computing power for your cloud configuration, your Cloud Servers are the heart of your Rackspace Cloud environment. Connect them to your data stored on Cloud Files or Cloud Block Storage, your Cloud Databases, and more. Use them with Cloud Load Balancers to deliver high availability. Plus, all Cloud Servers customers get free access to Cloud DNS, for easy management of your DNS records. ” [5]

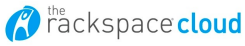


Figure 2.5: Rackspace Cloud Logo, Copyright : Rackspace

2.4 Overview of OSS Hypervisor Technologies

2.4.1 Linux KVM

This is the hypervisor I have chosen for running virtual machines on OpenStack.

Official Description

” KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM also requires a modified QEMU although work is underway to get the required changes upstream.

Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

The kernel component of KVM is included in mainline Linux, as of 2.6.20.

KVM is Open-Source software. ” [6]

Figure 2.6: KVM Logo, Copyright : Open Virtualization Alliance

2.4.2 LXC

This is the OS-level virtualization technology used by JuJu for running charms locally.

Official Description

” LXC is the userspace control package for Linux Containers, a lightweight virtual system mechanism sometimes described as “chroot on steroids”.

LXC builds up from chroot to implement complete virtual systems, adding resource management and isolation mechanisms to Linux’s existing process management infrastructure.

Linux Containers (lxc) implement:

[5] Rackspace Cloud Products Page – http://www.rackspace.com/cloud/products/
[6] KVM Main Page – http://www.linux-kvm.org/page/Main_Page

> Resource management via "process control groups" (implemented via the cgroup filesystem) Resource isolation via new flags to the clone(2) system call (capable of create several types of new namespaces for things like PIDs and network routing) Several additional isolation mechanisms (such as the "-o newinstance" flag to the devpts filesystem).

The LXC package combines these Linux kernel mechanisms to provide a userspace container object, a lightweight virtual system with full resource isolation and resource control for an application or a system.

Linux Containers take a completely different approach than system virtualization technologies such as KVM and Xen, which started by booting separate virtual systems on emulated hardware and then attempted to lower their overhead via paravirtualization and related mechanisms. Instead of retrofitting efficiency onto full isolation, LXC started out with an efficient mechanism (existing Linux process management) and added isolation, resulting in a system virtualization mechanism as scalable and portable as chroot, capable of simultaneously supporting thousands of emulated systems on a single server while also providing lightweight virtualization options to routers and smart phones.

The first objective of this project is to make the life easier for the kernel developers involved in the containers project and especially to continue working on the Checkpoint/Restart new features. The lxc is small enough to easily manage a container with simple command lines and complete enough to be used for other purposes. " [7]

2.4.3 OpenVZ

This is the virtualization technology I chose years ago for running servers of telecommunications laboratory at hepia.

Official Description

" OpenVZ is container-based virtualization for Linux. OpenVZ creates multiple secure, isolated Linux containers (otherwise known as VEs or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict. Each container performs and executes exactly like a stand-alone server; a container can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files. For more information about the technology and how it differs from the others like Xen, VMware etc.

OpenVZ software consists of an optional custom Linux kernel and command-line tools (mainly vzctl). Our kernel developers work hard to merge containers functionality into the Linux kernel, making OpenVZ team the biggest contributor to Linux Containers (LXC) kernel, with features such as PID and network namespaces, memory controller, checkpoint-restore etc. While OpenVZ can be used with recent upstream kernel, we recommend using OpenVZ kernel for security, stability and features.

OpenVZ is free Open-Source software, available under GNU GPL. " [8]

Figure 2.7: OpenVZ Logo, Copyright : Parallels

2.5 Overview of OSS Private Cluster/Cloud IaaS

2.5.1 Proxmox VE

This is the platform I choose years ago for running servers of telecommunications laboratory at hepia.

[7] LXC Main Page – http://lxc.sourceforge.net/
[8] OpenVZ Main Page – http://openvz.org/Main_Page

Official Description

"Proxmox VE is a complete virtualization management solution for servers. You can virtualize even the most demanding application workloads running on Linux and Windows Servers. It is based on the leading Kernel-based Virtual Machine (KVM) hypervisor and OpenVZ, the number one solution for container based virtualization. The best alternative to organizations looking for better total cost of ownership (TCO) and no vendor lock-in. " [9]

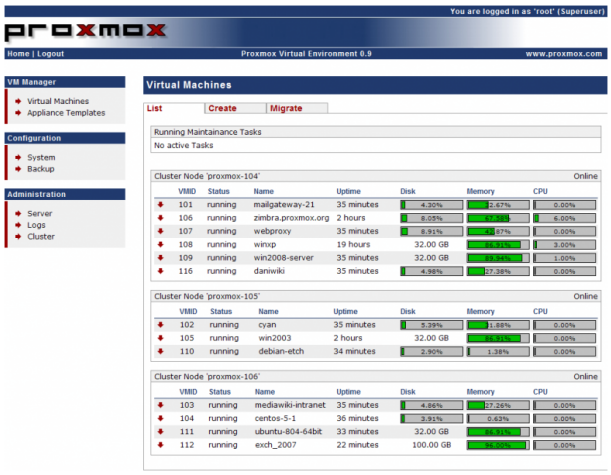


Figure 2.8: Integrated console view to the Virtual Machines, Copyright : YahyaNursalim

2.5.2 CloudStack

Official Description

"Apache CloudStack (Incubating) is Open-Source software designed to deploy and manage large networks of virtual machines, as a highly available, highly scalable Infrastructure as a Service (IaaS) cloud computing platform. CloudStack is used by a number of service providers to offer public cloud services, and by many companies to provide an on-premises (private) cloud offering, or as part of a hybrid cloud solution.

CloudStack is a turnkey solution that includes the entire "stack" of features most organizations want with an IaaS cloud: compute orchestration, Network-as-a-Service, user and account management, a full and open native API, resource accounting, and a first-class User Interface (UI).

CloudStack currently supports the most popular hypervisors: VMware, KVM, XenServer and Xen Cloud Platform (XCP).

Users can manage their cloud with an easy to use Web interface, command line tools, and/or a full-featured RESTful API. In addition, CloudStack provides an API that's compatible with AWS EC2 and S3 for organizations that wish to deploy hybrid clouds." [10]

[9] ProxmoxVE Main Page – http://www.proxmox.com/products/proxmox-ve
[10] CloudStack Main Page – http://incubator.apache.org/cloudstack/

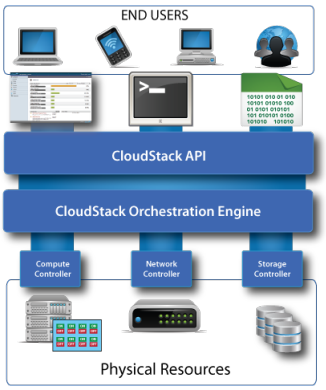


Figure 2.9: CloudStack conceptual infrastructure, Copyright : Apache Foundation

2.5.3 OpenStack

This is the Infrastructure as a Service I have chosen for this project.

Official Description

> ” OpenStack is an Infrastructure as a Service (IaaS) cloud computing project that is free Open-Source software released under the terms of the Apache License. The project is managed by the OpenStack Foundation, a non-profit corporate entity established in September 2012 to promote, protect and empower OpenStack software and its community.
>
> More than 150 companies have joined the project among which are AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP, IBM, NEC, VMware and Yahoo!. It is portable software, but is mostly developed and used on the Linux operating system.
>
> The technology consists of a series of interrelated projects that control large pools of processing, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.
>
> OpenStack is committed to an open design and development process. The community operates around a six-month, time-based release cycle with frequent development milestones. During the planning phase of each release, the community gathers for the OpenStack Design Summit to facilitate live developer working sessions and assemble the roadmap. ” source Wikipedia (OpenStack_IaaS)

Figure 2.10: OpenStack Logo, Copyright : OpenStack Foundation

2.6 Overview of OSS Cloud Orchestration Tools

2.6.1 HP CloudSystem

> ” Everyone has their own vision of how cloud computing will solve business problems. Why should you choose ours? HP delivers the most complete integrated system for enterprise and service providers to build and manage services across private, public and hybrid cloud environments.

- Unmatched automation & and orchestration
- The broadest support of applications, leading hypervisors, and operating systems
- Unified services management across cloud & and traditional IT
- Advanced application deployment, intelligent resource, & advanced configuration management
- Secure, scalable and extensible solutions built on proven and market leading Converged Infrastructure and Cloud Service Automation
- Integrated and automated application to infrastructure management for the cloud

HP CloudSystem is built on proven HP Cloud Service Automaton and Converged Infrastructure technologies. With support for the broadest set of applications, CloudSystem provides IT with a unified way to offer, provision and manage services across private clouds, public cloud providers, and traditional IT. It enables the flexibility to scale capacity within and outside your data center. And it's extensible to your existing IT infrastructure and can support heterogeneous environments. " [11]

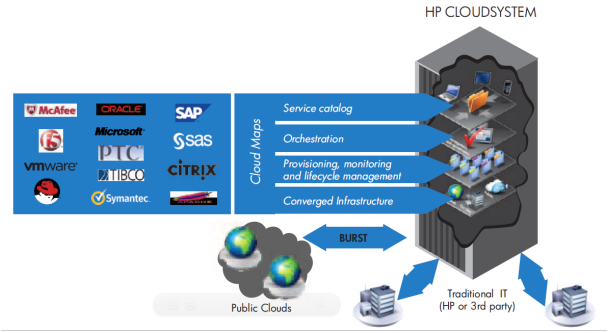


Figure 2.11: HP CloudSystem automation, Copyright : En Pointe

2.6.2 SlipStream

Official Description

" Automated provisioning and creation of cloud resources. SlipStream™ provides a simpler access to clouds, yet lets users do much more. For example automated, on-demand, creation of multi-machine runtime environments and version control the creation of custom machine images based on certified base images. SlipStream™ can also be used as your software engineering PaaS solution, as product or service. SlipStream will soon be released under an Open-Source license. This means that all SixSq cloud solutions will be available under a coherent Open-Source license.

Our customers use SlipStream™ to:

- Provision pre-certified production systems, as part of an overall vertical solution
- Cluster provisioning in the cloud - e.g. pre-configured clusters of user defined sizes
- Version control the creation of reference images, on which to base virtual machine deployments
- Software engineering Platform as a Service to speed-up project inception with provisioning of development tools (e.g. Jenkins/Hudson, Yum repository, Maven, Nexus)
- Single access to federated cloud, where users can switch between clouds yielding identical results on each

We are constantly amazed by new ways our customers come-up with using SlipStream™. If this is your case, please share them with us. SlipStream™ currently supports Amazon EC2 and StratusLab. We are actively adding several more, which will be announced soon. If you would like your cloud to be supported, please drop us a line. " [12]

[11] En Pointe HP CloudSystem – http://www.enpointe.com/hp/cloud
[12] SixSq. SlipStream Page – http://sixsq.com/products/slipstream.html

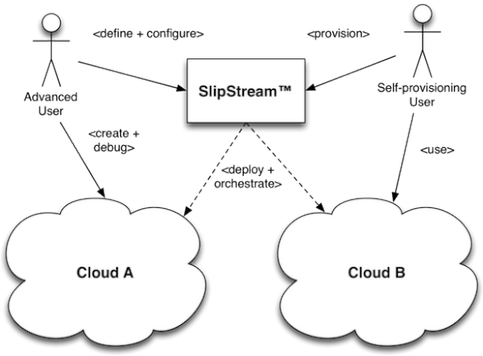


Figure 2.12: Shows how different user profiles can interact with SlipStream, Copyright : SiqSq.

2.6.3 JuJu

This is the cloud orchestration technology I have chosen for this project.

Official Description

" JuJu (formerly Ensemble) is a service orchestration management tool developed by Canonical Ltd.. It is an Open-Source project hosted on Launchpad released under the GNU Affero GPLv3. JuJu concentrates on the notion of service, abstracting the notion of machine or server, and defines relations between those services that are automatically updated when two linked services observe a notable modification. This allows for services to very easily be up and down scaled through the call of a single command. For example, a web service described as a JuJu charm that has an established relation with a load balancer can be scaled horizontally with a single `juju add-unit` call, without having to worry about re-configuring the load-balancer to declare the new instances: the charm's event based relations will take care of that. JuJu's charms can be written in any executable language.

What is JuJu ?

- Is DevOps Distilled. Through the use of charms, JuJu provides you with shareable, re-usable, and repeatable expressions of DevOps best practices. You can use them unmodified, or easily change and connect them to fit your needs. Deploying a charm is similar to installing a package on Ubuntu: ask for it and it's there, remove it and it's completely gone. With over 100 services ready to deploy, JuJu enables you to build entire environments in the cloud with only a few commands on public clouds like Amazon AWS, HP Cloud and Rackspace, to private clouds built on OpenStack, or raw bare metal via MAAS.
- Is a community of DevOps expertise. Most of the applications you want will be available in JuJu thus provides direct and free access to a DevOps community-contributed collection of charms.
- Provides service orchestration. JuJu focuses on managing the service units you need to deliver a single solution, above simply configuring the machines or cloud instances needed to run them. Charms developed, tested, and deployed on your own hardware will operate the same in an EC2 API compatible cloud.
- Is intelligent. JuJu exposes re-usable service units and well-defined interfaces that allow you to quickly and organically adjust and scale solutions without repeating yourself.
- Is easy. There's no need to learn a domain specific language (DSL) to use JuJu or create charms. You can be up and running with your own charm in minutes.

JuJu GUI Live Demo Available here : http://uistage.jujucharms.com:8080/ " source Wikipedia (JuJu_stoftware) + [13]

[13] JuJu FAQ Page – https://juju.ubuntu.com/docs/faq.html#why-is-juju-useful

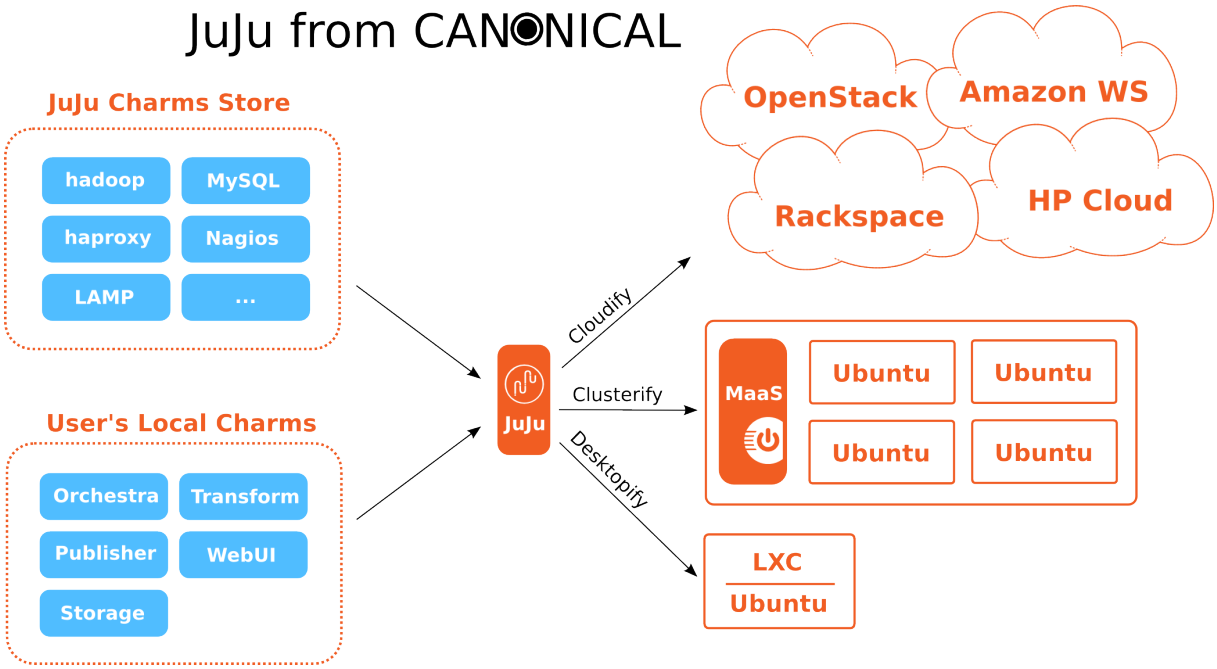


Figure 2.13: JuJu can deploy charms to a wide variety of environments, why not implementing your own provider ? Copyright : David Fischer (License : CC)

2.7 Overview of OSS Storage Technologies

See also:

Please see Ticket 122 for further interesting links.

2.7.1 Swift

Official Description

" The OpenStack Object Store project, known as Swift, offers cloud storage software so that you can store and retrieve lots of data in virtual containers. It's based on the Cloud Files offering from Rackspace.

When you install Swift, you can install multiple copies services that will track and retrieve the objects you want to store. Here's a description of what you get with OpenStack Object Store:

- object server that stores objects (files less than 5 GB currently, support for large objects is in the works)
- a container server that keeps track of the objects
- a proxy server that handles all requests from the other server
- an authorization server so that your cloud storage is contained and authorized
- an account server that keeps track of all the containers
- Since Rackspace already has this system in production, we share our configuration but you can determine your own best performance and availability based on your hardware and networking capabilities. " [14]

[14] Swift Wiki Page – http://wiki.openstack.org/Swift

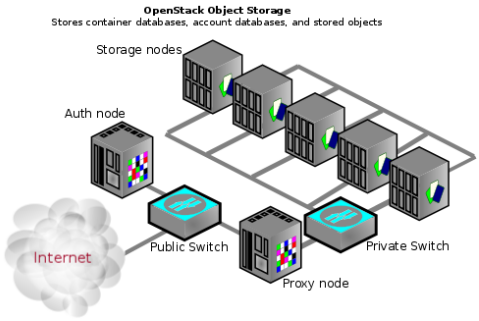


Figure 2.14: Example Swift installation architecture, Copyright : OpenStack Foundation

2.7.2 Ceph

Official Description

" The power of Ceph can transform your organization's IT infrastructure and your ability to manage vast amounts of data. If your organization runs applications with different storage interface needs, Ceph is for you! Ceph's foundation is the Reliable Autonomic Distributed Object Store (RADOS), which provides your applications with object, block, and file system storage in a single unified storage cluster—making Ceph flexible, highly reliable and easy for you to manage. Ceph's RADOS provides you with extraordinary data storage scalability-thousands of client hosts or KVMs accessing petabytes to exabytes of data. Each one of your applications can use the object, block or file system interfaces to the same RADOS cluster simultaneously, which means your Ceph storage system serves as a flexible foundation for all of your data storage needs. You can use Ceph for free, and deploy it on economical commodity hardware. Ceph is a better way to store data. " [15]

Figure 2.15: Ceph Logo, Copyright : Inktank Storage Inc.

2.7.3 GlusterFS

This is the storage technology I have chosen based on my *Decision Matrix*.

Official Description

" GlusterFS is an Open-Source, distributed file system capable of scaling to several petabytes (actually, 72 brontobytes!) and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design and can deliver exceptional performance for diverse workloads.

GlusterFS supports standard clients running standard applications over any standard IP network. Figure 1, above, illustrates how users can access application data and files in a Global namespace using a variety of standard protocols.

No longer are users locked into costly, monolithic, legacy storage platforms. GlusterFS gives users the ability to deploy scale-out, virtualized storage – scaling from terabytes to petabytes in a centrally managed and commoditized pool of storage.

[15] Ceph Storage Home Page – http://ceph.com/ceph-storage/

Attributes of GlusterFS include:

- No limit on files sizes as compared to 5GB object size limit of OpenStack Swift [16]
- A unified view of data across NAS and Object Storage technologies
- Scalability and Performance
- High Availability
- Global Namespace
- Elastic Hash Algorithm
- Elastic Volume Manager
- Gluster Console Manager
- Standards-based
- Geo Replication " [17] [18]

Figure 2.16: GlusterFS Logo, Copyright : Gluster

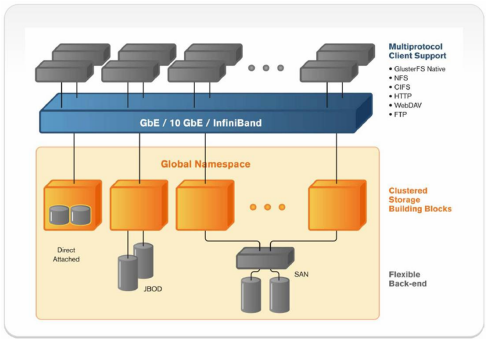


Figure 2.17: Gluster Open-Source Scalable NAS Implementation, Copyright : Gluster

2.7.4 LS4

Official Description

" LS4 is an Open-Source distributed storage system designed to store objects like photos, musics or movies.

- LS4 stores a set of objects identified by a key. Each object consists of data and attributes where data is a raw bytes and attributes are associative pairs. Objects are distributed to servers for scalability and copied on multiple servers for availability
- Each object can have multiple versions
- Replica set is a set of data servers that stores same objects
- Fail-back can be done without stopping the cluster

[16] GlusterFS + OpenStack – http://www.gluster.org/wp-content/uploads/2011/07/Gluster-Openstack-VM-storage-v1-shehjar.pdf
[17] GlusterFS About Page – http://www.gluster.org/about/
[18] GlusterFS Admin Guide – http://www.gluster.org/wp-content/uploads/2012/05/Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf

- With LS4 and nginx, contents can be transferred without passing through application servers while the application server proceeds HTTP requests. Thus you can reduce CPU load and network traffics. It's implemented using nginx's X-Accel-Redirect feature. See the HowTo to configure the bypass. Additionally, LVS's Direct Routing may be useful on the proxy
- You can configure LS4 to replicate data over remote datacenters while applications get data from the local datacenter " [19]

Figure 2.18: LS4 Logo, Copyright : FURUHASHI Sadayuki

2.7.5 NFS

Official Description

" Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol. " Source Wikipedia (NFS)

2.7.6 MongoDB

Official Description

" MongoDB (from "hu**mongo**us") is a scalable, high-performance, Open-Source NoSQL database. Written in C++, MongoDB features:

- Document-Oriented Storage – JSON-style (BSON) documents with dynamic schemas offer simplicity and power
- Full Index Support – Index on any attribute, just like you're used to
- Replication & High Availability – Mirror across LANs and WANs for scale and peace of mind
- Auto-Sharding – Scale horizontally without compromising functionality
- Querying – Rich, document-based queries
- Fast In-Place Updates – Atomic modifiers for contention-free performance
- Map/Reduce – Flexible aggregation and data processing
- GridFS – Store files of any size without complicating your stack
- Commercial Support – Enterprise class support, training, and consulting available " [20]

Figure 2.19: MongoDB Logo, Copyright : 10gen, Inc.

[19] LS4 Home Page – http://ls4.sourceforge.net/
[20] MongoDB Home Page – http://www.mongodb.org/

2.7.7 Decision Matrix

This is a table comparing main features of the set of Open-Source storage or database technologies I introduced earlier.

As you can see, MongoDB is actually the only database technology I selected. This is mainly due to the fact that MongoDB handles json natively and it was also easy for me to integrate it to my Python application with pyMongo !

Using GlusterFS was another great choice, thanks to the excellent Gluster File System 3.3.0 Administration Guide.

Note:

- DB = Database, FS = Filesystem, BS = Block storage, OS = Object storage
- Compl(exity) = Is the design complex ?
- Scal(ability) = Is it scalable ?
- Manag(ement) = Ease of management
- Sec(urity) = Secured access ?

Database technologies

Name	Description	DB Access	Complexity	Stability	Management	Scalability	Security	Charm
MongoDB	NoSQL database	JSON documents	Low	Yes	Easy	High	Yes	Yes

Storage technologies

Name	Description	FS Access	BS Access	OS Access	Compl.	Regions	Stability	Manag.	Scal.	Sec.	Miscellaneous	Charm
Swift	S3 storage	–	–	S3 API	High	(?)	(?)	(?)	High	(?)	Object < 5GB	~Yes
Ceph	All in one	Linux mount	Linux mount	S3 API	Medium	(?)	(?)	(?)	High	(?)	–	~Yes
LS4	Media storage	–	–	REST API	High	Yes	Unit test	(?)	High	(?)	No updates	No
NFS	Filesystem	Linux mount	–	–	Low	No	Stable	Simple	Low	Yes	–	~Yes
GlusterFS	Two in one	Linux mount	–	Yes	Low	Yes	Stable	Simple	High	Yes	–	~Yes

2.8 Conclusion

Choosing tools and technologies in order to build an Open-Source application is one of the most interesting and important preliminary step. In fact, most of the required functionnalities of OSCIED can be implemented by developing code using or gluing together Open-Source softwares or libraries. The design of OSCIED implies to select the technologies that provides required features (e.g. scalable storage, RESTful API) and at the same time fit the technical criterias (e.g. strong community, scalability, KISS, ...).

This preliminary step was guided by the *Project Specifications*, the chapter that follows.

CHAPTER

THREE

PROJECT SPECIFICATIONS

3.1 Contents

This chapter details the specifications of the project. This chapter begins by exposing the goal of the project and the work to do during the thesis. It also describes the organization of the project in high-level, long-term developments cycles. Then more details are given about the first cycle, this thesis, in the form of development phases describing the high-level tasks to complete during the cycle. Finally, this chapter ends with details about the planning and the management tools used during this thesis.

EBU Technology & Innovation

Project EBU OSCIED part 1: The basic set up

Title Open-Source Cloud Infrastructure for Encoding and Distribution

Main Developer *David Fischer*

Project Leader Bram Tullemans

3.2 Motivation

This project is aimed by the goal of providing a scalable media Open-Source platform to members of the European Broadcasting Union (EBU/UER).

This platform, based on cloud-era Open-Source technologies, would be dedicated and designed based on broadcasters specific needs such as transcoding of a wide collection of medias or online publication of popular medias, two basic use cases of this preliminary demonstrator.

This Open-Source platform would be freely available for broadcasters to promote interchange of knowledge and drive the project's developments by a wider community of experts.

3.3 Work to be done

The focus will be to develop an Open-Source scalable cloud infrastructure for encoding and distribution of on demand video using H.264 video codec.

A professional management layer for the system as a whole will be at the core of this system.

Nice to haves are in order of importance the support of MPEG-DASH, both for Live and VoD and play out to different devices (laptop, HbbTV, Tablet and Smartphone both for Android and iOS).

3.4 Development cycles

The EBU OSCIED project consist out of different development cycles. Within the *first cycle* the basic setup of the project will be realized.

All the core functionalities will be ready and can be upgraded either by adding them in the available code and / or by adding or exchanging modules.

Furthermore OSCIED will consist out of three separate scalable environments for development, test and production allowing different developers to work on new cycles simultaneously.

3.5 Time span first cycle

The project is the Master Thesis of *David Fischer* who will spend 0.4 fte for 6 months on the project as a minimum. The first development cycle starts on the 25 September 2012 and is ends on 25 March 2013. The Master Thesis itself will starts on the 20 September 2012 and is ends on 8 February 2013 (cf. *Planning*).

3.6 Development phases

1. **investigation** will consist of defining which Open-Source tools should be used in order to make this project successful. The main decision criteria are the size of the community behind the tool, the release of fixes & features, the quality of the documentation, the licensing, ...
2. **architecture** will consist of dividing the platform's architecture in specialized components (orchestrator, encoding and publishing units, ...) able to work together and to scale-up/down as easier as possible. During this key process, it is necessary to think about the media asset management and the future extensions
3. **servers** will consist of choosing appropriate server's configuration based on constraints such a price tag for a setup of 4 servers
4. **private cloud** will consist of installing Open-Source private cloud environment using OpenStack. This long-run task should be as automated as possible and the output will consist of setup scripts in parallel to the setup itself
5. **environment** will consist of defining what kind of professional management is needed (performance servers or virtual machines, reboot, input selector, feed check, encoding settings, triggering of processes, meta-data, xml-in feed from master control room, xml-output for MaM systems, scheduler, expand encoding or distribution parts to cloud) and develop basic interface
6. **application** will consist of implementing application-level components of the platform [1] :
   - **storage** : To store medias in a scalable way
   - **transform** : To transcode medias in a scalable way
   - **publisher** : To made medias available to end-users
   - **webui** : To provide a user interface for broadcasters
   - **orchestra** : To provide a RESTful API [2] and to orchestrate the other components

3.7 Set up

The development and distribution environment runs on the Open-Source cloud software OpenStack. The project involves 4 servers consisting of a Controller and Nodes (Computer/Storage) machines. On this private cloud run virtual machines that enable separate controllable environments for development cycle :

1. **Development** : Programmers access their own playground
2. **Test** : New code is tested in relation to the rest of the software
3. **Production** : Approved code is implemented in the live services

[1] This listing wasn't complete at the beginning of the project as it was part of *phase 2*.
[2] This API will be used by MAM to integrate the platform in their (automated) workflow !

These different environments can be accessed from the outside allowing external approved developers, for example from EBU members, to work on new code, to test the code so it can be added to the production environment.

Up scaling of capacity can be done by adding servers in the private (local) cloud or by using the public cloud. For example within a separate DTP-environment one can define virtual machines allowing for example for adding encoding or distribution entities. Within the management interface one should be able to identify if hardware needs to be added in the private cloud or if one needs to define the extra virtual machines in a public cloud.

The total environment is flexible and redundant [3]:

- Easily expandable by adding servers. Via the management layer one can add the server to the cluster ideally with a self install procedure
- Expandable via external Public Cloud Solutions (Parts can be up scaled via the public cloud, for example the encoding is scaled in the private cloud and the distribution is up scaled in the public cloud)
- Redundancy is accomplished by the fact that if one server is taken out, for example when it is broke, the rest will take over

3.8 Documentation

During the development process the Open-Source integrated ticket and project management tool TRAC (OSCIED Project's TRAC Environment) will be used to gather descriptions of the code and functionalities. The goal is to gather all necessary descriptions organically. This should reduce the effort to generate documentation after the project.

3.9 Organizational consequences

First of all we will have to decide which Open-Source legal infrastructure is going to be adopted (MIT, Creative Commons, GNU GPLv3, GNU LGPLv3 or LGPL Europe). Together with the legal department we should also investigate responsibilities for the EBU when this software is distributed. Furthermore a organizational / legal order like a Swiss association would perhaps be appropriate to attract external developers and funding.

3.10 Future cycles

Nice to haves are in order of importance the support of MPEG-DASH, both for Live and VoD and play out to different devices (laptop, HbbTV, Tablet and Smartphone both for Android and iOS). If this is not realized in *cycle 1* it will be the core of cycle 2.

3.11 Future development

Here are some of the preliminary ideas [4] :

- Open Broadcast Encoder for delivering MPEG2TS feed
- Adding profiles for public clouds allowing to use different clouds at the same time and automatic scalability functions
- Adding native imports of distribution files to reduce latency
- ... A lot of features we will think of during the process (cf. *Future Extensions*)

[3] Some of the features will be implemented after the end of my Master Thesis (cf. *Planning*)

3.12 Planning

To be honest with you, I don't planned the whole project on the pit-start (as soon as the project started) !

In fact, I spend the first days to install my work place and to setup the OSCIED Project's TRAC Environment.

We meet at least once a month with Mr. Bram Tullemans and Prof. Andrés Revuelta to manage this project. We discussed about the project to keep in sync what I have done and what I should do in order to be successful.

During my investigations I filled the OSCIED Project's TRAC Environment with links to the most interesting resources I founded on Internet. I also created tickets related to the development tasks not only to see what should be implemented but also to backup some of the future features I thought of. Of course, there where bugs and I noticed them as *defect* tickets and then I fixed them (as soon as possible).

All of these tickets are explicitly linked to a *component* of the application (e.g. Orchestra, Master Thesis Report, ...) and most of the time I set the *priority* field.

In order to reduce the entropy of this growing set of tickets I created *milestones* and grouped tickets.

This project started with my Master Thesis and hopefully it will not end with it ...

I planned my project as such :

- **25 September 2012 - 25 March 2012 :**
  - Investigations helped with necessary resources such as Internet, books, ...
  - Developments of automation scripts for easiness of tests, setup, ...
  - Large amount of bugs fixed mainly of my own creation but not only ...
  - Project's management with team members and with dedicated tools ...
- **25 September 2012 - 10 November 2012 :**
  - Setup of development and management tools at (EBU, hepia, home)
  - Project's specifications refinement with Mr. B. Tullemans and Prof. A. Revuelta
  - Preliminary design decisions based on early investigations
  - Bill of material of the servers (based on project's CAPEX)
- **25 September 2012 - 31 November 2012 :**
  - Intensive scripting and readings of OpenStack documentation
  - Initial OpenStack setup of the server based on automation scripts
- **1 December - 20 January 2012 :**
  - Intensive development of the application, release of the first demonstrator
  - Various deployments scenarios tested, application successfully deploy on Amazon AWS
- **21 January 2012 - 8 February 2012 :**
  - Selection of best tools to increase speed and easiness of writing
  - Cleanup and reordering of the project's tickets under TRAC
  - Writing of the following report with required content [5]

[5] This report is actually not the only source of documentation for this project, see OSCIED Project's TRAC Environment.

3.13 Management

Here will be introduced the tools used to manage code and tasks such as Subversion and TRAC.

3.13.1 Source Code Versioning : SVN (and GIT)

What is SVN & GIT ?

Subversion is a versioning and revision control software based on a client/server architecture. Developers use this kind of tool to maintain current and historical versions (called revisions) of their work such as source code and documentation of their project(s).

Subversion is one of the most widely used versioning system on the Open-Source community.

The SVN server hosts the projects repositories and the developers will use a SVN client to get a local copy of the project. The developer will work locally on his copy (add, remove, rename, modify files and directories) and then propagate (commit) these modifications to the repository.

GIT is a versioning and revision control software based on a distributed architecture. GIT was initially designed and developed by Linus Torvalds for the development of the Linux kernel, the biggest Open-Source project ever made. Unlike Subversion, every GIT local copy is a full-fledged repository with complete history. This design permit to maintain large distributed projects in a efficient manner.

Why using SVN & GIT ?

MPEG-DASH is a cutting-edge standard and the Open-Source community is actively implementing MPEG-DASH in their favorite Open-Source piece of software. This is why it is necessary to access to the latest revision of the source code of these softwares to understand what is implemented and what is not (profiles, ...) !

It is also necessary to manage backups of this project and Subversion is the best tool for that. Every backup (*commit*) is done manually and every single revision has a purpose, such a new functionality (code), a bug fix (code) or some files to backup (documents). The revision history permit revert changes if necessary and it also permit to get a statistical overview of the work (with StatSVN for example).

This tool is also a must have to synchronize the contributions (work units) of the team members !

We can also uses any Version Control System branching capabilities to manage release policy of the project. For example, we can create branches like :

- **trunk** related to latest features, here we can found the cutting-edge / development version of the software
- **testing** related to latest version to test before releasing it in production
- **production <version>** related to stable releases [6]

And then *checkout* any of the following branches anywhere you want, the upgrade of running *local copy* will be simply done by calling the *update* method of the Version Control System !

Note: Most of the Open-Source repositories are hosted on SourceForge (SVN) or GitHub (GIT).

[6] With a version number, I actually like the version numbering of Ubuntu's releases.

3.13.2 Project Management : TRAC

What is TRAC ?

TRAC is a simple tickets tracking system aimed to provide a management and documentation layer to any project based on software development. This tool is accessed by users through a web user interface.

TRAC can be interfaced with some of the most used Version Control System and for this project, we thanks the Open-Source community to provide an interface for Subversion !

Here are the main features :

- **Wiki** to add collaborative documentation to the project
- **Timeline** to see what happens to the source-code or to TRAC itself
- **Roadmap** to manage Milestones (e.g. grouping of tickets into higher-level features with a delivery deadline)
- **Source Browser** to browse the source code of the project hosted by the VCS
- **Tickets** to filter tickets by clicking on specific Reports, e.g. *Active Tickets by Milestone* ...
- **Search** to search something into the Wiki, the Tickets, the Milestones, ...
- **Admin** to configure the tool, add/edit Users, Components, Milestones, ...

... And a lot of plugins you can add to TRAC !

Tickets in a nutshell

They are fields (in **bold** the fields I really take care about) [7] :

- **Summary**
- Reporter
- **Description**
- Keywords
- Owner
- Cc
- **Type** (defect, enhancement, reference, task)
- **Priority** (none, trivial, minor, major, critical, blocker)
- **Component** (..., Cloud, Development, FIMS, MPEG-DASH, Master Thesis report, Orchestra, ...)
- **Milestone** (Demonstrator ready, Master Thesis Report, ...)
- **Status** (new, assigned, accepted, duplicate, fixed, invalid, wontfix, worksome, reopened)
- Version (1.0, 2.0)

They are three built-in type of tickets :

- **defect** typically used to report a bug or a missing feature
- **enhancement** typically used to describe interesting new features
- **task** typically used to specify what should be implemented and when

Why using TRAC ?

Mostly TRAC was used as a smart notepad (tasks, bookmarks, bugs) for my ideas and I actually have a lot of ideas for this project ;-)

For example, I created the ticket type called *reference* with related *None* priority to save (grouped by topic) bookmarks of the most interesting resources I founded on the Internet.

This tool is also useful to create *task* tickets reflecting any non-trivial unit of work. For example, add a feature (code), add a chapter to documentation (report), ...

[7] *Reference Type*, *Component*, *Milestone* and *Owner* fields values are not built-in but created in the *Admin* tab

This ticket will be accepted by *someone* and when the work is done, the person who has done the work will update the ticket's *status* (e.g. close the ticket with status *fixed*).

3.14 Conclusion

This project started with a motivating, well defined set of uses-cases based on realistic challenges the broadcasters face to provide new *connected* services to their audience. The detailed specification of the project OSCIED is then defined to fit the uses-cases that motivated the development of OSCIED.

The set of cycles, phases and tasks permit to create a roadmap for the project, at a macro- (cycles) and micro-level (phases). First cycle specified the work to be done during this thesis.

Another key element is the decision to use specialized tools to manage the project (TRAC) and to use a Version Control System like Subversion.

Chapter that follows will give more details about the builded demonstrator – a preliminary version of OSCIED.

CHAPTER

FOUR

OSCIED DEMONSTRATOR

4.1 Contents

This chapter introduce you with the demonstrator in the form of logical layers, from physical servers to developed application. Eeach of those layers is the described in dedicated sections of the chapter. The section about the application shows and details the main components of developed application. A FAQ about the demonstrator is also available in this chapter. Finally, the chapter ends with few example deployment scenarios (including example configuration files) and the tests and results section.

4.2 Introduction to Layers

The following chapter will introduce you with the demonstrator in the form of logical layers.

Conceptually, you can picture the demonstrator as such :

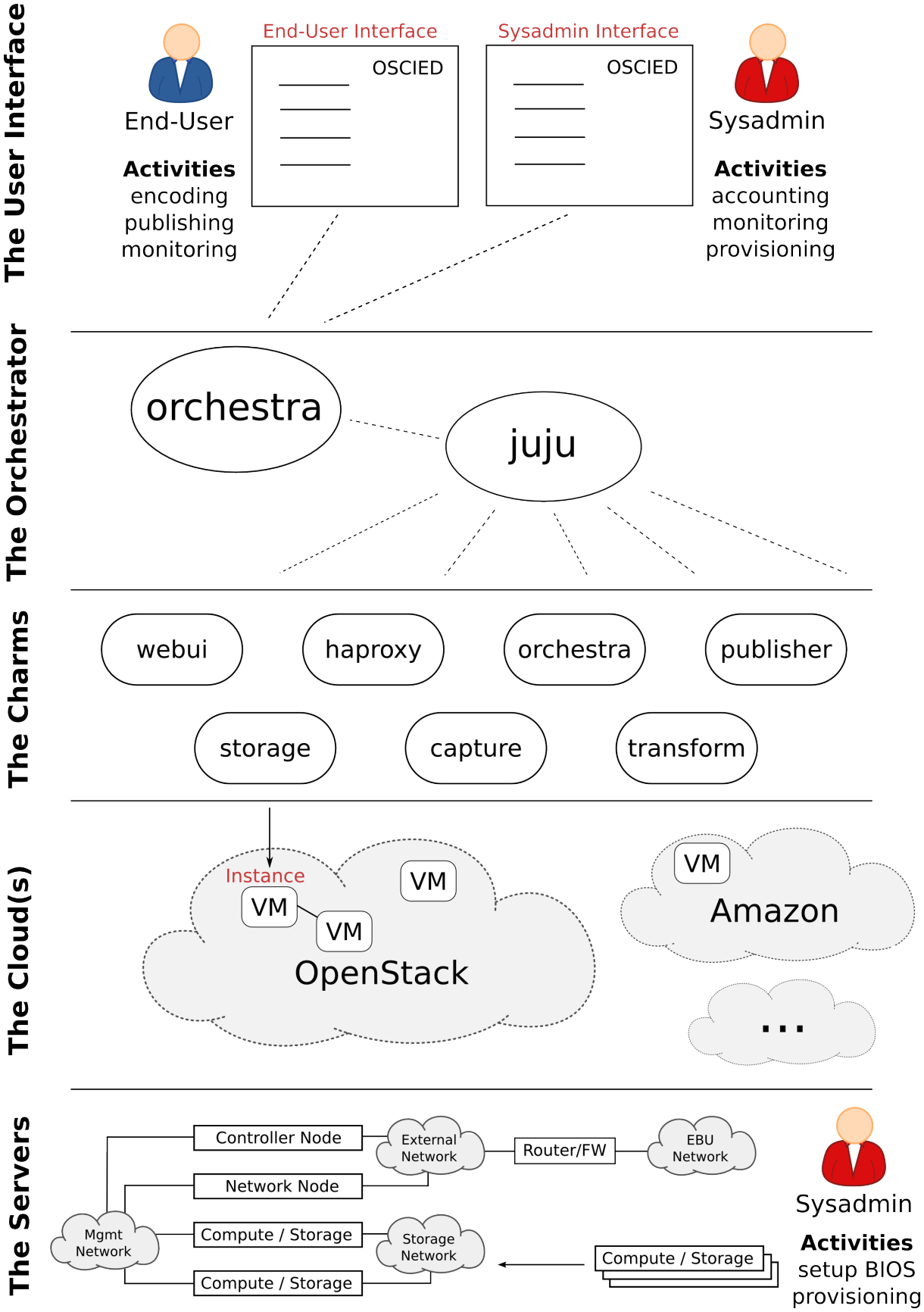


Figure 4.1: The project separated in logical layers, from the servers to the end-user

4.3 Server Layer

4.3.1 Specifications

In order to run the demonstrator's private cloud based on OpenStack we need servers.

A setup of four 1U servers where chosen and configured based on the following :

- The price-tag specified in the project's CAPEX
- The optimization of the configuration based on project's requirements
- The chosen OS Ubuntu Quantal Server -> Ubuntu Server certified hardware (Dell)

Here is the technical specifications of the servers.

> " Get an energy-efficient, dense 1U server for your applications with the PowerEdge™ R420, featuring next-generation processing and flexible I/O options. " source Dell

Characteristic	Description
Model	PowerEdge R420
Processors	2x Intel® Xeon® E5-2430 2.20GHz, 15M Cache, 7.2GT/s QPI, Turbo, 6C, 95W, Max Mem 1333MHz
Memory Configuration Type	Performance Optimized
Memory DIMM Type and Speed	1333 MHz RDIMMs
Memory Capacity	4GB RDIMM, 1333 MT/s, Low Volt, Dual Rank, x8 Data Width
Operating System	No Operating System
OS Media kits	No Operating System Media Kit
Chassis Configuration	3.5 Chassis with up to 4 Cabled Hard Drives and Embedded SATA"
RAID Configuration	HW RAID 0
RAID Controller	Dell PERC H310 Mini
Hard Drives	2x 1TB 7.2K RPM SATA 3.5in Cabled Hard Drive
Power Supply	Single Cabled Power Supply 550W
Power Cords	NEMA 5-15P to C13 Wall Plug, 125 Volt, 15 AMP, 10 Feet (3m), Power Cord
Power Management BIOS Settings	Performance BIOS Setting
Embedded Systems Management	Basic Management
Add-in Network Adapter	On-Board Dual Gigabit Network Adapter
Rack Rails	No Rack Rails or Cable Management Arm
Bezel	No Bezel
Internal Optical Drive	No Internal Optical Drive for 4HD Chassis
System Documentation	Electronic System Documentation and OpenManage DVD Kit for R420
PCIe Riser	PCIE Riser for Chassis with 2 Proc
Shipping	Shipping Material,PowerEdge R420
Hardware Support Services	3Yr Basic Hardware Warranty Repair: 5x10 (HW-Only, 5x10 NBD Onsite)
Installation Services	No Installation
Proactive Maintenance	Maintenance Declined

4.4 Cloud Layer

4.4.1 OpenStack in a Nutshell

OpenStack is an Open-Source Infrastructure as a Service project initially launched by NASA ( ) and Rackspace ( ) in July 2010 and released under the Apache 2 license and now managed by the non-profit OpenStack Foundation. OpenStack is also the name of the initiative related to this project, aimed by the goal of providing to any organization a way to deploy a private cloud (IaaS) on top of their (commodity) IT infrastructure.

This cloud technology consist of a bunch of inter-related Open-Source components (each of them running a specific service) working together to abstract and control hardware computing resources such as processing, storage and networking.

Th cloud infrastructure's users will consume computing resources in their abstracted form such as virtual machines, virtual networks, ...

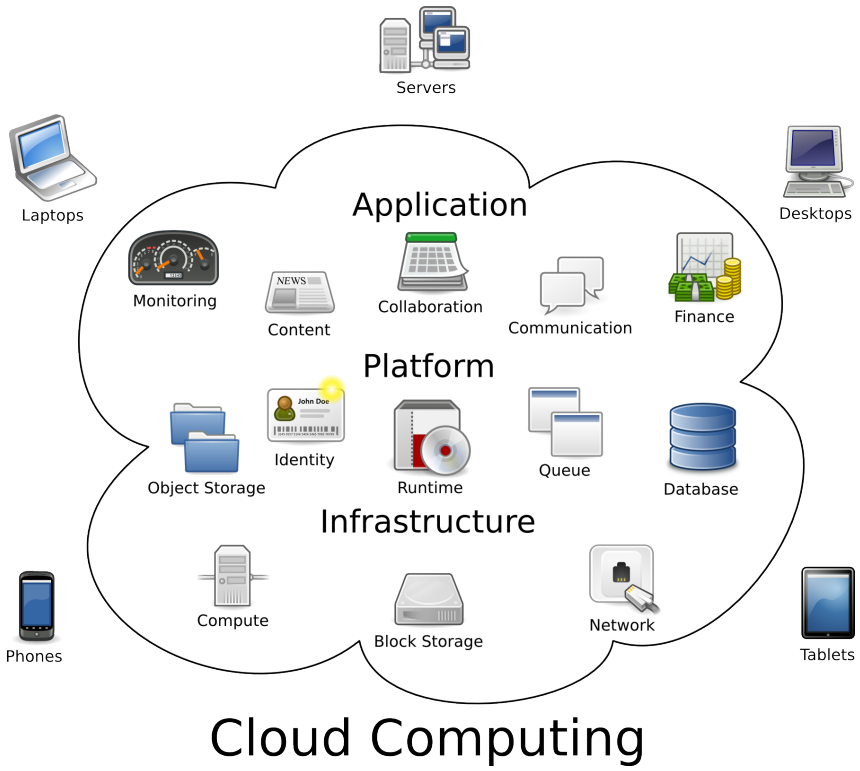


Figure 4.2: Source : Wikipedia - Cloud computing (Created by Sam Johnston)

Cloud Advantages

The main advantage of such approach is that the enterprise services are uncoupled from underlying hardware. The cloud infrastructure can improve the availability of services by enabling new way of managing IT resources.

One of them is called the **live-migration**, with it you can :

- Ensure the failover of services in case of hardware failure
- Replace computing resources without any service interruption
- Schedule the load of servers by specifying operational rules

Another advantage is that is easier to manage the usage of your IT infrastructure, here are some examples of what you can do :

- Easier scale up/down by adding/removing computing resources to the platform
- Specify platform's resources usage quotas, this is related to users
- Optimize hardware usage thus decrease the OPEX of your IT infrastructure

Nowadays, cloud-era tools (eg. JuJu) let you deploying and managing your services in any compatible cloud as easy as `juju deploy mysql` ! So why not the same for your own application on your own infrastructure ?

Then if you need more computing resources, you may decide to *seamlessly* use public cloud provider's resources complementary to your own cloud infrastructure.

And OpenStack

The strengths and weaknesses of the project called OpenStack are :

- **The modular architecture add flexibility to any deployment :**
  - (+) The components can be deployed in standalone (testing purposes) or in any form you want
  - (+) The underlying technologies are interchangeable, you may decide to use LXC instead of KVM
  - (+) Not all components are necessary (e.g. Swift), you may decide to do not use it or replace it by GlusterFS
- (+) The services collaborate by exchanging asynchronous messages, there is no locking synchronous call
- **This IaaS is designed to being deployed on commodity hardware :**
  - (+) You don't need to buy costly highly-available hardware such as RAID-based SAN or high performance fiber-channel
  - (+) Any node of any services can fail, the others will continue handling requests, so no single point of failure here
- (+) The Open-Source licensing means this project is community driven, no more vendor lock-in
- (+) The community of 150+ companies that have joined the project (Intel, Canonical, Cisco, Red Hat, ...)
- **The high rate of releases means :**
  - (+) The project gain rapidly in maturity and features
  - (-) You may need to upgrade you infrastructure as fast as the releases are
  - (-) The documentation needs to be updated according to release (seriously !)
  - **(-) Configuration files are some kind of chaotic :** Not based on the same template, please why *–thing* called flags & why files like api-patch.ini ? This fact will not remains in future releases as there is some encouraging re-ordering
- **OpenStack is not a single apt-get'able package but a complex system based on services working together :**
  - (-) The learning curve is high, especially if you are not an expert on this domain
  - (-) You need to understand and configure technologies involved in this IaaS
- (+) The services are configured via the config files and/or the API's calls, the setup can potentially be automated

For all that reasons, I chose OpenStack !

4.4.2 The Main Components

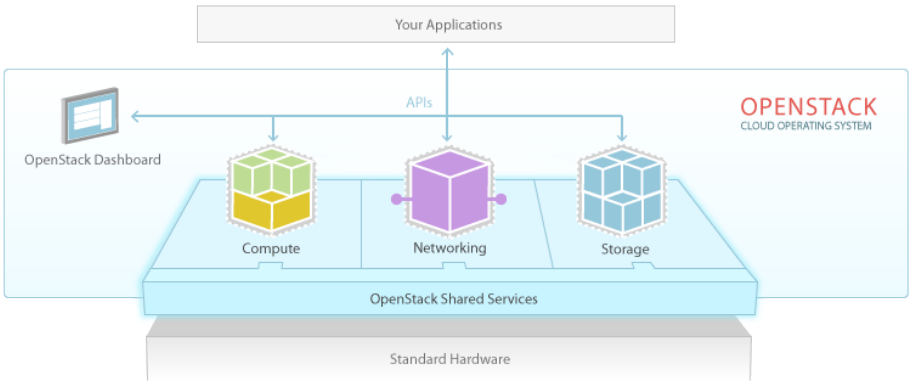


Figure 4.3: Original OpenStack's services-level diagram

- Identity Service (codename Keystone). This components provides a *centralized* Identity, Token, Catalog and Policy services intensively used by other components of OpenStack. One can access to Keystone functionalities through the keystone command-line client or by using Keystone RESTful API directly. This component provides a way for users and services to authenticate using token based OAuth mechanisms. This feature is especially useful to add a security layer to any RESTful API.

- Images Service (codename Glance). This component provides a way to register, discover and retrieve virtual machine images and images metadata. One can access to Glance functionalities through the glance command-line client or by using the RESTful API directly. Glance can use various storage technologies to store the images, from simple filesystem to object-based storages like Swift or AmazonS3.
- Block Storage (codename Cinder) originally developed by NASA and known as nova-volume in previous OpenStack releases. This component provides persistent, reusable storage volumes available for virtual machine instances. The volumes can be attached and detached to instances in the form of iSCSI mounted block storages. One can access to Cinder functionalities through the cinder command-line client or by using the RESTful API directly. This component is ideal for performance-critical storage mounted by virtual machines running low latency I/O applications such as a databases. Cinder has a snapshot management capabilities to volumes, this powerful functionality is available thanks to the underlying technology used by Cinder, called LVM.
- Object Storage (codename Swift) originally developed by Rackspace. This component allow users to store, update and retrieve files in the form of objects (e.g. hash table) stored in a highly available, distributed, eventually consistent object/blob store. One can access to Swift functionalities through a simple RESTful API or a AmazonS3 compatibility layer. This not a traditional filesystem with folders and files but rather a sort of key-value store ideal for static data such as pictures, audio files, ... This store can be installed on commodity hardware in order to create a storage cluster based on Swift.

- Network Service (codename Quantum) originally developed by NASA and known as nova-network in previous OpenStack releases. This component provides flexible, virtual/physical networks connectivity to virtual machines instances. Quantum is a tool aimed by the goal of providing network administrators with a simple but yet powerful approach to manage next-generation networks. One can access to Quantum functionalities through the quantum command-line client or by using the RESTful API directly. This component is ideal for managing highly-complex networking models mixing physical and virtual network and equipments. The pluggable design of Quantum (and of OpenStack in general) allow administrators to choose tools around Quantum such as the underlying network virtualization technology like Open-vSwitch or Bridge-Utils.

- Compute Service (codename Nova) originally developed by NASA. This component provides on-demand computing resources in the form of virtual machines instances managed by this cloud computing fabric controller (the main part an IaaS). One can access to Nova functionalities through the nova/nova-manage command-line clients or by using the RESTful API directly. In previous release, this component was also responsible of the network and volume services, each of these two services are now the responsibility of Quantum and Cinder projects. The flexible design of Nova let you choose tools and hardware around Nova such as the underlying hypervisor and the kind of computer's configuration (e.g. bare metal / HPC ...). The hypervisor choice is really a good thing, you may choose to use the widely used full-(para)virtualization hypervisor called KVM or to switch to a low overhead, high-density container-based isolation called LXC.

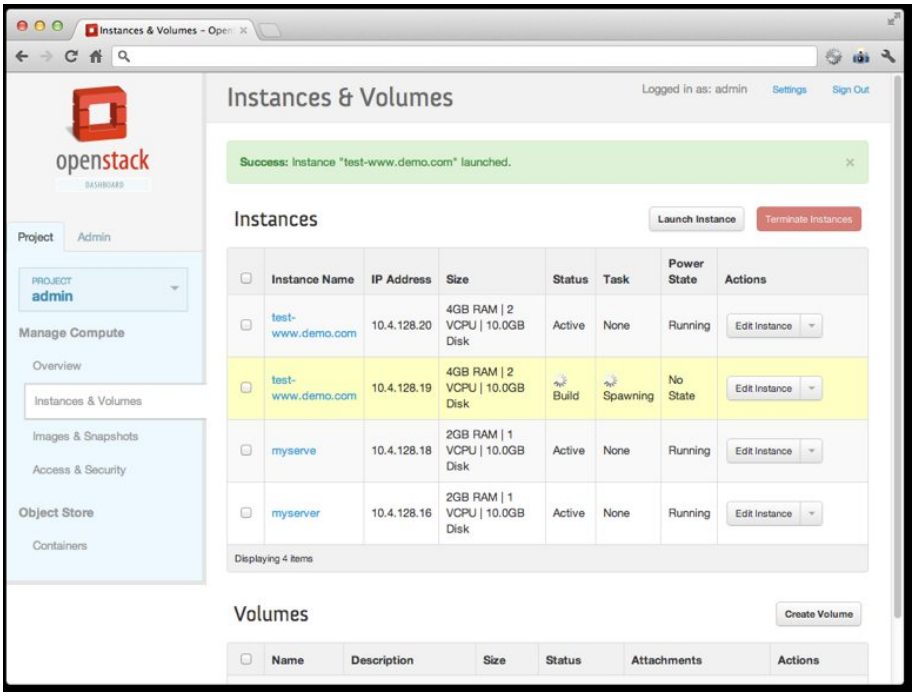


Figure 4.4: Launching an instance with OpenStack Horizon

- Web User Interface (codename Horizon). This component provides a web user interface to manage and access to other OpenStack components functionalities by mapping interface interaction to OpenStack component's APIs calls. Third-party extensions can be plugged-in to this interface to extend and add features and services such as real-time monitoring, resources consumption billing ....

4.4.3 The Conceptual Architecture

The services briefly introduced in previous chapter needs to collaborate in order to build a complete cloud infrastructure, an IaaS. For this integration to be successful, OpenStack is designed as such :

- **The services works together by calling other services :**
  - Each service functionalities are callable through corresponding REStful API
  - Each service act as an user, it means that Keystone authentication mechanisms applies for users & services
- **The API's calls are handled in the form of messages handled by a message broker such as RabbitMQ, ... :**
  - Each request is queued into the message's brokers queues and asynchronously consumed by OpenStack service's nodes

– Any available node of any service is able to handle service's requests, there is no single point of failure here

So, conceptually, you can picture the relationships between the services as such :

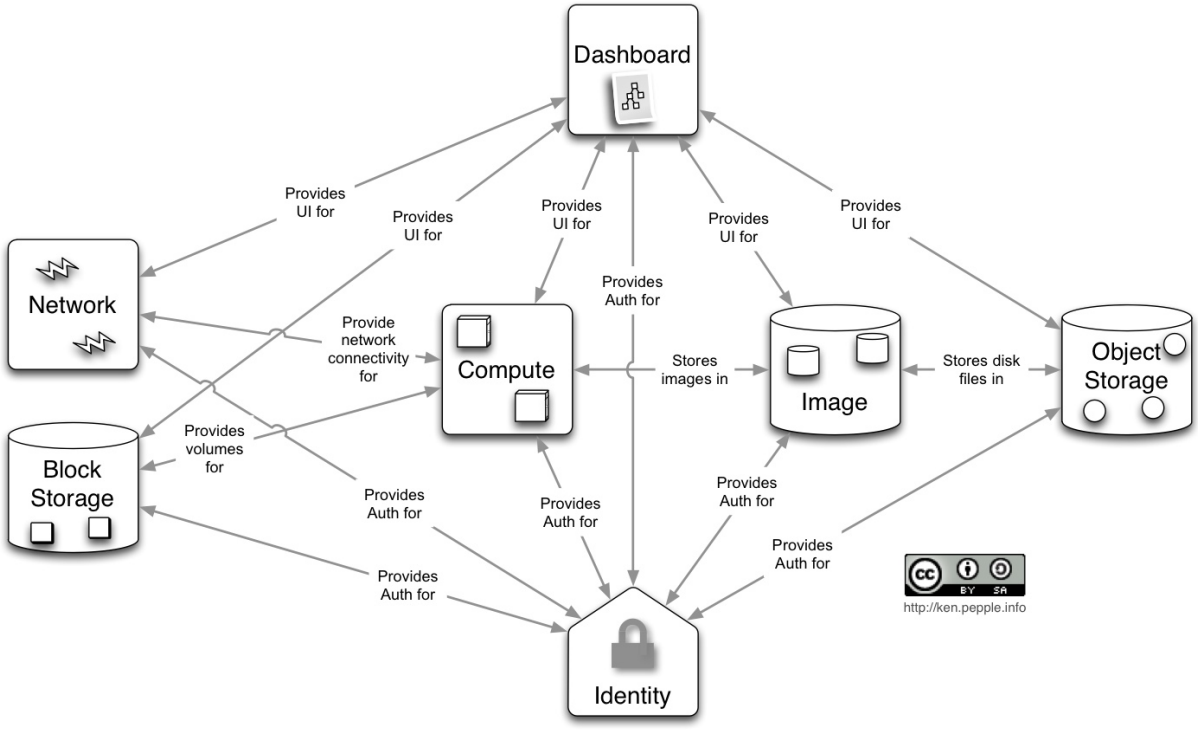


Figure 4.5: Nice diagram showing the main components of OpenStack

" This is a stylized and simplified view of the architecture, assuming that the implementer is using all of the services together in the most common configuration. It also only shows the *operator* side of the cloud – it does not picture how consumers of the cloud may actually use it. For example, many users will access object storage heavily (and directly). "

4.4.4 Logical Architecture

" As you can imagine, the logical architecture is far more complicated than the conceptual architecture shown above. As with any service-oriented architecture, diagrams quickly become "messy" trying to illustrate all the possible combinations of service communications. The diagram below, illustrates the most common architecture of an OpenStack-based cloud. However, as OpenStack supports a wide variety of technologies, it does not represent the only architecture possible. "

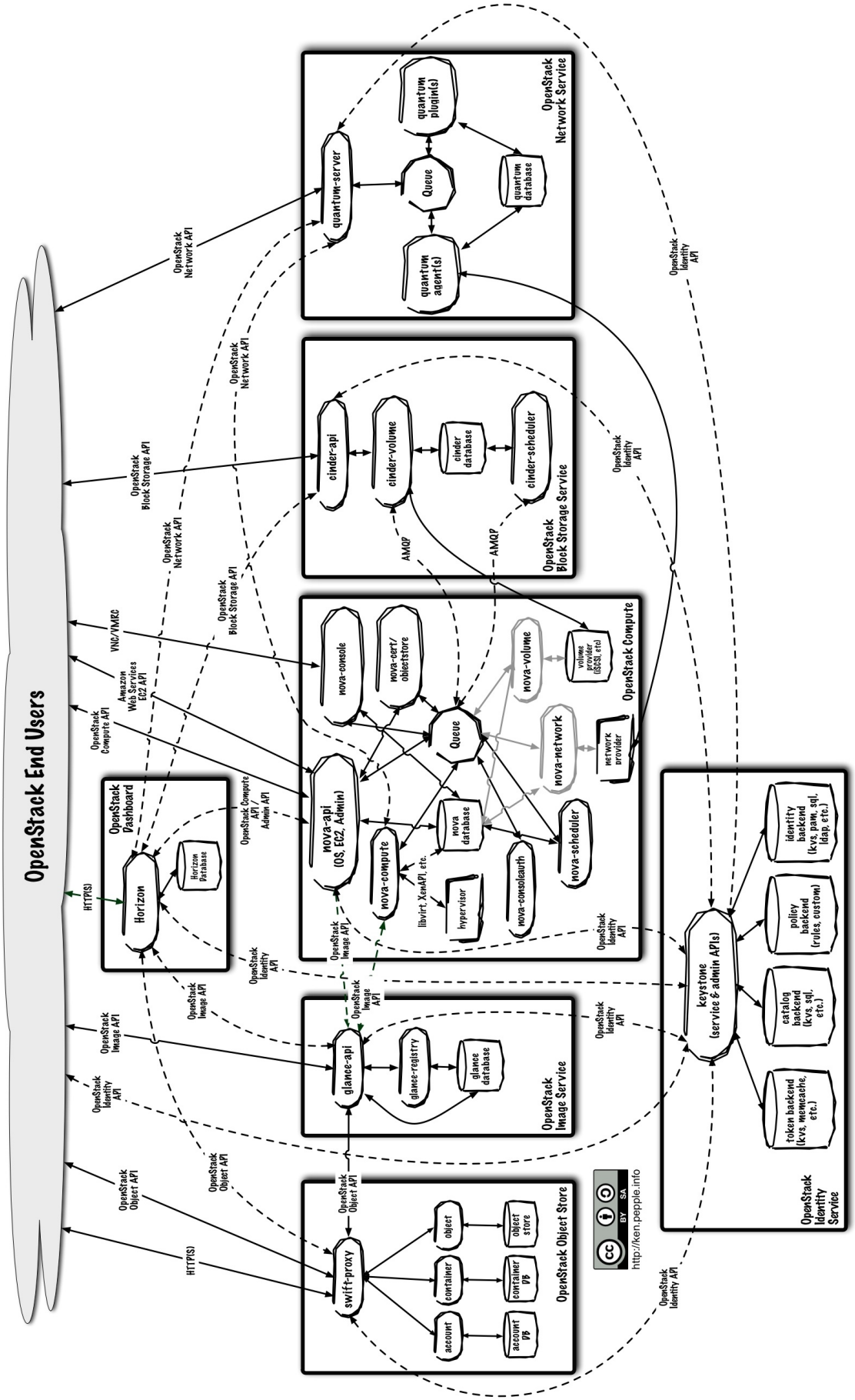


Figure 4.6: Nice diagram showing softwares involved in an OpenStack setup

4.4.5 A Standalone Setup

Difficulty o . . .

Use case Development

Hardware 1 Desktop Computer

Software Ubuntu

The easiest way to test OpenStack is to install all components in one desktop computer running Ubuntu !

The fastest way to deploy this setup is to use an already available installation script e.g. DevStack.

Remark: I really like the idea behind this script, as installing OpenStack is a really complex task requiring a lot of trials and debugs (thanks to documentation ...). You actually can explain how to setup something or you can write scripts and add some documentation around it : Documentations become tools !

4.4.6 A Typical Setup

Difficulty o o o .

Use case Development - Production

Hardware 2+ Desktop/Server Computers

Software Ubuntu

The following multi-node deployment is designed to separate the critical services (called controller) of the computing services (called compute nodes). This setup was pretty well documented and it seem that the more scalable setup is getting more popular now.

4.4.7 A More Scalable Setup

Difficulty o o o o

Use case Production

Hardware 3+ Desktop/Server Computers

Software Ubuntu

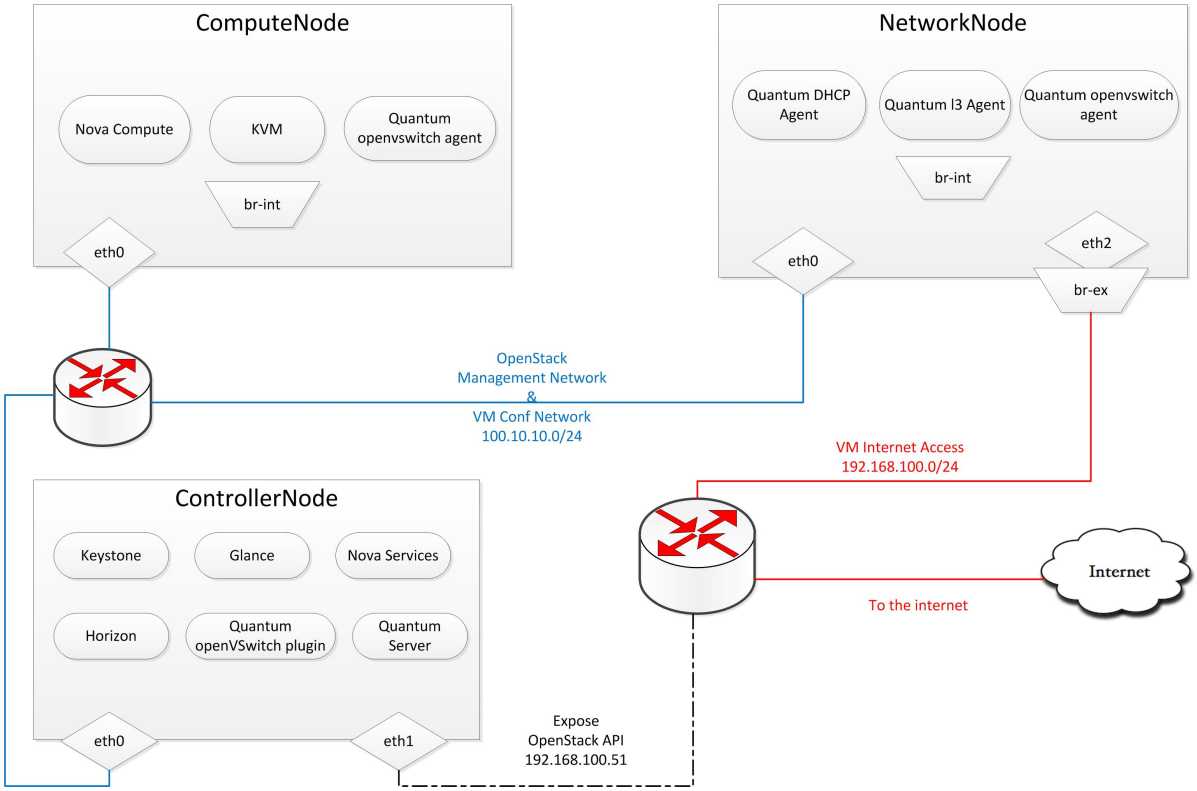


Figure 4.7: This multi-node setup separate the critical services, the compute service and the network GW

The setup of the guide from where this diagram comes from, specify that the controller node will host the following services :

- The shared database MySQL
- The message broker RabbitMQ
- The identity service Keystone
- The images service Glance
- The volumes service Cinder
- The compute service Nova (api, scheduler, ...)
- The network service Quantum (server, ...)
- The web user interface Horizon

The network node will host the following services :

- The network service Quantum (dhcp & layer3 agents, ...)
- The virtual switching technology Open-vSwitch

The compute nodes will host the following services :

- The network service Quantum (open-vswitch agent, ...)
- The virtual switching technology Open-vSwitch
- The compute service : Nova (compute-kvm, ...)
- The hypervisor technology KVM

4.4.8 A MAAS/JuJu Powered Setup

Difficulty o o . .

Use case Production

Hardware 6+ Desktop/Server Computers

Software Ubuntu

See also:

Please see *Clouds Orchestration Layer* for further details about JuJu.

If you plan to deploy OpenStack on a larger scale, you may be interested by using the metal-automation tool called MAAS from Canonical. When you install the first server with Ubuntu Quantal Server you specify to setup this servers as the MAAS master.

As an administrator, you will get access to a simple but rather efficient web user interface in order to *plug-in* new servers to the setup. Typically you only need to configure servers BIOS in order to enable Wake-on-LAN, enable remote-boot via PXE and take note of the network interfaces MAC addresses. Then, you will only specify servers MAC addresses to the MAAS master. The nodes will automatically being handled by the master and configured with Ubuntu.

Finally, you will uses JuJu cloud orchestrator in order to deploy OpenStack components on your setup !

At time of writing this report, JuJu cannot merge charms [1], it means that for any service (~charm) you want to deploy, a separated instance is required (e.g. a VM for each service). The nodes handled by the MAAS provider maps the deployment charms to the server itself, without encapsulating the instance into a virtual machine. This is why at least 6 of them are required for OpenStack to be installed !

4.4.9 The setup at EBU

The demonstrator will be deployed on a small setup of 4 servers. This is the main reason why MAAS + JuJu were not used in order to deploy OpenStack.

I started my work by reading (a lot) of documentation about the topic and followed some of the best tutorials I founded. The resulting scripts are available as appendices at the end of this report, the link: `appendices-stack`. I developed the scripts in order to make my work repeatable, as the setup will not work at the first try for sure !

During my Thesis, I faced a lot of problems during my trials with OpenStack, partially due to the quality of the documentation (not the quantity). Trying to install latest release of OpenStack, Folsom, is not as easy as expected. Unfortunately I stopped my work on the setup to concentrate on the application when it was scheduled to do so.

Here is the setup I suggest to deploy at European Broadcasting Union (EBU/UER), strongly inspired by A full guide for OpenStack Folsom with Quantum (GRE 2NICs).

[1] Charms are DevOps distilled. In brief, they are encapsulated services to be connected and scaled on demand.

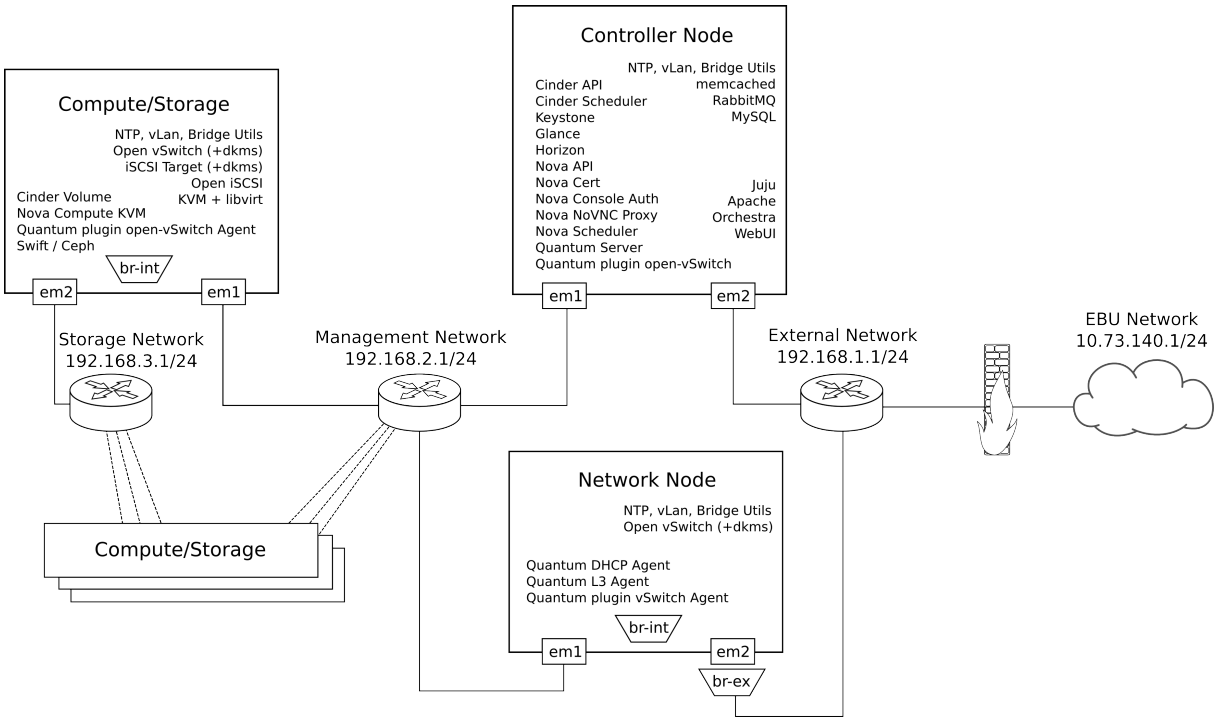


Figure 4.8: The proposed OpenStack setup

4.4.10 Documentation

Note: Please see Ticket 37 for further documentations (30+ links).

- Official Documentation : OpenStack Documentation
- A full installation guide for OpenStack Folsom with Quantum : This step-by-step guide is one of the most interesting guide mostly explained with **real** code snippets
- A full guide for OpenStack Folsom with Quantum (GRE 2NICs) : This is the guide that inspired my for the OpenStack production setup
- DevStack : This is a documented shell script to setup a development OpenStack environment
- TryStack : This is a cluster setup running OpenStack and available online for free to try

4.5 Clouds Orchestration Layer

4.5.1 Introduction

This project cannot being successful without automation !

We choose to use cloud technologies in order to automate the usage of computing resources. Next step is to automate usage of cloud(s) (IaaS) in order to deploy services of the demonstrator. Moreover it is not only necessary to use clouds in an automated manner, but also to automate OSCIED itself.

Remaining the fact that the project's application is split into components to being scalable ...

So, each of these components must be able to automatically :

- Install and configure any required service (e.g. FFmpeg)
- Manage the internal service's daemons (e.g. start, stop)
- Handle the relation with other components of the application

This is a rather complex task that requires a lot of work ... We need JuJu !

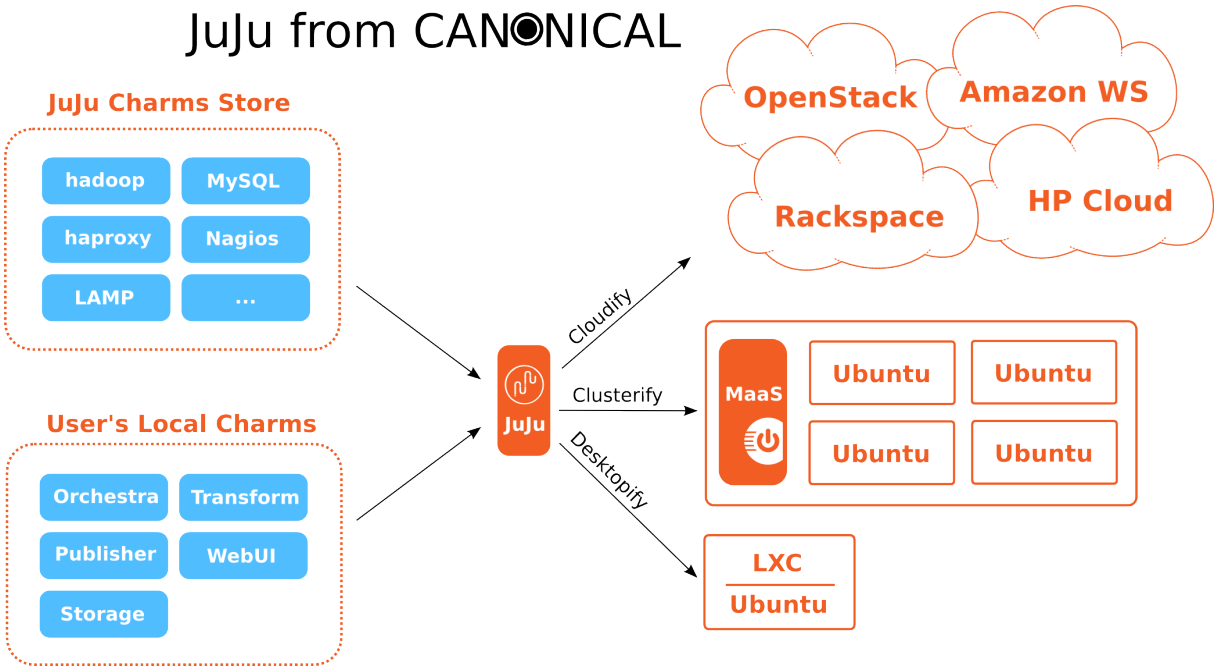


Figure 4.9: JuJu can deploy charms to a wide variety of environments, why not implementing your own provider ?

The goal of the following section is not to duplicate the official documentation from Canonical. I prefer to introduce you with the **how** and **why** OSCIED actually integrate JuJu.

See also:

Please see *JuJu* has some of the notions related to JuJu are necessary.

4.5.2 OSCIED + JuJu = ?

Integrating JuJu to OSCIED improved the automation of the project, by :

- **Packaging each of the application's component in a charm, requiring implementation of charm's :**
  - Automation scripts, called hooks (install, config-changed, start, stop, ...-relation-...)

    - Metadata (`metadata.yaml`, e.g. name, description, ...)
    - Configuration (`config.yaml`, e.g. mysql root password ...)
    - The service source code itself (e.g. `orchestra.py`)
- Creating scripts to improve easiness of use of the preliminary demonstrator
- Future : Integrating JuJu to the app. itself, thus permit features like auto-scale, ...

All that work to finally, helped with JuJu, make OSCIED able to be :

- **Deployed on a wide variety of environments, thanks to JuJu's providers :**
    - Clouds : OpenStack, Amazon AWS, *HP Cloud*, Rackspace
    - Clusters : MAAS
    - Computers : LXC
- **Linked to 100+ already available charms, thanks to JuJu Charms Store :**
    - Databases : CouchDB, MongoDB, MySQL
    - Message brokers : RabbitMQ, ...
    - Monitoring : Ganglia, Nagios, ...
    - Proxies : HAProxy, Nginx, ...
    - Storages : Cassandra, Ceph, GlusterFS, Hadoop, ...
    - Websites : Apache 2, LAMP, Django, Drupal6, ...
    - OpenStack (!) : Cinder, Glance, Keystone, Nova, ...
- **Managed easily (Of course this is JuJu that triggers the scripts) :**
    - Life-cycle of components are handled by automated charm's hooks
    - Relations between components are handled by automated charm's hooks
- **Future-proof, based on continuously improved tools with strong community support :** Ubuntu founder Mark Shuttleworth and former CEO of Canonical Ltd. [2] revealed that 2013 will be the year of the cloud in the sense that the enterprise will focus on making the cloud easier to use. Concretely, they will improve the already innovative projects called JuJu and MAAS. Canonical also contributes to OpenStack project by distributing and supporting OpenStack and are a Platinum Member of the OpenStack Foundation board. [3]

4.5.3 JuJu Tips & Facts

Here are some of the tricks & facts I collected by using JuJu, maybe helpful ...

- `$HOME/.juju/environments.yaml` : Here are configured the hosting environments for your services
- The local provider is quite useful to develop and test charms
- When you `juju bootstrap` an environment, a dedicated unit (juju) is deployed in order to manage the services
- When you use the `juju` command-line tool to manage an environment (status of running instances, ...), you interact with the orchestration unit running on this environment
- The communications (ssh) are secured with your private certificate `$HOME/.ssh/id_rsa`
- The services can be connected (e.g. *lamp -> mysql*) this is also handled by the hooks of the charms
- Each of the charms are sort of packaged services, the setup, start/stop of the service is handled by the hooks of the charm.

[2] In December 2009, he stepped down as the CEO of Canonical, Ltd. to focus energy on product design, partnership and customers.
[3] Ubuntu UDS R – Mark Shuttleworth Keynote [Ubuntu 13.04] - http://www.youtube.com/watch?NR=1&feature=endscreen&v=0voGsibCjHE

- You can choose the type of instance, e.g. `juju deploy --constraints instance-type=c1.medium mysql`
- You can use your own configuration, e.g. `juju deploy --config your_config.yaml mysql`
- You can get a graphical representation of an environment `juju status --format svg --output status.svg`
- When only one service is erroneous (e.g. *mysql* install failed) you do not need to destroy the whole environment `juju destroy-service mysql`
- Do not hesitate to open another terminal and `juju debug-log`, if something fail, you have the log !
- You can ssh any running unit with `juju ssh <unit_name>` or `ssh ubuntu@<unit_public_ip>`
- Do not forget to `juju expose` service you want to expose !
- Just try to ssh to any running unit and go to path `/var/lib/juju/units/<unit_name>/`, interesting right ?
- You can access to the environments you deployed from any computer, you only need your rsa key and your environments file

4.5.4 Charm's Life-cycle

Here is shown in a state machine the life-cycle of a charm (relation's triggers are not represented) :

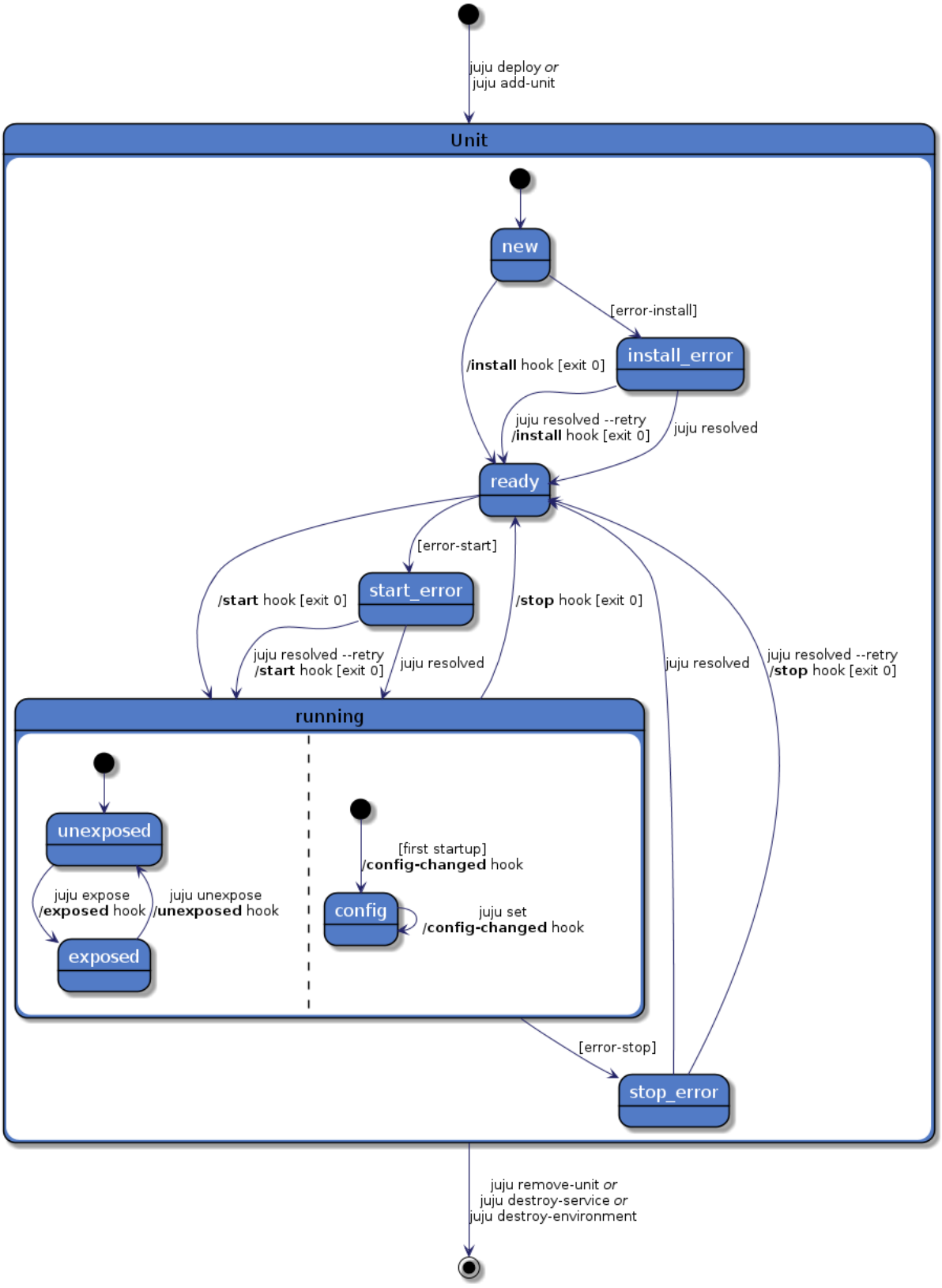


Figure 4.10: State machine representing the life-cycle of an unit (instance of a charm)

4.6 Application Layer

4.6.1 Structure of the Application

The application is composed of charms deployable with JuJu the cloud orchestrator.

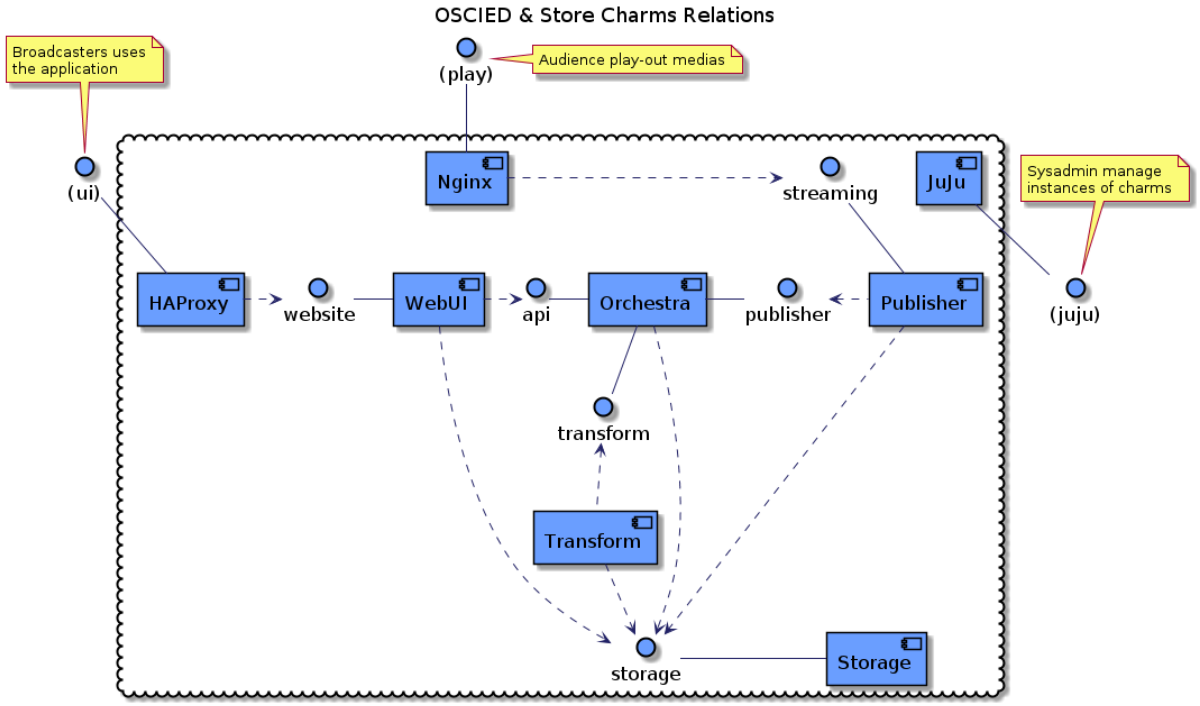


Figure 4.11: Charms of the project including charms from the store

This is an UML components diagram showing the architecture of the application deployed into a cloud ().

Here are the charms developed for this application :

Charm	Short description (services and goals)	Provides	Requires
WebUI	Provides a web based interface for the users of the platform (e.g. broadcasters)	website	storage api
Orches-tra	Provides the RESTful API and handles the DB & jobs scheduling (the brain)	api transform publisher	storage
Trans-form	Handles media encoding jobs to transform medias from/to various formats	(nothing)	storage transform
Pub-lisher	Handles media publication jobs to make medias available for the audience	(nothing)	storage publisher
Storage	Provides a shared medias storage mounted by other components of the application	storage	(nothing)

The *provides* and *requires* columns are the name of the relations required or provided by the charm.

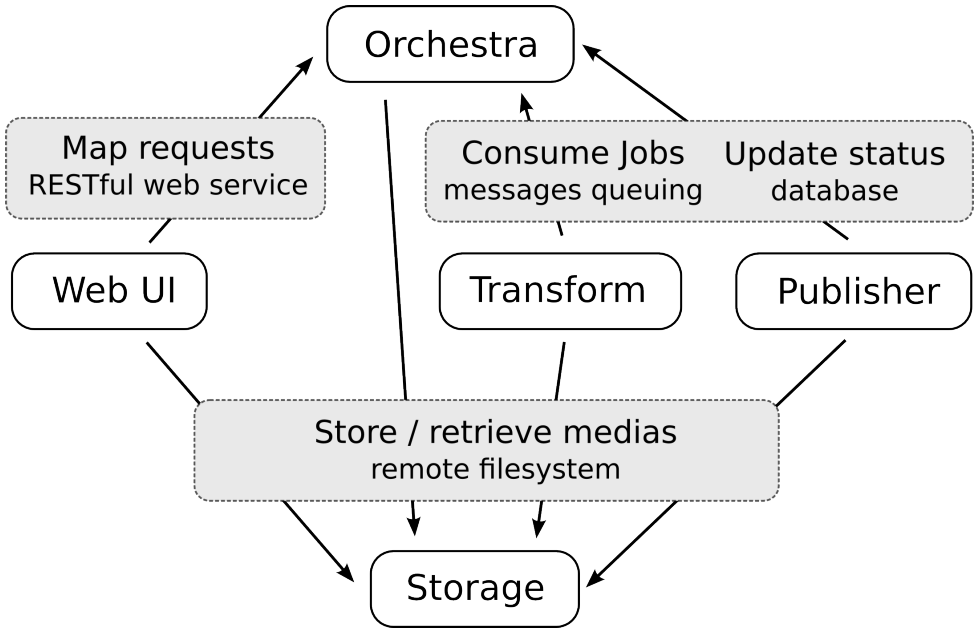


Figure 4.12: Purpose of the relations between components of OSCIED

Here are the relations :

Relation	Interface	Provider side	Requirer side
api	orchestra	Send RESTful API url	Update config. with RESTful API url
website	http	Update config. & add proxy to white list	Enable HTTP redirection
transform	subordinate	Send database & message broker connections	Update config. & connect to broker
publisher	subordinate	Send database & message broker connections	Update config. & connect to broker
storage	mount	Send parameters required to mount the storage	Update config. & mount the storage

Remark: Don't panic, they are only the required credentials to access to jobs related database !

The application's charms can be connected to charms of the JuJu Charms Store thanks to the nice contributors behind every of the charms. For example, one can imagine to plug Nginx charm in front of (e.g. 5x) publication points and closer to the user to reduce load and network traffic of backend publication points !

As you can imagine, next diagram will be a little bit more complicated, as it enter into charms to show you the Open-Source tools used internally by them.

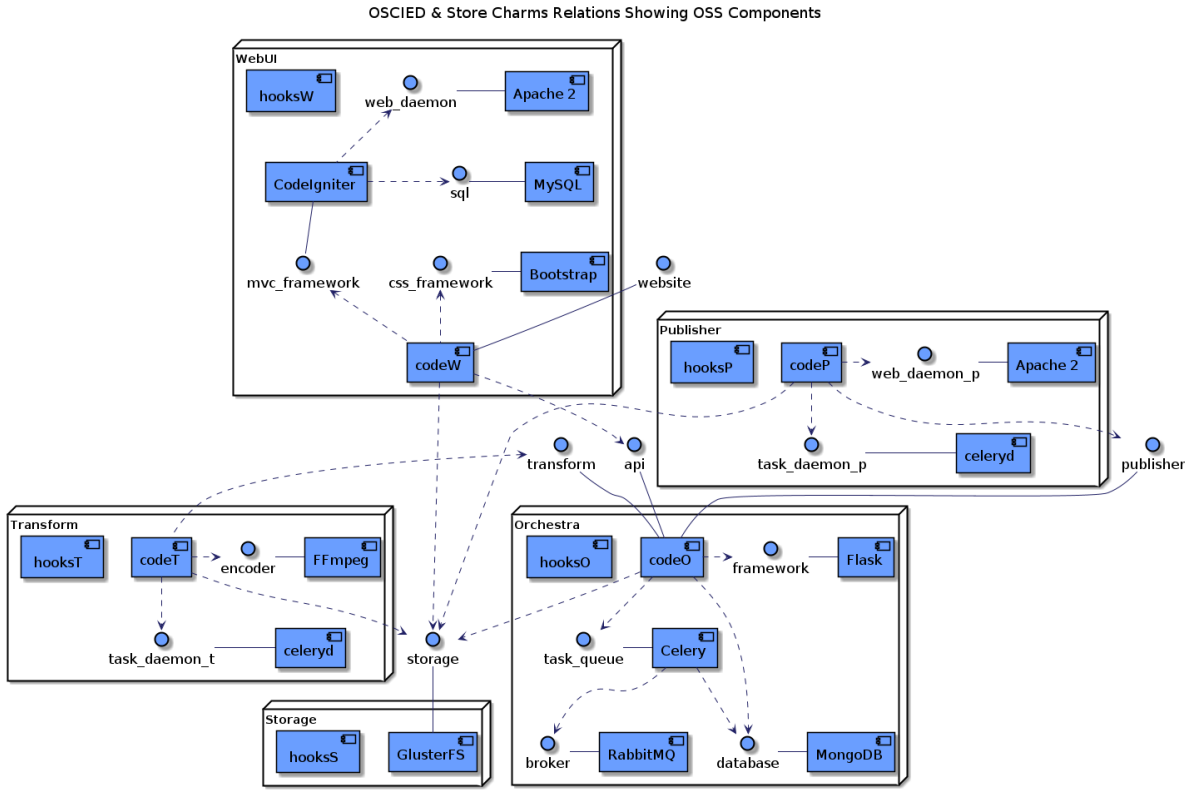


Figure 4.13: Charms of the project including involved sub-components

The charm's hooks (bash scripts) are represented by *hooks<X>* and the service's implementation (python source-code) is represented by *code<X>*.

4.6.2 OSCIED Advantages

Pluggable Design

The components of the application are defined in the form of pluggable charms and the relation between them are defined in the form of interfaces. One can potentially implement a compatible charm (eg. *StorageV2*) with the required interface and behavior and plug-in this new charm to any of the components requiring the implemented service.

For example, the *Storage* charm implements the storage service and provides the *storage* relation based on the *mount* interface. The *Orchestra* charm requires the *storage* relation based on the *mount* interface and you can plug the *Storage* units to the *Orchestra* as they are compatible !

Note: Please see *Application Layer* for further details.

So, for the needs of this preliminary demonstrator, the simplest form of a GlusterFS server is encapsulated into the *Storage* charm. This charm actually isn't capable of handling the scale-up/down of the service (adding or removing of instances, e.g. `juju add-unit`).

However, thanks to JuJu and to the pluggable design of this application, it is actually possible to go beyond this limiting factor by using your own network storage (see examples) !

Here are some example of what one can use for the storage service :

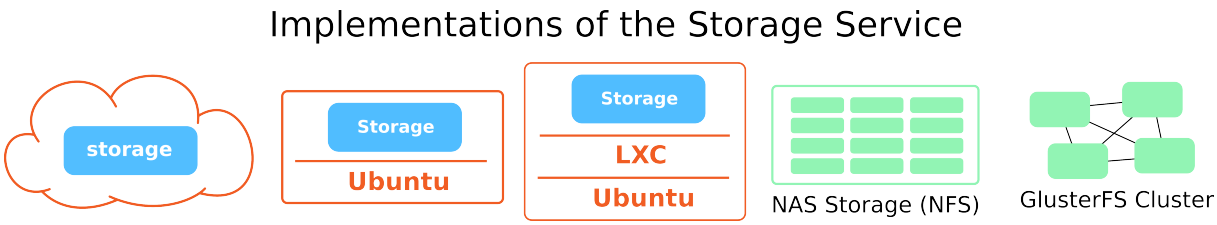


Figure 4.14: The project's storage charm can be replaced by any compatible physical or virtual storage

---

Note: If you prefer to use any external storage in place of the proposed charm, you need to specify the options related to storage in the configuration files used by juju to deploy the other components !

Strengths & Weaknesses

The strengths and weaknesses of this project are :

- (+) The project is based on Open-Source cloud-era tools in order to be elastic and scalable !
- (+) The Open-Source licensing means this project is community driven, no more vendor lock-in
- **The project is handled in a professional manner :**
  - (+) It is under revision control called subversion
  - (+) It is managed with a ticket system called TRAC
  - (+) The RESTful API methods responses are tested with JSONLint
  - (+) The RESTful API methods are tested with scripted unit-tests
- **The modular architecture add flexibility to any deployment :**
  - (+) The components can be deployed in standalone (testing purposes) or in any form you want
  - (+) The underlying technologies are interchangeable, you may decide to use NFS instead of GlusterFS
- (+) The services collaborate by exchanging asynchronous messages, there is no locking synchronous call
- **The application is designed to being deployed on commodity hardware :**
  - (+) You don't need to buy costly highly-available hardware such as RAID-based SAN or high performance fiber-channel
  - (+) Units of the services can fail, the others will continue handling requests, so no single point of failure here [4]
- **The cloud orchestrator JuJu add some kind of magic to the project, the application :**
  - (+) Is easy to deploy thanks to automation handled by application's charms hooks scripts
  - (+) Is scalable as it is easy to adapt services to load by adding or removing units to services
  - (+) Is flexible and deployable to a wide variety of targets, such as clouds, clusters, servers ...
  - (+) Is pluggable to charms developed by the community such as haproxy, nginx, nagios, ...
- (+) The orchestrator provides a RESTful API, one can implement a higher-level tools based on OSCIED !
- **The distributed tasks queue Celery add some kind of magic to the orchestrator :**
  - (+) The enterprise business rules can be implemented by connecting the workers [5] to the right tasks queues and sending jobs to the right queues.

[4] This is true for the services that actually can scale-up/down such as the transform, publisher and webui charms.
[5] They are actually two kind of workers, the transform (encoding jobs) and publisher (publication jobs).

- **The preliminary demonstrator is not perfect, some work is required to make it better, actually :**
  - (-) The storage charms doesn't handle clustering (not scalable)
  - (-) The orchestrator charm cannot be highly-available (not scalable)
  - (-) The orchestrator charm cannot auto-scale the workers
  - (-) The orchestrator only uses the basic features of Celery !

4.6.3 Various OSS Tools Involved

Operating system : Ubuntu Quantal Server from Canonical

> " Ubuntu is a computer operating system based on the Debian Linux distribution and distributed as free and Open-Source software, using its own desktop environment. ... Ubuntu is sponsored by the UK-based company Canonical Ltd., owned by South African entrepreneur Mark Shuttleworth. Canonical generates revenue by selling technical support and services related to Ubuntu, while the operating system itself is entirely free of charge. The Ubuntu project is committed to the principles of free software development; people are encouraged to use free software, improve it, and pass it on. " source Wikipedia (UbuntuOS)

4.6.4 Media's State Machine

It may seem obvious but the application not only stores media files into the shared storage but it also stores informations about it into database such as metadata (title, add_date, ... whatever you need), the id of the user who registered the media, ... And the actual state of the media :

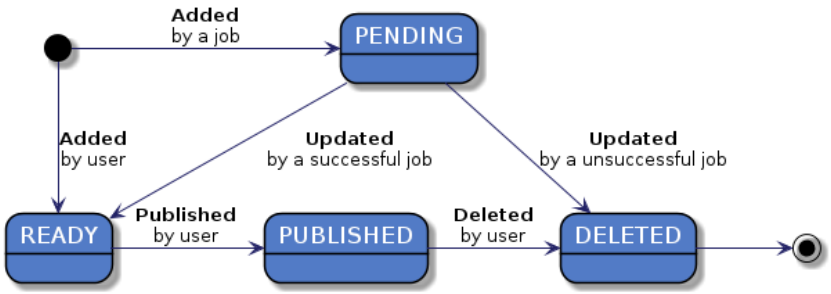


Figure 4.15: State machine of a media from registration to deletion

4.6.5 OSCIED-Orchestra : The Orchestrator

See also:

You can browse the source of the Orchestrator or browse *OSCIED - Orchestra RESTful API* for further details.

OSS Tools

- Flask Python Micro Web Framework
- PyMongo Python module for working with MongoDB
- MongoDB Scalable, High Performance NoSQL Database from 10gen
- RabbitMQ AMQP Message Broker from vmware
- Celery Distributed Task Queue
- JuJu Cloud Orchestrator from Canonical

Introduction

This component is the brain of the application, responsible of :

- the RESTful API, to expose application's functionalities to user
- the database, to store application's data (users, profiles, jobs, ...)
- the message broker, to communicate with workers (transform & publisher)
- the cloud orchestrator, to manage other components [6]

The main advantage of providing an API *and* a separated web user interface is that one can use functionalities of the application by programming an higher-level tool using the API directly. It was also really useful for me to develop & test the API by scripting uses cases helped with cURL.

The orchestrator is developed in Python and I actually choose this programming language for the following reasons:

- The tools used for this project, OpenStack, JuJu, ... are developed in Python
- The most interesting Open-Source tools involved in this charm are also developed in Python
- I like writing lesser code and I actually really want to practice this language !

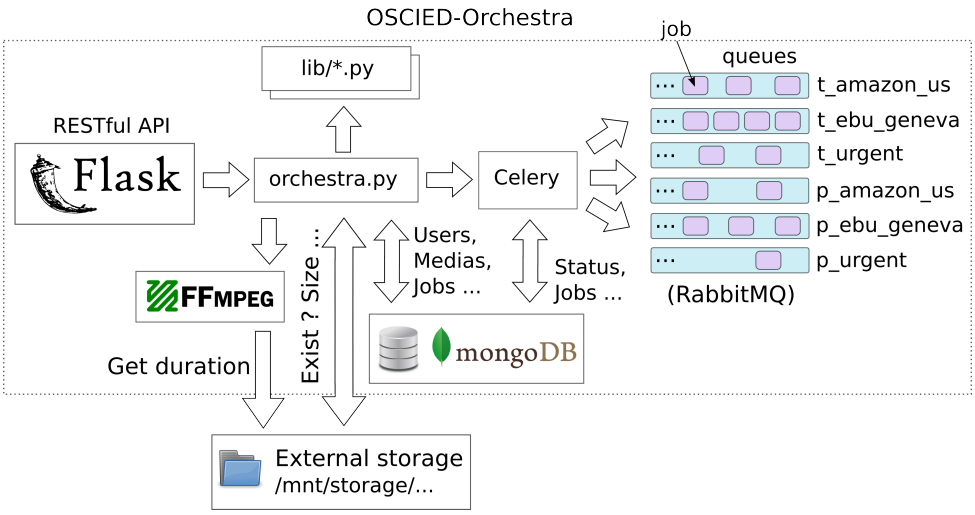


Figure 4.16: Architecture of the Orchestrator

[6] This feature will be available on a future release.

Charm's Configuration

You can start the charm without specifying any configuration (default values will be used, see `appendices-orchestra`) but I strongly recommend to specify your own values in production !

- **verbose** Set verbose logging
- **root_secret** Secret key used by API clients to manage users
- **nodes_secret** Secret key used by workers/nodes to callback API when they finish their job
- **repositories_user** OSCIED charms repositories client username
- **repositories_pass** OSCIED charms repositories client password
- **webui_repository** OSCIED Web UI charm will be checked out locally under ~/charms/(release)/oscied-webui
- **transform_repository** OSCIED Transform charm will be checked out locally under ~/charms/(release)/oscied-transform
- **publisher_repository** OSCIED Publisher charm will be checked out locally under ~/charms/(release)/oscied-publisher
- **mongo_admin_password** Database administrator password
- **mongo_nodes_password** Database nodes password [7]
- **rabbit_password** Messaging queue user's password [2]
- **storage_ip** Shared storage hostname / IP address (see interface mount of NFS charm) [8]
- **storage_fstype** Shared storage filesystem type (e.g. NFS) [3]
- **storage_mountpoint** Shared storage mount point (e.g. for NFS - /srv/data) [3]
- **storage_options** Shared storage options (e.g. for NFS - rw,sync,no_subtree_check)

[7] This secret is forwarded by the coordinator to managed units (transform, publish)
[8] If all options are set this will override and disable storage relation

Charm's Hooks Activity Diagrams

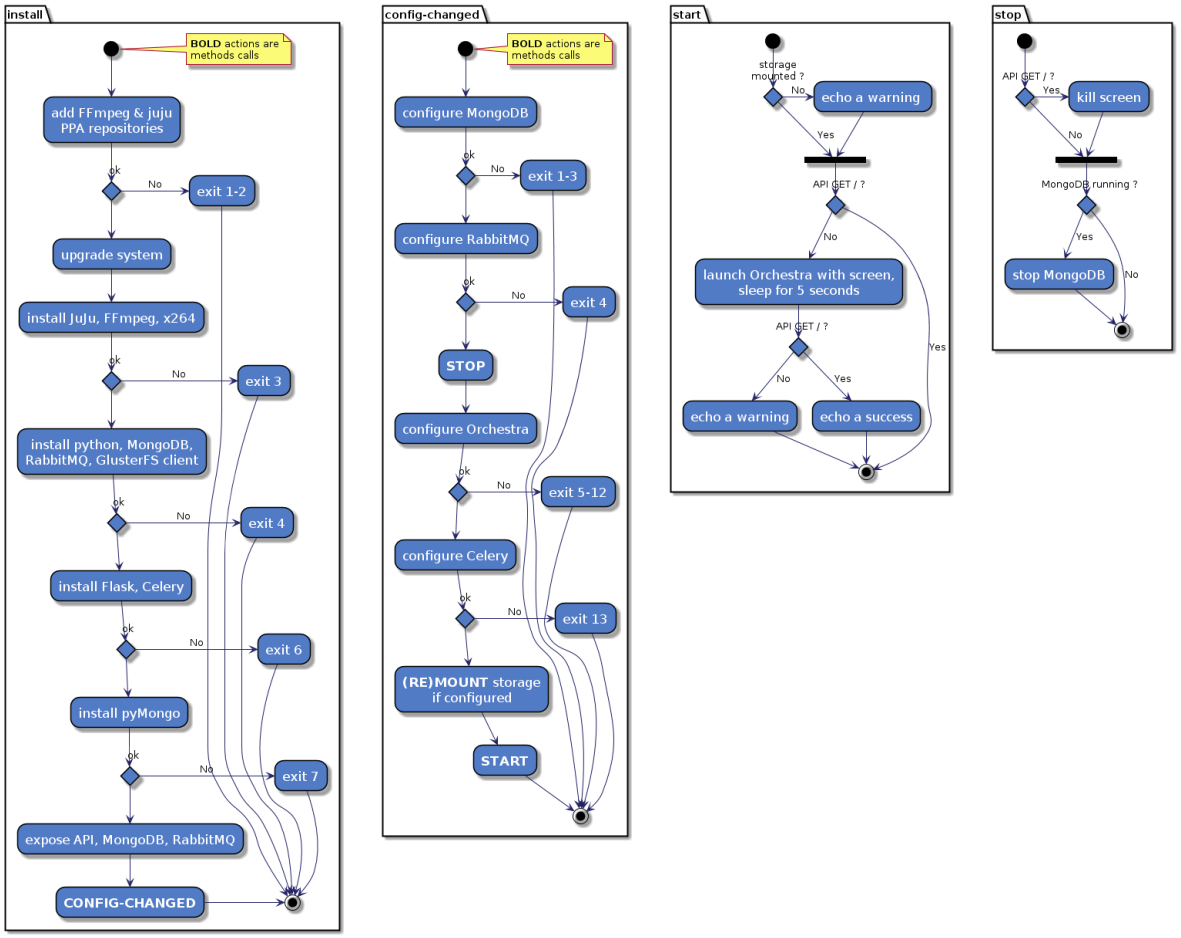


Figure 4.17: Activity diagram of Orchestra unit life-cycle hooks

Charm's Relations

- Provides : API [Orchestra], Transform [Subordinate], Publisher [Subordinate]
- Requires : Storage [Mount]

Warning: The unit's daemons will not start until a shared storage is mounted (via the storage relation or by specifying it into configuration).

4.6.6 OSCIED-Transform : The Transcoder

See also:

You can browse the source of the Transcoder

OSS Tools

- Celery Distributed Task Queue
- FFmpeg Complete Multimedia Framework from the FFmpeg Foundation

Introduction

This component is the worker specialized in handling transformation jobs. In fact this is *celeryd* daemon that handles the requests and maps jobs to transform functions calls. This charm's start hook will launch and connect the daemon to the message broker's queue(s) specified in configuration [9].

For example, one can choose that the workers running on his private OpenStack cloud will handle *private* & *high priority* transformation requests by setting worker's *rabbit_queues* option to "t_priv,t_high". Then one only need to launch jobs of such kind in one of the defined queues (*t_priv*, *t_high*) and that's it !

Moreover, one can choose to explicitly target a unique worker (e.g. *myWorker1*) by sending jobs to the queue *myWorker1*, this is another interesting feature offered by the application.

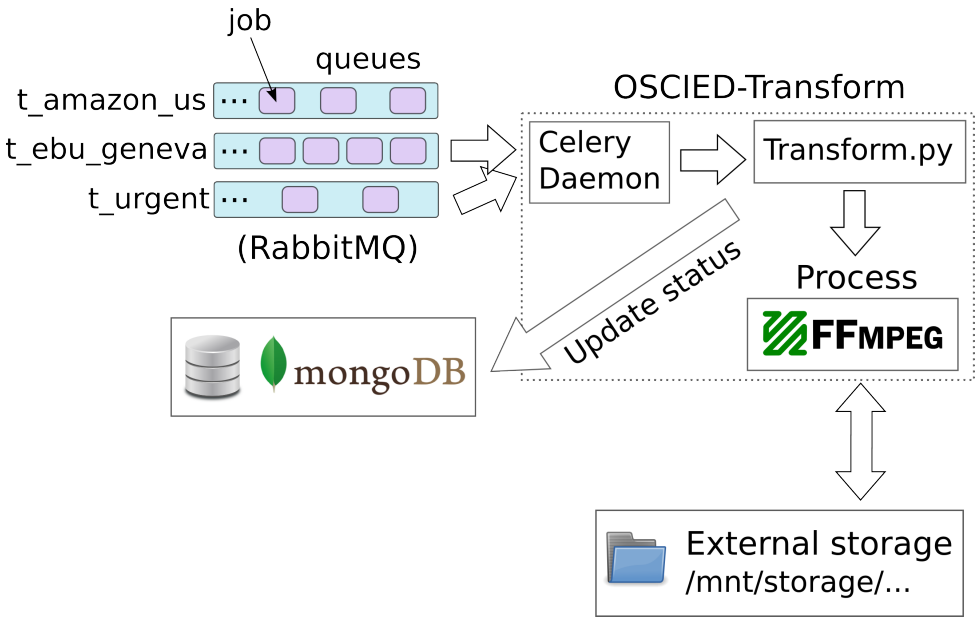


Figure 4.18: Architecture of the Transform Unit

Charm's Configuration

You can start the charm without specifying any configuration (default values will be used, see `appendices-transform`) but I strongly recommend to specify your own values in production !

- **verbose** Set verbose logging
- **concurrency** Amount of tasks the worker can handle simultaneously
- **rabbit_queues** Worker connect to queues to receive jobs

[9] Add worker's name to queues list, this make possible to launch jobs to this specific worker

- **mongo_connection** Orchestrator database connection [10]
- **rabbit_connection** Orchestrator message broker connection [2]
- **storage_ip** Shared storage hostname / IP address (see interface mount of NFS charm) [11]
- **storage_fstype** Shared storage filesystem type (e.g. NFS) [3]
- **storage_mountpoint** Shared storage mount point (e.g. for NFS - /srv/data) [3]
- **storage_options** Shared storage options (e.g. for NFS - rw,sync,no_subtree_check)

Charm's Hooks Activity Diagrams

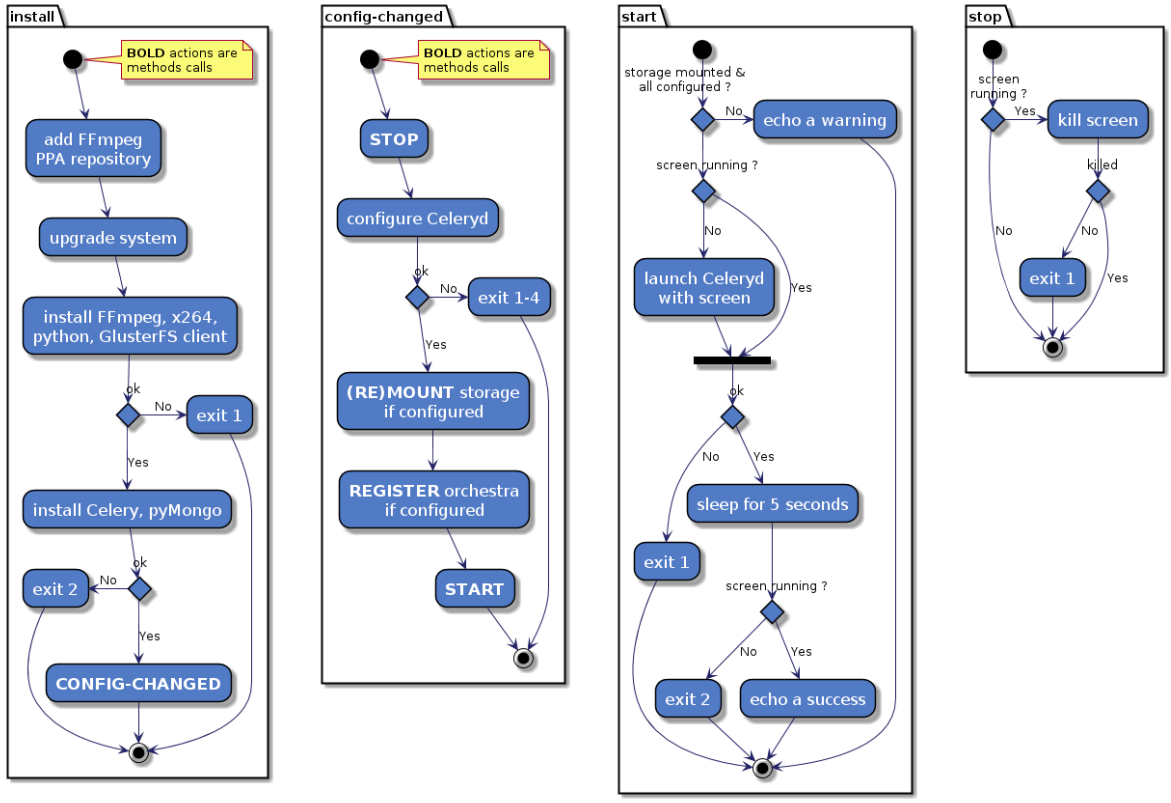


Figure 4.19: Activity diagram of Transform unit life-cycle hooks

Charm's Relations

- Provides : (nothing)
- Requires : Storage [Mount], Transform [Subordinate]

Warning: The unit's daemon will not start until both conditions are fulfilled :
- A shared storage is mounted (via the storage relation or by specifying it into configuration)
- An orchestrator is registered (via the transform relation or by specifying it into configuration)

[10] If all options are set this will override and disable transform relation
[11] If all options are set this will override and disable storage relation

Job State Machine

The orchestrator stores informations about the transformation jobs into database in parallel to the informations that Celery also stores in. It may seem as duplicate however I choose to do as such for good reasons :

- Additional informations about the jobs can be stored.
- One can choose to replace Celery to use any other task queuing technology.
- Listing of jobs is easy to implement, no needs of Celery's inspect & filtering.

Celery's keep track of jobs and stores informations about jobs into a *backend*, the orchestrator's database (MongoDB).

> " During its lifetime a task will transition through several possible states, and each state may have arbitrary metadata attached to it. When a task moves into a new state the previous state is forgotten about, but some transitions can be deducted, (e.g. a task now in the FAILED state, is implied to have been in the STARTED state at some point). " [12]

So, the transformation jobs stored in database has a *statistic* field that is filled with values mainly generated by the orchestrator such as *add_date*. The state machine diagram shows what is store in this field plus the values that are appended to job's state metadata.

Remark: The orchestrator RESTful API transformation methods responses contains job's metadata, appended into *statistic* field.

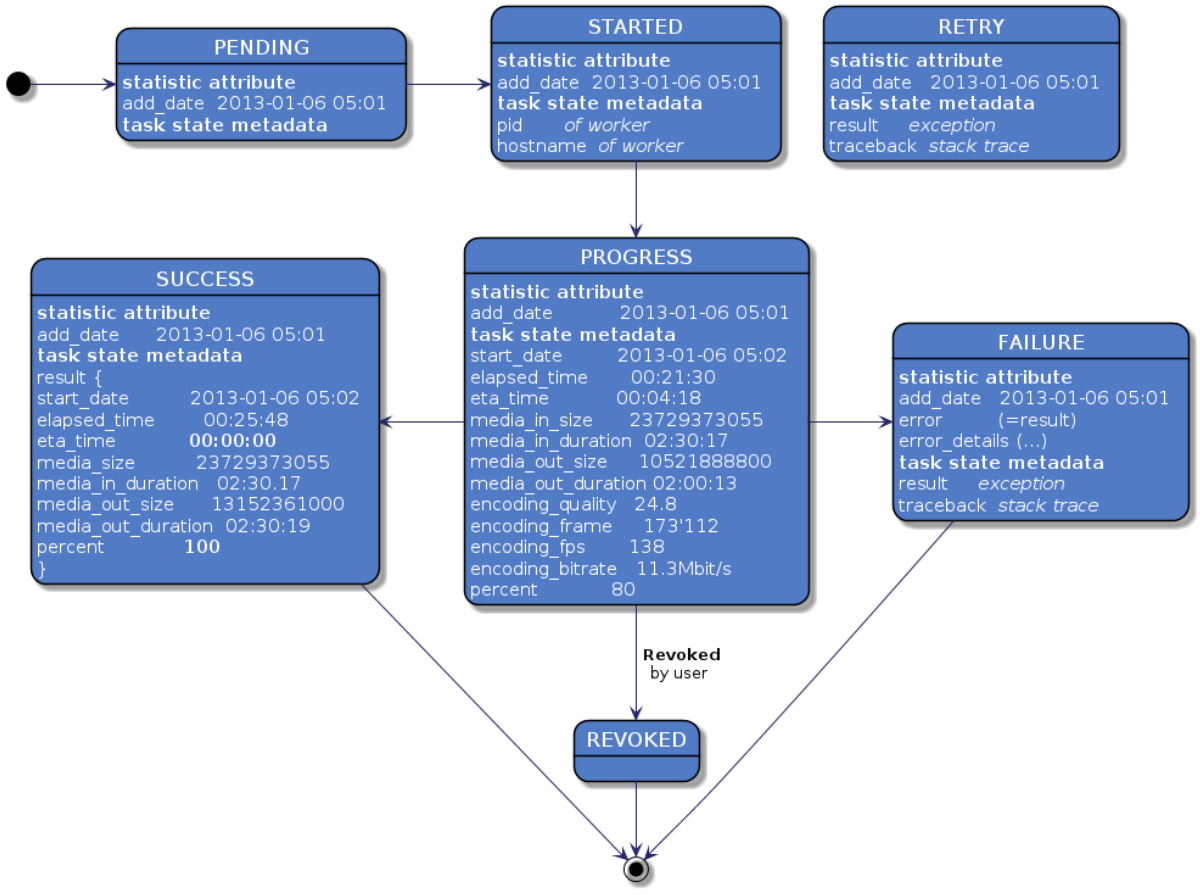


Figure 4.20: State machine of an encoding job (transform -> transform)

[12] Celery Tasks Page – http://docs.celeryproject.org/en/latest/userguide/tasks.html

Job Sequence Diagrams

A Successful Job

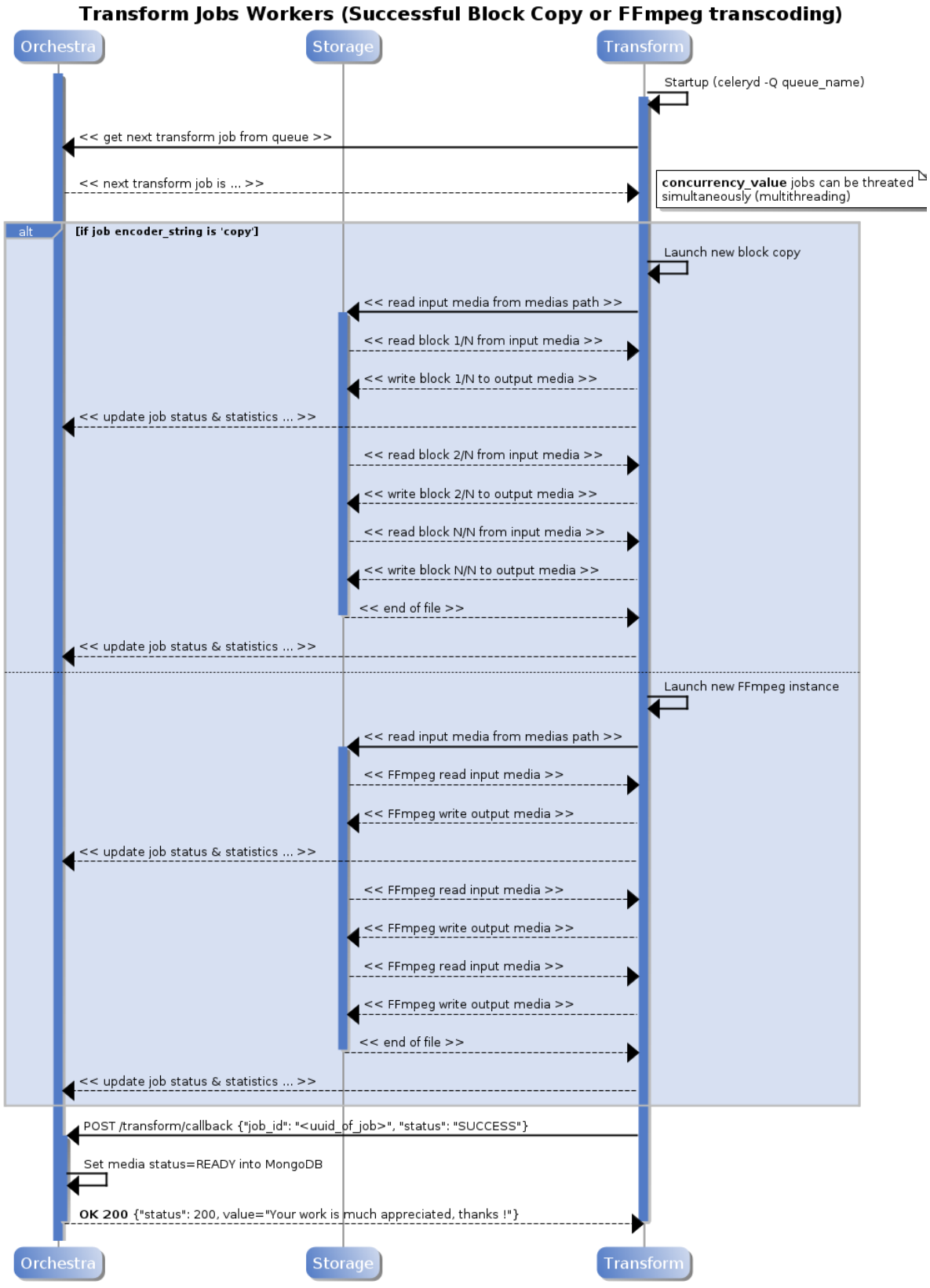


Figure 4.21: Sequence diagram of a successful encoding job (transform -> transform)

4.6.7 OSCIED-Publisher : The Publication Point

See also:

You can browse the source of the Publication Point

OSS Tools

- Celery Distributed Task Queue
- Apache 2 HTTP Server from the Apache Software Foundation
- H264 Streaming Module from CodeShop

Introduction

This component is the worker specialized in handling publication jobs. In fact this is *celeryd* daemon that handles the requests and maps jobs to publisher functions calls. This charm's start hook will launch and connect the daemon to the message broker's queue(s) specified in configuration [13].

For example, one can choose that the workers running on Amazon AWS cloud will handle *public* & *low priority* publication requests by setting worker's *rabbit_queues* option to "t_pub,t_low". Then one only need to launch jobs of such kind in one of the defined queues (*t_pub*, *t_low*) and that's it !

Moreover, one can choose to explicitly target a unique worker (e.g. *myWorker2*) by sending jobs to the queue *myWorker2*, this is another interesting feature offered by the application.

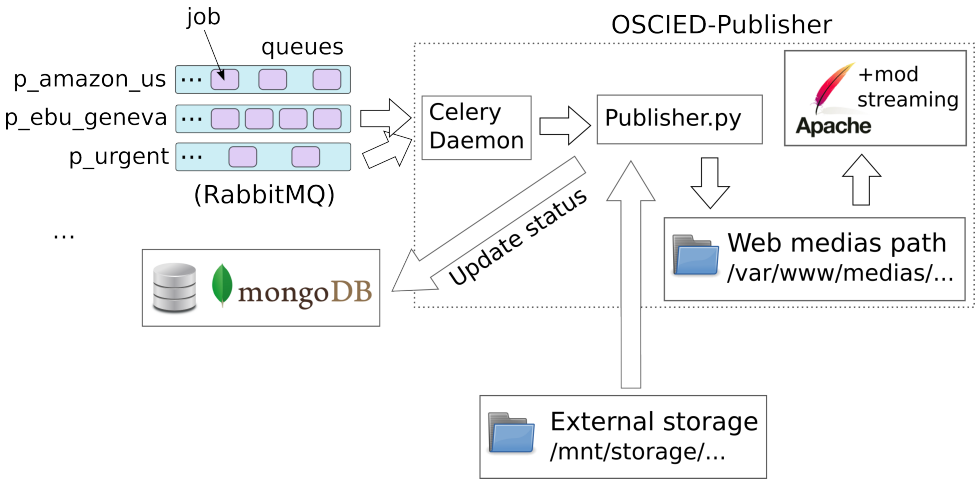


Figure 4.22: Architecture of the Publication Point

Charm's Configuration

You can start the charm without specifying any configuration (default values will be used, see `appendices-publisher`) but I strongly recommend to specify your own values in production !

- **verbose** Set verbose logging
- **concurrency** Amount of tasks the worker can handle simultaneously
- **rabbit_queues** Worker connect to queues to receive jobs
- **max_upload_size** Maximum size for file uploads
- **max_execution_time** Maximum time for PHP scripts

[13] Add worker's name to queues list, this make possible to launch jobs to this specific worker

- **max_input_time** Maximum time for HTTP post
- **mongo_connection** Orchestrator database connection [14]
- **rabbit_connection** Orchestrator message broker connection [2]
- **storage_ip** Shared storage hostname / IP address (see interface mount of NFS charm) [15]
- **storage_fstype** Shared storage filesystem type (e.g. NFS) [3]
- **storage_mountpoint** Shared storage mount point (e.g. for NFS - /srv/data) [3]
- **storage_options** Shared storage options (e.g. for NFS - rw,sync,no_subtree_check)

Charm's Hooks Activity Diagrams

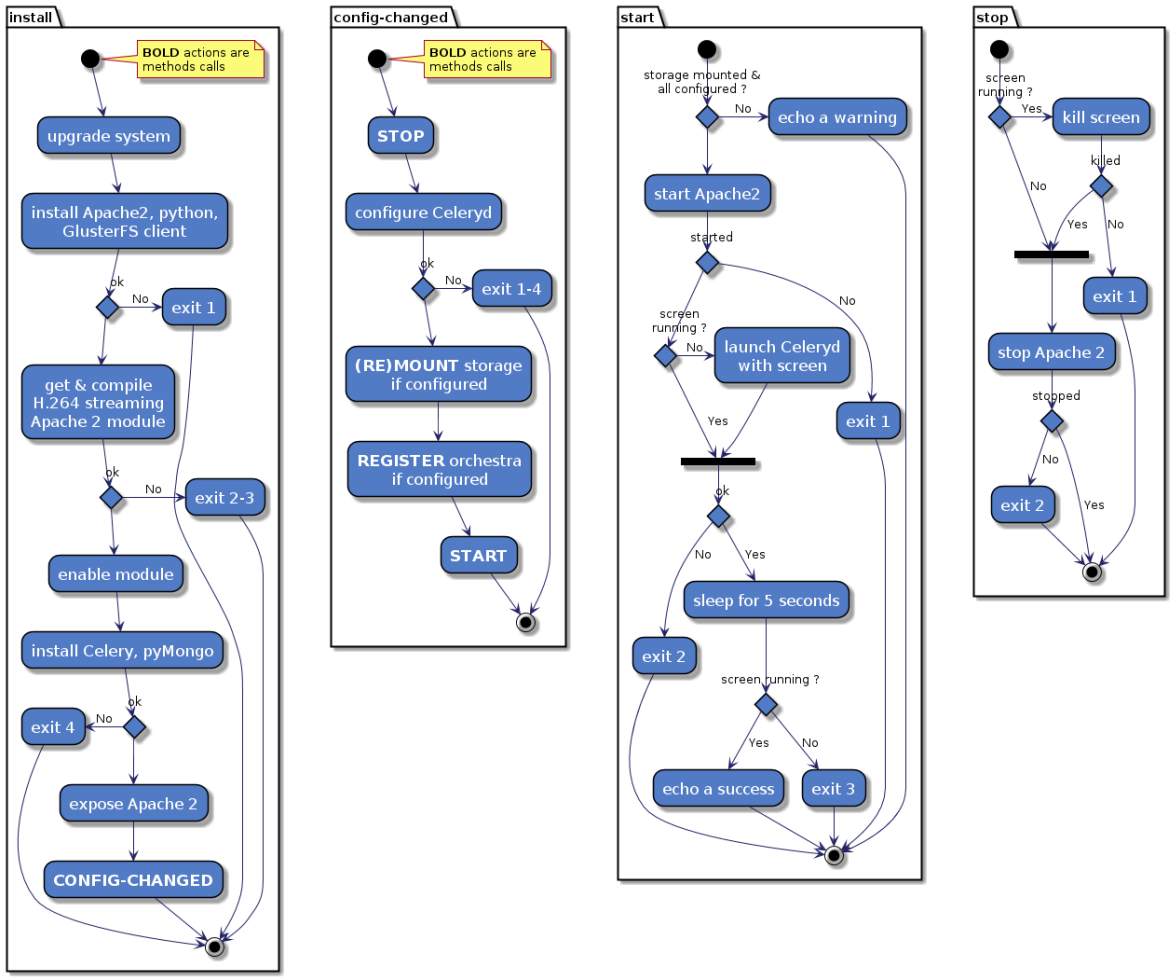


Figure 4.23: Activity diagram of Publisher unit life-cycle hooks

Charm's Relations

- Provides : (nothing)
- Requires : Storage [Mount], Publisher [Subordinate]

[14] If all options are set this will override and disable publisher relation
[15] If all options are set this will override and disable storage relation

Warning: The unit's daemon will not start until both conditions are fulfilled :
- A shared storage is mounted (via the storage relation or by specifying it into configuration)
- An orchestrator is registered (via the publisher relation or by specifying it into configuration)

Job State Machine

The orchestrator stores informations about the publication jobs into database in parallel to the informations that Celery also stores in. It may seem as duplicate however I choose to do as such for good reasons :

- Additional informations about the jobs can be stored.
- One can choose to replace Celery to use any other task queuing technology.
- Listing of jobs is easy to implement, no needs of Celery's inspect & filtering.

Celery's keep track of jobs and stores informations about jobs into a *backend*, the orchestrator's database (MongoDB).

> " During its lifetime a task will transition through several possible states, and each state may have arbitrary metadata attached to it. When a task moves into a new state the previous state is forgotten about, but some transitions can be deducted, (e.g. a task now in the FAILED state, is implied to have been in the STARTED state at some point). " [16]

So, the publication jobs stored in database has a *statistic* field that is filled with values mainly generated by the orchestrator such as *add_date*. The state machine diagram shows what is store in this field plus the values that are appended to job's state metadata.

Remark: The orchestrator RESTful API publication methods responses contains job's metadata, appended into *statistic* field.

[16] Celery Tasks Page – http://docs.celeryproject.org/en/latest/userguide/tasks.html

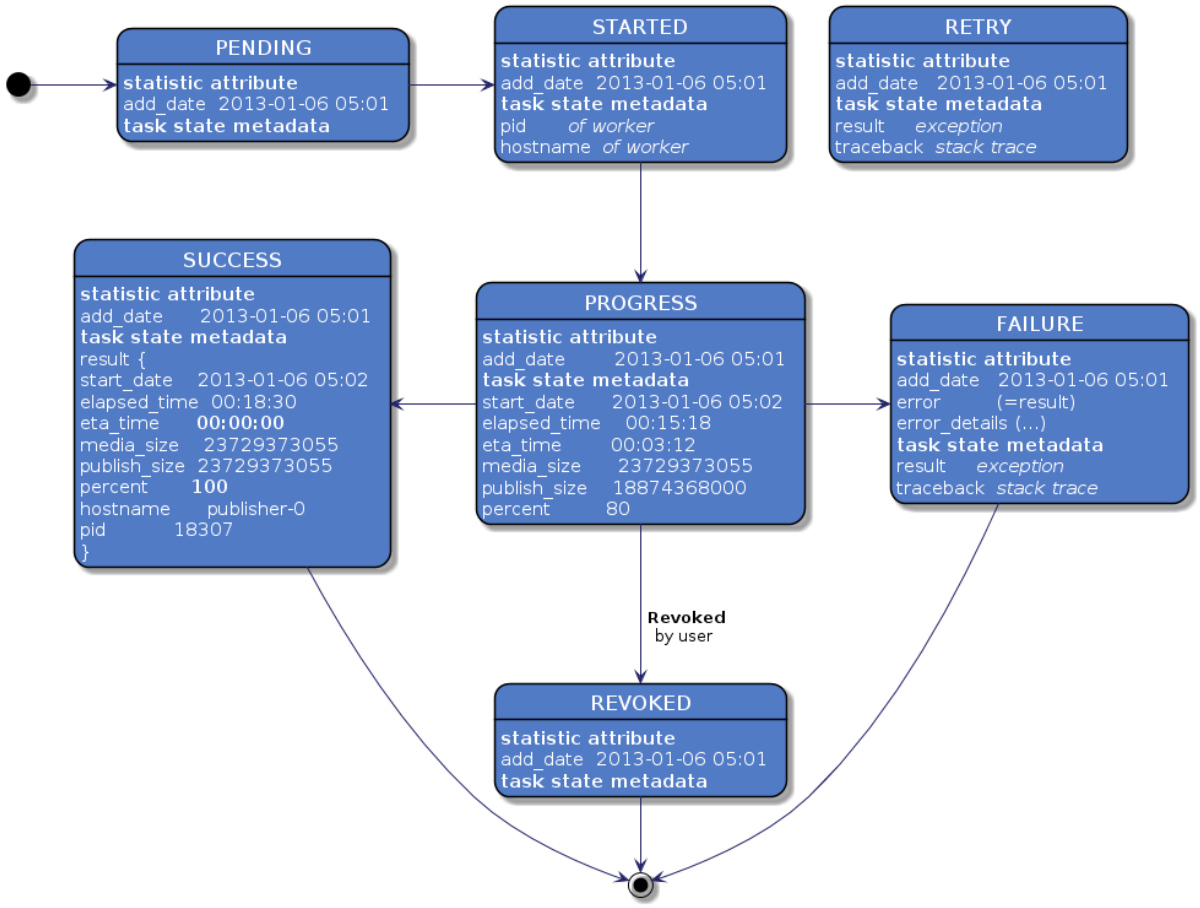


Figure 4.24: State machine of a publication job (publisher -> publish)

Job Sequence Diagrams

A Successful Job

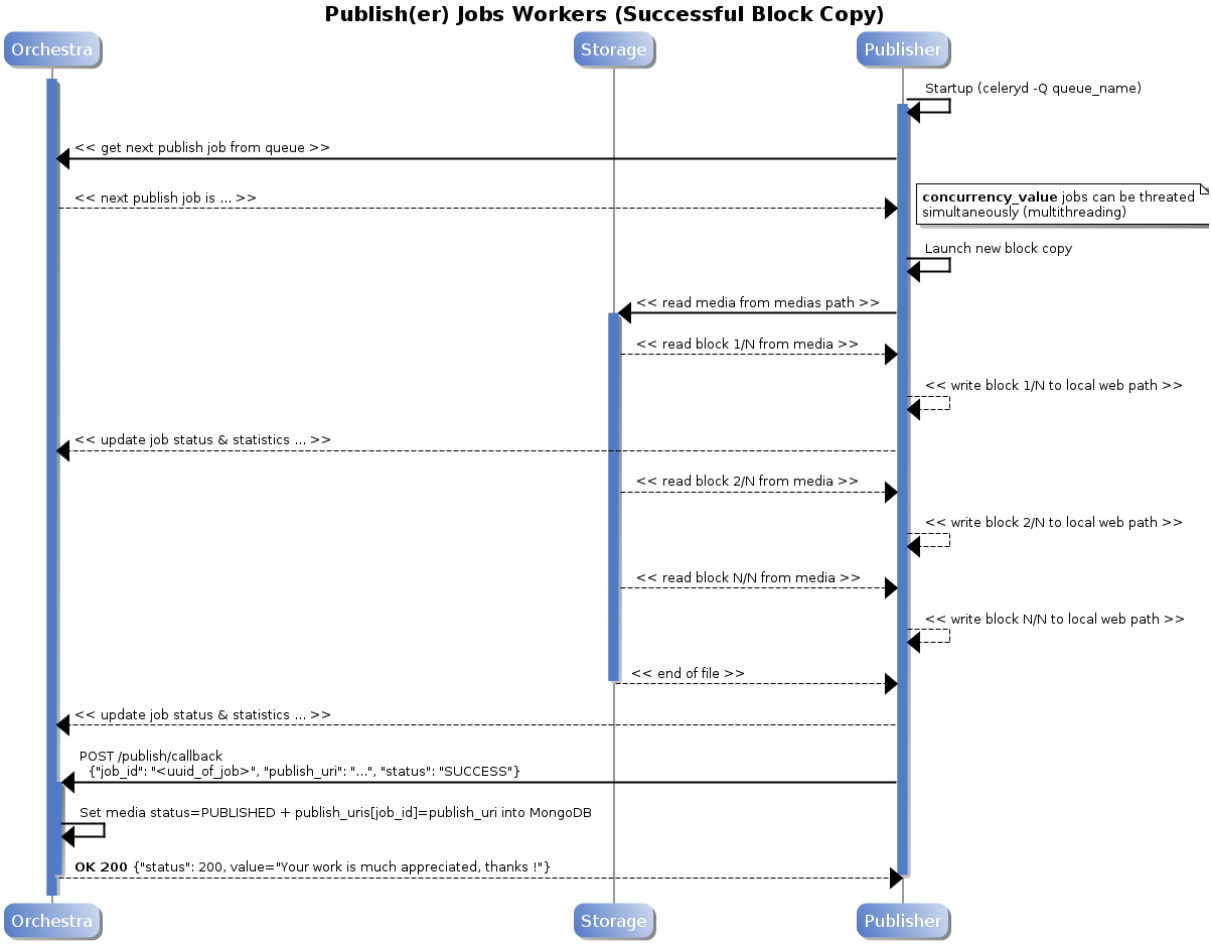


Figure 4.25: Sequence diagram of a successful publication job (publisher -> publish)

4.6.8 OSCIED-Storage : The Media Storage

See also:

You can browse the source of the Medias Storage

OSS Tools

- GlusterFS Highly Scalable Distributed Filesystem

Introduction

This component is the application's scale-out network attached storage responsible of the medias collection.

Actual version of the charm is keep simple (KISS) for the purposes of the demonstrator. However the charm encapsulate a GlusterFS server and they are numerous advantages of using this technology, as listed in *GlusterFS*.

Charm's Configuration

You can start the charm without specifying any configuration (default values will be used, see `appendices-storage`) but I strongly recommend to specify your own values in production !

- **verbose** Set verbose logging
- **concurrency** Amount of tasks the worker can handle simultaneously
- **rabbit_queues** Worker connect to queues to receive jobs
- **max_upload_size** Maximum size for file uploads
- **max_execution_time** Maximum time for PHP scripts
- **max_input_time** Maximum time for HTTP post
- **mongo_connection** Orchestrator database connection [17]
- **rabbit_connection** Orchestrator message broker connection [1]
- **storage_ip** Shared storage hostname / IP address (see interface mount of NFS charm) [18]
- **storage_fstype** Shared storage filesystem type (e.g. NFS) [2]
- **storage_mountpoint** Shared storage mount point (e.g. for NFS - /srv/data) [2]
- **storage_options** Shared storage options (e.g. for NFS - rw,sync,no_subtree_check)

[17] If all options are set this will override and disable publisher relation
[18] If all options are set this will override and disable storage relation

Charm's Hooks Activity Diagrams

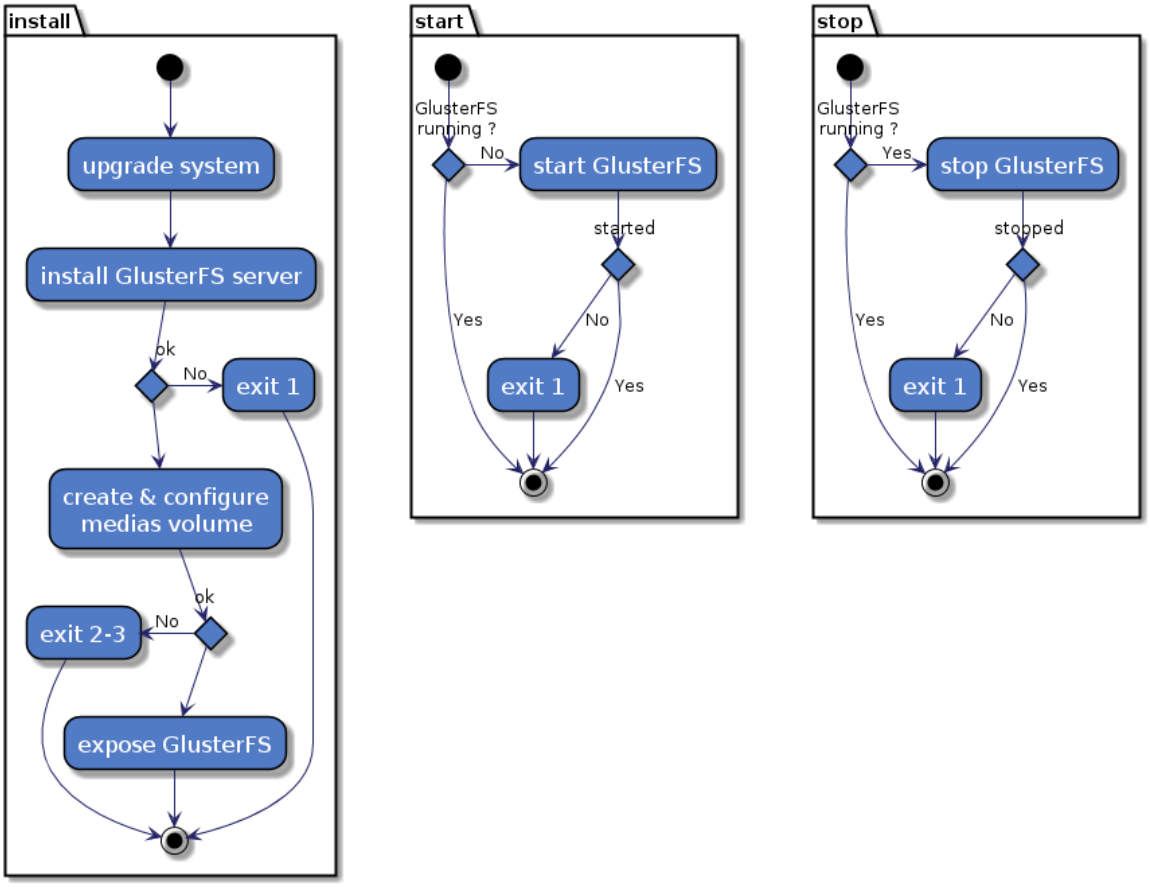


Figure 4.26: Activity diagram of Storage unit life-cycle hooks

Charm's Relations

- Provides : Storage [Mount]
- Requires : (nothing)

4.6.9 OSCIED-WebUI : The Web User Interface

See also:

You can browse the source of the Web User Interface

OSS Tools

- Apache 2 HTTP Server from the Apache Software Foundation
- CodeIgniter Powerful PHP MVC Framework from EllisLab Ticket 117
- CSS Bootstrap Front-end Framework from Twitter Ticket 36

Introduction

This component is the user interface of the application providing an uncluttered, user-friendly web interface for using the functionalities of the application. The user's actions are mapped to orchestrator's RESTful API calls.

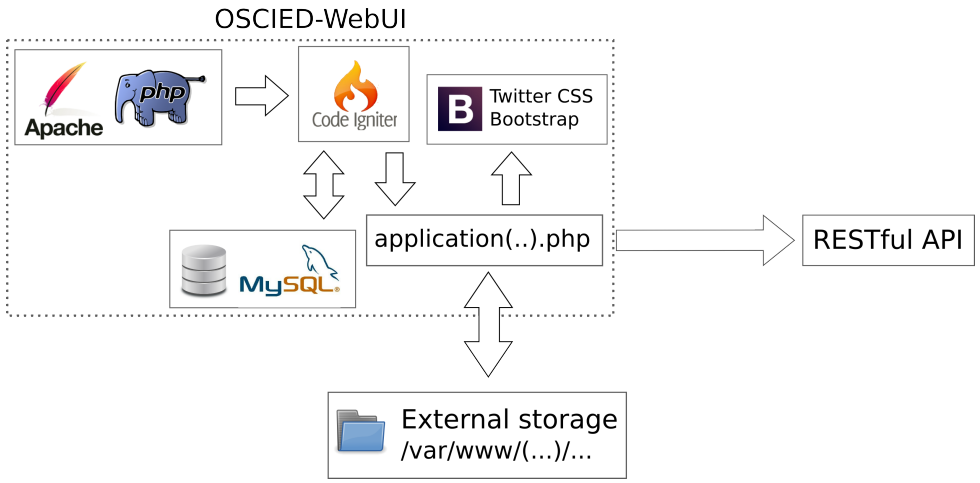


Figure 4.27: Architecture of the Web User Interface

Charm's Configuration

You can start the charm without specifying any configuration (default values will be used, see `appendices-webui`) but I strongly recommend to specify your own values in production !

- **verbose** Set verbose logging
- **max_upload_size** Maximum size for file uploads
- **max_execution_time** Maximum time for PHP scripts
- **max_input_time** Maximum time for HTTP post
- **mysql_my_password** Password for phpmyadmin
- **mysql_root_password** Password of MySQL root user
- **mysql_user_password** Password of MySQL webui user
- **api_url** Orchestrator REST API address [19]
- **storage_ip** Shared storage hostname / IP address (see interface mount of NFS charm) [20]

[19] If all options are set this will override and disable api relation
[20] If all options are set this will override and disable storage relation

- **storage_fstype** Shared storage filesystem type (e.g. NFS) [2]
- **storage_mountpoint** Shared storage mount point (e.g. for NFS - /srv/data) [2]
- **storage_options** Shared storage options (e.g. for NFS - rw,sync,no_subtree_check)

Charm's Hooks Activity Diagrams

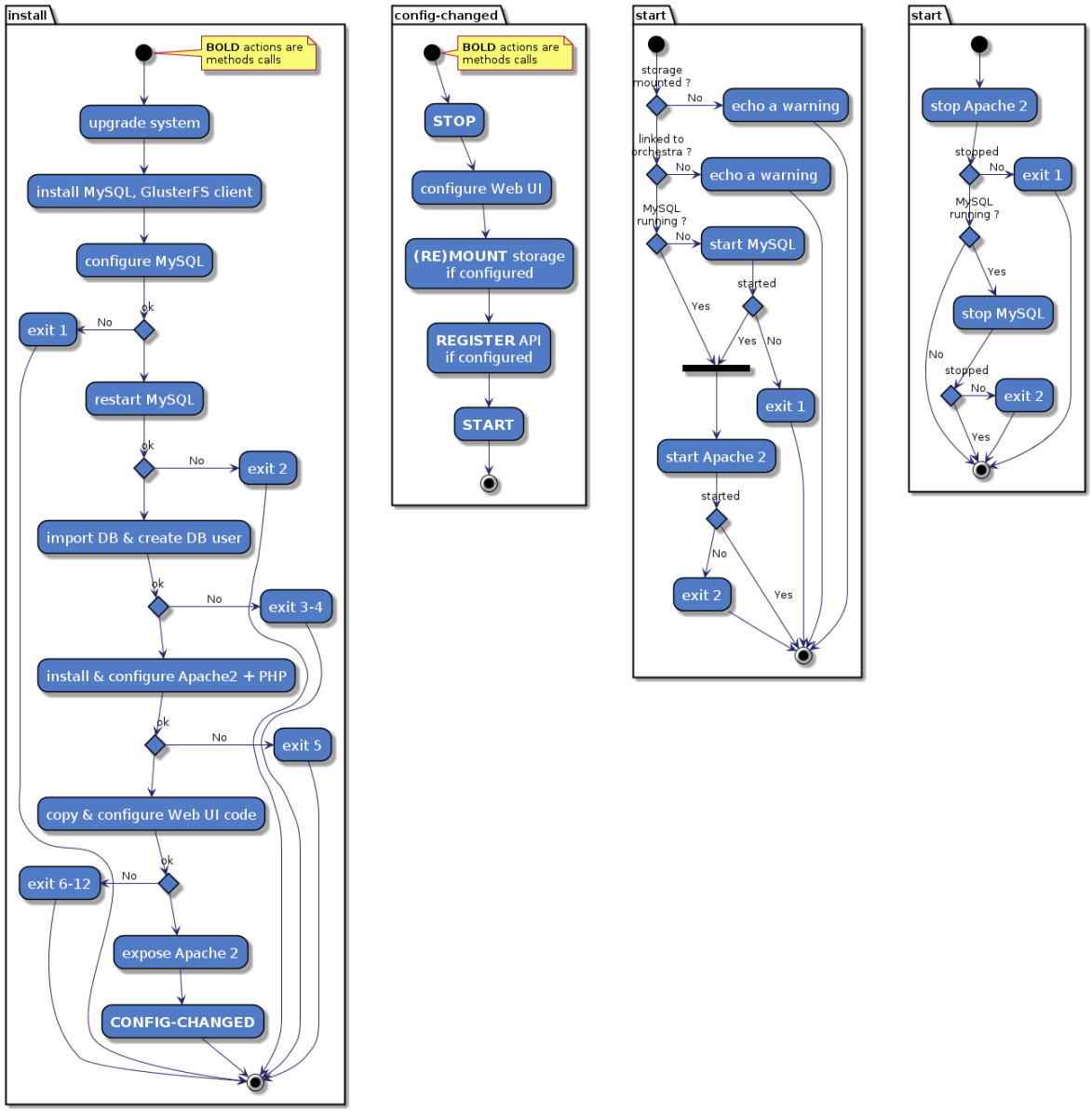


Figure 4.28: Activity diagram of WebUI unit life-cycle hooks

Charm's Relations

- Provides : Website [HTTP]
- Requires : Storage [Mount], API [Orchestra]

Warning: The unit's daemon will not start until both conditions are fulfilled :

- A shared storage is mounted (via the storage relation or by specifying it into configuration)
- An orchestrator is registered (via the api relation or by specifying it into configuration)

Users Tab

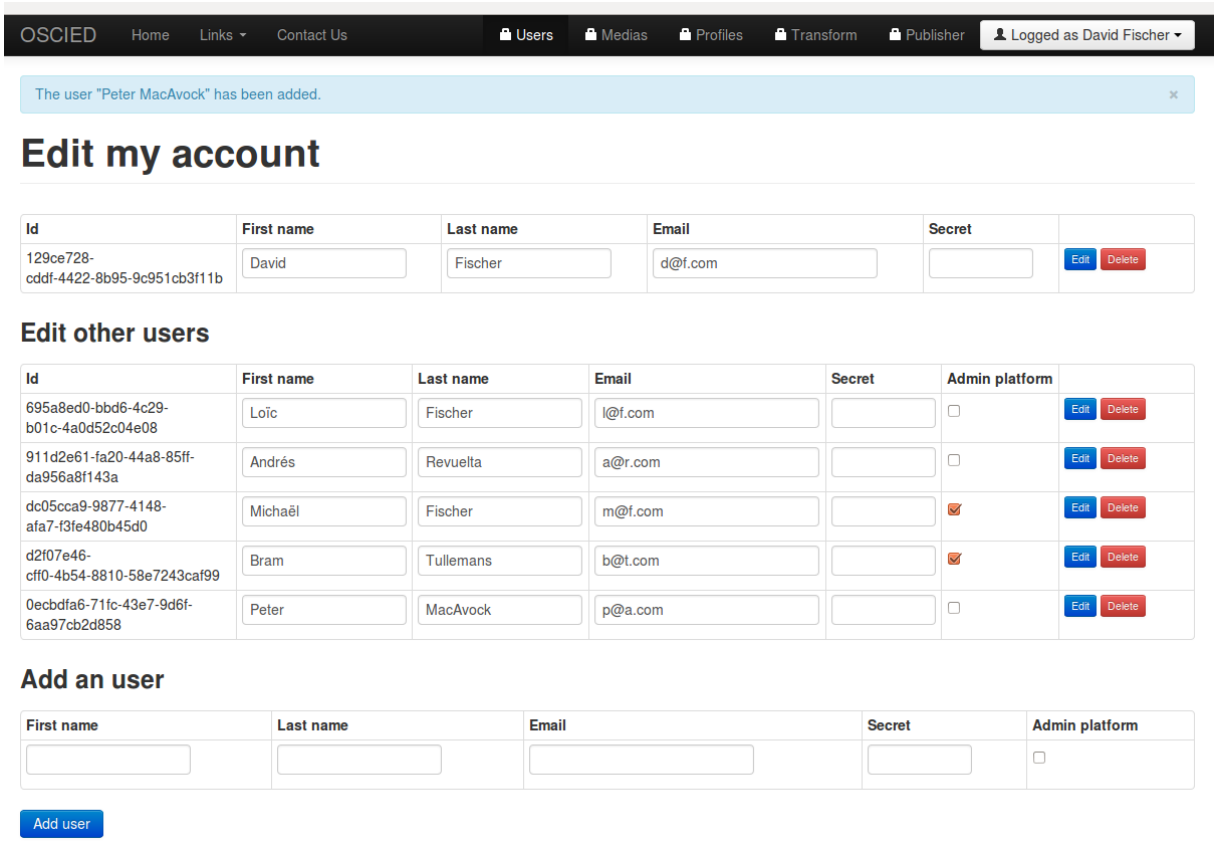


Figure 4.29: Adding an user with the users add form (WebUI -> Users)

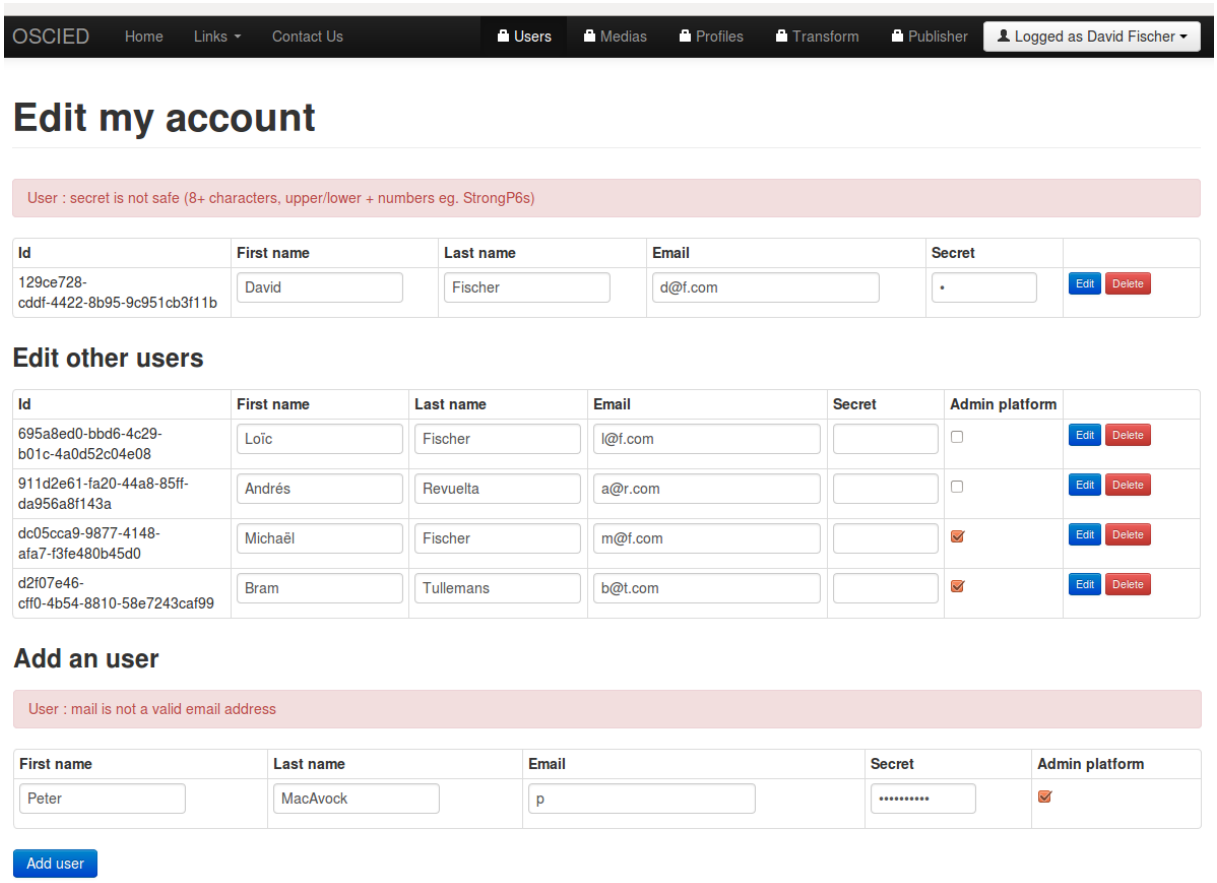


Figure 4.30: The API check validity of all inputs, e.g. weak secret + bad email format (WebUI -> Users)

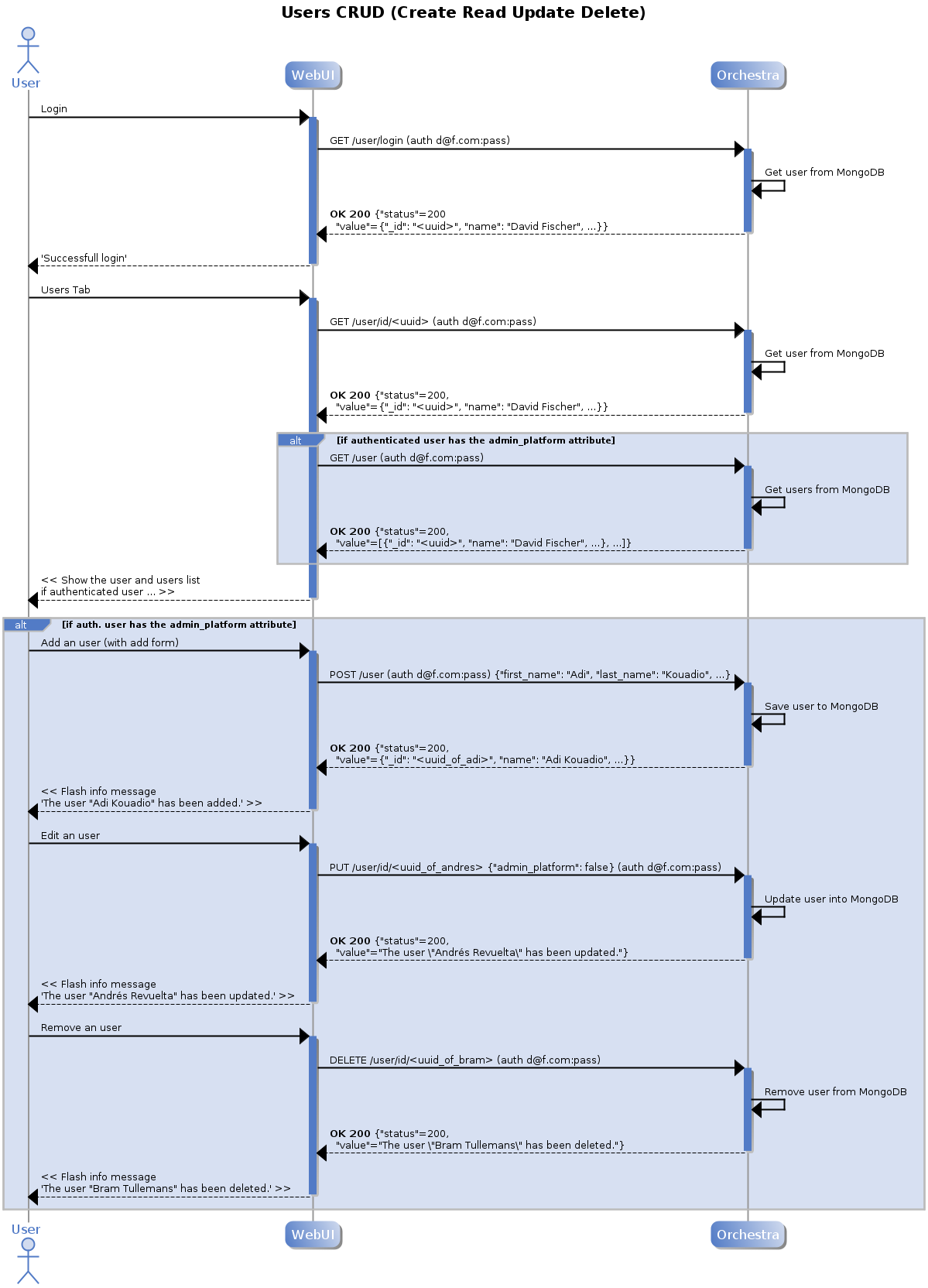


Figure 4.31: Sequence diagram of WebUI Users Tab CRUD (Create Read Update Delete)

Medias Tab

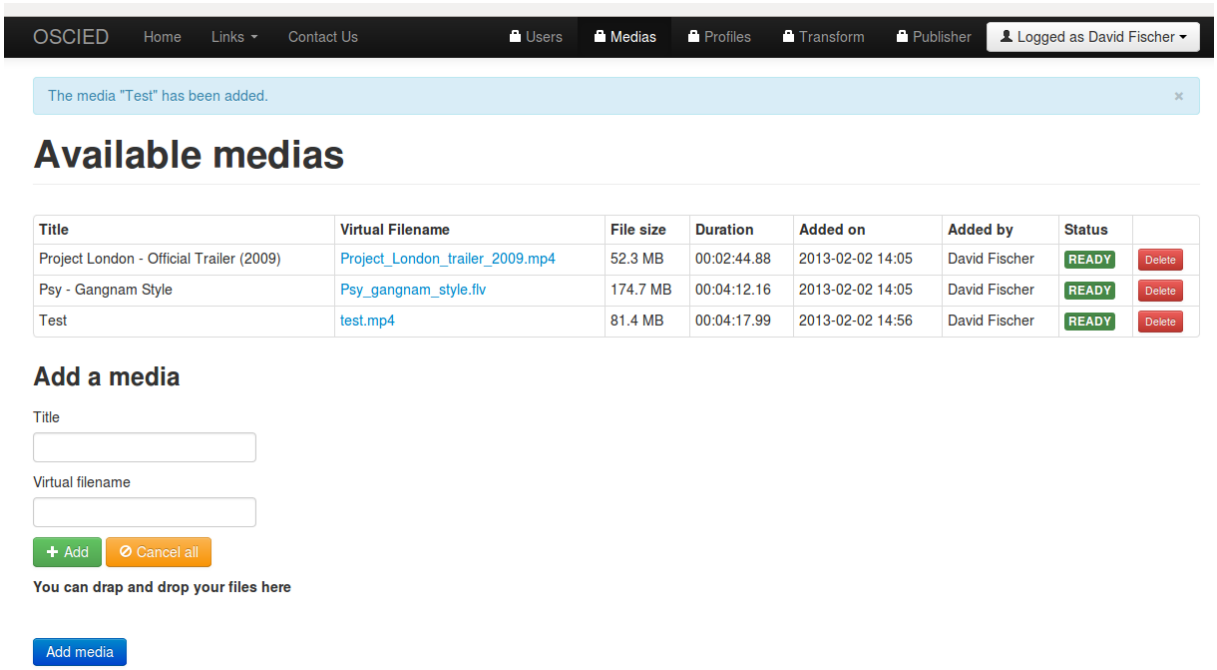


Figure 4.32: Adding a media with the medias upload form (WebUI -> Media)

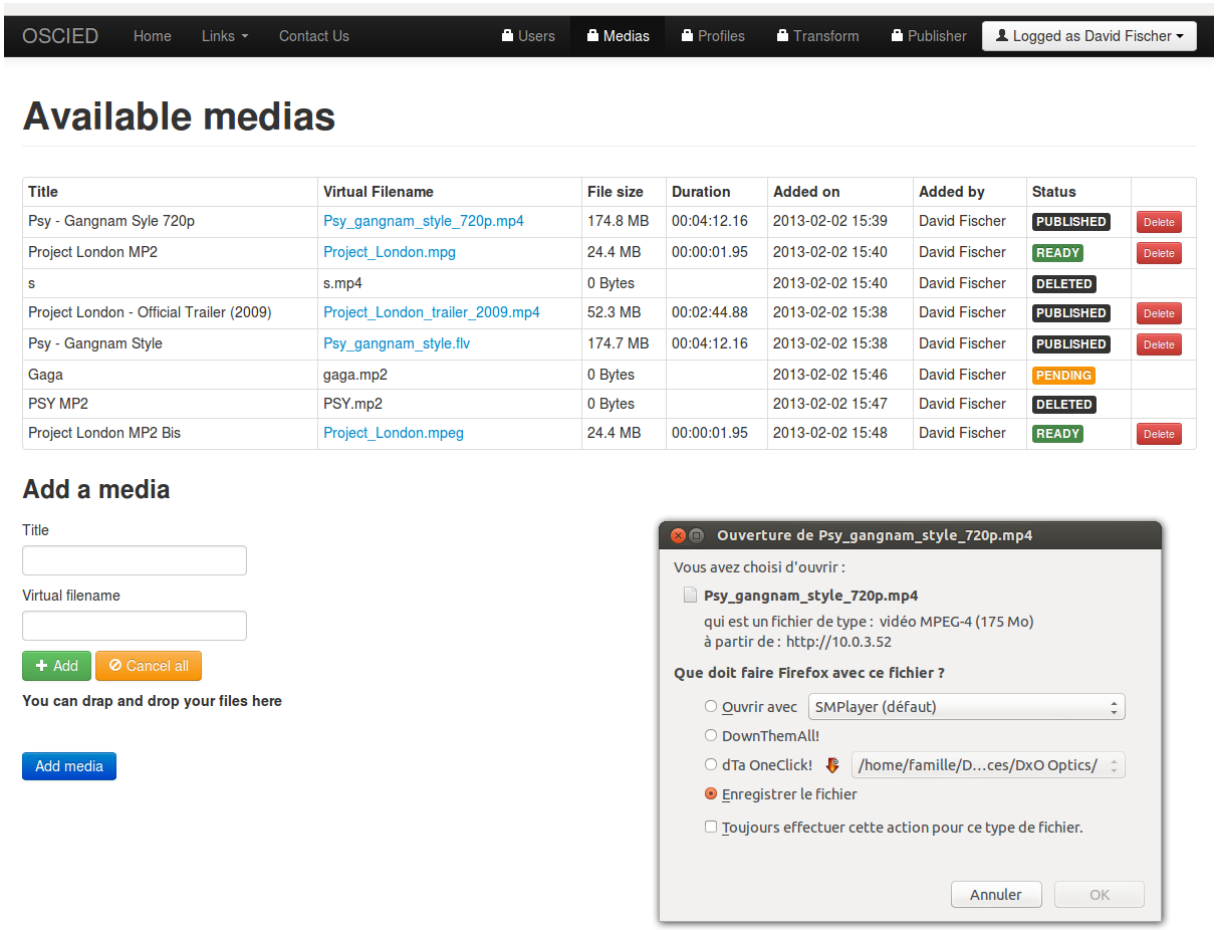


Figure 4.33: Downloading a media by clicking on the hyperlink (WebUI -> Media)

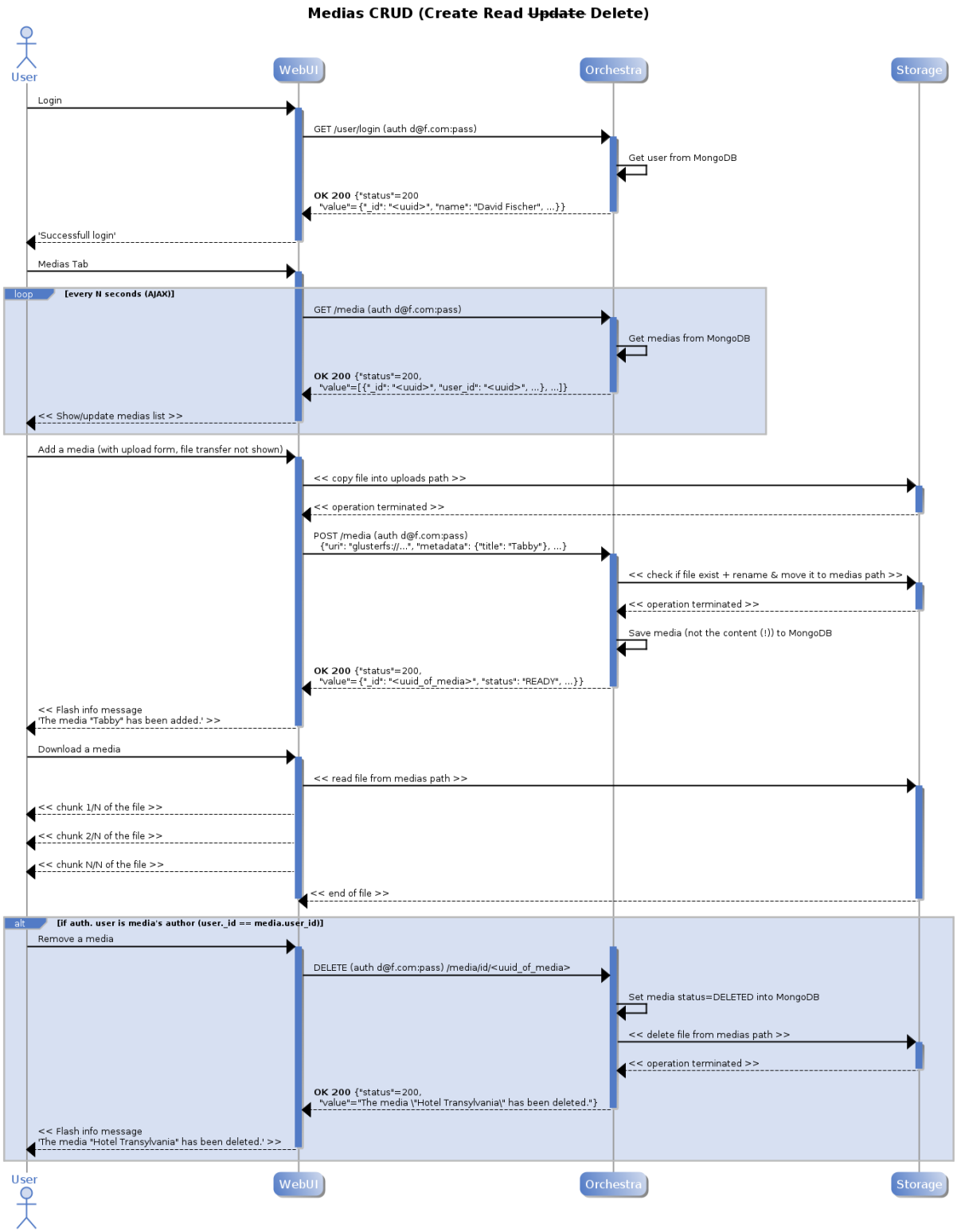


Figure 4.34: Sequence diagram WebUI Medias Tab CRUD (Create Read (update) Delete)

Transform Profiles Tab

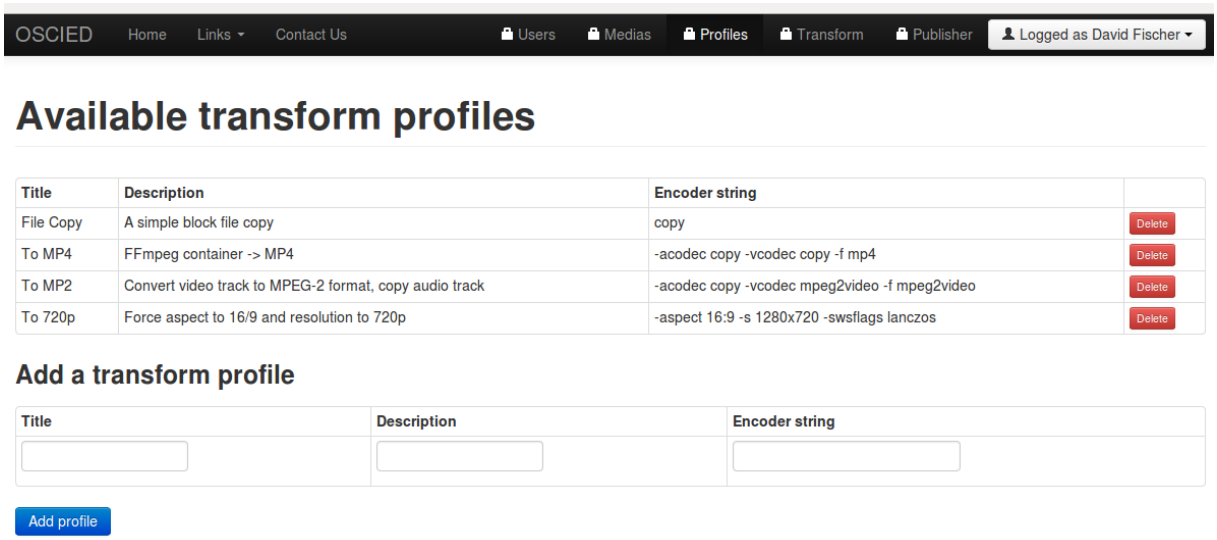


Figure 4.35: List of available profiles that transform jobs can pick from (WebUI -> Profile)

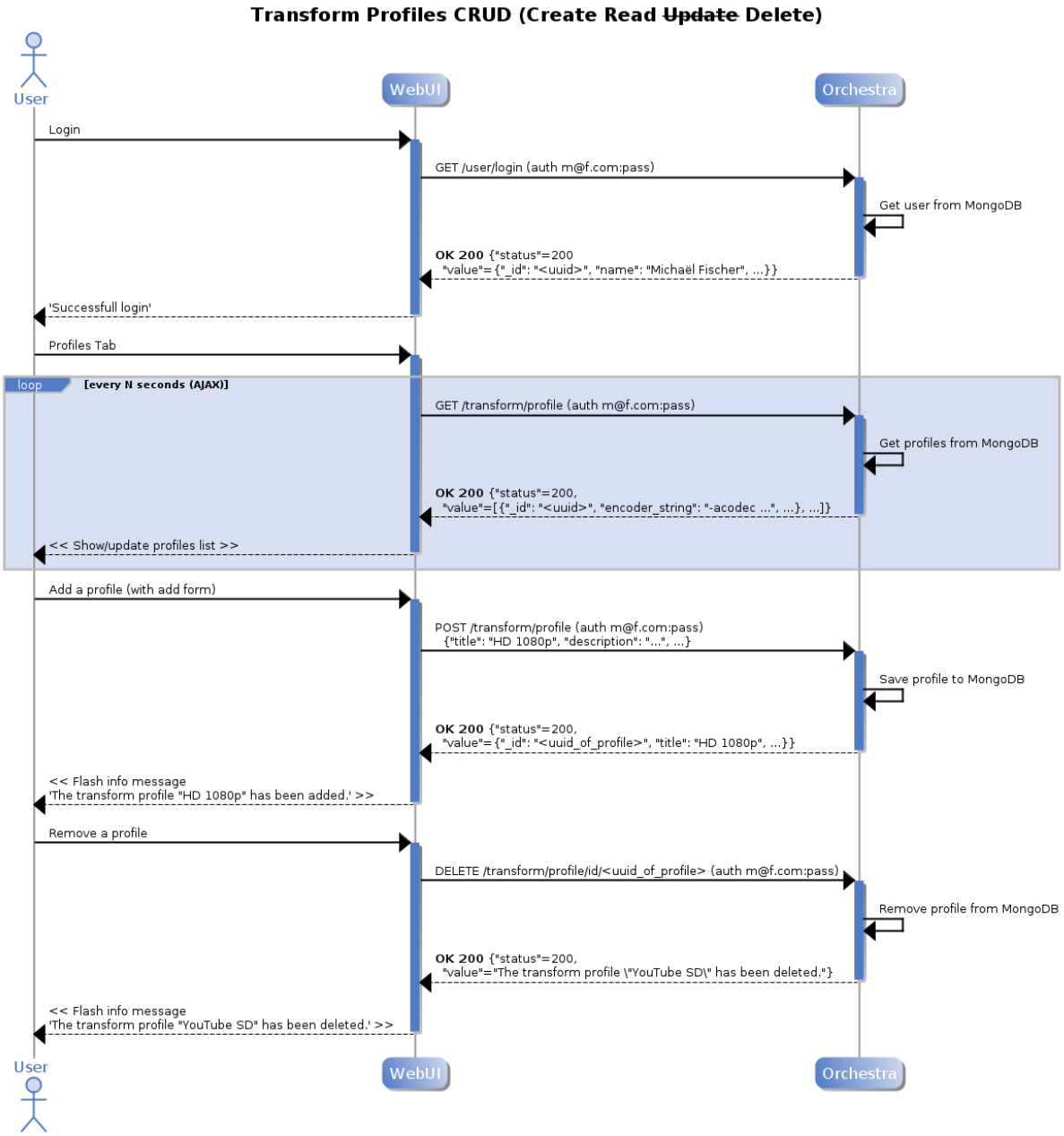


Figure 4.36: Sequence diagram of WebUI Transform Profiles Tab CRUD (Create Read (update) Delete)

Transform Jobs Tab

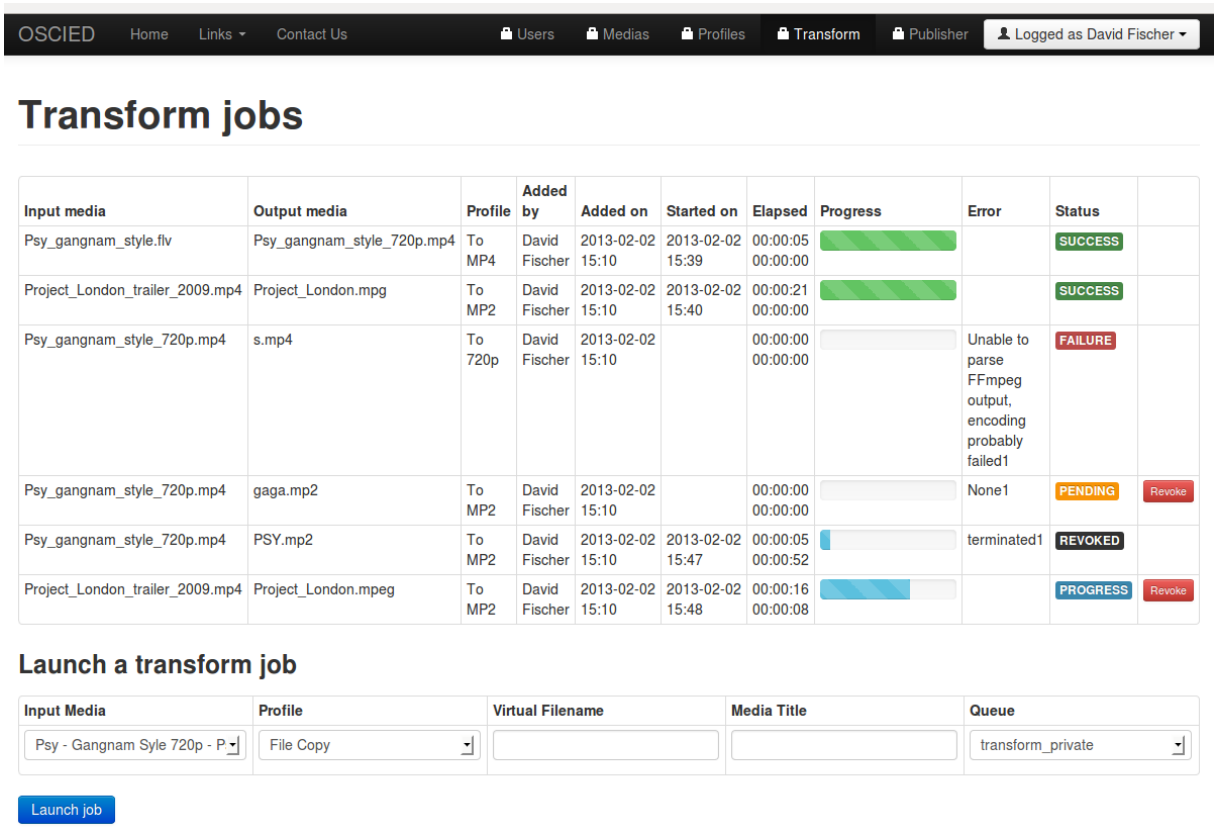


Figure 4.37: List of transform jobs (encoding) with various status (WebUI -> Transform)

Note: As you can see, here the components needs NTP time synchronization !

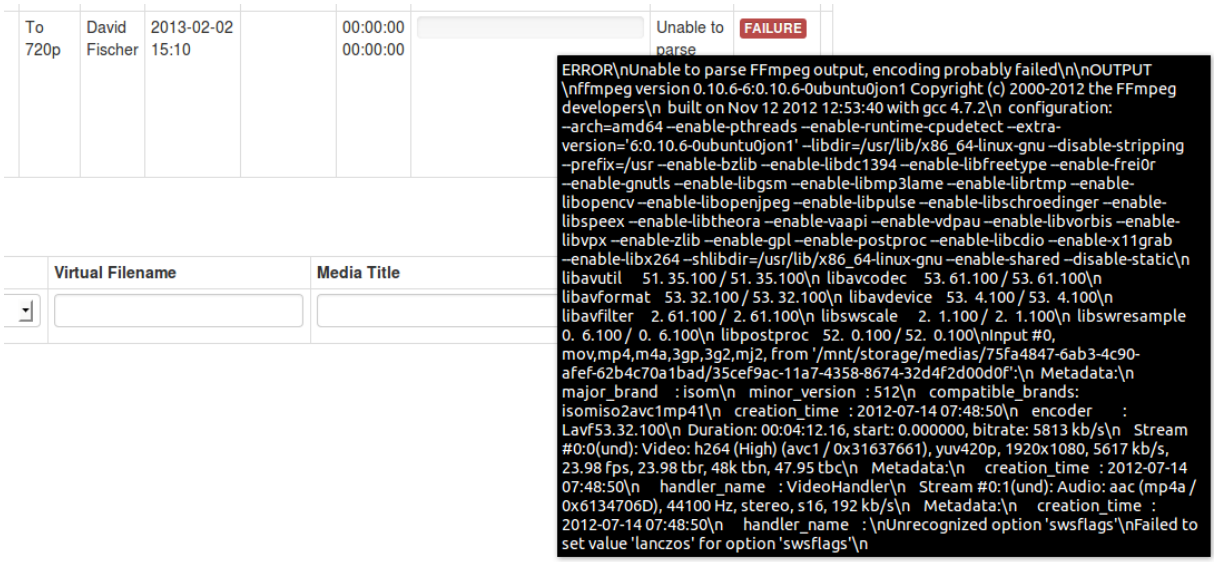


Figure 4.38: Details of the erroneous transform job (WebUI -> Transform)

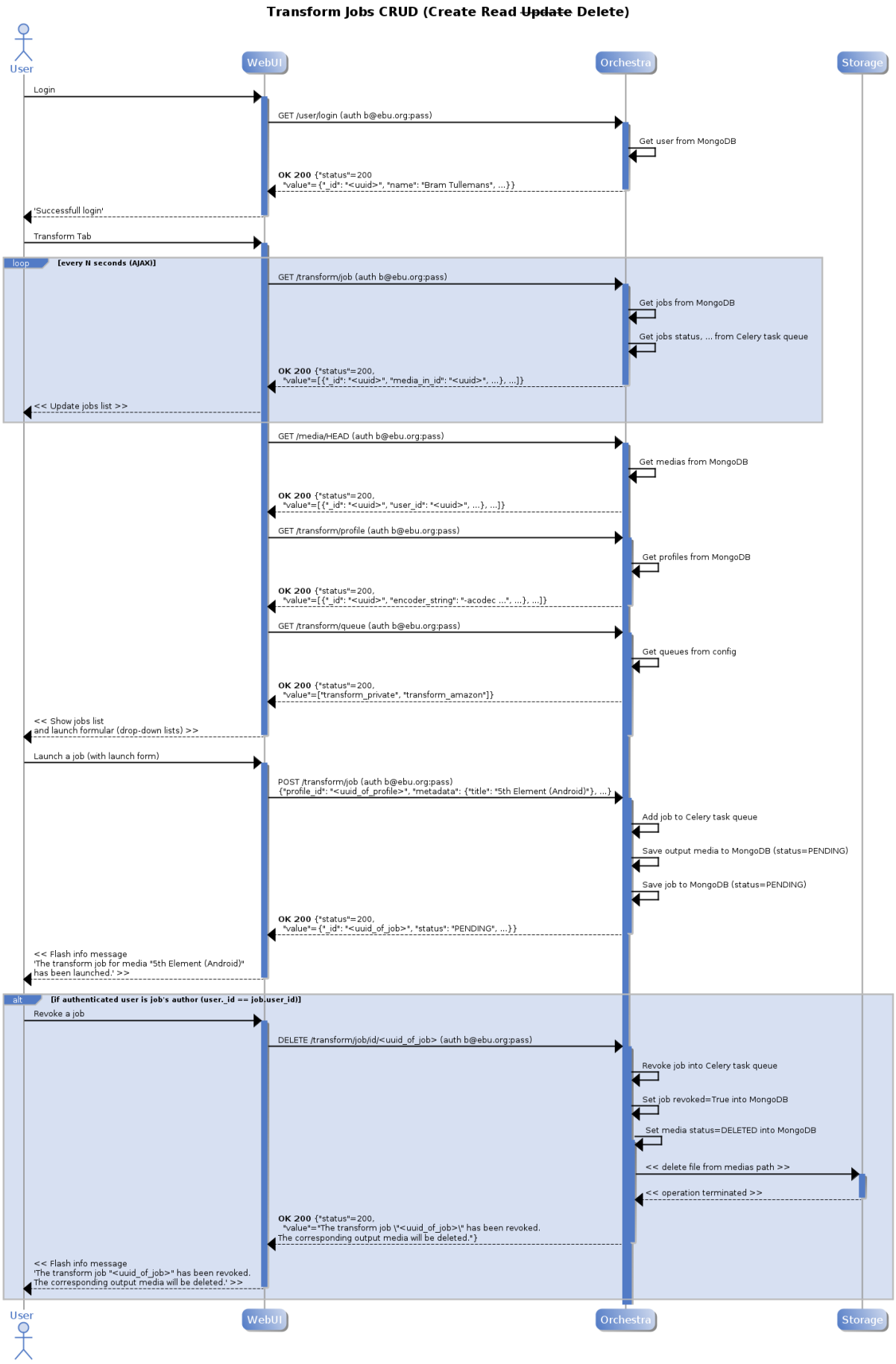


Figure 4.39: Sequence diagram of WebUI Transform Jobs Tab CRUD (Create Read (update) Delete)

Publish Jobs Tab

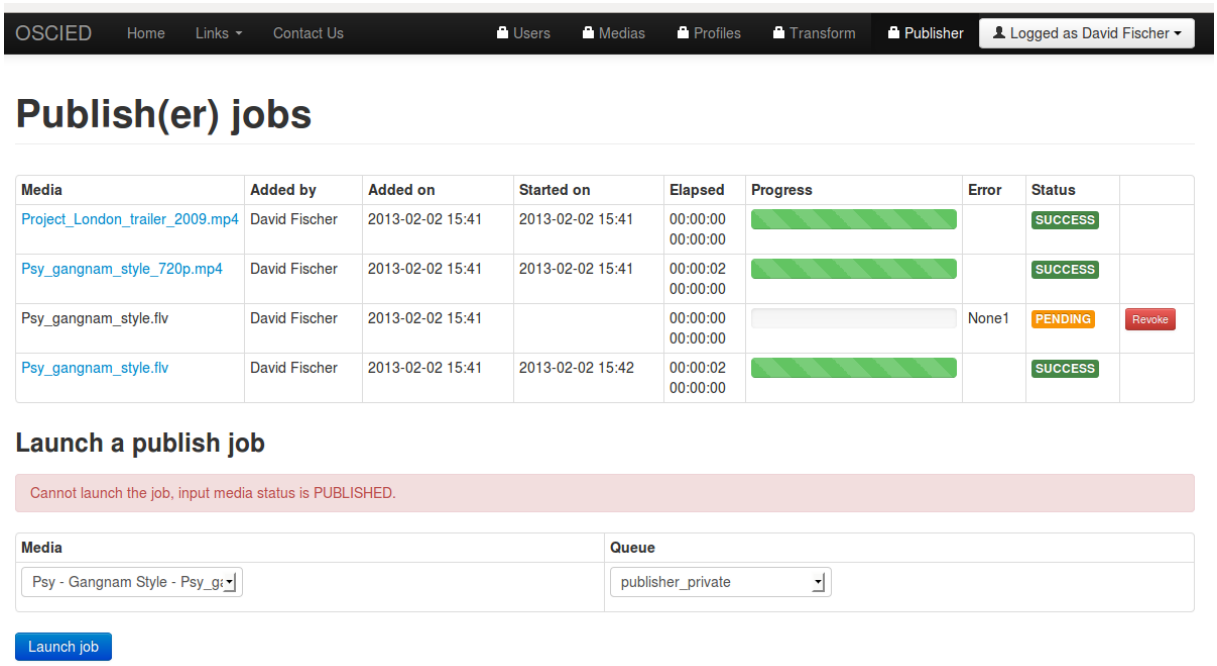


Figure 4.40: List of publish jobs (publication) with API input validity error shown (WebUI -> Publisher)

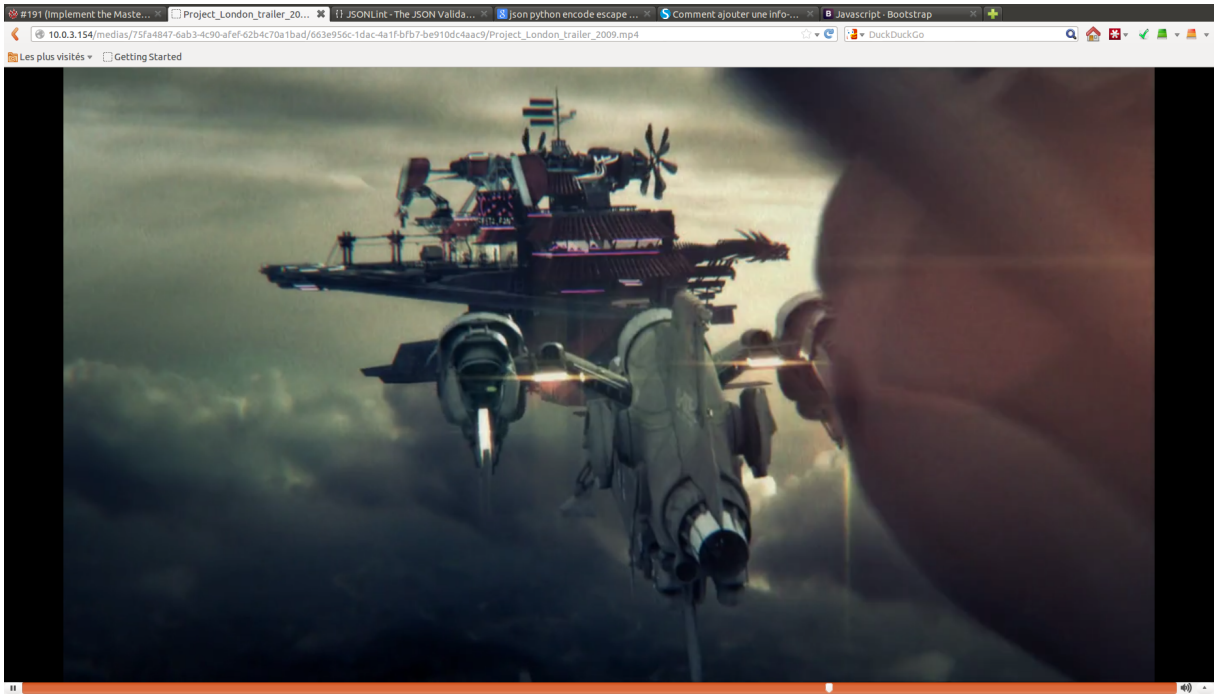


Figure 4.41: Play-out of a published media thanks to H.264 Streaming mod (WebUI -> Publisher)

Figure 4.42: Sequence diagram WebUI Publish(er) Jobs Tab CRUD (Create Read (update) Delete)

4.7 Demonstrator FAQ

Note: This documentation is intended to be read by anyone with sufficient GNU/Linux / Ubuntu skills to understand what happens when executing the example of code snippets !

4.7.1 How to get a copy of the project ?

At time of writing this documentation (7 May 2013) the project is hosted on GitHub.

So, to get a development copy of the project, you only need to open a terminal and run the following:

```
>>> ~$ cd
>>> ~$ sudo apt-get install git
>>> ~$ git clone https://github.com/EBU-TI/OSCIED
```

Then, I invite you to open a terminal and run the nice old-fashioned project's main menu and select **install**:

```
>>> ~$ cd $HOME/OSCIED/scripts/
>>> ~/OSCIED/scripts$ sh menu.sh
```

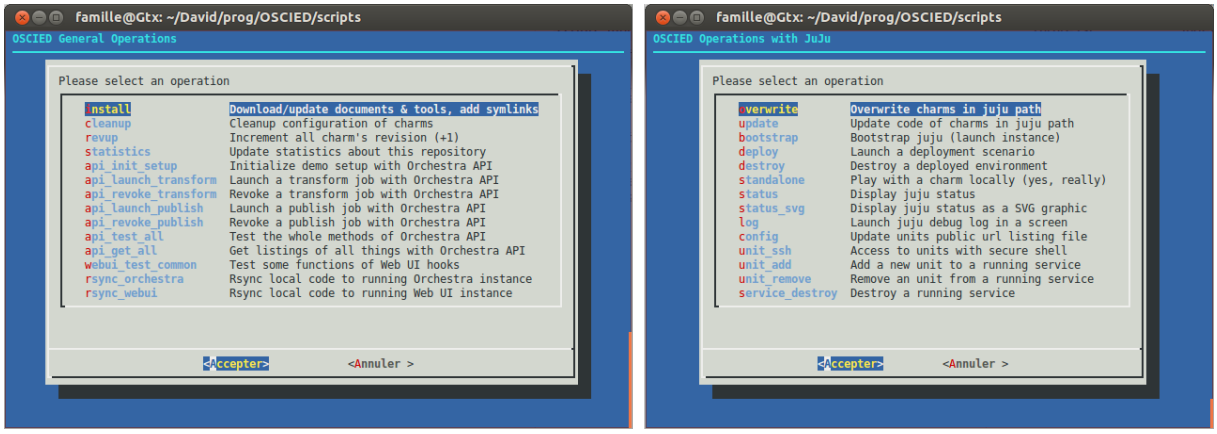


Figure 4.43: On the left : OSCIED Main Menu // On the right : OSCIED JuJu Menu

Warning: TODO update screenshots

This will install required packages, update some reference documents, populate `tools/` paths with Open-Source tools source-code (e.g. Celery's one), ...

4.7.2 What are the functionalities of project's scripts ?

I prefer to explain you the functionalities offered by the script by describing some of the typical uses cases.

So, please browse this FAQ to find the right answer to your case. If you don't find the right answer, don't hesitate to contact me *About* !

4.7.3 How to configure the demonstrator ?

It is important to understand that are four main layers involved in this project :

1. The host computing resources :

1.1) Any (desktop) computer running Ubuntu ;

1.2) A bare-metal set of servers running Ubuntu Quantal Server ;

1.3) A bare-metal set of servers running Ubuntu Quantal Server with MAAS ;

1.4) ... The cloud providers computing resources running an IaaS ;

2. The host cloud or virtualization technology :

   2.1) A private cloud running on top of (1.1, 1.2, 1.3) eg. OpenStack ;

   2.2) A public cloud running on top of (1.4), eg. Amazon AWS, HP Cloud ;

   2.3) An OS Level Virtualization technology running on top of (1.1, 1.2, 1.3), eg. LXC ;

   2.4) ... or even no virtualization at all (see *Deployment Scenarios*) ;

3. The clouds orchestrator called JuJu ;
4. The application itself called *OSCIED* ;

And all of them must be configured according to your needs.

So, here will be only introduced what to configure for 2 out of the 4 layers.

Note: In a future release JuJu will be embedded into the *Orchestra* charm to add auto-scaling features and improve easiness of deployment !

See also:

Please see Ticket 131 for quick start with Amazon AWS and Ticket 40 for further details about JuJu and MAAS.

Configuration files of JuJu

Please read *Deployment Scenarios* to gather the configuration files corresponding to the scenario you want to deploy. Then save your configuration files into `config/juju/` path of the demonstrator :

```
>>> ~$ cd $HOME/OSCIED/config/juju/
>>> ~/OSCIED/config/juju$ ls -1
environments.yaml
id_rsa
id_rsa.pub
orchestra.yaml
publisher.yaml
storage.yaml
transform.yaml
webui.yaml
```

Note: You can generate the certificate used to connect to JuJu's remote unit with `ssh-keygen -t rsa -f id_rsa`.

Configuration files of the demo

When launched, the demonstrator database is empty : The application should be initialized !

A utility script is available in order to allow you specifying the configuration into simple CSV files. You only need to fill these files with required content before launching the demonstrator itself.

Here are some example of API configuration files. Copy, paste, update and save them into `config/api/` path of the demonstrator :

```
>>> ~$ cd $HOME/OSCIED/config/
>>> ~/OSCIED/config$ ls juju -1 | grep 'yaml' | grep -v 'environments.yaml'
orchestra.yaml
publisher.yaml
storage.yaml
transform.yaml
```

```
webui.yaml
>>> ~/OSCIED/config$ ls api -1
medias.csv
tprofiles.csv
users.csv
```

api/medias.csv

```
Project London - Official Trailer [2009].mp4;Project_London_trailer_2009.mp4;Project London - Official Trailer
Psy - Gangnam Style.flv;Psy_gangnam_style.flv;Psy - Gangnam Style
big_buck_bunny_720p_h264.mov;big_buck_bunny_720p_h264.mov;Big Buck Bunny 720p H.264 MOV
big_buck_bunny_720p_h264.mp4;big_buck_bunny_720p_h264.mp4;Big Buck Bunny 720p H.264 MP4
tears_of_steel_720p.mkv;tears_of_steel_720p.mkv;Tears of Steel 720p MKV
tears_of_steel_720p.mov;tears_of_steel_720p.mov;Tears of Steel 720p MOV
tears_of_steel_720p.mp4;tears_of_steel_720p.mp4;Tears of Steel 720p MP4
Agronomie.flv;hepia_Agronomie.flv;hepia - Agronomie
Architecture.flv;hepia_Architecture.flv;hepia - Architecture
Architecture du paysage.flv;hepia_Architecture_du_paysage.flv;hepia - Architecture du paysage
Génie civil.flv;hepia_Genie_civil.flv;hepia - Génie civil
Gestion de la nature.flv;hepia_Gestion_de_la_nature.flv;hepia - Gestion de la nature
Informatique.mp4;hepia_Informatique.mp4;hepia - Informatique
ITI.mov;hepia_ITI.mov;hepia - ITI
microtechniques.flv;hepia_Microtechniques.flv;hepia - Microtechniques
Télécommunications.avi;hepia_Telecommunications.avi;hepia - Télécommunications
MusiqueChinoise.mp4;MusiqueChinoise.mp4;Sounds of Shangai (2013) by Andrés Revuelta
deploy_lxc_bug-cannonical_path.mp4;deploy_lxc_bug-cannonical_path.mp4;JuJu deploy LXC bug 'Cannonical path'
deploy_lxc_bug-memtest86+.mp4;deploy_lxc_bug-memtest86+.mp4;JuJu deploy LXC bug 'memtest86+'
init_and_jobs.mp4;init_and_jobs.mp4;David deploying OSCIED r1110 -> LXC
```

api/tprofiles.csv

```
File Copy;A simple block file copy;copy
To MP4;Change container -> MP4;-acodec copy -vcodec copy -f mp4
To MP2 (buggy);Convert video track to MPEG-2 format, copy audio track;-acodec copy -vcodec mpeg2video -sameq -
Force 720p MP4;Force aspect to 16/9 and resolution to 720p;-aspect 16:9 -s 1280x720 -sameq -f mp4
To 480p MP4;Resize (only if height > 480) to 480p -> MP4;-strict experimental -vf 'scale=trunc(oh*a/2)*2:min(4
To H.264 MP4;Transcode video track to H.264 -> MP4;-acodec copy -vcodec libx264 -crf 10 -f mp4
Deinterlace MP4;Deinterlace video track -> MP4;-strict experimental -vf 'yadif=0.-1:0, scale=trunc(iw/2)*2:tru
Speed-up 2x MP4;Speed-up video track by a factor of 2, remove audio -> MP4;-strict experimental -vf 'setpts=0.
Negate MP4;Negate video track -> MP4;-strict experimental -vf 'negate' -sameq -f mp4
```

api/users.csv

```
David;Fischer;d@f.com;oscied3D1;true
Loïc;Fischer;l@f.com;oscied3D3;false
Andrés;Revuelta;a@r.com;oscied3D4;false
Michaël;Fischer;m@f.com;oscied3D2;false
Bram;Tullemans;b@t.com;oscied3D5;true
Nabil;Abdennadher;n@a.com;oscied3D6;false
Thomas;Kernen;t@k.com;oscied3D7;false
Peter;Mac Avock;p@ma.com;oscied3D8;true
Cédric;De Carvalho;c@c.com;oscied3D9;false
Daniel;Maechler;d@m.com;oscied3D10;false
```

Finally, the input media files should be put into `medias/` path of the demonstrator. The filename must match the first column of *medias.csv* (eg. *Project London - Official Trailer [2009].mp4*).

4.7.4 How to launch the demonstrator ?

Note: Do not forget to update the configuration **before** any deployment !

In order to deploy the application (a scripted scenario), do the following:

```
>>> ~$ cd $HOME/OSCIED/scripts/
>>> ~/OSCIED/scripts$ sh juju-menu.sh
```

- Select **overwrite** to copy application's charms to the deployment path of JuJu ;
- Select **bootstrap** to bootstrap the environment (launch the JuJu's unit) ;
- Select **status** to ensure that the service is running ;
- (optional) Open a new tab to give an eye to the debug log of the deployment:

  ```
  >>> ~$ juju debug-log
  ```

- Select **deploy** then launch on of the deployment scenarios ;
- Answer to questions of the scenarios, most of the time you only need to answer y for yes ;

During the deployment, you may want to check the status, so:

- (optional) See what happens in the debug log (enter the matrix ;-)) ;
- Select **status** or even **status_svg** to get latest deployment status ;

If the deployment is unsuccessful, you can recover the situation by more than two different ways:

- Destroying the whole environment with **destroy** ;
- Destroying the faulty service (e.g. *oscied-webui*) with **service_destroy** ;
- Fix the charm's source code and upgrade charms ... NOT YET IMPLEMENTED ;

Then, you need to fix the charm's source code (see *How to update the code ?*) ... and retry your deployment:

- ... Do all steps if you destroyed the whole environment ;
- ... Only **update**, **deploy** and choose to launch only the faulty service again ;

Note: You can check status faster by filtering the output of JuJu's status:

```
>>> ~$ juju status | grep error
(nothing)
```

When all services are hopefully up and running you can setup the Orchestrator by doing the following:

- Select **config** to update generated configuration files used by scripts ;

Note: The actual Orchestrator can't understand that the storage's public/private IP are linked to the same host. This is a known limitation that will be fixed in a future release. It means that the private IP of the storage unit is hardcoded in `common.sh`. The **config** function update it automatically.

The function will connect to the orchestrator, grep the configuration file `/var/lib/.../charm/config.json` and extract *storage_ip* value. Last but not least, `common.sh` is then updated with auto-detected value.

- Start another script ;-)

  ```
  >>> ~$ cd $HOME/OSCIED/scripts/
  >>> ~/OSCIED/scripts$ sh menu.sh
  ```

- Select **api_init_setup** this will (most of the time by calling the Orchestra's RESTful API) :
  - Flush Orchestrator's database ;
  - Add the users of `config/api/users.csv` ;
  - **Send and add the medias of `config/api/medias.csv` :**
    - Send the media file from `medias/` to storage's unit home path [21] ;
    - Copy the media from storage's home path to storage's upload path [22] ;

[21] I known that is not realistic, HTTP media injest is currently not implemented.
[22] It is done this way to avoid sending medias files every time the Orchestrator is (re)initialized !

        * Add the media ;
    - Add the transform profiles of `config/api/tprofiles.csv` ;

Voila, your demonstrator is ready !

4.7.5 How to scale up/down a service of the demonstrator ?

Note: Do not forget to update the configuration **before** any deployment !

In order to add/remove an unit to/from a service do the following:

```
>>> ~$ cd $HOME/OSCIED/scripts/
>>> ~/OSCIED/scripts$ sh juju-menu.sh
```

- Select **unit_add** to add an unit to any service able to scale up/down ;
- Select **unit_remove** to remove an unit from any service able to scale up/down ;

Warning: At time of writing this report, only transform and publisher services can scale up/down !

Note: In some of the deployment scenarios, you may need to bypass this helper script to directly uses juju in order to specify the environment.

4.7.6 How to stop the demonstrator ?

Note: Do not forget to update the configuration **before** any deployment !

In order to stop the application do the following:

```
>>> ~$ cd $HOME/OSCIED/scripts/
>>> ~/OSCIED/scripts$ sh juju-menu.sh
```

- Select **destroy** to destroy the whole environment (files in *storage* charm are **lost**) ;

4.7.7 How to update the code ?

- Modify the source-code of charms (e.g. for *oscied-orchestra*) :

```
>>> ~$ cd $HOME/OSCIED/charms/oscied-orchestra/
>>> ~/OSCIED/charms/oscied-orchestra/$ nano orchestra.py
>>> ~/OSCIED/charms/oscied-orchestra/$ nano hooks_lib/common.sh.lu-dep
>>> ~/OSCIED/charms/oscied-orchestra/$ lu-importUtils . no
>>> ~/OSCIED/charms/oscied-orchestra/$ echo $(($(cat revision)+1)) > revision
```

Note: If you'll have a lot of components to update, you may find useful to skip the revision's increment step and do the following:

```
>>> ~$ cd $HOME/OSCIED/scripts/
>>> ~/OSCIED/scripts/$ sh menu.sh
```

- Select **revup** to increment all charm's revision otherwise JuJu will not deploy latest version of the code !

4.7.8 How to efficiently add features ?

Note: You need at least a running deployment to test the real behavior of components as described here *How to launch the demonstrator ?*.

At the early stage of the development, it was easy to test the behavior of the Orchestrator without too much effort. In fact, it was easy because on previous versions it wasn't necessary to deploy the shared storage nor the transform/publisher units.

Now, this is not as easy as before as they are really nice features to test !

Here is explained the steps I followed during the developments (be warned, they are a lot):

1. Deploy the demonstrator on a local environment for speed or on a real cloud, as you prefer ;
2. The Orchestrator
   - Edit the source code :

     ```
     >>> ~$ cd $HOME/OSCIED/charms/oscied-orchestra
     >>> ~/OSCIED/charms/oscied-orchestra$ nano ...
     ...
     ```

   - Update the running unit :

     ```
     >>> ~$ cd $HOME/OSCIED/scripts
     >>> ~/OSCIED/scripts$ sh menu.sh
     ```

     - Select **rsync_orchestra** to update running unit source code ;

   - Commit any relevant modification to code :

     ```
     >>> ~$ ~/OSCIED/charms/oscied-orchestra$ svn st
     M orchestra.py
     M lib/Orchestra.py
     >>> ~/OSCIED/charms/oscied-orchestra$ svn commit -m 'I implemented something cool'
     ```

3. The Transform
   - Add subversion to the running unit :

     ```
     >>> ~$ ssh oscied-transform/0
     >>> (transform) ~$ sudo su
     >>> (transform) ~# ln -s /var/lib/juju/units/oscied-transform-0/charm
     >>> (transform) ~# ln -s /var/lib/juju/units/oscied-transform-0/charm.log
     >>> (transform) ~# apt-get install subversion
     >>> (transform) ~# cd charm
     >>> (transform) ~/charm# svn co https://claire-et-david.dyndns.org/prog/OSCIED/
                                                          charms/oscied-transform/
     >>> (transform) ~/charm# mv charm/.svn .
     >>> (transform) ~/charm# rm -rf charm
     >>> (transform) ~/charm# svn st
     ? celeryconfig.py
     ```

   - Attach to screen and edit code :

     ```
     >>> (transform) ~/charm# screen -r
     (you will see celeryd output 'log')
     (CTRL+A C to create a new tab into screen)
     >>> (transform) ~/charm# nano lib/Transform.py
     (CTRL+A N to see celeryd output 'log')
     (CTRL+C to stop celeryd)
     >>> (transform) ~/charm# celeryd -Q transform_private
     ...
     ```

   - Commit any relevant modification to code :

     ```
     >>> (transform) ~/charm# svn st
     ? celeryconfig.py
     M lib/Transform.py
     M lib/Medias.py
     >>> (transform) ~/charm# svn commit lib/ -m 'I implemented something cool'
     ```

4. The Web User Interface
   - Edit the source code :

```
>>> ~$ cd $HOME/OSCIED/charms/oscied-webui/www
>>> ~/OSCIED/charms/oscied-webui/www$ nano ...
...
```

- Update the running unit :

```
>>> ~$ cd $HOME/OSCIED/scripts
>>> ~/OSCIED/scripts$ sh menu.sh
```

  - Select **rsync_webui** to update running unit source code ;

- Commit any relevant modification to code :

```
>>> ~$ ~/OSCIED/charms/oscied-webui/www$ svn st
M application/controllers/medias.php
>>> ~/OSCIED/charms/oscied-webui/www$ svn commit -m 'I implemented something cool'
```

5. Add automated test calls to Orchestra's RESTful API in order to *unit test* the Orchestrator :

   - Edit the script :

```
>>> ~$ cd $HOME/OSCIED/scripts
>>> ~/OSCIED/scripts$ nano menu.sh
... (update api_test_all method) ...
```

   - Launch the test :

```
>>> ~/OSCIED/scripts$ sh menu.sh
```

     - Select **api_test_all** to test features of the API ;

     Warning: This method will flush the Orchestrator's database !

6. Update OSCIED Project's TRAC Environment by creating / solving tickets and update **this documentation** too !

4.7.9 How to update documentation ?

Has you already noticed, in the `report/david/MA/` path of the project sit a lot of things and you will find in `source/` path some .rst files. If you wish to contribute to this the project documentation I must introduce you with reStructuredText.

reStructuredText is an easy-to-read plaintext markup syntax parseable by specific tools to produce documentation in a wide variety of formats. This markup syntax is really useful for developers to add *smart* in-line documentation to their code (such as Python docstrings).

For this project I wrote my documentation (my report) in .rst files and I used the powerful Sphinx parser to create a makefile for the documentation to be generated !

As always, a bash script is also provided to generate the documentation *label_generate_report*, so you only need to run it to produce corresponding html + pdf [23]:

```
>>> ~/OSCIED$ cd scripts/
>>> ~/OSCIED/scripts$ sh generate-report.sh
```

This script will install required packages, cleanup output paths and generate the documentation into `OSCIED/report/david/MA`

See also:

reStructuredText Rocks ! Please see Ticket 141 to be convinced.

[23] Be aware the warnings & errors of the command-line, this is a compilation process, it may fail !

4.8 Deployment Scenarios

4.8.1 JuJu++

At time of writing this report, one cannot uses JuJu to :

1. Deploy charms locally without using the providers such as MAAS or LXC.
2. Add relations between services running on different environments.

... Challenge accepted, let's make it possible !

First Challenge

I partially hacked this by bypassing JuJu and creating my own set of tools to install charms locally, here are the key elements of this hack :

Problem

Charms hooks requires JuJu's callable methods like `juju-log`, `config-get`, `unit-get`, ...

Solution

Home-made implementation of the missing methods :

- **juju-log** is mapped to a bash script with colorful echoes
- ***-get** are mapped to a bash script having as input the options *name* and echoing corresponding *value* based on a predefined *name* = *value* file

Problem

Charms hooks must be called in order to deploy the application.

Solution

A simple command-line dialog menu allowing the user to call charms hooks directly. This utility copy the home-made implementation of the missing JuJu's methods to local binaries path `/usr/local/bin/` before calling the hook.

See also:

Hacks are available here : *label_juju_plus_plus*.

Second Challenge

Problem

To connect services together (application's charms instances) one need to use `juju add-relation` however JuJu only allow you to connect services from the same environment.

Solution

In order to allow such kind of complex deployment the following have been added into charm's configuration file `config.yaml` :

- *storage* related options for charms requiring the *storage* relation.
- *publisher* related options for *Publisher* charm requiring the *publisher* relation.
- *transform* related options for *Transform* charm requiring the *transform* relation.

It may sound as redundant at first sight however it unlock the elasticity of the application !

For example, you only need to specify the *storage* related options into `orchestra.yaml` and then deploy the *Orchestra* charm with `juju deploy --config orchestra.yaml (...) oscied-orchestra` to make the orchestrator instance using the storage you specified. This will also disable the orchestrator instance's hooks related to *storage* relation.

4.8.2 Local Deployment

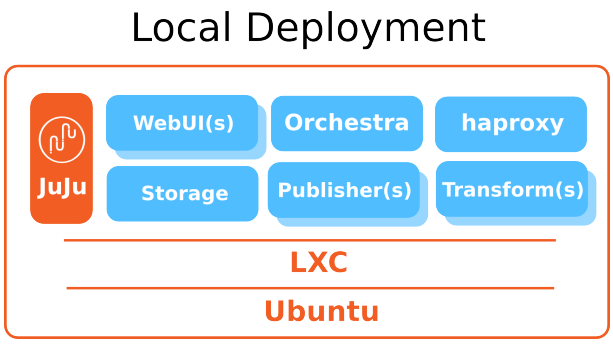


Figure 4.44: Local deployment is really useful for development & testing purposes

See also:

Please see official documentation for JuJu Local provider.

environments.yaml

```
default: local
environments:
  local:
    type: local
    control-bucket: juju-a14dfae3830142d9ac23c499395c2785999
    admin-secret: 6608267bbd6b447b8c90934167b2a294999
    data-dir: /home/<username>/.juju/storage
    default-series: quantal
    juju-origin: ppa
```

Procedure

```
juju bootstrap
juju deploy --repository=charms/ local:quantal/oscied-orchestra
juju deploy --repository=charms/ local:quantal/oscied-webui
juju deploy --repository=charms/ local:quantal/oscied-storage
juju deploy --repository=charms/ local:quantal/oscied-transform -n 3
juju deploy --repository=charms/ local:quantal/oscied-publisher -n 2
juju deploy cs:precise/haproxy
juju expose oscied-storage
juju expose oscied-orchestra
juju expose oscied-publisher
juju expose haproxy
juju add-relation oscied-storage             oscied-transform
juju add-relation oscied-storage             oscied-publisher
juju add-relation oscied-storage             oscied-orchestra
juju add-relation oscied-storage             oscied-webui
juju add-relation oscied-orchestra:transform oscied-transform:transform
juju add-relation oscied-orchestra:publisher oscied-publisher:publisher
juju add-relation oscied-orchestra:api       oscied-webui:api
juju add-relation haproxy                    oscied-webui
```

4.8.3 Cloud Deployment

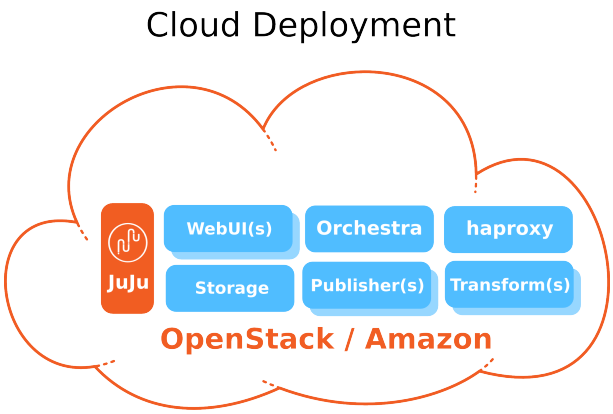


Figure 4.45: Cloud deployment is really interesting for his elasticity

See also:

Please see official documentation for JuJu OpenStack, Amazon AWS, HP Cloud, Rackspace providers.

environments.yaml

```
default: amazon
environments:
  amazon:
    type: ec2
    access-key: AKI*****************
    secret-key: Vl5g/QLi0*******************************
    control-bucket: juju-24etR***********.s3-website-us-east-1.amazonaws.com
    admin-secret: 81a1e7429e6847*****************
    default-series: quantal
    juju-origin: ppa
```

Procedure

```
tmicro='instance-type=t1.micro'
mmedium='instance-type=m1.medium'
cmedium='instance-type=c1.medium'
juju bootstrap
juju deploy --constraints "$tmicro"  --repository=charms/ local:quantal/oscied-orchestra
juju deploy --constraints "$tmicro"  --repository=charms/ local:quantal/oscied-webui
juju deploy --constraints "$mmedium" --repository=charms/ local:quantal/oscied-storage
juju deploy --constraints "$cmedium" --repository=charms/ local:quantal/oscied-transform -n 3
juju deploy --constraints "$mmedium" --repository=charms/ local:quantal/oscied-publisher -n 2
juju expose oscied-storage
juju expose oscied-orchestra
juju expose oscied-publisher
juju expose oscied-webui
juju add-relation oscied-storage             oscied-transform
juju add-relation oscied-storage             oscied-publisher
juju add-relation oscied-storage             oscied-orchestra
juju add-relation oscied-storage             oscied-webui
juju add-relation oscied-orchestra:transform oscied-transform:transform
juju add-relation oscied-orchestra:publisher oscied-publisher:publisher
juju add-relation oscied-orchestra:api       oscied-webui:api
```

4.8.4 Multi-Environment Deployment

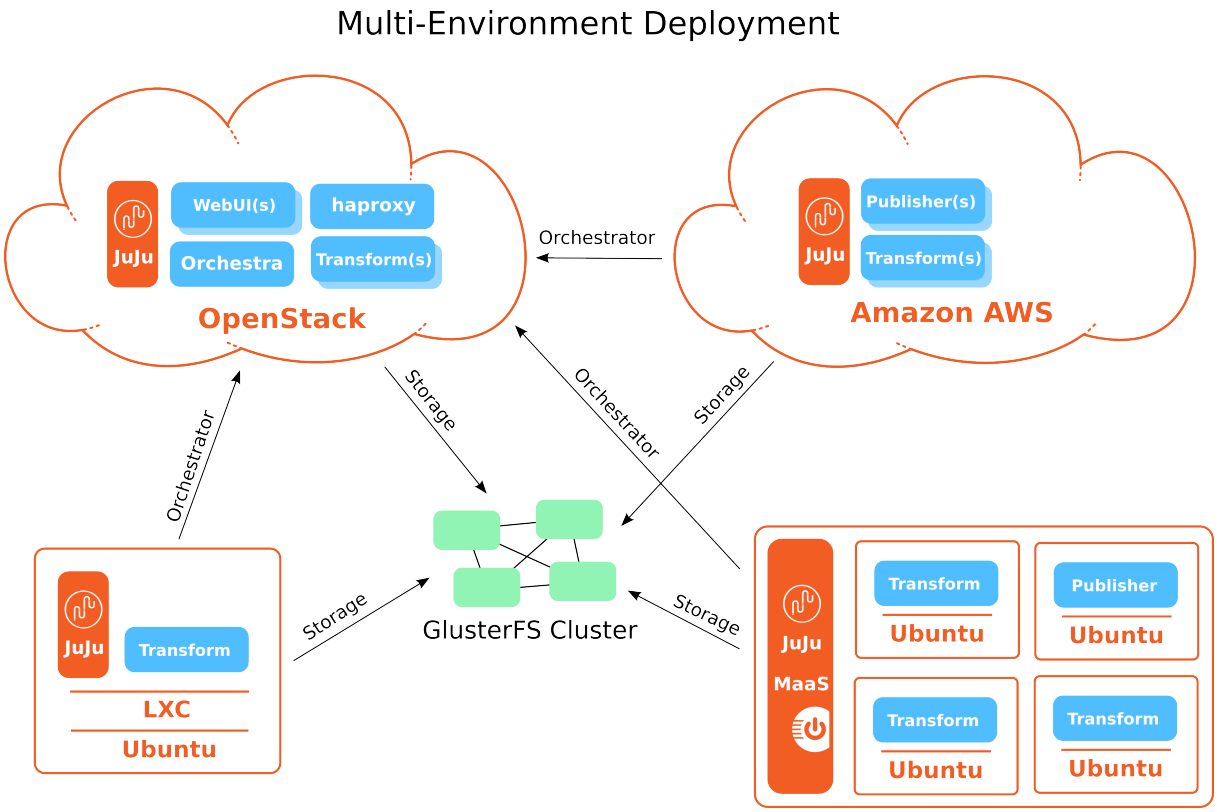


Figure 4.46: The application components can be deployed in parallel to any compatible environment !

See also:

Please see official documentation for JuJu Local, MAAS, OpenStack, Amazon AWS, HP Cloud, Rackspace providers.

environments.yaml

```
default: amazon
environments:
  local:
    type: local
    control-bucket: juju-a14dfae3830142d9ac23c499395c2785999
    admin-secret: 6608267bbd6b447b8c90934167b2a294999
    data-dir: /home/<username>/.juju/storage
    default-series: quantal
    juju-origin: ppa
  maas:
    type: maas
    maas-server: 'http://<maas_host>:5240'
    maas-oauth: '${maas-api-key}'
    admin-secret: 'nothing'
    juju-origin: ppa
  openstack:
    type: openstack_s3
    control-bucket: juju-05n2105318671zvmr9388d6db2725871636
    admin-secret: 6341958569854581158 4r7691832885p714
    auth-url: https://<yourkeystoneurl>:443/v2.0/
    default-series: quantal
    juju-origin: ppa
    ssl-hostname-verification: True
    default-image-id: bb636e4f-79d7-4d6b-b13b-c7d53419fd5a
```

```
    default-instance-type: m1.small
  amazon:
    type: ec2
    access-key: AKI*****************
    secret-key: Vl5g/QLi0*******************************
    control-bucket: juju-24etR***********.s3-website-us-east-1.amazonaws.com
    admin-secret: 81a1e7429e6847******************
    default-series: quantal
    juju-origin: ppa
```

publisher-amazon.yaml

```
oscied-publisher:
  verbose: "true"
  concurrency: 4
  rabbit_queues: "publisher_amazon"
  max_upload_size: 4294967296
  max_execution_time: 180
  max_input_time: 600
  mongo_connection: "mongodb://nodes:M2qlif8rdtKtBYil@<orchestra_ip>:27017/celery"
  rabbit_connection: "amqp://nodes:OZy23iO0D4UpYS2k@<orchestra_ip>:5672/celery"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

publisher-maas.yaml

```
oscied-publisher:
  verbose: "true"
  concurrency: 6
  rabbit_queues: "publisher_maas"
  max_upload_size: 4294967296
  max_execution_time: 180
  max_input_time: 600
  mongo_connection: "mongodb://nodes:M2qlif8rdtKtBYil@<orchestra_ip>:27017/celery"
  rabbit_connection: "amqp://nodes:OZy23iO0D4UpYS2k@<orchestra_ip>:5672/celery"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

orchestra-openstack.yaml

```
oscied-orchestra:
  verbose: "true"
  root_secret: "bXzZ6SFmh0Z5a8PQ"
  nodes_secret: "Y3v8rXTyPjTAQgI0"
  repositories_user: ""
  repositories_pass: ""
  webui_repository: ""
  transform_repository: ""
  publisher_repository: ""
  mongo_admin_password: "Iz85QVjdCJugEosz"
  mongo_nodes_password: "M2qlif8rdtKtBYil"
  rabbit_password: "IwmZk3F3suCH8rvC"
  juju_environment: ""
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

transform-local.yaml

```
oscied-transform:
  verbose: "true"
  concurrency: 2
  rabbit_queues: "transform_local"
  mongo_connection: "mongodb://nodes:M2qlif8rdtKtBYil@<orchestra_ip>:27017/celery"
  rabbit_connection: "amqp://nodes:OZy23iO0D4UpYS2k@<orchestra_ip>:5672/celery"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

transform-maas.yaml

```
oscied-transform:
  verbose: "true"
  concurrency: 12
  rabbit_queues: "transform_maas,transform_openstack"
  mongo_connection: "mongodb://nodes:M2qlif8rdtKtBYil@<orchestra_ip>:27017/celery"
  rabbit_connection: "amqp://nodes:OZy23iO0D4UpYS2k@<orchestra_ip>:5672/celery"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

transform-openstack.yaml

```
oscied-transform:
  verbose: "true"
  concurrency: 2
  rabbit_queues: "transform_openstack"
  mongo_connection: ""
  rabbit_connection: ""
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

transform-amazon.yaml

```
oscied-transform:
  verbose: "true"
  concurrency: 2
  rabbit_queues: "transform_amazon"
  mongo_connection: "mongodb://nodes:M2qlif8rdtKtBYil@<orchestra_ip>:27017/celery"
  rabbit_connection: "amqp://nodes:OZy23iO0D4UpYS2k@<orchestra_ip>:5672/celery"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

webui-openstack.yaml

```
oscied-webui:
  verbose: "true"
  max_upload_size: 4294967296
  max_execution_time: 180
  max_input_time: 600
  mysql_my_password: "mUzf4JUwTIa3AIXj"
  mysql_root_password: "mUzf4JUwTIa3AIXj"
  mysql_user_password: "SD1MwuxjMzck2ZCs"
  storage_ip: "<storage_ip>"
  storage_fstype: "glusterfs"
  storage_mountpoint: "medias_volume"
  storage_options: ""
```

Procedure

```
tmicro='instance-type=t1.micro'
mmedium='instance-type=m1.medium'
cmedium='instance-type=c1.medium'
repo='--repository=charms/'
path='local:quantal/oscied'
c='--constraints'
cfg='--config'
stack='openstack'

juju bootstrap -e openstack
juju deploy -e $stack $c "$tmicro"  $cfg orchestra-openstack.yaml $repo $oscied-orchestra
juju deploy -e $stack $c "$tmicro"  $cfg webui-openstack.yaml     $repo $oscied-webui
juju deploy -e $stack $c "$cmedium" $cfg transform-openstack.yaml $repo $oscied-transform -n 2
juju expose -e $stack oscied-orchestra
juju expose -e $stack oscied-webui
juju add-relation -e $stack oscied-orchestra:transform oscied-transform:transform
juju add-relation -e $stack oscied-orchestra:api       oscied-webui:api

juju bootstrap -e amazon
juju deploy -e amazon $c "$cmedium" $cfg transform-amazon.yaml $repo $oscied-transform -n 2
juju deploy -e amazon $c "$mmedium" $cfg publisher-amazon.yaml $repo $oscied-publisher -n 2
juju expose -e amazon oscied-publisher

juju bootstrap -e maas
juju deploy -e maas $cfg transform-maas.yaml $repo $oscied-transform -n 3
juju deploy -e maas $cfg publisher-maas.yaml $repo $oscied-publisher -n 1
juju expose -e maas oscied-publisher

juju bootstrap -e local
juju deploy -e local $cfg transform-local.yaml $repo $oscied-transform
```

4.9 Tests and Results

4.9.1 Foreword

You may have noticed they are missing *numbers* here. The reason is the only conclusions one would gather from *performance benchmarks* would be the following :

- Actual storage charm cannot scale, this is the bottleneck of any deployment using it
- Celery does work as expected, workers keep connection to queues and jobs are handled rapidly
- Encoding : Sure, a Pentium IV is slower than a i7
- ...

This is out of the scope of the project.

4.9.2 Tests of Orchestra API

The Orchestrator's API is my first RESTful API !

So I decided to create various scripts using cURL in to help me testing the functionalities of the API to test various inputs and corresponding outputs. Outputs of the API were also validated with a json string validator called JSONLint. The scripts are also useful to ensure security policies by testing responses (code + value) to scripted requests. One can call that unit testing ... OK it is.

4.9.3 Charms are Reliable

The application's components are charms, charms means hooks and hooks means code.

At the beginning of the charm's developments, it was an intensive period of deployment, debugging ... The charm's hooks where developed with the following algorithm :

```
counter = 0
while time < 03:00 AM:
    juju deploy charms
    juju debug-log
    if any error:
        reason = decipher (error)
        solution = find solution (reason)
    elif sufficient_features:
        say "Ouais !"
        break
    juju destroy
    counter++
    if counter mod 10 == 0:
        drink (water)
    foreach charm of the application:
        hooks += feature
    foreach faulty charm of the application:
        hooks += solution
sleep (8 hours)
```

The charms are now robust as they are developed, tested and debugged. ... And also successfully deployed for weeks on Amazon AWS and LXC.

4.9.4 Tested Deployment Scenarios

What was the procedure ?

- Deployment of the application's charms to the chosen environment
- Live checking of the juju's debug-log (are there any error ?)

- Remote access to unit's in order to update code in case of error
- Scaling of transform & publisher units by adding/removing units
- Usage of the orchestrator's API helped with scripts
- Usage of the platform with the web user interface

What was the results ?

- The scaling of transform and publisher actually works as expected
- The more you add transform units, the more you will encode in parallel
- The more you add publisher units, the more you will handle streaming sessions
- The API security policies are applied, e.g. one cannot revoke tasks of other users
- **The (error) messages & HTTP codes are accurate :**
  - 200 : User "Mathias Coinchon" successfully updated
  - 403 : Authentication failed
  - 404 : No media with id ...
  - ...

Nothing is perfect, they are room for improvements !

Here are the most interesting of the numerous live-tests (at least the one I keep records).

Amazon with the minimal setup

This scenario represent a typical *public cloud only* deployment of the plaftorm.

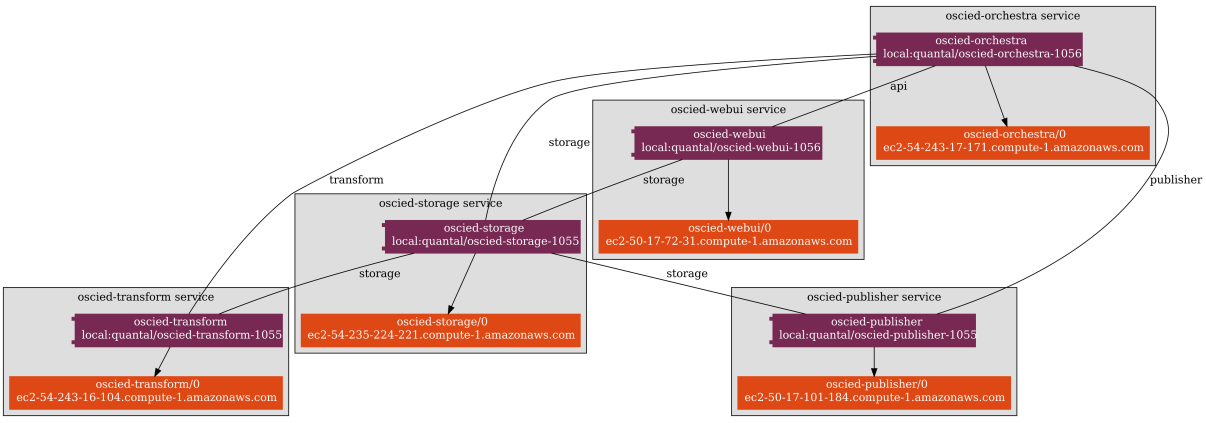


Figure 4.47: Status of the deployment by JuJu

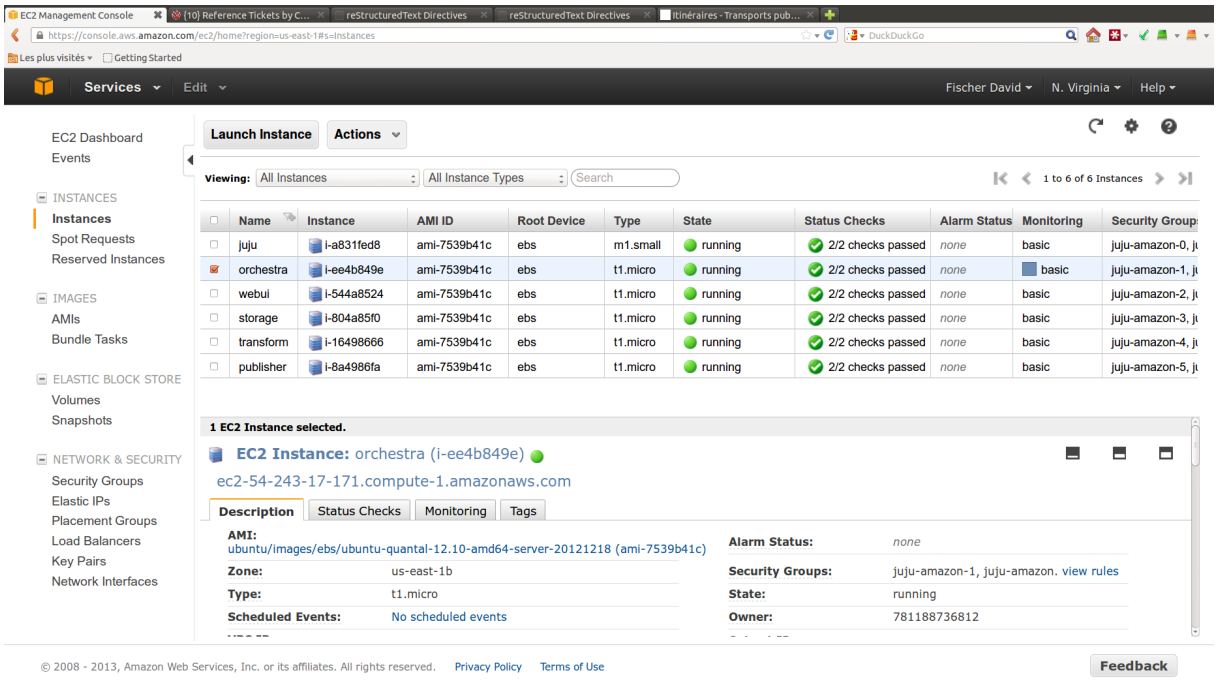


Figure 4.48: Instances running on Amazon EC2

date 2013-01-26 19:13:24,322

revision 1056

juju `debug.log`

menu `menu.log`

Local with 3 Transform and 2 Publisher

This scenario represent a typical *development purposes only* local deployment of the plaftorm.

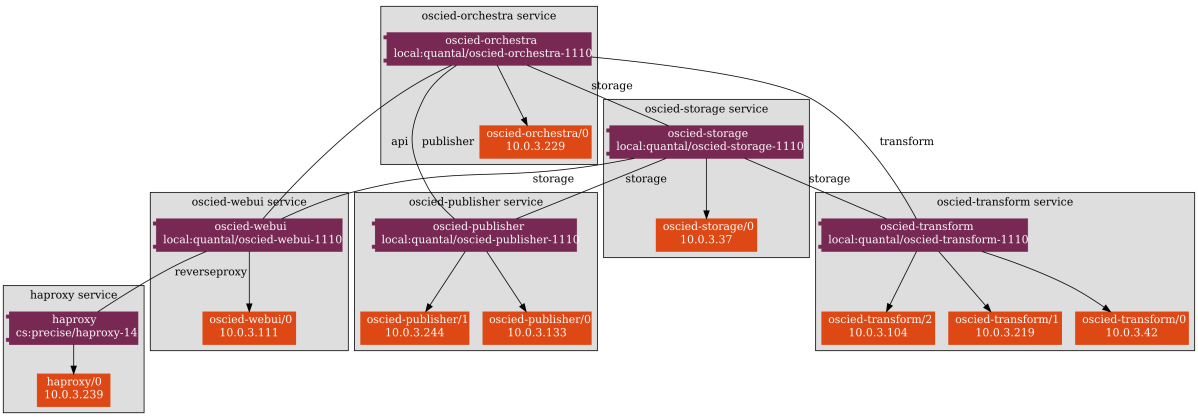


Figure 4.49: Status of the deployment by JuJu

date 2013-01-27 18:57:01,994

revision 1110

juju `debug.log`

screen `init_and_jobs.mp4`

Parallel deployment MaaS, Desktop, Amazon

This scenario represent a more powerful and realistic multi-environment deployment of the plaftorm. In parallel are running two completelly separated OSCIED : One of them is fully running on Amazon AWS, the other is the one detailed here.

date 2013-03-08

revision 1278

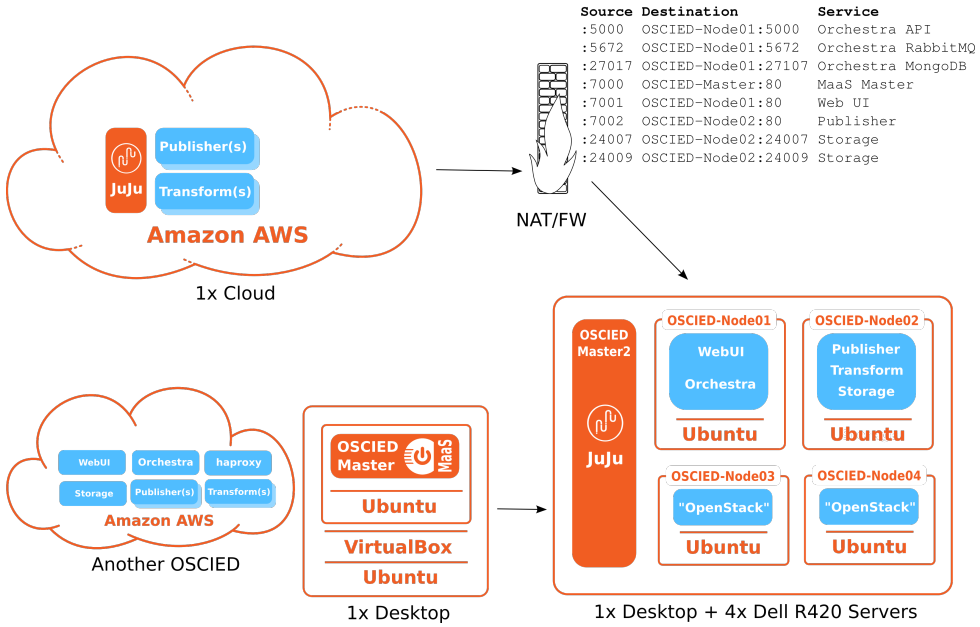


Figure 4.50: OSCIED made available as two completelly separated platforms running in parallel.

OSCIED-Master	Virtual Server	wan1	any	10.10.10.1	any	7000	80
OSCIED-Website	Virtual Server	wan1	any	10.10.10.2	any	7001	80
OSCIED-Publisher1	Virtual Server	wan1	any	10.10.10.3	any	7002	80
OSCIED-Publisher2	Virtual Server	wan1	any	10.0.3.40	any	7003	80
OSCIED-Storage	Virtual Server	wan1	any	10.10.10.3	any	24007	24007
OSCIED-Storage2	Virtual Server	wan1	any	10.10.10.3	any	24009	24009
OSCIED-MongoDB	Virtual Server	wan1	any	10.10.10.2	any	27017	27017
OSCIED-RabbitMQ	Virtual Server	wan1	any	10.10.10.2	any	5672	5672
OSCIED-API	Virtual Server	wan1	any	10.10.10.2	any	5000	5000

Figure 4.51: Configuration of our FW

MaaS

Main components of OSCIED were installed on EBU's setup of servers. The cluster of servers were provisioned and made available by Canonical's MaaS for JuJu to deploy OSCIED. OpenStack is not required in such scenario as Canonical's MaaS provisioning help us to deploy services quite easily.

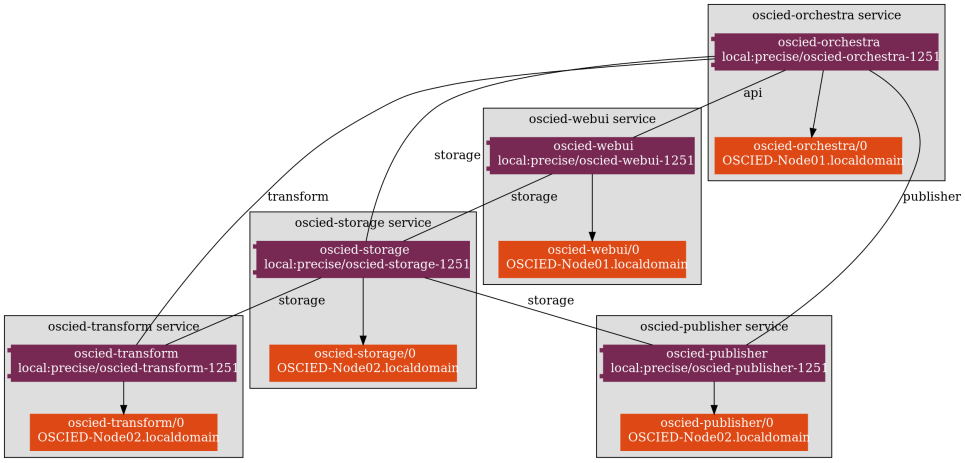


Figure 4.52: Status of the deployment by JuJu

menu maas `maas_menu.log`

juju maas `maas_juju.log`

Local

Some transform units were started on my workstation at hepia.

Figure 4.53: Status of the deployment by JuJu

menu local `local_menu.log`

juju local `local_juju.log`

Bernex

Some transform units were started on my personnal desktop computer at home.

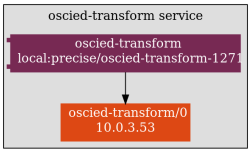


Figure 4.54: Status of the deployment by JuJu

menu bernex `bernex_menu.log`

juju bernex `bernex_juju.log`

Amazon

This public-cloud provider has been used to deploy transcoder and publication points.

Figure 4.55: Status of the deployment by JuJu

menu amazon `amazon_menu.log`

juju amazon `amazon_juju.log`

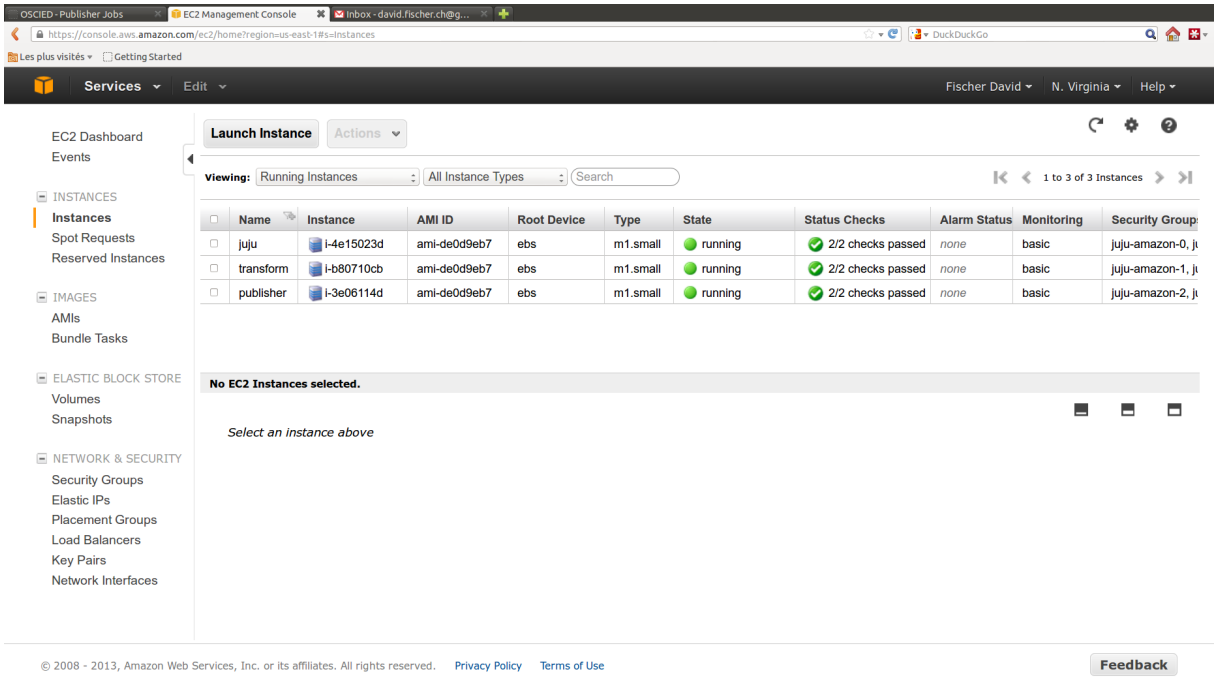


Figure 4.56: Amazon Web Services Console

4.10 Future Extensions

This section list the most interesting features to integrate to developed platform.

- See OSCIED Project's Roadmap

4.10.1 Demonstrator being Production Grade Tool

Collected ideas of improvement, for example :

- Storage : Enhance security + scalability
- Storage : Implement scalability
- Web UI : Improve tabs (add filters, ...)

4.10.2 Demonstrator with DASH Encoder

Adding MPEG-DASH encoding capabilities to OSCIED

4.10.3 Demonstrator with JuJu in Orchestra

- Improving ease of use of OSCIED
- Adding auto-scaling capabilities

4.10.4 Demonstrator with Easy Private-Cloud Provisioning

- Automating setup of new servers (maybe with MAAS)

4.10.5 Demonstrator with FIMS Interface

- Being compatible with FIMS MaM's

4.10.6 Demonstrator with Codem Interface

- Integrating Codem encoder to OSCIED

CHAPTER

FIVE

CONCLUSION

With OSCIED I proved that building a platform based on cloud-era Open-Source technologies can fix the scalability issue by providing a rather simple but yet powerful way to consume already existing enterprise's IT resources mixed with necessary amount of public cloud resources.

The hybrid-cloud model is the perfect approach to combine the highly available, low-cost, in-house IT infrastructure with scalable, on-demand public cloud infrastructures : You decide, OSCIED do !

Developed application made available the following to broadcasters :

- On-demand, scalable transcoding services by running virtualized transformation nodes
- On-demand, scalable distribution services by running virtualized publisher nodes

The platform can be used through an uncluttered, user-friendly web interface that actually maps the call of orchestrator's RESTful API.

This API is a key feature allowing broadcasters to automate usage of the platform and make possible the integration of OSCIED in broadcasters automated workflows.

Implemented features are tested and works well, the multi-cloud deployment, mixed with bare-metal storage works even better. I deployed the platform for weeks on my desktop computer, on Amazon AWS and on any server I was authorized to use.

The demonstrator is the proof of concept of something bigger, something not expected, something called OSCIED!

Personal

During the development of the application I not only learned Python but also the tools I never used before like Celery, MongoDB, ... It was a quite interesting challenge to start my work at the servers room level and finally designing the interface that makes the platform easy to use !

The only regrets I have, is that most of the improvements I think-ed-of are easy to implement, the only concern is the lack of time, as right after this thesis I will work at full time for another nice project called GaVi (guide audiovisuel interactif).

... I will be back !

CHAPTER

SIX

APPENDIX

6.1 Abbreviations

6.1.1 Entreprises & Institutions

EBU	European Broadcasting Union (EBU/UER)
HbbTV	Hybrid Broadcast Broadband Television
hepia	hepia : Haute école du Paysage d'Ingénierie et d'Architecture
IEEE	Institute of Electronical and Electronics Engineers
IETF	Internet Engineering Task Force
MPEG	Motion Picture Experts Group
MSE	HES-SO Master of Science in Engineering
OIPF	Open IPTV Forum
VCEG	Video Coding Experts Group
W3C	World Wide Web Consortium

6.1.2 Open-Source & OSS License

AGPL	GNU Affero GPLv3
CC	Creative Commons
GPL	GNU GPLv3
LGPL	GNU LGPLv3
MIT	MIT License
OSS	Open-Source

6.1.3 Networking

CDN	Content Delivery Network
DNS	Domain Name System
ISP	Internet Service Provider
LAN	Local Area Network

6.1.4 Programming

API	Application Programming Interface
GIT	Git (version control system)
IDE	Integrated Development Environment
SVN	Subversion (version control system)

6.1.5 Operational & Cloud

PXE	Preboot Execution Environment
SLA	Service-Level Agreement
MAAS	Metal as a Service
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service

6.1.6 Multimedia

HbbTV	Hybrid Broadcast Broadband Television
QoE	Quality of Experience
QoS	Quality of Service
SDI	Serial Digital Interface
STB	Set-Top Box
TV	Television
VoD	Video On Demand

6.1.7 Formats & Protocols

DASH	MPEG	Dynamic Adaptive Streaming over HTTP
FLV	Adobe	Flash Video
H264	VCEG/MPEG	Advanced Video Coding
H265	VCEG/MPEG	High Efficiency Video Coding
HLS	Apple Inc.	HTTP Live Streaming
HTML	W3C	Hypertext Mark-up Language
HTTP	IETF	Hypertext Transfer Protocol
IP	IETF	Internet Protocol
MMS	Microsoft	Microsoft Media Services
MP4	MPEG	MPEG-4 File Format
MSS	Microsoft	Microsoft Smooth Streaming
RDT	RealNetworks	Real Data Transport
rst	Python	reStructuredText
RTCP	IETF	Real-Time Control Protocol
RTMP	Adobe	Real-Time Messaging Protocol
RTP	IETF	Real-Time Transport Protocol
RTSP	IETF	Real-Time Streaming Protocol
TCP	IETF	Transmission Control Protocol
TS	MPEG	Transport Stream (e.g. MPEG-2 TS)
UDP	IETF	User Datagram Protocol
XML	W3C	Extensible Markup Language

6.2 References

6.2.1 OSCIED TRAC Management

- [trac_home] http://claire-et-david.dyndns.org/OSCIED/
- [trac_roadmap] http://claire-et-david.dyndns.org/OSCIED/milestone
- [trac_source] http://claire-et-david.dyndns.org/OSCIED/browser
- [trac_tickets_active] http://claire-et-david.dyndns.org/OSCIED/report/3
- [trac_tickets_all] http://claire-et-david.dyndns.org/OSCIED/report/6
- [trac_tickets_references] http://claire-et-david.dyndns.org/OSCIED/report/10

6.2.2 OSCIED TRAC Source-Code Browser

- [browse_orchestra] http://claire-et-david.dyndns.org/OSCIED/browser/components/orchestra
- [browse_publisher] http://claire-et-david.dyndns.org/OSCIED/browser/components/publisher
- [browse_storage] http://claire-et-david.dyndns.org/OSCIED/browser/components/storage
- [browse_transform] http://claire-et-david.dyndns.org/OSCIED/browser/components/transform
- [browse_webui] http://claire-et-david.dyndns.org/OSCIED/browser/components/webui

6.2.3 OSCIED TRAC Tickets

- [Ticket 25] http://claire-et-david.dyndns.org/OSCIED/ticket/25
- [Ticket 36] http://claire-et-david.dyndns.org/OSCIED/ticket/36
- [Ticket 37] http://claire-et-david.dyndns.org/OSCIED/ticket/37
- [Ticket 40] http://claire-et-david.dyndns.org/OSCIED/ticket/40
- [Ticket 117] http://claire-et-david.dyndns.org/OSCIED/ticket/117
- [Ticket 120] http://claire-et-david.dyndns.org/OSCIED/ticket/120
- [Ticket 122] http://claire-et-david.dyndns.org/OSCIED/ticket/122
- [Ticket 131] http://claire-et-david.dyndns.org/OSCIED/ticket/131
- [Ticket 141] http://claire-et-david.dyndns.org/OSCIED/ticket/141

6.2.4 Entreprises & Institutions

- [Akamai] http://www.akamai.com/
- [Amazon AWS] http://aws.amazon.com/
- [AmazonCDN] http://aws.amazon.com/fr/cloudfront/
- [AmazonEC2] http://aws.amazon.com/fr/ec2/
- [AmazonS3] http://aws.amazon.com/fr/s3/
- [Canonical] http://www.canonical.com/
- [EBU] http://www.ebu.ch/fr/
- [EBU_TECH] http://tech.ebu.ch/
- [EBU_Cloud] http://tech.ebu.ch/cloudworkshop

- [HbbTV] http://www.hbbtv.org/
- [hepia] http://hepia.hesge.ch/
- [HP Cloud] https://www.hpcloud.com/
- [IEEE] http://www.ieee.org/index.html
- [IETF] http://www.ietf.org/
- [MPEG] http://mpeg.chiariglione.org/
- [MSE] http://www.hes-so.ch/en/master-engineering.html
- [NASA] http://www.nasa.gov/
- [OIPF] http://www.oipf.tv/
- [Rackspace] http://www.rackspace.com/
- [W3C] http://www.w3.org/
- [Zencoder] http://zencoder.com/en/

6.2.5 Open-Source Licenses

- [AGPL] http://www.gnu.org/licenses/agpl-3.0.html
- [CC] http://creativecommons.org/
- [GPL] http://www.gnu.org/licenses/gpl.html
- [LGPL] http://www.gnu.org/copyleft/lesser.html
- [MIT] http://en.wikipedia.org/wiki/MIT_License
- [OSS] http://en.wikipedia.org/wiki/Open_source

6.2.6 Codecs, Formats & Protocols

- [BSON] http://bsonspec.org/
- [DASH] http://dashif.org/mpeg-dash/
- [FLV] http://en.wikipedia.org/wiki/Flash_Video
- [H264] http://fr.wikipedia.org/wiki/H.264
- [H265] http://en.wikipedia.org/wiki/High_Efficiency_Video_Coding
- [HLS] http://en.wikipedia.org/wiki/HTTP_Live_Streaming
- [HTML] http://en.wikipedia.org/wiki/HTML
- [HTTP] http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- [IP] http://en.wikipedia.org/wiki/Internet_Protocol
- [JSON] http://www.json.org/
- [JSONLint] http://jsonlint.com/
- [MMS] http://fr.wikipedia.org/wiki/Microsoft_Media_Services
- [MP4] http://en.wikipedia.org/wiki/MPEG-4_Part_14
- [MSS] http://www.iis.net/downloads/microsoft/smooth-streaming
- [RDT] http://en.wikipedia.org/wiki/Real_Data_Transport
- [rst] http://docs.python.org/3.0/documenting/rest.html
- [RTCP] http://en.wikipedia.org/wiki/RTP_Control_Protocol

- [RTMP] http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol
- [RTP] http://en.wikipedia.org/wiki/Real-time_Transport_Protocol
- [RTSP] http://en.wikipedia.org/wiki/Rtsp
- [TCP] http://en.wikipedia.org/wiki/Transmission_Control_Protocol
- [TS] http://en.wikipedia.org/wiki/Transport_stream
- [UDP] http://en.wikipedia.org/wiki/User_Datagram_Protocol
- [XML] http://en.wikipedia.org/wiki/XML

6.2.7 Database & Storage

- [Ceph] http://ceph.com/ceph-storage/
- [GlusterFS] http://www.gluster.org/
- [GlusterFS_admin_guide] http://www.gluster.org/wp-content/uploads/2012/05/Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf
- [LS4] http://ls4.sourceforge.net/
- [LVM] http://en.wikipedia.org/wiki/Logical_Volume_Manager_%28Linux%29
- [MongoDB] http://www.mongodb.org/
- [MySQL] http://www.mysql.com/
- [NFS] http://en.wikipedia.org/wiki/Network_File_System

6.2.8 Multimedia

- [Distribution] http://en.wikipedia.org/wiki/Digital_distribution
- [FFmpeg] http://ffmpeg.org/
- [OBE] http://www.ob-encoder.com/index.html
- [QoE] http://en.wikipedia.org/wiki/Quality_of_experience
- [QoS] http://en.wikipedia.org/wiki/Quality_of_service
- [SDI] http://en.wikipedia.org/wiki/Serial_digital_interface
- [STB] http://en.wikipedia.org/wiki/Set-top_box
- [Transcoding] http://en.wikipedia.org/wiki/Transcoding
- [TV] http://en.wikipedia.org/wiki/Television
- [VoD] http://en.wikipedia.org/wiki/Video_on_demand

6.2.9 Networking

- [CDN] http://en.wikipedia.org/wiki/Content_delivery_network
- [DNS] http://en.wikipedia.org/wiki/Domain_Name_System
- [ISP] http://en.wikipedia.org/wiki/Internet_service_provider
- [LAN] http://en.wikipedia.org/wiki/Local_area_network

6.2.10 Operational & Cloud

- [Cloud computing] http://en.wikipedia.org/wiki/Cloud_computing
- [PXE] http://fr.wikipedia.org/wiki/Preboot_Execution_Environment
- [SLA] http://en.wikipedia.org/wiki/Service-level_agreement
- [MAAS] http://www.ubuntu.com/cloud/orchestration/deployment
- [IaaS] http://en.wikipedia.org/wiki/Infrastructure_as_a_service
- [PaaS] http://en.wikipedia.org/wiki/Platform_as_a_service
- [SaaS] http://en.wikipedia.org/wiki/Software_as_a_service

6.2.11 JuJu

- [JuJu] https://juju.ubuntu.com/
- [JuJu_stoftware] http://en.wikipedia.org/wiki/Juju_%28software%29
- [juju_charms_store] http://jujucharms.com/charms
- [juju_unit_startup] https://juju.ubuntu.com/docs/internals/unit-agent-startup.html
- [juju_local_provider] https://juju.ubuntu.com/get-started/local/
- [juju_maas_provider] https://juju.ubuntu.com/get-started/maas/
- [juju_openstack_provider] https://juju.ubuntu.com/get-started/openstack/
- [juju_amazon_provider] https://juju.ubuntu.com/get-started/amazon/
- [juju_hpcloud_provider] https://juju.ubuntu.com/get-started/hp-cloud/
- [juju_rackspace_provider] https://juju.ubuntu.com/get-started/rackspace/
- [Nginx Charm] http://jujucharms.com/~imbrandon/precise/nginx

6.2.12 OpenStack

- [OpenStack] http://www.openstack.org/
- [OpenStack Foundation] https://www.openstack.org/join
- [OpenStack Documentation] http://docs.openstack.org/
- [OpenStack_IaaS] http://en.wikipedia.org/wiki/OpenStack
- [Cinder] http://docs.openstack.org/developer/cinder/
- [Glance] http://docs.openstack.org/developer/glance/
- [Horizon] http://docs.openstack.org/developer/horizon/
- [Keystone] http://docs.openstack.org/developer/keystone/
- [Nova] http://docs.openstack.org/developer/nova/
- [Quantum] http://docs.openstack.org/developer/quantum/
- [Swift] http://docs.openstack.org/developer/swift/
- [openstack_folsom_gre_2nic] https://github.com/mseknibilel/OpenStack-Folsom-Install-guide
- [OS_folsom_install_guide] https://github.com/mseknibilel/OpenStack-Folsom-Install-guide
- [OS_folsom_gre_2nics] https://github.com/mseknibilel/OpenStack-Folsom-Install-guide/blob/GRE/2NICs/OpenStack_Folso
- [DevStack] http://devstack.org/

- [TryStack] http://trystack.org/

6.2.13 Programming

- [API] http://en.wikipedia.org/wiki/Application_programming_interface
- [GIT] http://git-scm.com/
- [GitHub] https://github.com/
- [IDE] http://en.wikipedia.org/wiki/Integrated_development_environment
- [Launchpad] https://launchpad.net/
- [snippets] http://en.wikipedia.org/wiki/Snippet_(programming)
- [SourceForge] http://sourceforge.net/
- [Subversion] http://subversion.apache.org/
- [SVN] http://subversion.apache.org/
- [StatSVN] http://www.statsvn.org/
- [TRAC] http://trac.edgewall.org/
- [TRACP] http://trac.edgewall.org/wiki/TracPlugins
- [VCS] http://en.wikipedia.org/wiki/Revision_control

6.2.14 Python and Sphinx

- [docstrings] http://www.python.org/dev/peps/pep-0257/
- [PyMongo] http://api.mongodb.org/python/current/
- [Python] http://www.python.org/
- [Sphinx_autodoc] http://sphinx-doc.org/ext/autodoc.html
- [Sphinx_autoflask] http://packages.python.org/sphinxcontrib-httpdomain/
- [Sphinx] http://sphinx-doc.org/

6.2.15 Virtualization

- [BridgeUtils] http://www.linuxfromscratch.org/blfs/view/svn/basicnet/bridge-utils.html
- [KVM] http://www.linux-kvm.org/page/Main_Page
- [LXC] http://lxc.sourceforge.net/
- [OpenVZ] http://openvz.org/Main_Page
- [vSwitch] http://openvswitch.org/
- [YahyaNursalim] http://www.yahyanursalim.com/linux/cloud-computing-with-proxmox-ve.html

6.2.16 Web Development

- [Apache 2] http://httpd.apache.org/
- [Celery] http://celeryproject.org/
- [Celery_Tasks] http://docs.celeryproject.org/en/latest/userguide/tasks.html
- [CodeIgniter] http://ellislab.com/codeigniter

- [CSS Bootstrap] http://twitter.github.com/bootstrap/
- [cURL] http://en.wikipedia.org/wiki/CURL
- [Flask] http://flask.pocoo.org/
- [H264 Streaming Module] http://h264.code-shop.com/trac/wiki/Mod-H264-Streaming-Apache-Version2
- [RabbitMQ] http://www.rabbitmq.com/

6.2.17 Miscellaneous

- [KISS] http://en.wikipedia.org/wiki/KISS_principle
- [Linux] http://www.gnu.org/gnu/linux-and-gnu.html
- [SSH] http://en.wikipedia.org/wiki/Secure_Shell
- [Ubuntu] http://www.ubuntu.com/
- [UbuntuOS] http://en.wikipedia.org/wiki/Ubuntu_%28operating_system%29
- [Ubuntu Quantal Server] http://www.canonical.com/content/ubuntu-server-1210-all-you-need-cloud
- [Wiki] http://en.wikipedia.org/wiki/Wiki

6.3 TRAC - An Overview

Next pages are screenshots of OSCIED Project's TRAC Environment to give you an overview of the functionalities that helped me during this Thesis.

Note: The screenshots are clickable (sorry for the readers who printed my report) !

6.3.1 Roadmap

Here you can see the milestones with the *estimated* progress (as every ticket has the same *weight* and they cannot be *partially* implemented). I noticed that the milestone *Some preliminary documentation* wasn't closed and I updated the status of tickets *5 weeks too late* ;-)

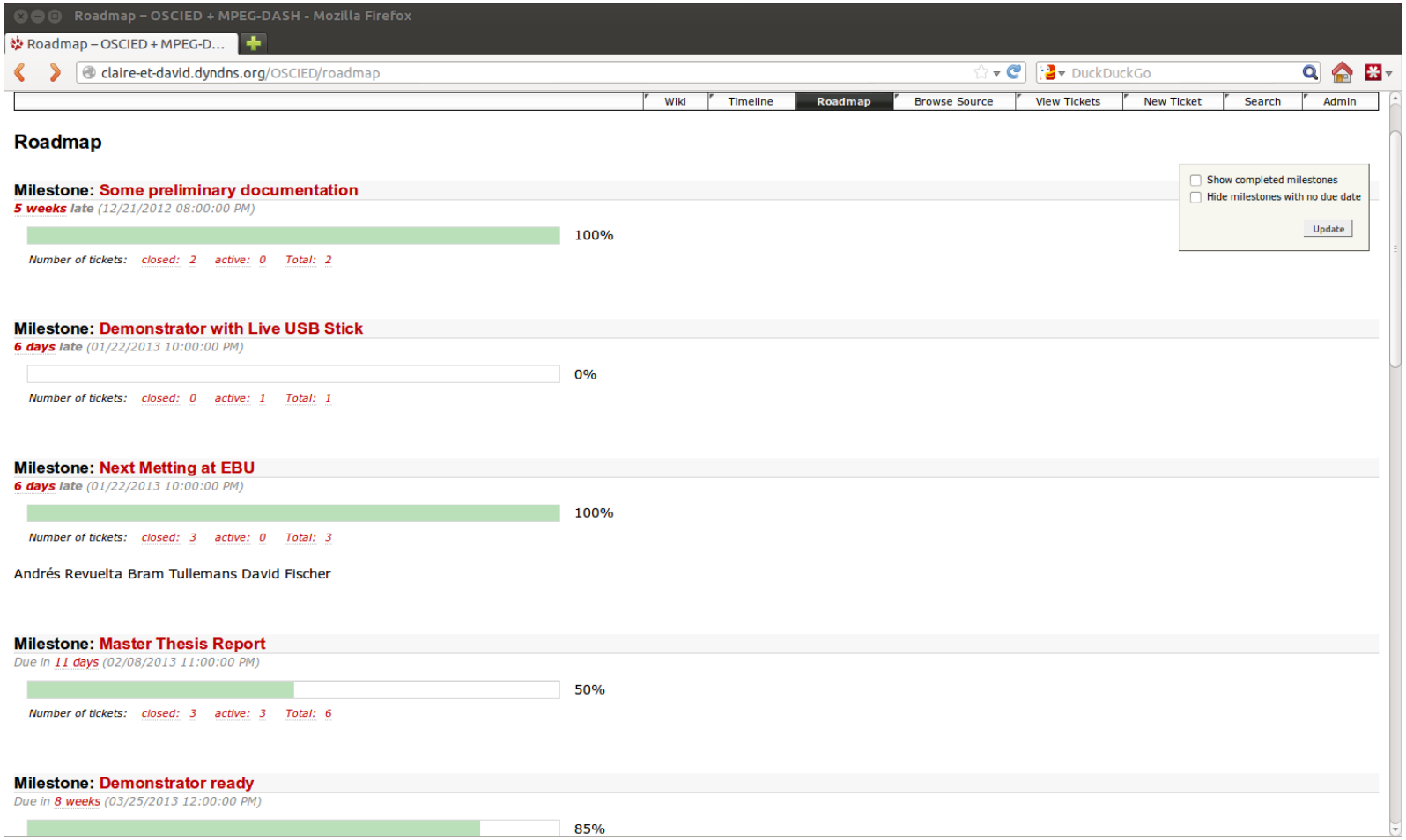


Figure 6.1: OSCIED Project's Roadmap

6.3.2 Source Browser

Here you can browse the Subversion repository of the project and you can travel back in time by viewing any of the thousand of revisions (not this is not a binary string 0b1110 but really the number 1‘110) !

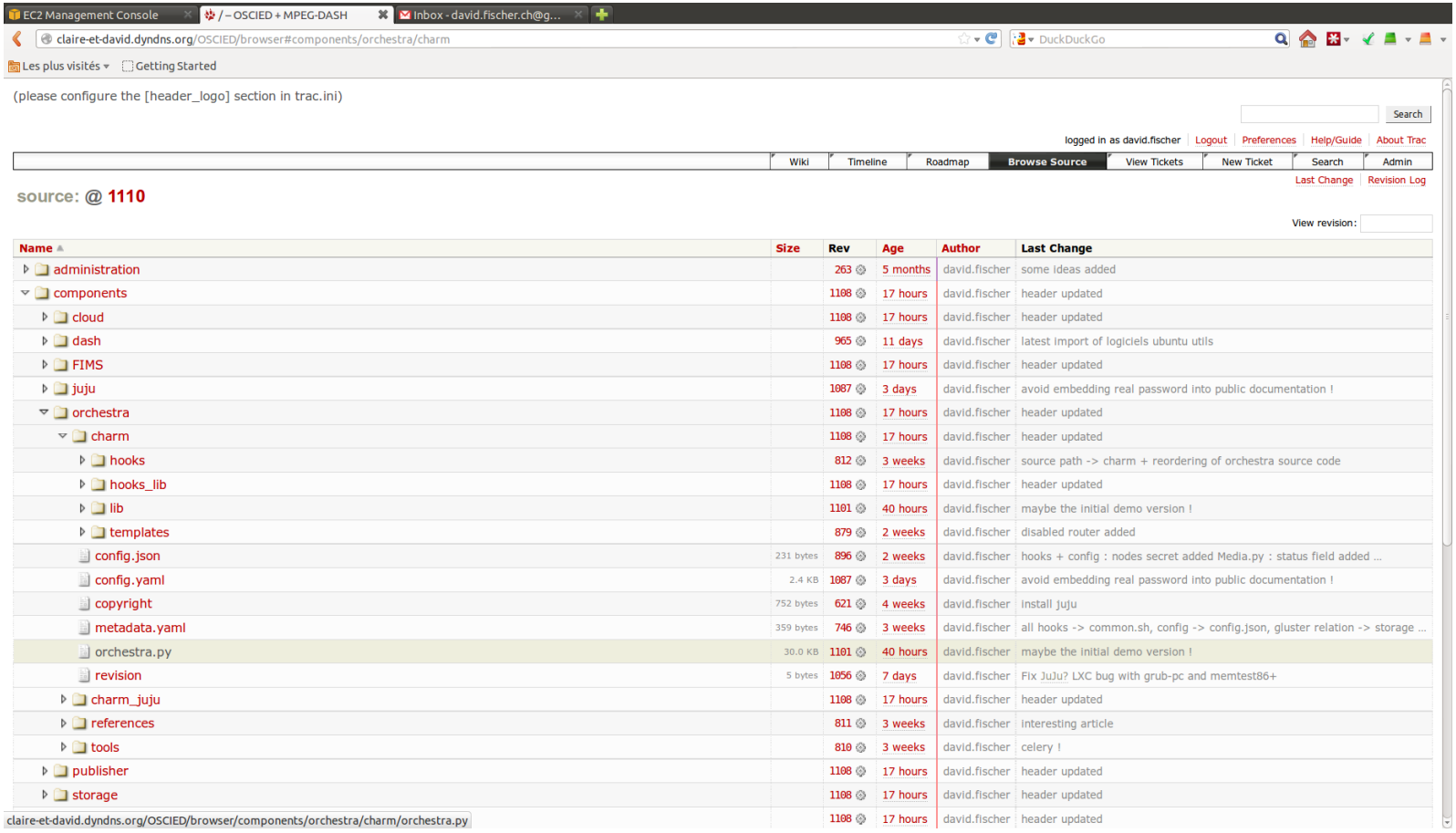


Figure 6.2: OSCIED Project’s Source Browser

6.3.3 Source Diff Tool

Here you can compare any versions of any files under Version Control System. This feature is linked to one of the most interesting and intensively used feature of Subversion always used before *committing* anything. Useful for any developer to recall itself what he does in order to write the message of the *commit*.

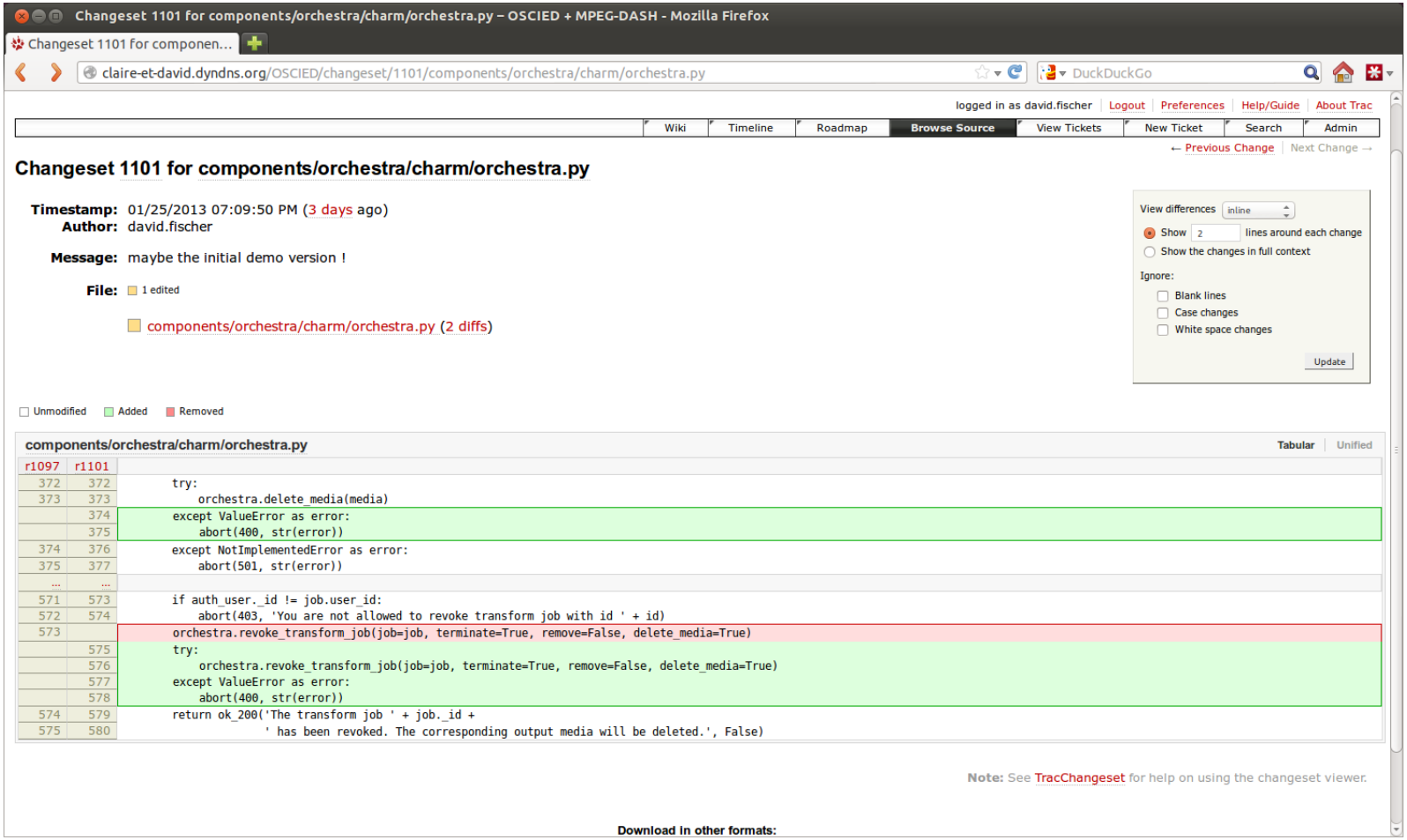
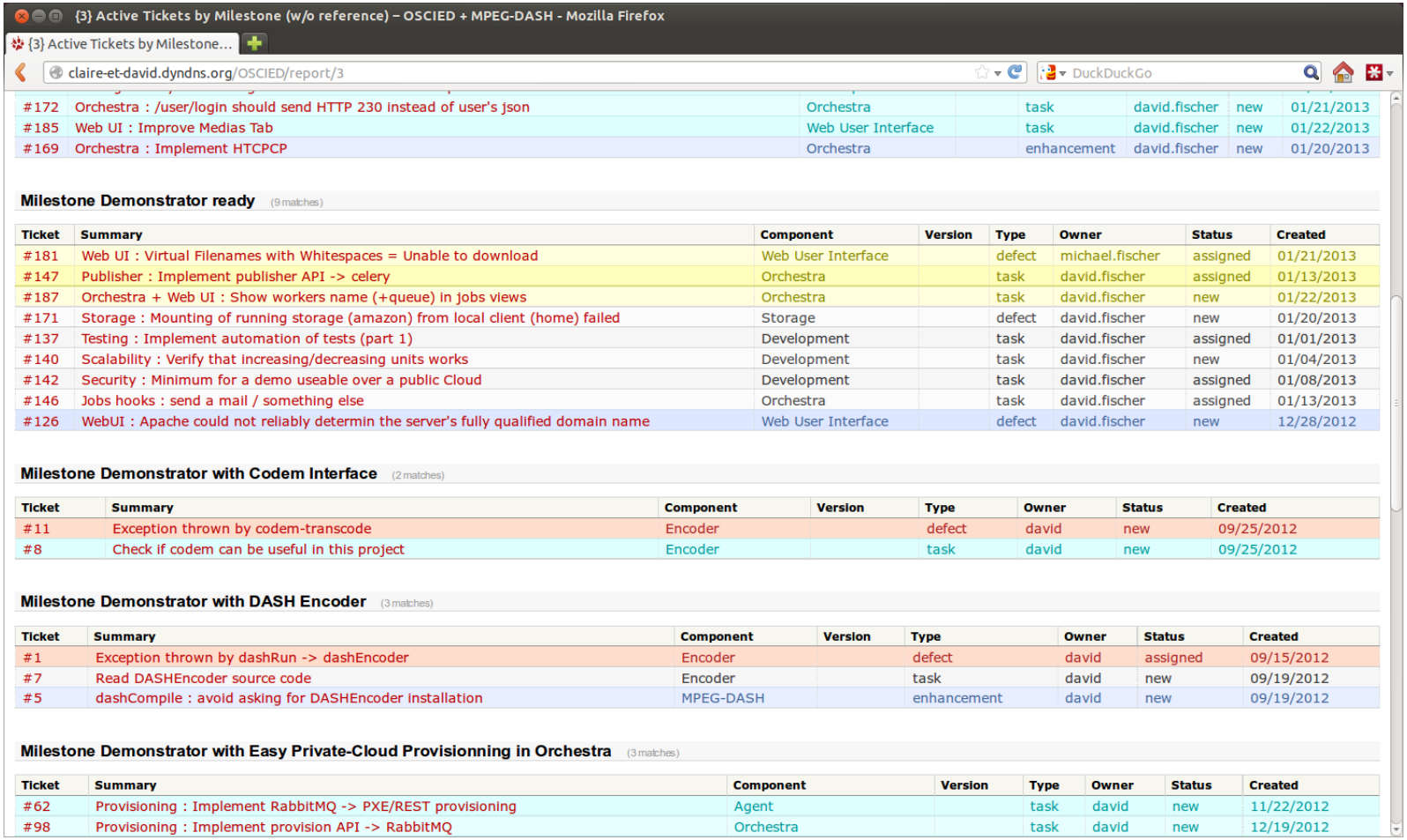


Figure 6.3: OSCIED Project's Source Diff Tool

6.3.4 Report : Active Tickets by Milestone (w/o reference)

Here you can browse through the pending tickets in order to select the next task you will work on or to show to your boss that the demonstrator is on the way to be ready on time.



#172	Orchestra : /user/login should send HTTP 230 instead of user's json	Orchestra		task	david.fischer	new	01/21/2013
#185	Web UI : Improve Medias Tab	Web User Interface		task	david.fischer	new	01/22/2013
#169	Orchestra : Implement HTCPCP	Orchestra		enhancement	david.fischer	new	01/20/2013

Milestone Demonstrator ready (9 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#181	Web UI : Virtual Filenames with Whitespaces = Unable to download	Web User Interface		defect	michael.fischer	assigned	01/21/2013
#147	Publisher : Implement publisher API -> celery	Orchestra		task	david.fischer	assigned	01/13/2013
#187	Orchestra + Web UI : Show workers name (+queue) in jobs views	Orchestra		task	david.fischer	new	01/22/2013
#171	Storage : Mounting of running storage (amazon) from local client (home) failed	Storage		defect	david.fischer	new	01/20/2013
#137	Testing : Implement automation of tests (part 1)	Development		task	david.fischer	assigned	01/01/2013
#140	Scalability : Verify that increasing/decreasing units works	Development		task	david.fischer	new	01/04/2013
#142	Security : Minimum for a demo useable over a public Cloud	Development		task	david.fischer	assigned	01/08/2013
#146	Jobs hooks : send a mail / something else	Orchestra		task	david.fischer	assigned	01/13/2013
#126	WebUI : Apache could not reliably determin the server's fully qualified domain name	Web User Interface		defect	david.fischer	new	12/28/2012

Milestone Demonstrator with Codem Interface (2 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#11	Exception thrown by codem-transcode	Encoder		defect	david	new	09/25/2012
#8	Check if codem can be useful in this project	Encoder		task	david	new	09/25/2012

Milestone Demonstrator with DASH Encoder (3 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#1	Exception thrown by dashRun -> dashEncoder	Encoder		defect	david	assigned	09/15/2012
#7	Read DASHEncoder source code	Encoder		task	david	new	09/19/2012
#5	dashCompile : avoid asking for DASHEncoder installation	MPEG-DASH		enhancement	david	new	09/19/2012

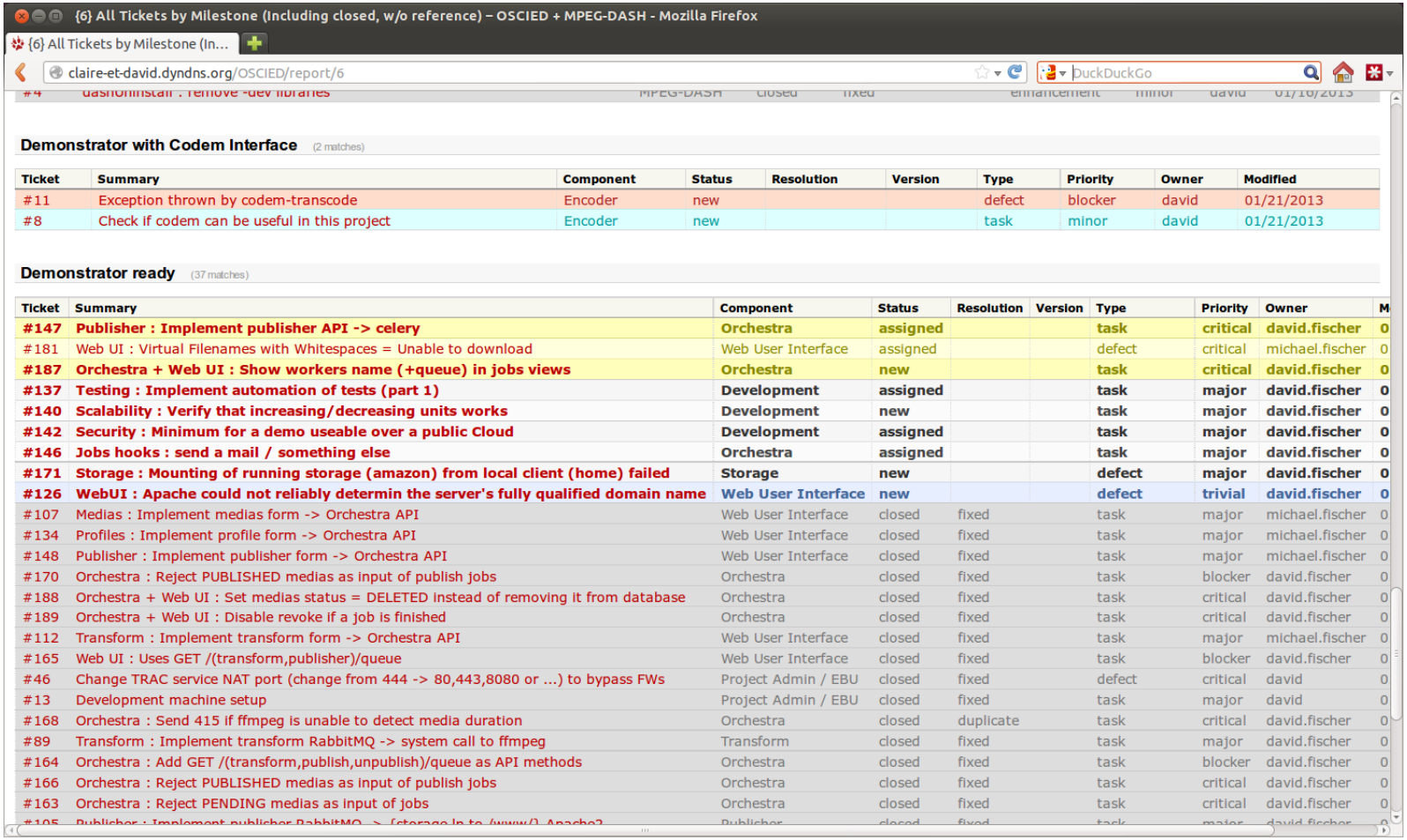
Milestone Demonstrator with Easy Private-Cloud Provisionning in Orchestra (3 matches)

Ticket	Summary	Component	Version	Type	Owner	Status	Created
#62	Provisioning : Implement RabbitMQ -> PXE/REST provisioning	Agent		task	david	new	11/22/2012
#98	Provisioning : Implement provision API -> RabbitMQ	Orchestra		task	david	new	12/19/2012

Figure 6.4: OSCIED Project's Active Tasks Tickets

6.3.5 Report : All Tickets by Milestone (Including closed, w/o reference)

Here you can browse through all tickets, including the ones which are specified as *closed*. A not so useful report except for this Thesis.



Demonstrator with Codem Interface (2 matches)

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	Modified
#11	Exception thrown by codem-transcode	Encoder	new			defect	blocker	david	01/21/2013
#8	Check if codem can be useful in this project	Encoder	new			task	minor	david	01/21/2013

Demonstrator ready (37 matches)

Ticket	Summary	Component	Status	Resolution	Version	Type	Priority	Owner	M
#147	**Publisher : Implement publisher API -> celery**	**Orchestra**	**assigned**			**task**	**critical**	**david.fischer**	**0**
#181	Web UI : Virtual Filenames with Whitespaces = Unable to download	Web User Interface	assigned			defect	critical	michael.fischer	0
#187	**Orchestra + Web UI : Show workers name (+queue) in jobs views**	**Orchestra**	**new**			**task**	**critical**	**david.fischer**	**0**
#137	**Testing : Implement automation of tests (part 1)**	**Development**	**assigned**			**task**	**major**	**david.fischer**	**0**
#140	**Scalability : Verify that increasing/decreasing units works**	**Development**	**new**			**task**	**major**	**david.fischer**	**0**
#142	**Security : Minimum for a demo useable over a public Cloud**	**Development**	**assigned**			**task**	**major**	**david.fischer**	**0**
#146	**Jobs hooks : send a mail / something else**	**Orchestra**	**assigned**			**task**	**major**	**david.fischer**	**0**
#171	**Storage : Mounting of running storage (amazon) from local client (home) failed**	**Storage**	**new**			**defect**	**major**	**david.fischer**	**0**
#126	**WebUI : Apache could not reliably determin the server's fully qualified domain name**	**Web User Interface**	**new**			**defect**	**trivial**	**david.fischer**	**0**
#107	Medias : Implement medias form -> Orchestra API	Web User Interface	closed	fixed		task	major	michael.fischer	0
#134	Profiles : Implement profile form -> Orchestra API	Web User Interface	closed	fixed		task	major	michael.fischer	0
#148	Publisher : Implement publisher form -> Orchestra API	Web User Interface	closed	fixed		task	major	michael.fischer	0
#170	Orchestra : Reject PUBLISHED medias as input of publish jobs	Orchestra	closed	fixed		task	blocker	david.fischer	0
#188	Orchestra + Web UI : Set medias status = DELETED instead of removing it from database	Orchestra	closed	fixed		task	critical	david.fischer	0
#189	Orchestra + Web UI : Disable revoke if a job is finished	Orchestra	closed	fixed		task	critical	david.fischer	0
#112	Transform : Implement transform form -> Orchestra API	Web User Interface	closed	fixed		task	major	michael.fischer	0
#165	Web UI : Uses GET /(transform,publisher)/queue	Web User Interface	closed	fixed		task	blocker	david.fischer	0
#46	Change TRAC service NAT port (change from 444 -> 80,443,8080 or ...) to bypass FWs	Project Admin / EBU	closed	fixed		defect	critical	david	0
#13	Development machine setup	Project Admin / EBU	closed	fixed		task	major	david	0
#168	Orchestra : Send 415 if ffmpeg is unable to detect media duration	Orchestra	closed	duplicate		task	critical	david.fischer	0
#89	Transform : Implement transform RabbitMQ -> system call to ffmpeg	Transform	closed	fixed		task	major	david.fischer	0
#164	Orchestra : Add GET /(transform,publish,unpublish)/queue as API methods	Orchestra	closed	fixed		task	blocker	david.fischer	0
#166	Orchestra : Reject PUBLISHED medias as input of publish jobs	Orchestra	closed	fixed		task	critical	david.fischer	0
#163	Orchestra : Reject PENDING medias as input of jobs	Orchestra	closed	fixed		task	critical	david.fischer	0

Figure 6.5: OSCIED Project's All Tasks Tickets

6.3.6 Report : Reference Tickets by Component

Here you can browse through my *online* bibliography. This is actually a set of the most interesting resources I founded during my investigations, grouped by topic.

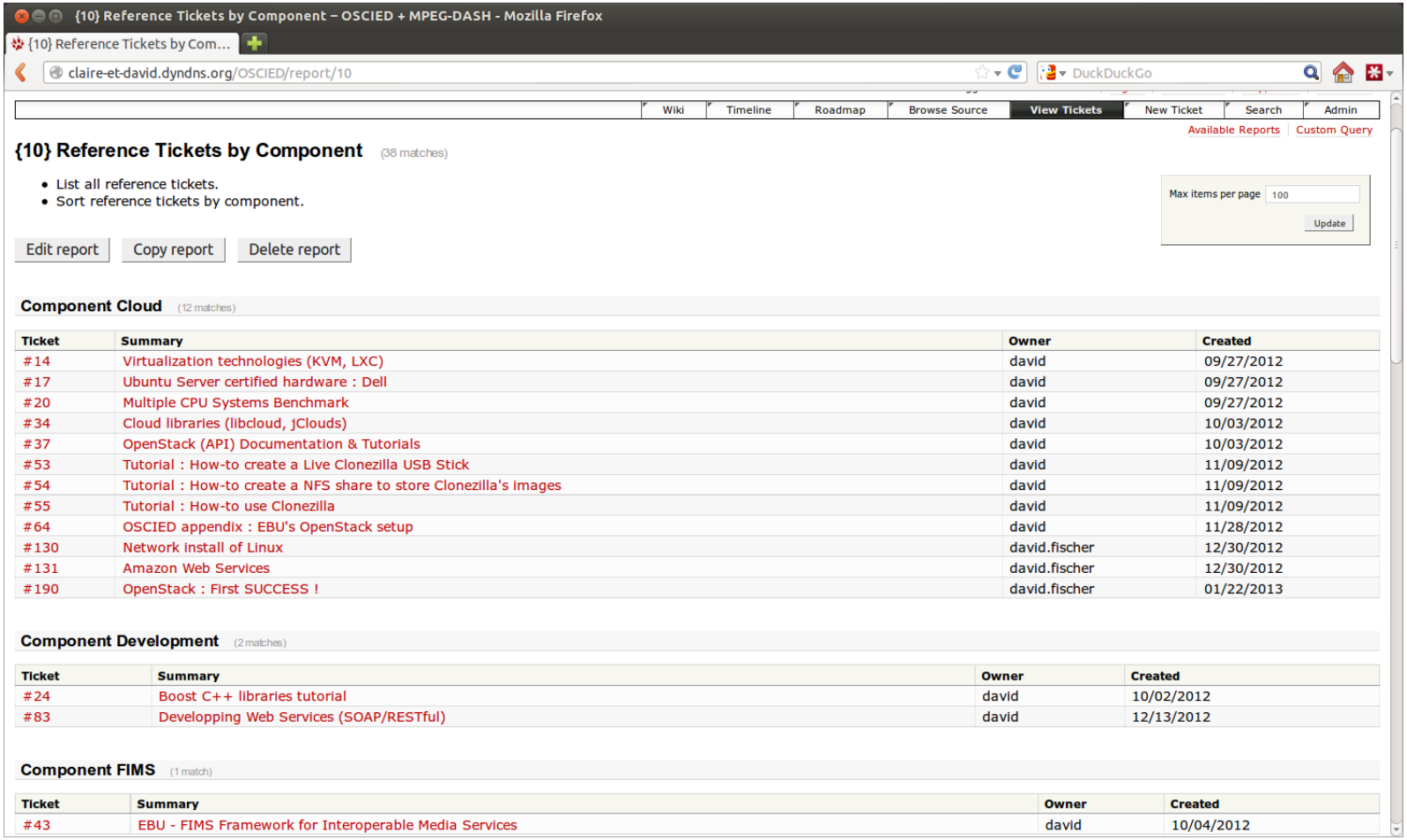


Figure 6.6: OSCIED Project's All Reference Tickets

6.3.7 Reference Ticket #120 : Developing with Python

Here you can see the content of one of the reference tickets. As you can see most of them will take at least one A4 page if embedded in my Thesis ...

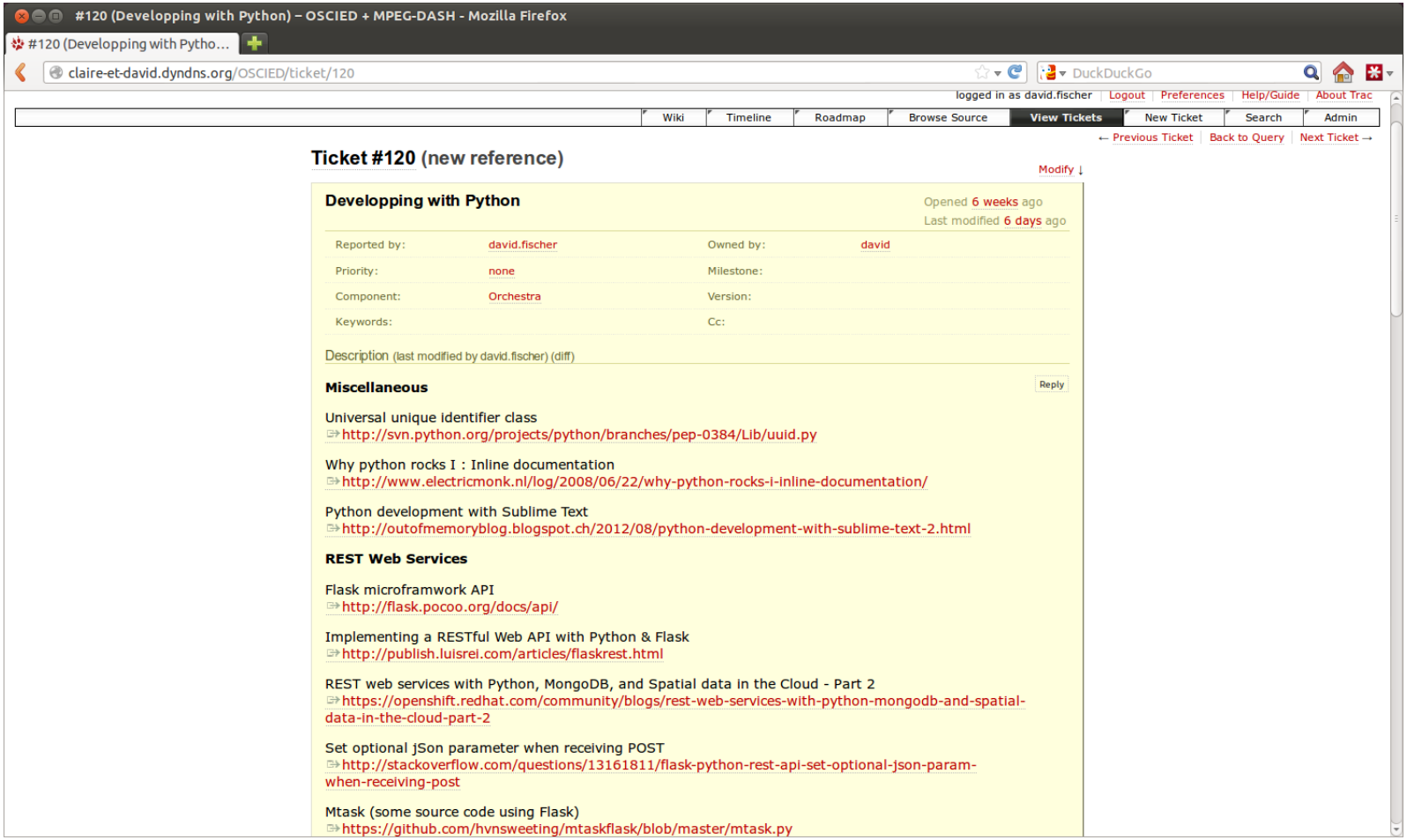


Figure 6.7: OSCIED Project's Reference Ticket #120

6.4 OSCIED - Source Code Licensing

6.4.1 Common HEADER (License)

```
#!/usr/bin/env bash

#**************************************************************************************************#
#              OPEN-SOURCE CLOUD INFRASTRUCTURE FOR ENCODING AND DISTRIBUTION : SCRIPTS
#
#  Authors   : David Fischer
#  Contact   : david.fischer.ch@gmail.com / david.fischer@hesge.ch
#  Project   : OSCIED (OS Cloud Infrastructure for Encoding and Distribution)
#  Copyright : 2012 OSCIED Team. All rights reserved.
#**************************************************************************************************#
#
# This file is part of EBU/UER OSCIED Project.
#
# This project is free software: you can redistribute it and/or modify it under the terms of the
# GNU General Public License as published by the Free Software Foundation, either version 3 of the
# License, or (at your option) any later version.
#
# This project is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without
# even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with this project.
# If not, see <http://www.gnu.org/licenses/>
#
# Retrieved from https://github.com/EBU-TI/OSCIED

set -o nounset # will exit if an unitialized variable is used
```

6.5 OSCIED - Orchestra RESTful API

See also:

Automated documentation. Please see Sphinx_autodoc and Sphinx_autoflask !

6.5.1 Utility Methods

orchestra.**requires_auth**(*request*, *allow_root=False*, *allow_node=False*, *allow_any=False*, *role=None*, *id=None*, *mail=None*)

This method implements Orchestra's RESTful API authentication logic. Here is ensured that an access to a method of the API is filtered based on rules (this method's parameters). HTTP user agent must authenticate through HTTP basic access authentication. The username must be user's email address and password must be user's secret. This not apply for system-users like root or node as they do not have any e-mail address.

> **Warning:** Username and password are passed as plaintext, SSL/TLS is one of the way to improve security although this was not tested during my thesis.

This method will abort request with HTTP 401 error if HTTP user agent doesn't authenticate.

Parameters

- **request** – the request itself, credentials are retrieved from request authorization header
- **allow_root** – if set to *True* root system-user will be allowed
- **allow_node** – if set to *True* node system-user will be allowed
- **allow_any** – if set to *True* any authenticated user will be allowed
- **role** – if set to <name>, any user will "name" role set to *True* will be allowed

- **id** – if set to <uuid>, any user with _id equal to "uuid" will be allowed
- **mail** – if set to <mail>, any user with mail equal to "mail" will be allowed

This method will abort request with HTTP 403 error if none of the following conditions are met.

Example:

```
# Allow any authenticated user
@app.route('/my/example/route', methods=['GET'])
def api_my_example_route():
    if request.method == 'GET':
        auth_user = requires_auth(request=request, allow_any=True)
        ...
        return ok_200('my return value', True)

# Allow root system-user or any user with admin attribute set
@app.route('/my/restricted/route', methods=['GET'])
def api_my_restricted_route():
    if request.method == 'GET':
        auth_user = requires_auth(request=request, allow_root=True, allow_role='admin')
        ...
        return ok_200('my return value', True)
```

6.5.2 The RESTful API

GET /transform/profile/count
Return profiles count.
Example request:

```
GET /transform/profile/count HTTP/1.1
Host: somewhere.com
Header: nabil@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": 100}
```

Allowed Any user
Status Codes
- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /transform/job/count
Return transform jobs count.
Example request:

```
GET /transform/job/count HTTP/1.1
Host: somewhere.com
Header: marylene@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": 15260}
```

Allowed Any user
Status Codes
- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /transform/job/HEAD
Return an array containing the transform jobs serialized as JSON. The transform jobs attributes are appended with the Celery's `async result` of the jobs.
Example request:

```
GET /transform/job/HEAD HTTP/1.1
Host: somewhere.com
Header: thomas@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user
Status Codes
- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /publish/job/count
Return publish jobs count.
Example request:

```
GET /publish/job/count HTTP/1.1
Host: somewhere.com
Header: sophie@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": 3904}
```

Allowed Any user
Status Codes
- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /publish/job/HEAD

Return an array containing the publish jobs serialized as JSON.
The publish jobs attributes are appended with the Celery's `async result` of the jobs.
Example request:

```
GET /publish/job/HEAD HTTP/1.1
Host: somewhere.com
Header: antonin@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user
Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /transform/callback

This method is called by transform workers when they finish their work.
If job is successful, the orchestrator will set media's status to READY. Else, the orchestrator will append `error_details` to `statistic` attribute of job.
The media will be deleted if job failed (even the worker already take care of that).
Example request:

```
POST /transform/callback HTTP/1.1
Host: somewhere.com
Header: node:abcdef
Accept: application/json
Content-Type: application/json

{
  "job_id": "1b96dcd6-7460-11e2-a06d-3085a9accb47",
  "status": "SUCCESS"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": "Your work is much appreciated, thanks !"}
```

Allowed Node
Query Parameters

- **job_id** – Job's id (required)
- **status** – Job's status (SUCCESS) or error's details (required)

Status Codes

- **200** – OK
- **400** – Key `key` not found. *or* on type or value error
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No transform job with id `id`.
- **404** – Unable to find output media with id `id`.
- **415** – Requires (valid) json content-type.

GET /transform/profile

Return an array containing the transform profiles serialized to JSON.
Example request:

```
GET /transform/profile HTTP/1.1
Host: somewhere.com
Header: michel@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]}
```

Allowed Any user
Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /transform/profile

Add a transform profile.

The transform profile's `encoder_string` attribute can be a keyword like :

- **copy** to bypass FFmpeg and do a simple file block copy ;
- ... or it can be a valid string containing FFmpeg options (see example) ;

Example request:

```
POST /transform/profile HTTP/1.1
Host: somewhere.com
Header: daniel@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{
  "title": "To MP4",
  "description": "Convert to MP4 (container)",
  "encoder_string": "-acodec copy -vcodec copy -f mp4"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c316ff1a-74f8-11e2-82d4-3085a9accd33",
    "title": "To MP4",
    "description": "Convert to MP4 (container)",
    "encoder_string": "-acodec copy -vcodec copy -f mp4"
  }
}
```

Allowed Any user

Query Parameters

- **title** – New profile's title (required)
- **description** – New profile's description (required)
- **encoder_string** – New profile's (FFmpeg) encoder string (required)

Status Codes

- **200** – OK
- **400** – Key `key` not found. *or* on type or value error
- **400** – Duplicate transform profile title `profile`.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **415** – Requires (valid) json content-type.

GET /transform/queue

Return an array containing the transform queues serialized to JSON.

Example request:

```
GET /transform/queue HTTP/1.1
Host: somewhere.com
Header: marco@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": ["transform_amazon", "transform_ebu_geneva"]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /unpublish/queue

Return an array containing the publish queues.

Example request:

```
GET /publish/queue HTTP/1.1
Host: somewhere.com
Header: jean-claude@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": ["publisher_london", "publisher_ebu_geneva"]
}
```

Allowed Any user

Status Codes

- **200** – OK

- **401** – Authenticate.
- **403** – Authentication Failed.

GET /publisher/queue

Return an array containing the publish queues.

Example request:

```
GET /publish/queue HTTP/1.1
Host: somewhere.com
Header: jean-claude@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": ["publisher_london", "publisher_ebu_geneva"]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /transform/job

Return an array containing the transform jobs serialized to JSON.

The transform jobs attributes are appended with the Celery's `async result` of the jobs.

All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.

Example request:

```
GET /transform/job HTTP/1.1
Host: somewhere.com
Header: antoinette@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /transform/job

Launch a transform job.

Any user can launch a transform job using any media as input and any transform profile. This is linked to media and transform profile API methods access policies.

The output media is registered to the database with the PENDING status and the `parent_id` field is set to input media's `id`. This permit to know relation between medias !

The orchestrator will automatically add `add_date` to `statistic`.

Note: Interesting enhancement would be to :

•Schedule obs by specifying start time (...) ;

•Handle the registration of jobs related to PENDING medias ;

Example request:

```
POST /transform/job HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
Content-Type: application/json

{
  "media_in_id": "a396fe66-74ee-11e2-89ad-3085a9accbb8",
  "profile_id": "c316ff1a-74f8-11e2-82d4-3085a9accd33",
  "virtual_filename": "avatar.mp4",
  "metadata": {"title": "Avatar (1080p)"},
  "queue": "transform_ebu-geneva"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": "ea9088f0-74f8-11e2-b780-3085a9acccb2a"}
```

Allowed Any user

Query Parameters

- **media_in_id** – New job input media's id (required)
- **profile_id** – New job profile's id (required)
- **virtual_filename** – New job output media's virtual_filename (required)
- **metadata** – New job output media's metadata (required)
- **queue** – The transform queue used to route the new job (required)

Status Codes

- **200** – OK
- **400** – Key `key` not found. *or* on type or value error
- **400** – Unable to transmit job to workers of queue `queue`.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`.
- **404** – No media with id `media_in_id`.
- **404** – No profile with id `profile_id`.
- **404** – No transform queue with name `queue`.
- **415** – Required (valid) json content-type.
- **501** – Cannot launch the job, input media status is `status`.

POST /publish/callback

This method is called by publisher workers when they finish their work.
If job is successful, the orchestrator will update `publish_uri` attribute of job, set media's status to SUCCESS and update `public_uris` attribute. Else, the orchestrator will append `error_details` to `statistic` attribute of job.

Example request:

```
POST /publish/callback HTTP/1.1
Host: somewhere.com
Header: node:abcdef
Accept: application/json
Content-Type: application/json

{
  "job_id": "1b96dcd6-7460-11e2-a06d-3085a9accb47",
  "publish_uri": "http://<address>/medias/<user_id>/<media_id>/
                  Project_London_trailer_2009.mp4",
  "status": "SUCCESS"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": "Your work is much appreciated, thanks !"}
```

Allowed Node

Query Parameters

- **job_id** – Job's id (required)
- **publish_uri** – Publication URI of the media (required)
- **status** – Job's status (SUCCESS) or error's details (required)

Status Codes

- **200** – OK
- **400** – Key `key` not found. *or* on type or value error
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No publish job with id `id`.
- **404** – Unable to find media with id `id`.
- **415** – Requires (valid) json content-type.

GET /publish/queue

Return an array containing the publish queues.

Example request:

```
GET /publish/queue HTTP/1.1
Host: somewhere.com
Header: jean-claude@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": ["publisher_london", "publisher_ebu_geneva"]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /publish/job

Return an array containing the publish jobs serialized to JSON.
The publish jobs attributes are appended with the Celery's `async result` of the jobs.
All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.

Example request:

```
GET /publish/job HTTP/1.1
Host: somewhere.com
Header: melanie@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /publish/job

Launch a publish job.

Any user can launch a publish job using any media as input. This is linked to media API methods access policy.
The orchestrator will automatically add `add_date` to `statistic`.

Note: Interesting enhancements would be to :
•Schedule jobs by specifying start time (...)
•Handle the registration of jobs related to PENDING medias
•Permit to publish a media on more than one (1) publication queue
•Permit to unpublish a media vbia a unpublish (broadcast) message

---

Example request:

```
POST /publish/job HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
Content-Type: application/json

{
  "media_id": "a396fe66-74ee-11e2-89ad-3085a9accbb8",
  "queue": "publish_london"
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "73abcf7e-74ef-11e2-9322-3085a9accc9b9"
}
```

Allowed Any user

Query Parameters

- **media_id** – New job input media's id (required)
- **queue** – The publish queue used to route job (required)

Status Codes

- **200** – OK
- **400** – Key `key` not found. *or* on type or value error
- **400** – Unable to transmit job to workers of queue `queue`.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`.

- **404** – No media with id `media_id`.
- **404** – No publish queue with name `queue`.
- **415** – Required (valid) json content-type.
- **501** – Cannot launch the job, input media status is `status`.
- **501** – Cannot launch the job, input media will be published by another job with id `id`.

GET /media/count

Return medias count.

Example request:

```
GET /media/count HTTP/1.1
Host: somewhere.com
Header: tewfiq@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": 8000}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /media/HEAD

Return an array containing the medias serialized to JSON.

Example request:

```
GET /media/HEAD HTTP/1.1
Host: somewhere.com
Header: andres@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /user/login

Return authenticated user serialized to JSON if authentication passed (without `secret` field).

This method is useful for WebUI to simulate stateful login scheme and get informations about the user.

Note: This is kind of duplicate with API's GET /user/id/*id* method ...

Example request:

```
GET /user/login HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c4daa8a6-6be4-11e2-ae91-3085a9accb47",
    "first_name": "Tabby",
    "last_name": "Fischer",
    "name": "Tabby Fischer",
    "mail": "tabby@bernex.ch",
    "admin_platform": false
  }
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /user/count

Return users count.

Example request:

```
GET /user/count HTTP/1.1
Host: somewhere.com
Header: bram@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": 5000}
```

Allowed Root and any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /index

Return an about string.
This method is actually used by Orchestra charm's hooks to check API's status.

Example request:

```
GET / HTTP/1.1
Host: example.com
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200, "value":
  "Orchestra : EBU's OSCIED Orchestrator by David Fischer 2012\n"
}
```

Allowed Any user (including unauthenticated)

Status Codes

- **200** – OK

POST /flush

Flush Orchestrator's database.
This method is useful for test/development purposes.

Example request:

```
POST /flush HTTP/1.1
Host: example.com
Header: root:password
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{"status": 200, "value": "Orchestra database flushed !"}
```

Allowed Only root

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

GET /media

Return an array containing the medias serialized to JSON.
All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.

Example request:

```
GET /media HTTP/1.1
Host: somewhere.com
Header: nabil@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Any user

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /media

Add a media.

This method handle registration of already uploaded media to the shared storage. For example, the WebUI will upload a media to uploads path **before** registering it with this method.

Medias in the shared storage are renamed with the following convention:

`storage_root/medias/user_id/media_id`

When published or downloaded, media will be renamed to `virtual_filename`. Spaces ( ) are not allowed and they will be converted to underscores (_).

Media's `metadata` must contain any valid JSON string. Only the `title` key is required. The orchestrator will automatically add `add_date` and `duration` to `metadata`.

Note: Registration of external media (aka. http://) will be an interesting improvement.

Example request:

```
POST /media HTTP/1.1
Host: somewhere.com
Header: d@f.com:oscied
Accept: application/json
Content-Type: application/json

{
  "uri": "glusterfs://<address>/medias_volume/uploads/
          Project London - Official Trailer [2009].mp4",
  "virtual_filename": "Project_London_trailer_2009.mp4",
  "metadata":
  {
    "title": "Project London - Official Trailer (2009)"
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "96590fdc-74f8-11e2-8c58-3085a9acc651",
    "user_id": "1298f206-74f8-11e2-9b82-3085a9acc11b",
    "parent_id": null,
    "uri": "glusterfs://<address>/medias_volume/medias/
            <user_id>/<media_id>",
    "public_uris": null,
    "virtual_filename": "Project_London_trailer_2009.mp4",
    "metadata": {
      "add_date": "2013-02-02 14:05",
      "duration": "00:02:44.88", "size": 54871886,
      "title": "Project London - Official Trailer (2009)"
    },
    "status": "READY"
  }
}
```

Allowed Any user can do that

Query Parameters

- **uri** – Media's source URI, actually only shared storage's URI are handled (required)
- **virtual_filename** – Media's filename when downloaded or published (required)
- **metadata** – JSON string containing metadatas about the media (required)

Status Codes

- **200** – OK
- **400** – on type or value error
- **400** – Key `key` not found.
- **400** – The media uri `uri` is already used by another media.
- **400** – Title key is required in media metadata.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – An error occured : `OSError`
- **415** – Requires (valid) json content-type.
- **501** – FIXME Add of external uri not implemented.

GET /user

Return an array containing the users serialized to JSON (without `secret` fields).
Example request:

```
GET /user HTTP/1.1
Host: somewhere.com
Header: peter@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
```

```
  "value": [{"_id": "...", "...": "..."}, {"_id": "..."}]
}
```

Allowed Root and user with admin_platform attribute set

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.

POST /user

Add an user.

Example request:

```
POST /user HTTP/1.1
Host: somewhere.com
Header: kouadi@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{
  "first_name": "Laurent",
  "last_name": "Nicolet",
  "mail": "laurent@comique.ch",
  "secret": "genevois_style",
  "admin_platform": false
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "8bda488c-6be8-11e2-89b7-3085a9accb47",
    "first_name": "Laurent",
    "last_name": "Nicolet",
    "name": "Laurent Nicolet",
    "mail": "laurent@comique.ch",
    "admin_platform": false
  }
}
```

Allowed Root and user with admin_platform attribute set

Query Parameters

- **first_name** – New user's first name (required)
- **last_name** – New user's last name (required)
- **mail** – New user's email address (required)
- **secret** – New user's secret (required)
- **admin_platform** – New user's admin_platform (required)

Status Codes

- **200** – OK
- **400** – on type or value error
- **400** – Key `key` not found.
- **400** – The email address `mail` is already used by another user.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **415** – Requires (valid) json content-type.

GET /

Return an about string.

This method is actually used by Orchestra charm's hooks to check API's status.

Example request:

```
GET / HTTP/1.1
Host: example.com
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200, "value":
  "Orchestra : EBU's OSCIED Orchestrator by David Fischer 2012\n"
}
```

Allowed Any user (including unauthenticated)

Status Codes

- **200** – OK

GET /transform/job/id/ (*id*) /HEAD

Return a transform job serialized to JSON.

The transform job attributes are appended with the Celery's `async result` of the job.

Example request:

```
GET /transform/job/id/48c111c8-74f8-11e2-a7a8-3085a9acc6c4/HEAD HTTP/1.1
Host: somewhere.com
Header: edoardo@oscied.org:oscied
Accept: application/json
Content-Type: application/json
```

Example response:
Floating numbers are here with “” for autoflask to work !!

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "48c111c8-74f8-11e2-a7a8-3085a9acc6c4",
    "user_id": "4e8a1bce-74f3-11e2-9660-3085a9acce0b",
    "media_in_id": "a8a165b8-74f7-11e2-a59e-3085a9acc049",
    "media_out_id": "52ea73ac-74f3-11e2-afdb-3085a9acc5ff",
    "profile_id": "55da66d6-74f3-11e2-9dff-3085a9acce4e",
    "statistic": {
      "add_date": "2013-02-11 22:44",
      "start_date": "2013-02-11 22:44",
      "elapsed_time": "19.241864919662476",
      "eta_time": 0, "percent": 100,
      "media_in_size": 54871886, "media_in_duration": "00:02:44.88",
      "media_out_size": 25601528, "media_out_duration": "00:00:01.95"
    },
    "revoked": false,
    "status": "SUCCESS"
  }
}
```

Allowed Any user

Parameters

- **id** – id of job to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No transform job with id `id`.
- **415** – Wrong id format `id`.

GET /publish/job/id/ (*id*) **/HEAD**

Return a publish job serialized to JSON.
The publish job attributes are appended with the Celery’s `async result` of the job.

Example request:

```
GET /publish/job/id/c697f528-74f7-11e2-96a3-3085a9accc5d/HEAD HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
```

Example response:
Floating numbers are here with “” for autoflask to work !!

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c697f528-74f7-11e2-96a3-3085a9accc5d"
    "user_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
    "media_id": "a8a165b8-74f7-11e2-a59e-3085a9acc049",
    "publish_uri": "http://<publish_uri>/medias/<user_id>/<media_id>/
                    Project_London_trailer_2009.mp4",
    "statistic":
    {
      "add_date": "2013-02-11 22:38",
      "start_date": "2013-02-11 22:38",
      "elapsed_time": "0.5068690776824951",
      "eta_time": 0, "percent": 100,
      "media_size": 54871886, "publish_size": 54871886,
      "pid": 18307, "hostname": "famille-local-oscied-publisher-0"
    },
    "revoked": false,
    "status": "SUCCESS"
  }
}
```

Allowed Any user

Parameters

- **id** – id of job to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No publish job with id `id`.

GET /transform/profile/id/ (*id*)

Return a transform profile serialized to JSON.

Example request:

```
GET /transform/profile/id/c316ff1a-74f8-11e2-82d4-3085a9accd33 HTTP/1
Host: somewhere.com
Header: francois@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c316ff1a-74f8-11e2-82d4-3085a9accd33",
    "title": "To MP4",
    "description": "Convert to MP4 (container)",
    "encoder_string": "-acodec copy -vcodec copy -f mp4"
  }
}
```

Allowed Any user

Parameters

- **id** – id of profile to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No transform profile with id `id`.
- **415** – Wrong id format `id`.

PUT /transform/profile/id/ (*id*)

Update a transform profile.

> **Warning:** All fields can be updated, maybe not a good idea (??)

Example request:

```
PUT /transform/profile/id/c316ff1a-74f8-11e2-82d4-3085a9accd33 H
Host: somewhere.com
Header: dimitri@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{"description": "Convert any container to MP4"}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The transform profile \"To MP4\" has been updated."
}
```

Allowed Any user

param id id of profile to get

query title Transform profile's title (optional)

query description Transform profile's description (optional)

query encoder_string Transform profile's (FFmpeg) encoder string (optional)

statuscode 200 OK

statuscode 400 Key `key` not found. *or* on type or value error

statuscode 400 Duplicate transform profile title `profile`.

statuscode 401 Authenticate.

statuscode 403 Authentication Failed.

statuscode 404 No transform profile with id `id`.

statuscode 415 Wrong id format `id`.

statuscode 415 Requires (valid) json content-type.

PATCH /transform/profile/id/ (*id*)

Update a transform profile.

> **Warning:** All fields can be updated, maybe not a good idea (??)

Example request:

```
PUT /transform/profile/id/c316ff1a-74f8-11e2-82d4-3085a9accd33 HTTP/1
Host: somewhere.com
Header: dimitri@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{"description": "Convert any container to MP4"}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The transform profile \"To MP4\" has been updated."
```

```
}
```

Allowed Any user
param id id of profile to get
query title Transform profile's title (optional)
query description Transform profile's description (optional)
query encoder_string Transform profile's (FFmpeg) encoder string (optional)
statuscode 200 OK
statuscode 400 Key `key` not found. *or* on type or value error
statuscode 400 Duplicate transform profile title `profile`.
statuscode 401 Authenticate.
statuscode 403 Authentication Failed.
statuscode 404 No transform profile with id `id`.
statuscode 415 Wrong id format `id`.
statuscode 415 Requires (valid) json content-type.

DELETE /transform/profile/id/ (*id*)

Delete a transform profile.

Example request:

```
DELETE /transform/profile/id/c316ff1a-(...)-3085a9accd33 HTTP/1.1
Host: somewhere.com
Header: dimitri@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The transform profile \"To MP4\" has been deleted."
}
```

Allowed Any user

Parameters

- **id** – id of profile to delete

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No transform profile with id `id`.
- **415** – Wrong id format `id`.

GET /transform/job/id/ (*id*)

Return a transform job serialized to JSON.
The transform job attributes are appended with the Celery's `async result` of the job.
All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.

Example request:

```
GET /transform/job/id/ea9088f0-74f8-11e2-b780-3085a9acccb2a HTTP/1.1
Host: somewhere.com
Header: claire@oscied.org:oscied
Accept: application/json
Content-Type: application/json
```

Example response:
Floating numbers are here with "" for autoflask to work !!

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "48c111c8-74f8-11e2-a7a8-3085a9acc6c4",
    "user": {
      "_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
      "first_name": "David",
      "last_name": "Fischer",
      "name": "David Fischer",
      "mail": "d@f.com",
      "admin_platform": true
    },
    "media_in": {
      "_id": "a8a165b8-74f7-11e2-a59e-3085a9acc049",
      "user_id": "4e8a1bce-74f3-11e2-9660-3085a9acce0b",
      "parent_id": null,
      "uri": "glusterfs://<address>/medias_volume/medias/
              <user_id>/<media_id>",
      "public_uris": {
        "c697f528-74f7-11e2-96a3-3085a9accc5d":
        "http://10.0.3.254/medias/<user_id>/<media_id>/
         Project_London_trailer_2009.mp4"
      },
      "virtual_filename": "Project_London_trailer_2009.mp4",
      "metadata": {
        "add_date": "2013-02-11 22:37",
```

```
        "duration": "00:02:44.88", "size": 54871886,
        "title": "Project London - Official Trailer (2009)"
      },
      "status": "PUBLISHED"
    },
    "media_out": {
      "_id": "52ea73ac-74f3-11e2-afdb-3085a9acc5ff",
      "user_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
      "parent_id": "a8a165b8-74f7-11e2-a59e-3085a9acc049",
      "uri": "glusterfs://<address>/medias_volume/medias/
              <user_id>/<media_id>",
      "public_uris": null,
      "virtual_filename": "project_london.mp2",
      "metadata": {
        "add_date": "2013-02-11 22:44",
        "duration": "00:00:01.95", "size": 25601528,
        "title": "Project London MP2"
      }
      "status": "READY"
    },
    "profile": {
      "_id": "55da66d6-74f3-11e2-9dff-3085a9acce4e",
      "title": "To MP2",
      "description":
        "Convert video track to MPEG-2 format, copy audio track",
      "encoder_string":
        "-acodec copy -vcodec mpeg2video -f mpeg2video"
    },
    "statistic": {
        "add_date": "2013-02-11 22:44",
        "start_date": "2013-02-11 22:44",
        "elapsed_time": "19.241864919662476",
        "eta_time": 0, "percent": 100,
        "media_in_size": 54871886, "media_in_duration": "00:02:44.88",
        "media_out_size": 25601528, "media_out_duration": "00:00:01.95"
    },
    "revoked": false,
    "status": "SUCCESS"
  }
}
```

Allowed Any user

Parameters

- **id** – id of job to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No transform job with id `id`.
- **415** – Wrong id format `id`.

`DELETE /transform/job/id/` (*id*)

Revoke a transform job.

This method do not delete jobs from jobs database but set `revoked` attribute in jobs database and broadcast revoke request to transform units with Celery. If the job is actually running it will be canceled. The output media will be deleted.

Example request:

```
DELETE /transform/job/id/ea9088f0-74f8-11e2-b780-3085a9acccb2a HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
Content-Type: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The transform job \"<job_id>\" has been revoked.
            Corresponding output media will be deleted."
}
```

Allowed Only author of the job

Parameters

- **id** – id of job to delete

Status Codes

- **200** – OK
- **400** – on value error
- **400** – Transform job `id` is already revoked !
- **400** – Cannot revoke a transform job with status `status`.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **403** – You are not allowed to revoke transform job with id `id`.
- **404** – No transform job with id `id`.
- **415** – Wrong id format `id`.

`GET /publish/job/id/` (*id*)

Return a publish job serialized to JSON.
The publish job attributes are appended with the Celery's `async result` of the job.
All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.
Example request:

```
GET /publish/job/id/c697f528-74f7-11e2-96a3-3085a9accc5d HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
```

Example response:
Floating numbers are here with "" for autoflask to work !!

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c697f528-74f7-11e2-96a3-3085a9accc5d",
    "publish_uri": "http://<address>/medias/<user_id>/<media_id>/
                    Project_London_trailer_2009.mp4",
    "media": {
      "_id": "a8a165b8-74f7-11e2-a59e-3085a9acc049",
      "user_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
      "parent_id": null,
      "uri": "glusterfs://<address>/medias_volume/medias/
              <user_id>/<media_id>",
      "public_uris": {
        "c697f528-74f7-11e2-96a3-3085a9accc5d":
        "http://<address>/medias/<user_id>/<media_id>/
         Project_London_trailer_2009.mp4"
      },
      "virtual_filename": "Project_London_trailer_2009.mp4",
      "metadata": {
        "duration": "00:02:44.88", "add_date": "2013-02-11 22:37",
        "size": 54871886,
        "title": "Project London - Official Trailer (2009)"
      },
      "status": "PUBLISHED"
    },
    "user": { "name": "David Fischer", "...": "..." },
    "statistic": {
      "add_date": "2013-02-11 22:38", "start_date": "2013-02-11 22:38"
      "media_size": 54871886, "publish_size": 54871886,
      "elapsed_time": "0.5068690776824951",
      "eta_time": 0, "percent": 100,
      "pid": 18307, "hostname": "famille-local-oscied-publisher-0"
    },
    "revoked": false,
    "status": "SUCCESS"
  }
}
```

Allowed Any user
Parameters

- **id** – id of job to get

Status Codes

- **200** – OK

- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No publish job with id `id`.

`DELETE /publish/job/id/` (*id*)

Revoke a publish job.

This method do not delete jobs from jobs database but set `revoked` attribute in jobs database and broadcast revoke request to publisher units with Celery. If the job is actually running it will be canceled. The output publication media will be deleted.

Example request:

```
DELETE /pulish/job/id/c697f528-74f7-11e2-96a3-3085a9accc5d HTTP/1.1
Host: somewhere.com
Header: tabby@bernex.ch:miaow
Accept: application/json
Content-Type: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The publish job \"<job_id>\" has been revoked.
            Corresponding media will be unpublished from here."
}
```

Allowed Only author of the job

Parameters

- **id** – id of job to delete

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **403** – You are not allowed to revoke publish job with id `id`.
- **404** – No publish job with id `id`.
- **415** – Wrong id format `id`.

`GET /media/id/` (*id*) **`/HEAD`**

Return a media serialized to JSON.

Example request:

```
GET /media/id/96590fdc-74f8-11e2-8c58-3085a9acc651/HEAD HTTP/1.1
Host: somewhere.com
Header: monique@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "b8b9ae78-74f8-11e2-8dae-3085a9accbc1",
    "user_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
    "parent_id": null,
    "uri": "glusterfs://<address>/medias_volume/medias/
            <user_id>/<media_id>",
    "public_uris": null,
    "virtual_filename": "Psy_gangnam_style.flv",
    "metadata": {
      "duration": "00:04:12.16",
      "add_date": "2013-02-11 22:37",
      "title": "Psy - Gangnam Style",
      "size": 183190475
    },
    "status": "READY"
  }
}
```

Allowed Any user

Parameters

- **id** – id of media to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No media with id `id`.
- **415** – Wrong id format `id`.

`GET /media/id/` (*id*)

Return a media serialized to JSON.

All `thing_id` fields are replaced by corresponding `thing`. For example `user_id` is replaced by `user`'s data.

Example request:

```
GET /media/id/96590fdc-74f8-11e2-8c58-3085a9acc651 HTTP/1.1
Host: somewhere.com
Header: estelle@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "b8b9ae78-74f8-11e2-8dae-3085a9accbc1",
    "user": {
    "_id": "4e8ec55e-74f7-11e2-8451-3085a9acc8e0b",
      "first_name": "David",
      "last_name": "Fischer",
      "name": "David Fischer",
      "mail": "d@f.com",
      "admin_platform": true
    },
    "parent": null,
    "uri": "glusterfs://<address>/medias_volume/medias/
            <user_id>/<media_id>",
    "public_uris": null,
    "virtual_filename": "Psy_gangnam_style.flv",
    "metadata": {
      "duration": "00:04:12.16",
      "add_date": "2013-02-11 22:37",
      "title": "Psy - Gangnam Style",
      "size": 183190475
    },
    "status": "READY"
  }
}
```

Allowed Any user

Parameters

- **id** – id of media to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No media with id `id`.
- **415** – Wrong id format `id`.
- **415** – Requires json content-type.

PUT /media/id/ (*id*)

Update a media (only virtual_filename and metadata field can be updated).

Example request:

```
PUT /media/id/a396fe66-74ee-11e2-89ad-3085a9accbb8 HTTP/1.1
Host: somewhere.com
Header: anthony@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{"virtual_filename": "the_fifth_element.mp4"}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The media \"fifth_element.mp4\" has been updated."
}
```

Allowed Only the author of the media

param id media's id

query virtual_filename Media's filename when downloaded or published (optional)

query metadata JSON string containing metadatas about the media (optional)

statuscode 200 OK

statuscode 400 Key `key` not found. *or* on type or value error

statuscode 401 Authenticate.

statuscode 403 Authentication Failed.

statuscode 403 You are not allowed to modify media with id `id`.

statuscode 404 No media with id `id`.

statuscode 415 Wrong id format `id`.

statuscode 415 Requires (valid) json content-type.

PATCH /media/id/ (*id*)

Update a media (only virtual_filename and metadata field can be updated).

Example request:

```
PUT /media/id/a396fe66-74ee-11e2-89ad-3085a9accbb8 HTTP/1.1
Host: somewhere.com
Header: anthony@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{"virtual_filename": "the_fifth_element.mp4"}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The media \"fifth_element.mp4\" has been updated."
}
```

Allowed Only the author of the media
param id media's id
query virtual_filename Media's filename when downloaded or published (optional)
query metadata JSON string containing metadatas about the media (optional)
statuscode 200 OK
statuscode 400 Key `key` not found. *or* on type or value error
statuscode 401 Authenticate.
statuscode 403 Authentication Failed.
statuscode 403 You are not allowed to modify media with id `id`.
statuscode 404 No media with id `id`.
statuscode 415 Wrong id format `id`.
statuscode 415 Requires (valid) json content-type.

DELETE /media/id/ (*id*)

Delete a media.

The media file is removed from the shared storage and media's status is set to DELETED.

Example request:

```
DELETE /media/id/a396fe66-74ee-11e2-89ad-3085a9accbb8 HTTP/1.1
Host: somewhere.com
Header: sandro@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The media \"fifth_element.mp4\" has been deleted."
}
```

Allowed Only the author of the media
param id id of media to delete
statuscode 200 OK
statuscode 400 Cannot delete the media, it is actually in use by transform job with id `id` and status `status`.
statuscode 400 Cannot delete the media, it is actually in use by publish job with id `id` and status `status`.
statuscode 401 Authenticate.
statuscode 403 Authentication Failed.
statuscode 403 You are not allowed to delete media with id `id`.
statuscode 404 No media with id `id`.
statuscode 415 Wrong id format `id`.
statuscode 501 FIXME Delete of external uri not implemented.

GET /user/id/ (*id*)

Return an user serialized to JSON (without `secret` field).

Example request:

```
GET /user/id/c4daa8a6-6be4-11e2-ae91-3085a9accb47 HTTP/1.1
Host: somewhere.com
Header: michael@oscied.org:oscied
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": {
    "_id": "c4daa8a6-6be4-11e2-ae91-3085a9accb47",
    "first_name": "Tabby",
```

```
    "last_name": "Fischer",
    "name": "Tabby Fischer",
    "mail": "tabby@bernex.ch",
    "admin_platform": false
  }
}
```

Allowed Root and any user

Parameters

- **id** – id of user to get

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`.
- **415** – Wrong id format `id`.

PUT /user/id/ (*id*)

Update an user.

User's admin_platform attribute can only be modified by root or any authenticated user with admin_platform attribute set.

Example request:

```
PUT /user/id/8bda488c-6be8-11e2-89b7-3085a9accb47 HTTP/1.1
Host: somewhere.com
Header: loic@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{
  "mail": "laurent.nicolet@comiques.ch",
  "secret": "gnevois_style",
  "admin_platform": true
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The user \"Laurent Nicolet\" has been updated."
}
```

Allowed Root, user with admin_platform attribute set or the user itself

Parameters

- **id** – id of user to get

Query Parameters

- **first_name** – User's first name (optional)
- **last_name** – User's last name (optional)
- **mail** – User's email address (optional)
- **secret** – User's secret (optional)
- **admin_platform** – User's admin_platform (optional)

Status Codes

- **200** – OK
- **400** – on type or value error
- **400** – Key `key` not found.
- **400** – The email address `mail` is already used by another user.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`
- **415** – Wrong id format `id`.
- **415** – Requires (valid) json content-type.

PATCH /user/id/ (*id*)

Update an user.

User's admin_platform attribute can only be modified by root or any authenticated user with admin_platform attribute set.

Example request:

```
PUT /user/id/8bda488c-6be8-11e2-89b7-3085a9accb47 HTTP/1.1
Host: somewhere.com
Header: loic@oscied.org:oscied
Accept: application/json
Content-Type: application/json

{
  "mail": "laurent.nicolet@comiques.ch",
  "secret": "gnevois_style",
  "admin_platform": true
}
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
```

```
  "value": "The user \"Laurent Nicolet\" has been updated."
}
```

Allowed Root, user with admin_platform attribute set or the user itself

Parameters

- **id** – id of user to get

Query Parameters

- **first_name** – User's first name (optional)
- **last_name** – User's last name (optional)
- **mail** – User's email address (optional)
- **secret** – User's secret (optional)
- **admin_platform** – User's admin_platform (optional)

Status Codes

- **200** – OK
- **400** – on type or value error
- **400** – Key `key` not found.
- **400** – The email address `mail` is already used by another user.
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`
- **415** – Wrong id format `id`.
- **415** – Requires (valid) json content-type.

DELETE /user/id/(*id*)

Delete an user.

Example request:

```
DELETE /user/id/8bda488c-6be8-11e2-89b7-3085a9accb47 HTTP/1.1
Host: somewhere.com
Header: laurent.nicolet@comiques.ch:gnevois_style
Accept: application/json
```

Example response:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
  "status": 200,
  "value": "The user \"Laurent Nicolet\" has been deleted."
}
```

Allowed Root, user with admin_platform attribute set or the user itself

Parameters

- **id** – id of user to delete

Status Codes

- **200** – OK
- **401** – Authenticate.
- **403** – Authentication Failed.
- **404** – No user with id `id`.
- **415** – Wrong id format `id`.

6.6 FFmpeg Documentation

6.6.1 Available Codecs

```
Codecs:
 D..... = Decoding supported
 .E.... = Encoding supported
 ..V... = Video codec
 ..A... = Audio codec
 ..S... = Subtitle codec
 ...S.. = Supports draw_horiz_band
 ....D. = Supports direct rendering method 1
 .....T = Supports weird frame truncation
 ------
 D V D  4xm              4X Movie
 D V D  8bps             QuickTime 8BPS video
 D A D  8svx_exp         8SVX exponential
 D A D  8svx_fib         8SVX fibonacci
  EV    a64multi         Multicolor charset for Commodore 64
  EV    a64multi5        Multicolor charset for Commodore 64, extended with 5th color (colram)
 DEA D  aac              Advanced Audio Coding
 D A D  aac_latm         AAC LATM (Advanced Audio Codec LATM syntax)
 D V D  aasc             Autodesk RLE
 DEA D  ac3              ATSC A/52A (AC-3)
  EA    ac3_fixed        ATSC A/52A (AC-3)
 D A D  adpcm_4xm        ADPCM 4X Movie
 DEA D  adpcm_adx        SEGA CRI ADX ADPCM
 D A D  adpcm_ct         ADPCM Creative Technology
 D A D  adpcm_ea         ADPCM Electronic Arts
 D A D  adpcm_ea_maxis_xa ADPCM Electronic Arts Maxis CDROM XA
 D A D  adpcm_ea_r1      ADPCM Electronic Arts R1
 D A D  adpcm_ea_r2      ADPCM Electronic Arts R2
 D A D  adpcm_ea_r3      ADPCM Electronic Arts R3
 D A D  adpcm_ea_xas     ADPCM Electronic Arts XAS
 D A D  adpcm_ima_amv    ADPCM IMA AMV
 D A D  adpcm_ima_apc    ADPCM IMA CRYO APC
 D A D  adpcm_ima_dk3    ADPCM IMA Duck DK3
 D A D  adpcm_ima_dk4    ADPCM IMA Duck DK4
 D A D  adpcm_ima_ea_eacs ADPCM IMA Electronic Arts EACS
 D A D  adpcm_ima_ea_sead ADPCM IMA Electronic Arts SEAD
 D A D  adpcm_ima_iss    ADPCM IMA Funcom ISS
 DEA D  adpcm_ima_qt     ADPCM IMA QuickTime
 D A D  adpcm_ima_smjpeg ADPCM IMA Loki SDL MJPEG
 DEA D  adpcm_ima_wav    ADPCM IMA WAV
 D A D  adpcm_ima_ws     ADPCM IMA Westwood
 DEA D  adpcm_ms         ADPCM Microsoft
 D A D  adpcm_sbpro_2    ADPCM Sound Blaster Pro 2-bit
 D A D  adpcm_sbpro_3    ADPCM Sound Blaster Pro 2.6-bit
 D A D  adpcm_sbpro_4    ADPCM Sound Blaster Pro 4-bit
 DEA D  adpcm_swf        ADPCM Shockwave Flash
 D A D  adpcm_thp        ADPCM Nintendo Gamecube THP
 D A D  adpcm_xa         ADPCM CDROM XA
 DEA D  adpcm_yamaha     ADPCM Yamaha
 DEA D  alac             ALAC (Apple Lossless Audio Codec)
 D A D  als              MPEG-4 Audio Lossless Coding (ALS)
 D A D  amrnb            Adaptive Multi-Rate NarrowBand
 D A D  amrwb            Adaptive Multi-Rate WideBand
 DEV    amv              AMV Video
 D V D  anm              Deluxe Paint Animation
 D V D  ansi             ASCII/ANSI art
 D A D  ape              Monkey's Audio
 DES    ass              Advanced SubStation Alpha subtitle
 DEV D  asv1             ASUS V1
 DEV D  asv2             ASUS V2
 D A D  atrac1           Atrac 1 (Adaptive TRansform Acoustic Coding)
 D A D  atrac3           Atrac 3 (Adaptive TRansform Acoustic Coding 3)
 D V D  aura             Auravision AURA
 D V D  aura2            Auravision Aura 2
 DEV D  avrp             Avid 1:1 10-bit RGB Packer
 D V D  avs              AVS (Audio Video Standard) video
 D V D  bethsoftvid      Bethesda VID video
```

```
 D V D  bfi             Brute Force & Ignorance
 D A D  binkaudio_dct   Bink Audio (DCT)
 D A D  binkaudio_rdft  Bink Audio (RDFT)
 D V    binkvideo       Bink video
 D V D  bintext         Binary text
 DEV D  bmp             BMP image
 D A D  bmv_audio       Discworld II BMV audio
 D V    bmv_video       Discworld II BMV video
 D V D  c93             Interplay C93
 D V D  camstudio       CamStudio
 D V D  camtasia        TechSmith Screen Capture Codec
 D V D  cavs            Chinese AVS video (AVS1-P2, JiZhun profile)
 D V D  cdgraphics      CD Graphics video
 D V D  cinepak         Cinepak
 DEV D  cljr            Cirrus Logic AccuPak
 D A D  cook            COOK
 D V D  cyuv            Creative YUV (CYUV)
 DEA D  dca             DCA (DTS Coherent Acoustics)
 D V D  dfa             Chronomaster DFA
 D V    dirac           BBC Dirac VC-2
 DEV D  dnxhd           VC3/DNxHD
 DEV    dpx             DPX image
 D A D  dsicinaudio     Delphine Software International CIN audio
 D V D  dsicinvideo     Delphine Software International CIN video
 DES    dvbsub          DVB subtitles
 DES    dvdsub          DVD subtitles
 DEV D  dvvideo         DV (Digital Video)
 D V D  dxa             Feeble Files/ScummVM DXA
 D V D  dxtory          Dxtory
 DEA D  eac3            ATSC A/52 E-AC-3
 D V D  eacmv           Electronic Arts CMV video
 D V D  eamad           Electronic Arts Madcow Video
 D V D  eatgq           Electronic Arts TGQ video
 D V    eatgv           Electronic Arts TGV video
 D V D  eatqi           Electronic Arts TQI Video
 D V D  escape124       Escape 124
 D V D  escape130       Escape 130
 DEV D  ffv1            FFmpeg video codec #1
 DEVSD  ffvhuff         Huffyuv FFmpeg variant
 DEA D  flac            FLAC (Free Lossless Audio Codec)
 DEV D  flashsv         Flash Screen Video
 DEV D  flashsv2        Flash Screen Video Version 2
 D V D  flic            Autodesk Animator Flic video
 DEVSD  flv             Flash Video (FLV) / Sorenson Spark / Sorenson H.263
 D V D  fraps           Fraps
 D V D  frwu            Forward Uncompressed
 DEA D  g722            G.722 ADPCM
 DEA    g723_1          G.723.1
 DEA D  g726            G.726 ADPCM
 D A D  g729            G.729
 DEV D  gif             GIF (Graphics Interchange Format)
 D A D  gsm             GSM
 D A D  gsm_ms          GSM Microsoft variant
 DEV D  h261            H.261
 DEVSDT h263            H.263 / H.263-1996
 D VSD  h263i           Intel H.263
  EV    h263p           H.263+ / H.263-1998 / H.263 version 2
 D V D  h264            H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
 D V D  h264_vdpau      H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 (VDPAU acceleration)
 DEVSD  huffyuv         Huffyuv / HuffYUV
 D V D  idcinvideo      id Quake II CIN video
 D V D  idf             iCEDraw text
 D V D  iff_byterun1    IFF ByteRun1
 D V D  iff_ilbm        IFF ILBM
 D A D  imc             IMC (Intel Music Coder)
 D V D  indeo2          Intel Indeo 2
 D V    indeo3          Intel Indeo 3
 D V    indeo4          Intel Indeo Video Interactive 4
 D V    indeo5          Intel Indeo Video Interactive 5
 D A D  interplay_dpcm  DPCM Interplay
 D V D  interplayvideo  Interplay MVE video
 DEV    j2k             JPEG 2000
 DEV D  jpegls          JPEG-LS
```

```
 D V D  jv              Bitmap Brothers JV video
 D V    kgv1            Kega Game Video
 D V D  kmvc            Karl Morton's video codec
 D V D  lagarith        Lagarith lossless
 DEA D  libgsm          libgsm GSM
 DEA D  libgsm_ms       libgsm GSM Microsoft variant
  EA    libmp3lame      libmp3lame MP3 (MPEG audio layer 3)
 DEV D  libopenjpeg     OpenJPEG based JPEG 2000 encoder
 DEV    libschroedinger libschroedinger Dirac 2.2
 DEA D  libspeex        libspeex Speex
  EV    libtheora       libtheora Theora
  EA    libvorbis       libvorbis Vorbis
 DEV    libvpx          libvpx VP8
  EV    libx264         libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
  EV    libx264rgb      libx264 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10 RGB
  EV    ljpeg           Lossless JPEG
 D V D  loco            LOCO
 D A D  mace3           MACE (Macintosh Audio Compression/Expansion) 3:1
 D A D  mace6           MACE (Macintosh Audio Compression/Expansion) 6:1
 D V D  mdec            Sony PlayStation MDEC (Motion DECoder)
 D V D  mimic           Mimic
 DEV D  mjpeg           MJPEG (Motion JPEG)
 D V D  mjpegb          Apple MJPEG-B
 D A D  mlp             MLP (Meridian Lossless Packing)
 D V D  mmvideo         American Laser Games MM Video
 D V D  motionpixels    Motion Pixels video
 D A D  mp1             MP1 (MPEG audio layer 1)
 D A D  mp1float        MP1 (MPEG audio layer 1)
 DEA D  mp2             MP2 (MPEG audio layer 2)
 D A D  mp2float        MP2 (MPEG audio layer 2)
 D A D  mp3             MP3 (MPEG audio layer 3)
 D A D  mp3adu          ADU (Application Data Unit) MP3 (MPEG audio layer 3)
 D A D  mp3adufloat     ADU (Application Data Unit) MP3 (MPEG audio layer 3)
 D A D  mp3float        MP3 (MPEG audio layer 3)
 D A D  mp3on4          MP3onMP4
 D A D  mp3on4float     MP3onMP4
 D A D  mpc7            Musepack SV7
 D A D  mpc8            Musepack SV8
 DEVSDT mpeg1video      MPEG-1 video
 D V DT mpeg1video_vdpau MPEG-1 video (VDPAU acceleration)
 DEVSDT mpeg2video      MPEG-2 video
 DEVSDT mpeg4           MPEG-4 part 2
 D V DT mpeg4_vdpau     MPEG-4 part 2 (VDPAU)
 D VSDT mpegvideo       MPEG-1 video
 D V DT mpegvideo_vdpau MPEG-1/2 video (VDPAU acceleration)
 D VSDT mpegvideo_xvmc  MPEG-1/2 video XvMC (X-Video Motion Compensation)
 DEVSD  msmpeg4         MPEG-4 part 2 Microsoft variant version 3
 D VSD  msmpeg4v1       MPEG-4 part 2 Microsoft variant version 1
 DEVSD  msmpeg4v2       MPEG-4 part 2 Microsoft variant version 2
 D V D  msrle           Microsoft RLE
 DEV D  msvideo1        Microsoft Video-1
 D V D  mszh            LCL (LossLess Codec Library) MSZH
 D V D  mxpeg           Mobotix MxPEG video
 DEA D  nellymoser      Nellymoser Asao
 D V D  nuv             NuppelVideo/RTJPEG
 DEV D  pam             PAM (Portable AnyMap) image
 DEV D  pbm             PBM (Portable BitMap) image
 DEA D  pcm_alaw        PCM A-law
 D A D  pcm_bluray      PCM signed 16|20|24-bit big-endian for Blu-ray media
 D A D  pcm_dvd         PCM signed 20|24-bit big-endian
 DEA D  pcm_f32be       PCM 32-bit floating point big-endian
 DEA D  pcm_f32le       PCM 32-bit floating point little-endian
 DEA D  pcm_f64be       PCM 64-bit floating point big-endian
 DEA D  pcm_f64le       PCM 64-bit floating point little-endian
 D A D  pcm_lxf         PCM signed 20-bit little-endian planar
 DEA D  pcm_mulaw       PCM mu-law
 DEA D  pcm_s16be       PCM signed 16-bit big-endian
 DEA D  pcm_s16le       PCM signed 16-bit little-endian
 D A D  pcm_s16le_planar PCM 16-bit little-endian planar
 DEA D  pcm_s24be       PCM signed 24-bit big-endian
 DEA D  pcm_s24daud     PCM D-Cinema audio signed 24-bit
 DEA D  pcm_s24le       PCM signed 24-bit little-endian
 DEA D  pcm_s32be       PCM signed 32-bit big-endian
```

```
 DEA D  pcm_s32le       PCM signed 32-bit little-endian
 DEA D  pcm_s8          PCM signed 8-bit
 D A D  pcm_s8_planar   PCM signed 8-bit planar
 DEA D  pcm_u16be       PCM unsigned 16-bit big-endian
 DEA D  pcm_u16le       PCM unsigned 16-bit little-endian
 DEA D  pcm_u24be       PCM unsigned 24-bit big-endian
 DEA D  pcm_u24le       PCM unsigned 24-bit little-endian
 DEA D  pcm_u32be       PCM unsigned 32-bit big-endian
 DEA D  pcm_u32le       PCM unsigned 32-bit little-endian
 DEA D  pcm_u8          PCM unsigned 8-bit
 D A D  pcm_zork        PCM Zork
 DEV D  pcx             PC Paintbrush PCX image
 DEV D  pgm             PGM (Portable GrayMap) image
 DEV D  pgmyuv          PGMYUV (Portable GrayMap YUV) image
 D S    pgssub          HDMV Presentation Graphic Stream subtitles
 D V D  pictor          Pictor/PC Paint
 DEV D  png             PNG image
 DEV D  ppm             PPM (Portable PixelMap) image
 DEV D  prores          Apple ProRes
 D V D  prores_lgpl     Apple ProRes (iCodec Pro)
 D V D  ptx             V.Flash PTX image
 D A D  qcelp           QCELP / PureVoice
 D A D  qdm2            QDesign Music Codec 2
 D V D  qdraw           Apple QuickDraw
 D V D  qpeg            Q-team QPEG
 DEV D  qtrle           QuickTime Animation (RLE) video
 DEV D  r10k            AJA Kona 10-bit RGB Codec
 DEV D  r210            Uncompressed RGB 10-bit
 DEV    rawvideo        raw video
 DEA D  real_144        RealAudio 1.0 (14.4K) encoder
 D A D  real_288        RealAudio 2.0 (28.8K)
 D V D  rl2             RL2 video
 DEA D  roq_dpcm        id RoQ DPCM
 DEV D  roqvideo        id RoQ video
 D V D  rpza            QuickTime video (RPZA)
 DEV D  rv10            RealVideo 1.0
 DEV D  rv20            RealVideo 2.0
 D V D  rv30            RealVideo 3.0
 D V D  rv40            RealVideo 4.0
 D A D  s302m           SMPTE 302M
 DEV    sgi             SGI image
 D A D  shorten         Shorten
 D A D  sipr            RealAudio SIPR / ACELP.NET
 D A D  smackaud        Smacker audio
 D V D  smackvid        Smacker video
 D V D  smc             QuickTime Graphics (SMC)
 DEV D  snow            Snow
 D A D  sol_dpcm        DPCM Sol
 DEA D  sonic           Sonic
  EA    sonicls         Sonic lossless
 D V D  sp5x            Sunplus JPEG (SP5X)
 DES    srt             SubRip subtitle
 D V D  sunrast         Sun Rasterfile image
 DEV D  svq1            Sorenson Vector Quantizer 1 / Sorenson Video 1 / SVQ1
 D VSD  svq3            Sorenson Vector Quantizer 3 / Sorenson Video 3 / SVQ3
 DEV D  targa           Truevision Targa image
 D VSD  theora          Theora
 D V D  thp             Nintendo Gamecube THP video
 D V D  tiertexseqvideo Tiertex Limited SEQ video
 DEV D  tiff            TIFF image
 D V D  tmv             8088flex TMV
 D A D  truehd          TrueHD
 D V D  truemotion1     Duck TrueMotion 1.0
 D V D  truemotion2     Duck TrueMotion 2.0
 D A D  truespeech      DSP Group TrueSpeech
 D A D  tta             True Audio (TTA)
 D A D  twinvq          VQF TwinVQ
 D V D  txd             Renderware TXD (TeXture Dictionary) image
 D V D  ultimotion      IBM UltiMotion
 D V D  utvideo         Ut Video
 DEV D  v210            Uncompressed 4:2:2 10-bit
 D V D  v210x           Uncompressed 4:2:2 10-bit
 DEV D  v308            Uncompressed packed 4:4:4
```

```
 DEV D  v410            Uncompressed 4:4:4 10-bit
 D V    vb              Beam Software VB
 D V D  vble            VBLE Lossless Codec
 D V D  vc1             SMPTE VC-1
 D V D  vc1_vdpau       SMPTE VC-1 VDPAU
 D V D  vc1image        Windows Media Video 9 Image v2
 D V D  vcr1            ATI VCR1
 D A D  vmdaudio        Sierra VMD audio
 D V D  vmdvideo        Sierra VMD video
 D V D  vmnc            VMware Screen Codec / VMware Video
 DEA D  vorbis          Vorbis
 D VSD  vp3             On2 VP3
 D V D  vp5             On2 VP5
 D V D  vp6             On2 VP6
 D V D  vp6a            On2 VP6 (Flash version, with alpha channel)
 D V D  vp6f            On2 VP6 (Flash version)
 D V D  vp8             On2 VP8
 D V D  vqavideo        Westwood Studios VQA (Vector Quantized Animation) video
 D A D  wavesynth       Wave synthesis pseudo-codec
 D A D  wavpack         WavPack
 D A    wmalossless     Windows Media Audio 9 Lossless
 D A D  wmapro          Windows Media Audio 9 Professional
 DEA D  wmav1           Windows Media Audio 1
 DEA D  wmav2           Windows Media Audio 2
 D A D  wmavoice        Windows Media Audio Voice
 DEVSD  wmv1            Windows Media Video 7
 DEVSD  wmv2            Windows Media Video 8
 D V D  wmv3            Windows Media Video 9
 D V D  wmv3_vdpau      Windows Media Video 9 VDPAU
 D V D  wmv3image       Windows Media Video 9 Image
 D V D  wnv1            Winnov WNV1
 D A D  ws_snd1         Westwood Audio (SND1)
 D A D  xan_dpcm        DPCM Xan
 D V D  xan_wc3         Wing Commander III / Xan
 D V D  xan_wc4         Wing Commander IV / Xxan
 D V D  xbin            eXtended BINary text
 D V D  xl              Miro VideoXL
 DES    xsub            DivX subtitles (XSUB)
 DEV D  xwd             XWD (X Window Dump) image
 DEV D  y41p            Uncompressed YUV 4:1:1 12-bit
 D V    yop             Psygnosis YOP Video
 DEV D  yuv4            Uncompressed packed 4:2:0
 DEV D  zlib            LCL (LossLess Codec Library) ZLIB
 DEV D  zmbv            Zip Motion Blocks Video

Note, the names of encoders and decoders do not always match, so there are
several cases where the above table shows encoder only or decoder only entries
even though both encoding and decoding are supported. For example, the h263
decoder corresponds to the h263 and h263p encoders, for file formats it is even
worse.
```

6.6.2 Available Filters

```
Filters:
aconvert         A->A       Convert the input audio to sample_fmt:channel_layout:packed_fmt.
aformat          A->A       Convert the input audio to one of the specified formats.
amerge           AA->A      Merge two audio streams into a single multi-channel stream.
anull            A->A       Pass the source unchanged to the output.
aresample        A->A       Resample audio data.
ashowinfo        A->A       Show textual information for each audio frame.
asplit           A->AA      Pass on the audio input to two outputs.
astreamsync      AA->AA     Copy two streams of audio data in a configurable order.
earwax           A->A       Widen the stereo image.
pan              A->A       Remix channels with coefficients (panning).
silencedetect    A->A       Detect silence.
volume           A->A       Change input volume.
abuffer          |->A       Buffer audio frames, and make them accessible to the filterchain.
aevalsrc         |->A       Generate an audio signal generated by an expression.
amovie           |->A       Read audio from a movie source.
anullsrc         |->A       Null audio source, return empty audio frames.
abuffersink      A->|       Buffer audio frames, and make them available to the end of the filter graph.
```

```
anullsink        A->|       Do absolutely nothing with the input audio.
blackframe       V->V       Detect frames that are (almost) black.
boxblur          V->V       Blur the input.
copy             V->V       Copy the input video unchanged to the output.
crop             V->V       Crop the input video to width:height:x:y.
cropdetect       V->V       Auto-detect crop size.
delogo           V->V       Remove logo from input video.
deshake          V->V       Stabilize shaky video.
drawbox          V->V       Draw a colored box on the input video.
drawtext         V->V       Draw text on top of video frames using libfreetype library.
fade             V->V       Fade in/out input video.
fieldorder       V->V       Set the field order.
fifo             V->V       Buffer input images and send them when they are requested.
format           V->V       Convert the input video to one of the specified pixel formats.
frei0r           V->V       Apply a frei0r effect.
gradfun          V->V       Debands video quickly using gradients.
hflip            V->V       Horizontally flip the input video.
hqdn3d           V->V       Apply a High Quality 3D Denoiser.
lut              V->V       Compute and apply a lookup table to the RGB/YUV input video.
lutrgb           V->V       Compute and apply a lookup table to the RGB input video.
lutyuv           V->V       Compute and apply a lookup table to the YUV input video.
mp               V->V       Apply a libmpcodecs filter to the input video.
negate           V->V       Negate input video.
noformat         V->V       Force libavfilter not to use any of the specified pixel formats for the input to t
null             V->V       Pass the source unchanged to the output.
ocv              V->V       Apply transform using libopencv.
overlay          VV->V      Overlay a video source on top of the input.
pad              V->V       Pad input image to width:height[:x:y[:color]] (default x and y: 0, default color:
pixdesctest      V->V       Test pixel format definitions.
scale            V->V       Scale the input video to width:height size and/or convert the image format.
select           V->V       Select frames to pass in output.
setdar           V->V       Set the frame display aspect ratio.
setpts           V->V       Set PTS for the output video frame.
setsar           V->V       Set the pixel sample aspect ratio.
settb            V->V       Set timebase for the output link.
showinfo         V->V       Show textual information for each video frame.
slicify          V->V       Pass the images of input video on to next video filter as multiple slices.
split            V->VV      Pass on the input to two outputs.
swapuv           V->V       Swap U and V components.
thumbnail        V->V       Select the most representative frame in a given sequence of consecutive frames.
tinterlace       V->V       Perform temporal field interlacing.
transpose        V->V       Transpose input video.
unsharp          V->V       Sharpen or blur the input video.
vflip            V->V       Flip the input video vertically.
yadif            V->V       Deinterlace the input image.
cellauto         |->V       Create pattern generated by an elementary cellular automaton.
color            |->V       Provide an uniformly colored input, syntax is: [color[:size[:rate]]].
frei0r_src       |->V       Generate a frei0r source.
life             |->V       Create life.
mandelbrot       |->V       Render a Mandelbrot fractal.
movie            |->V       Read from a movie source.
mptestsrc        |->V       Generate various test pattern.
nullsrc          |->V       Null video source, return unprocessed video frames.
rgbtestsrc       |->V       Generate RGB test pattern.
testsrc          |->V       Generate test pattern.
buffersink       V->|       Buffer video frames, and make them available to the end of the filter graph.
nullsink         V->|       Do absolutely nothing with the input video.
buffer           |->V       Buffer video frames, and make them accessible to the filterchain.
```

6.6.3 Available Formats

```
File formats:
 D. = Demuxing supported
 .E = Muxing supported
 --
  E 3g2             3GP2 format
  E 3gp             3GP format
 D  4xm             4X Technologies format
 D  IFF             IFF format
 D  ISS             Funcom ISS format
 D  MTV             MTV format
```

```
 DE RoQ            raw id RoQ format
  E a64            a64 - video for Commodore 64
 D  aac            raw ADTS AAC
 DE ac3            raw AC-3
 D  act            ACT Voice file format
 D  adf            Artworx Data Format
  E adts           ADTS AAC
 DE adx            CRI ADX
 D  aea            MD STUDIO audio
 DE aiff           Audio IFF
 DE alaw           PCM A-law format
 DE alsa           ALSA audio output
 DE amr            3GPP AMR file format
 D  anm            Deluxe Paint Animation
 D  apc            CRYO APC format
 D  ape            Monkey's Audio
 D  applehttp      Apple HTTP Live Streaming format
 DE asf            ASF format
  E asf_stream     ASF format
 DE ass            Advanced SubStation Alpha subtitle format
 DE au             SUN AU format
 DE avi            AVI format
  E avm2           Flash 9 (AVM2) format
 D  avs            AVS format
 D  bethsoftvid    Bethesda Softworks VID format
 D  bfi            Brute Force & Ignorance
 D  bin            Binary text
 D  bink           Bink
 DE bit            G.729 BIT file format
 D  bmv            Discworld II BMV
 D  c93            Interplay C93
 DE caf            Apple Core Audio Format
 DE cavsvideo      raw Chinese AVS video
 D  cdg            CD Graphics Format
  E crc            CRC testing format
 DE daud           D-Cinema audio format
 D  dfa            Chronomaster DFA
 DE dirac          raw Dirac
 DE dnxhd          raw DNxHD (SMPTE VC-3)
 D  dsicin         Delphine Software International CIN format
 DE dts            raw DTS
 DE dv             DV video format
 D  dv1394         DV1394 A/V grab
  E dvd            MPEG-2 PS format (DVD VOB)
 D  dxa            DXA
 D  ea             Electronic Arts Multimedia Format
 D  ea_cdata       Electronic Arts cdata
 DE eac3           raw E-AC-3
 DE f32be          PCM 32 bit floating-point big-endian format
 DE f32le          PCM 32 bit floating-point little-endian format
 DE f64be          PCM 64 bit floating-point big-endian format
 DE f64le          PCM 64 bit floating-point little-endian format
 D  fbdev          Linux framebuffer
 DE ffm            FFM (FFserver live feed) format
 DE ffmetadata     FFmpeg metadata in text format
 D  film_cpk       Sega FILM/CPK format
 DE filmstrip      Adobe Filmstrip
 DE flac           raw FLAC
 D  flic           FLI/FLC/FLX animation format
 DE flv            FLV format
  E framecrc       framecrc testing format
  E framemd5       Per-frame MD5 testing format
 DE g722           raw G.722
 DE g723_1         raw G.723.1
 D  g729           G.729 raw format demuxer
  E gif            GIF Animation
 D  gsm            raw GSM
 DE gxf            GXF format
 DE h261           raw H.261
 DE h263           raw H.263
 DE h264           raw H.264 video format
 D  ico            Microsoft Windows ICO
 D  idcin          id Cinematic format
```

```
 D  idf             iCE Draw File
 DE image2          image2 sequence
 DE image2pipe      piped image2 sequence
 D  ingenient       raw Ingenient MJPEG
 D  ipmovie         Interplay MVE format
  E ipod            iPod H.264 MP4 format
  E ismv            ISMV/ISMA (Smooth Streaming) format
 D  iv8             A format generated by IndigoVision 8000 video server
 DE ivf             On2 IVF
 D  jack            JACK Audio Connection Kit
 D  jv              Bitmap Brothers JV
 DE latm            LOAS/LATM
 D  lavfi           Libavfilter virtual input device
 D  libcdio
 D  libdc1394       dc1394 A/V grab
 D  lmlm4           lmlm4 raw format
 D  loas            LOAS AudioSyncStream
 D  lxf             VR native stream format (LXF)
 DE m4v             raw MPEG-4 video format
  E matroska        Matroska file format
 D  matroska,webm   Matroska/WebM file format
  E md5             MD5 testing format
 DE microdvd        MicroDVD subtitle format
 DE mjpeg           raw MJPEG video
  E mkvtimestamp_v2 extract pts as timecode v2 format, as defined by mkvtoolnix
 DE mlp             raw MLP
 D  mm              American Laser Games MM format
 DE mmf             Yamaha SMAF
  E mov             MOV format
 D  mov,mp4,m4a,3gp,3g2,mj2 QuickTime/MPEG-4/Motion JPEG 2000 format
  E mp2             MPEG audio layer 2
 DE mp3             MPEG audio layer 3
  E mp4             MP4 format
 D  mpc             Musepack
 D  mpc8            Musepack SV8
 DE mpeg            MPEG-1 System format
  E mpeg1video      raw MPEG-1 video
  E mpeg2video      raw MPEG-2 video
 DE mpegts          MPEG-2 transport stream format
 D  mpegtsraw       MPEG-2 raw transport stream format
 D  mpegvideo       raw MPEG video
  E mpjpeg          MIME multipart JPEG format
 D  msnwctcp        MSN TCP Webcam stream
 DE mulaw           PCM mu-law format
 D  mvi             Motion Pixels MVI format
 DE mxf             Material eXchange Format
  E mxf_d10         Material eXchange Format, D-10 Mapping
 D  mxg             MxPEG clip file format
 D  nc              NC camera feed format
 D  nsv             Nullsoft Streaming Video
  E null            raw null video format
 DE nut             NUT format
 D  nuv             NuppelVideo format
 DE ogg             Ogg
 DE oma             Sony OpenMG audio
 DE oss             Open Sound System playback
 D  pmp             Playstation Portable PMP format
  E psp             PSP MP4 format
 D  psxstr          Sony Playstation STR format
 D  pulse           Pulse audio input
 D  pva             TechnoTrend PVA file and stream format
 D  qcp             QCP format
 D  r3d             REDCODE R3D format
 DE rawvideo        raw video format
  E rcv             VC-1 test bitstream
 D  rl2             RL2 format
 DE rm              RealMedia format
 D  rpl             RPL/ARMovie format
 DE rso             Lego Mindstorms RSO format
 DE rtp             RTP output format
 DE rtsp            RTSP output format
 DE s16be           PCM signed 16 bit big-endian format
 DE s16le           PCM signed 16 bit little-endian format
```

```
 DE s24be           PCM signed 24 bit big-endian format
 DE s24le           PCM signed 24 bit little-endian format
 DE s32be           PCM signed 32 bit big-endian format
 DE s32le           PCM signed 32 bit little-endian format
 DE s8              PCM signed 8 bit format
 DE sap             SAP output format
 D  sbg             SBaGen binaural beats script
  E sdl             SDL output device
 D  sdp             SDP
  E segment         segment muxer
 D  shn             raw Shorten
 D  siff            Beam Software SIFF
 DE smjpeg          Loki SDL MJPEG
 D  smk             Smacker video
 D  sol             Sierra SOL format
 DE sox             SoX native format
 DE spdif           IEC 61937 (used on S/PDIF - IEC958)
 DE srt             SubRip subtitle format
  E svcd            MPEG-2 PS format (VOB)
 DE swf             Flash format
 D  thp             THP
 D  tiertexseq      Tiertex Limited SEQ format
 D  tmv             8088flex TMV
 DE truehd          raw TrueHD
 D  tta             True Audio
 D  tty             Tele-typewriter
 D  txd             Renderware TeXture Dictionary
 DE u16be           PCM unsigned 16 bit big-endian format
 DE u16le           PCM unsigned 16 bit little-endian format
 DE u24be           PCM unsigned 24 bit big-endian format
 DE u24le           PCM unsigned 24 bit little-endian format
 DE u32be           PCM unsigned 32 bit big-endian format
 DE u32le           PCM unsigned 32 bit little-endian format
 DE u8              PCM unsigned 8 bit format
 D  vc1             raw VC-1
 D  vc1test         VC-1 test bitstream format
  E vcd             MPEG-1 System format (VCD)
 D  video4linux2,v4l2 Video4Linux2 device grab
 D  vmd             Sierra VMD format
  E vob             MPEG-2 PS format (VOB)
 DE voc             Creative Voice file format
 D  vqf             Nippon Telegraph and Telephone Corporation (NTT) TwinVQ
 D  w64             Sony Wave64 format
 DE wav             WAV format
 D  wc3movie        Wing Commander III movie format
  E webm            WebM file format
 D  wsaud           Westwood Studios audio format
 D  wsvqa           Westwood Studios VQA format
 DE wtv             Windows Television (WTV)
 D  wv              WavPack
 D  x11grab         X11grab
 D  xa              Maxis XA File Format
 D  xbin            eXtended BINary text (XBIN)
 D  xmv             Microsoft XMV
 D  xwma            Microsoft xWMA
 D  yop             Psygnosis YOP Format
 DE yuv4mpegpipe    YUV4MPEG pipe format
```

O

O

R