

# C.I.R.R.U.S.<sup>1</sup>

## Cloud IaaS Réservoir de Ressources Unifiées pour Salles de TP -----oOo-----

**Jacques Landru ; Tovohérizo Rakotonavalona ; Martine Sion**

Institut Mines-Télécom / Université Lille 1 Sciences et Technologies

TELECOM Lille

Cité scientifique, rue G. Marconi BP 20145 59653 Villeneuve d'Ascq Cedex

### Résumé

*La croissance de notre infra-structure de virtualisation, conséquence de l'inflation du nombre de machines virtuelles, nous incite à nous poser la question du passage à l'échelle. En capitalisant sur notre expérience KVM/libvirt/virt-manager, base de notre architecture de virtualisation actuelle, nous envisageons le déploiement d'une infrastructure de type cloud privé IaaS (Infrastructure as a Service) s'appuyant sur l'environnement libre OpenStack. Adopté sur des déploiements d'envergure, OpenStack est une des références crédibles de l'alternative libre face aux ténors dominants du marché. La pérennité de l'outil semble assurée et repose sur une communauté solide. Fonctionnellement très riche, conçu pour pouvoir passer à l'échelle industrielle, l'environnement est assez complexe à appréhender. Afin d'acquérir les compétences nécessaires avant d'envisager un déploiement sur notre infrastructure de production, nous avons décidé de construire un mini cloud privé IaaS pour nos ressources de salles de TP : **CIRRUS** (Cloud IaaS Réservoir de Ressources Unifiées pour Salles de tp) sur lequel seront mises à disposition à la demande les appliances virtuelles de TP basées sur la plate-forme **VIMINAL** (Virtual Model for Ip Network Architecture Lab). Outre la création de salles virtuelles de TP, l'expérience et les connaissances acquises sur l'environnement OpenStack seront précieuses pour la « cloudification » de notre infrastructure d'exploitation. Elles enrichiront également les contenus des enseignements relatifs aux architectures système. A moyen terme CIRRUS pourrait bénéficier des évolutions de Keystone (service de gestion d'identités d'OpenStack), pour une ouverture sur les fédérations d'identités, afin de constituer un « **MOOL** » (Modern Open Online Lab) offrant un service de type **VaaS** (Viminal as a Service) accessible à une communauté plus large .*

### Mots-clefs

*cloud privé, IaaS, OpenStack, appliances virtuelles, plate-forme de TP, virtualisation, libvirt, nested KVM, clonage de VM, salles de TP virtuelles.*

## 1 Introduction

La maturité des technologies de virtualisation a permis l'industrialisation d'architectures virtuelles à grande échelle, débouchant sur la définition d'offres nouvelles de services à la demande, disponibles sur le nuage global Internet. Affublées du suffixe « **aaS** » (... **as a Service**) ces offres constituent la base des services de la « nébuleuse

---

1. Le **cirrus** est un genre de nuage troposphérique (entre 5 000 et 14 000 mètres d'altitude, dépendant de la latitude et de la saison), formé de cristaux de glace. Ces nuages ont l'apparence de filaments blancs et ne génèrent pas de précipitations. On le compare souvent à des cheveux d'ange. (wikipedia : <http://fr.wikipedia.org/wiki/Cirrus>)

inonuagique » (*cloud computing*). Bien que toutes ces offres de services ne reposent pas exclusivement sur la virtualisation, celle-ci contribue pour une bonne part à l'essor du *cloud*. Parmi ces services, l'**IaaS** (*Infrastructure as a Service*) et dans une moindre mesure le PaaS (*Platform as a Service*) sont bien souvent la transposition dans le *cloud* de la virtualisation. Les offres IaaS des éditeurs permettent de déployer à la demande des machines virtuelles sur lesquelles les clients choisissent d'installer leurs systèmes d'exploitation usuels et leurs environnements de service.

La virtualisation améliore effectivement le taux d'usage des machines d'infrastructure, en consolidant plusieurs serveurs virtuels sur un nombre restreint de serveurs physiques. Cependant les facilités nouvelles qu'elle apporte, couplées à sa banalisation, conduisent à la multiplication des machines virtuelles entraînant une reprise de l'accroissement du nombre d'équipements d'infrastructure. Sur notre infrastructure de production ces équipements sont aujourd'hui au nombre de cinq, avec une augmentation annuelle moyenne de deux équipements.

Nous avons misé dès les premières versions disponibles sur l'environnement de virtualisation libre qemu/KVM [1] (Kernel based Virtual Machine), couplé à la librairie libvirt [2] associée au gestionnaire virt-manager [3]. La librairie libvirt et ses outils (virsh : virtualization shell, virt-install, ...) apporte une couche d'abstraction de virtualisation commune à toute une panoplie d'hyperviseurs et d'environnements de virtualisation (KVM, Xen, LXC, UserMode Linux, Virtuoso, Virtualbox, VMWare, ...). Elle dispose d'un format XML de description des machines virtuelles, sur lequel s'appuient les outils associés (virsh, virt-install, ...) pour piloter ces machines virtuelles. Cette librairie fournit un socle générique de gestion sur lequel est développé tout un ensemble d'outils d'administration de haut niveau (virt-manager, ovirt, abicloud, archipel, ...) qui permettent l'exploitation d'un parc de VM. Virt-manager est une de ces applications de gestion de machines virtuelles développée sur libvirt. Elle offre une interface graphique pour la gestion d'un parc limité d'hyperviseurs et de leurs VM (Virtual Machine) associées.

## 2 Objectifs

La croissance du nombre de machines virtuelles nous amène à nous poser la question du passage à l'échelle. Les cinq hyperviseurs de production supportent, aujourd'hui, une vingtaine de machines virtuelles ; auxquels s'ajoutent trois hyperviseurs d'expérimentation sur lesquels cohabitent une petite dizaine de VM en pré-production ainsi qu'une série croissante de VM de test dont l'usage est sporadique. Virt-manager, commence à montrer ses limites. Adapté au poste de travail, il s'agit d'un outil basique qui ne permet pas la gestion d'un parc important de VM. L'application facilite l'accès à une série d'hyperviseurs mais n'offre pas une vision centralisée globale et homogène du parc. Installé sur plusieurs postes de travail, elle ne nous garantit pas la cohérence nécessaire au partage de l'administration avec les autres membres de l'équipe système. Nous souhaitons donc nous orienter vers une architecture de type *cloud IaaS* (*Infrastructure as a Service*) privé. Les solutions libres commencent à émerger. Parmi celles ci OpenStack [4] a retenu notre attention. Elle nous permet de capitaliser sur notre socle de virtualisation actuel ; KVM/libvirt étant l'un des systèmes d'hypervision supportés. Une communauté s'est constituée autour d'acteurs reconnus (at&t, hp, IBM, rackspace, Ubuntu, Redhat, ...). Adoptée sur des déploiements d'envergure (Cisco WebEx, NASA, CERN, CC-IN2P3, ...) elle est une des références crédibles de l'alternative libre face aux ténors du marché. Le projet de *cloud national* Cloudwatt (Orange, Thales) l'a adopté pour déployer son offre de *cloud* souverain. La pérennité d'OpenStack semble donc reposer sur des bases solides. Fonctionnellement très riche, conçu pour pouvoir passer à l'échelle industrielle, l'outil est assez complexe à appréhender. Afin d'acquérir les compétences nécessaires avant d'envisager un déploiement sur notre infrastructure de production, nous avons décidé de monter un mini *cloud* privé *IaaS* pour nos ressources de salles de TP : **CIRRUS** (**C**loud **I**aas **R**éservoir de **R**essources **U**nifiées pour **S**alles de tp). Dans le cadre de cette expérimentation, dont la taille reste modeste, deux hyperviseurs de notre environnement d'expérimentation et de R&D constitueront le socle d'exécution. Les machines de contrôle des services du *cloud* seront, quant à elles, virtualisées et déportées sur un hyperviseur dédié.

## 3 Socle de base : Eco-système OpenStack

OpenStack [4], s'est construit en agrégeant un ensemble d'outils système de base : cluster d'hyperviseurs (service Nova), gestion des images système (service Glance), gestion du stockage d'objets (service Swift), gestion de l'authentification et des autorisations (service Keystone), gestion de l'infrastructure réseau (service Quantum/Neutron), interface web d'accès et tableau de bord (service Horizon). Cette liste d'outils, non exhaustive, continue de s'enrichir au fur et à mesure de la publication des versions majeures. En phase de développement rapide, OpenStack suit actuellement un

cycle semestriel de publication. C'est donc un projet jeune en dynamique de développement soutenue. Cette agrégation d'outils, d'origines diverses, est unifiée par un bus de communication asynchrone, basé sur un *middleware* en mode message (MOM *Message Oriented Middleware*) (Rabbitmq), couplé à un ensemble d'API assurant l'interopérabilité des différents services. L'environnement est donc modulaire et ouvert. La contrepartie de cette dynamique foisonnante est que la montée d'une version à l'autre s'apparente plus à une migration qu'à une simple mise à jour. Ce qui peut être délicat pour les déploiements en production, compte tenu du rythme de publication des versions majeures. Ainsi depuis le début du projet prospectif CIRRUS nous sommes passés de la version Folsom (2012-2) à la version Grizzly (2013-1) et prochainement Havana (2013-2). Cependant, il faut reconnaître qu'OpenStack dispose d'une documentation de qualité soutenue par une communauté active.

## 4 Appliances virtuelles

Dans le cadre de CIRRUS, une version des *liveDVD* de la collection VIMINAL (VIRtuel Model for Ip Network Architecture Lab) [5] [6] est transposée sous forme d'*appliances* virtuelles pour un usage à la demande. La plate-forme VIMINAL [5] est un environnement autonome de travaux pratiques système et réseau. Disponible sous forme de LiveDVD, elle met à disposition des maquettes système et réseau, constituées de machines virtuelles (VM), sur lesquelles vous pouvez interagir avec des droits étendus. Elle dispose de tous les éléments nécessaires pour effectuer de manière sûre des TP système et réseau IP sur des ordinateurs banalisés sans en altérer la configuration originelle. Elle se décline aujourd'hui sous la forme d'une collection de trois plate-formes de travaux pratiques (MEDI6 : TP IPv6, MOBIDIK : TP Kerberos et VODKA : TP virtualisation) [6]. Chaque position de travail VIMINAL est un bac à sable. L'utilisateur n'a pas d'accès privilégié aux composants du poste de travail sur lequel s'exécute le *liveDVD* (disque, carte réseau, ...). Par contre il dispose de droits étendus (en mode *root*) sur les VM de la maquette réseau du TP. Ainsi pour les TP actuels (MEDI6, MOBIDIK, VODKA), chaque machine de la salle de TP ne dispose d'aucun accès physique au réseau de la salle de TP. Chaque position de travail reste cloisonnée dans son bac à sable. En fin de séance, l'arrêt et le retrait du DVD, laisse le poste de travail dans l'état initial dans lequel il se trouvait avant la séance de TP. Par souci de sécurisation, les postes des salles de TP en accès libre ont le *boot* CD/DVD verrouillé au niveau du BIOS. Aussi les séances de TP de la plate-forme VIMINAL restent aujourd'hui confinées à deux laboratoires de TP réseaux disposant respectivement de 12 et 20 postes de travail. La mise à disposition des *lab* VIMINAL sur l'infrastructure CIRRUS facilitera l'accès aux plate-formes depuis les salles banalisées de TP, voire au delà au travers d'Internet l'ouvrira à une communauté plus large.

## 5 KVM<sup>2</sup> : la virtualisation gigogne

Les maquettes système et réseau des versions 1.2 et supérieures de VIMINAL sont également basées sur l'environnement de virtualisation KVM/libvirt. L'usage de VIMINAL comme *appliance* virtuelle sur des nœuds d'exécution (*compute-node* dans la terminologie OpenStack) nécessite l'exécution de « KVM dans KVM ». Cette fonctionnalité, dénommée « *nested KVM* », permet d'exécuter des environnements avec plusieurs niveaux de virtualisation imbriqués. *Nested KVM*, propage les extensions du jeu d'instructions du processeur dédiées à l'assistance matérielle à la virtualisation (VMX sur Intel, SVM sur AMD) sur les machines virtuelles. Celles-ci peuvent alors, à leur tour, exécuter un hyperviseur qui contrôle ses propres VM de deuxième niveau. Les nœuds d'exécution de l'infrastructure CIRRUS, sont donc configurés en mode *nested* afin que les *appliances* virtuelles VIMINAL puissent exécuter leurs propres VM. Les premiers tests du *liveDVD* VIMINAL sur une machine virtuelle KVM montrent que les performances, de cette double virtualisation, restent acceptables avec les maquettes actuelles (MEDI6, MOBIDIK, VODKA, ... [6]). La gestion de l'affichage reste cependant délicate. L'écran virtuel d'une VM est géré par le protocole RFB/VNC (*Virtual Network Computing*). La virtualisation à deux niveaux entraîne également l'imbrication de deux niveaux d'affichage VNC. A l'usage il est apparu que les performances d'affichage étaient peu convaincantes. L'affichage des écrans des machines virtuelles de deuxième niveau était lent, ce qui se traduit par un rafraîchissement saccadé et une réactivité dégradée.

L'hyperviseur KVM dispose, dans ses versions les plus récentes, d'un périphérique vidéo paravirtualisé nommé QXL, associé à un nouveau protocole de déport d'affichage, dénommé SPICE [7]. QXL/SPICE offre aux machines virtuelles des capacités multimédia (accélération 3D, audio, vidéo, déport de périphériques USB) bien plus avancées que le binôme RFB/VNC. Les flux des cartes video virtuelles QXL, peuvent s'afficher sur des clients d'affichage dédiés

SPICE, aussi bien que sur des clients VNC. Après divers combinaisons de tests, il apparaît qu'en l'état actuel des développements, c'est en démarrant l'*appliance* virtuelle VIMINAL en mode QXL/VNC et les machines virtuelles des maquettes de TP (VM de deuxième niveau) en mode RFB/VNC que l'on obtient le meilleur confort d'affichage. C'est donc ce mode d'affichage, qui sera déporté dans Horizon, (frontal web d'accès d'OpenStack). De plus, le pilote vidéo QXL dispose d'une large palette de résolutions d'écran, il autorise donc un ajustement confortable des écrans des *appliances* VIMINAL.

## 6 « pico-cloud »

Comparativement aux offres *cloud* public, hébergées dans les centres de données (*datacenters*) géants des opérateurs dominants (Amazon AWS, Google, Cloudwatt, OVH, ...) CIRRUS peut être qualifié de « pico-architecture », tant par son étendue que par le sous ensemble de services OpenStack déployés.

Pour cette expérimentation nous disposons deux nœuds d'exécution (*compute-node*), d'un nœud contrôleur OpenStack (services keystone, rabbitmq, nova-\*) et d'un nœud d'affichage (service horizon). De plus, les deux nœuds d'exécution ne sont pas des machines homogènes (2 processeurs AMD sexcoeurs et 32 Giga de RAM pour la première, 2 processeurs bicoeurs et 8 Giga de RAM pour la seconde). En première approximation, ces deux nœuds devraient permettre l'exécution de 5 à 7 positions de travail VIMINAL simultanées. Un test de charge devra confirmer ces capacités. Dans un premier temps il s'agit essentiellement de valider la configuration OpenStack de CIRRUS. La montée en charge viendra ultérieurement en remplaçant les nœuds d'exécution par un ensemble de machines homogènes ou par hybridation *cloud* privé/public.

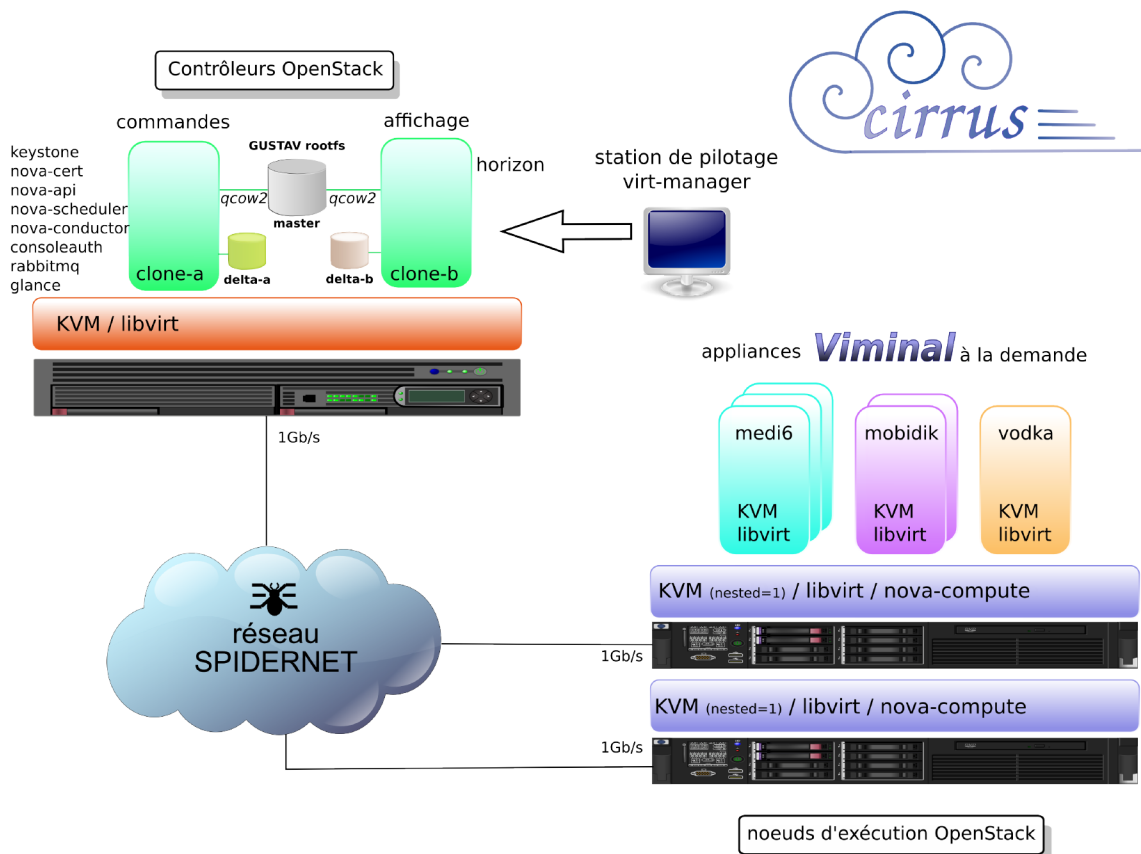


Figure 1 - architecture CIRRUS

Le nœud contrôleur OpenStack, ainsi que le nœud d'affichage sont virtualisés sur un hyperviseur distinct. L'architecture système de ces deux clones est en mode GUSTAV [8] (Gestion Unifiée des Systèmes de fichiers Transposées aux Appareillages Virtuels). GUSTAV isole l'ensemble des éléments spécifiques CIRRUS (fichiers de configuration, base

de données, journaux, ...) du clone contrôleur et du clone d'affichage sur un espace de stockage dédié (disque delta : fichier disque virtuel, distinct du fichier disque système commun nommé disque master). Ce mode de fonctionnement facilite les montées de versions, l'archivage de points de reprise et le retour arrière éventuel lors des mises à jour. Le passage de la version Folsom à la version Grizzly d'OpenStack en a été simplifiée. Il devrait en être de même pour la prochaine montée de version, Havana (2013-2). Le mode GUSTAV, conçu pour gérer des populations de clones virtuels se partageant une base système commune, permet une mise à jour (kernel, logiciel de base, pile de services, ...) en une seule passe de toute la lignée de clones. Cela facilitera la montée en charge de CIRRUS. Les fonctions du contrôleur (Keystone, base de données, services Nova-\*, Glance, *middleware* de messagerie Rabbitmq, ...) sont, à ce jour, centralisées sur un seul contrôleur virtualisé ; elles pourront être réparties sur un ensemble de contrôleurs clonés, sur une base Openstack commune, et différenciés par leur propre disque delta. Cela garantit l'homogénéité de la version d'OpenStack commune à l'ensemble des contrôleurs clonés.

Sur CIRRUS, seul un sous ensemble restreint des services de base d'OpenStack a été déployé (base de données, messagerie asynchrone : Rabbitmq, gestion d'identités : Keystone, services Nova-\*, gestion d'images système : Glance, gestion de l'affichage : Horizon). Les IaaS classiques mettent à disposition une infrastructure de virtualisation socle au déploiement à la demande de machines virtuelles de service distinctes. Même si elles dérivent d'un patron (*template*) commun, chacune de ces machines virtuelles a des caractéristiques qui lui sont propres (CPU, mémoire, OS, cartes réseau, système de fichiers, stockage en mode fichiers ou en mode blocs sur un SAN, ...). CIRRUS diffère de ce mode classique. Toutes les instances virtuelles VIMINAL d'un même TP (MEDI6, MOBIDIK, VODKA) se caractérisent par une machine virtuelle sans disque (boot liveDVD) et sans carte réseau (cf notion de bac à sable au paragraphe 4) disposant de 4 Giga octets de RAM et de deux CPU. Les instances des *appliances* VIMINAL sont donc des clones se partageant un DVD commun, sous forme d'un fichier iso stocké en ramdisk sur le nœud d'exécution (*compute-node*) pour en améliorer les accès. Les services d'OpenStack suivants :

- gestion des accès réseau : service Quantum/Neutron ;
- gestions des objets : service Swift ;
- gestion du stockage en mode blocs : service Cinder ;

ne sont donc pas, à ce jour, déployés sur le contrôleur. Nul doute que la maîtrise de ces services sera nécessaire pour la « cloudification » de notre architecture d'exploitation, mais pour le moment ils ne sont pas nécessaires pour notre expérimentation.

## 7 Perspectives d'évolution

Le frontal d'accès à OpenStack (service Horizon) s'appuie sur les standards du web. Sa « Shibbolethisation » ouvrirait l'accès de CIRRUS à une communauté plus large d'utilisateurs. L'opération n'est probablement pas aisée, compte tenu du fait qu'Horizon délègue l'authentification et les autorisations au sous système Keystone, ce dernier jouant le rôle d'IdM (*Identity Manager*) d'OpenStack. Toutefois, il semble que des travaux aient débutés pour ouvrir Keystone à des mécanismes d'authentifications externes, voire aux fédérations d'identités. A terme CIRRUS pourrait bénéficier de ces évolutions, pour constituer l'infrastructure d'un « MOOL » (**Modern<sup>2</sup> Open Online Lab**) offrant à une communauté plus large un service d'un nouveau type : **VaaS** (**V**iminal **a**s **a** Service). Le MOOL pourrait être au MOOC (Massive Open Online Courses) ce que la séance de travaux pratiques est à l'UV (Unité de Valeur) d'enseignement.

Passée la phase d'expérimentation, il faudra envisager l'extension des capacités de CIRRUS. L'ajout de nœuds d'exécution (*compute-node*) supplémentaires peut s'envisager de deux manières. Une première approche purement privée, consiste à ajouter des machines supplémentaires. Elle engendre des coûts d'investissement, mais offre un contrôle total de l'architecture. De plus ces nouvelles machines dédiées sont mobilisées exclusivement à l'architecture CIRRUS quelque soit le taux d'utilisation, ce qui limite le périmètre de l'extension en cas de forte charge et consomme de l'énergie en période creuse. La seconde consiste à déléguer à un prestataire les pics de charge, approche qualifiée de *cloud* hybride. Il s'agit ponctuellement d'étendre les capacités de CIRRUS en ajoutant des nœuds d'exécution externes loués à un fournisseur de service *cloud* public. Cela permet de limiter les investissements mais engendre des coûts de fonctionnement. Ici aussi deux approches peuvent être envisagées, la location de machines dédiées (*bare-metal*) ou de machines virtuelles :

---

2. Compte tenu de l'étendue de CIRRUS, composé de deux nœuds d'exécution pour 5 à 7 positions de travail simultanées, il paraît difficile d'affubler le qualificatif « massive » à ce service de TP à la demande, le terme « modern » semble plus approprié.

- *bare metal* : location de nœuds dédiés chez un prestataire, on reste sur deux niveaux de virtualisation (KVM<sup>2</sup>), on conserve les avantages de l'extension privée sans la contrainte de l'investissement. Cependant les ressources sont mobilisées même si elles sont peu utilisées et le coût est plus important que la solution virtualisée.
- virtualisée : location à la demande de machines virtuelles pour héberger nœuds d'exécution supplémentaires. Sous réserve que les extensions du jeu d'instructions du processeur dédiées à l'assistance matérielle à la virtualisation (VMX sur Intel, SVM sur AMD) soient propagées sur les machines virtuelles louées. Dans ce cas on empile non plus deux mais trois niveaux de virtualisation (KVM<sup>3</sup>). Il faudra s'assurer que les performances des machines virtuelles de troisième niveau restent acceptables pour effectuer les séances de TP. Ce type de solution offre une flexibilité maximale, les ressources supplémentaires peuvent être mobilisées rapidement à la demande avec en général une facturation à l'usage.

## Bibliographie

- [1] Linux KVM : [http://www.linux-kvm.org/page/Main\\_Page](http://www.linux-kvm.org/page/Main_Page)
- [2] Libvirt : The virtualization API <http://libvirt.org>
- [3] Virt-manager : <http://virt-manager.org/>
- [4] OpenStack, Open source software for building private and public clouds : <http://www.openstack.org/>
- [5] Jacques Landru. VIMINAL Virtual Model for Ip Network Architecture Lab. Dans Actes du congrès JRES2005, Marseille, Décembre 2005. <http://2005.jres.org/paper/49.pdf>
- [6] VIMINAL : MEDI6 lab, MOBIDIK lab, VODKA lab, <http://www.telecom-lille.fr/people/landru/viminal/>
- [7] SPICE : Spice project, <http://spice-space.org/>
- [8] Jacques Landru. GUSTAV Gestion Unifiée des Systèmes de fichiers Transposées aux Appareillages Virtuels. JRES2009, Nantes, Décembre 2009 [https://2009.jres.org/planning\\_files/article/pdf/14.pdf](https://2009.jres.org/planning_files/article/pdf/14.pdf)