Rapport de stage :

Solution de monitoring (Zenoss)

Mathieu Lederman
Réseaux et Télécommunications 2e année

A l'attention de :

– M. Thierry Val, tuteur pédagogique
– M. James Regis, maître de stage

2009/2010
Promotiom 14

Rapport de stage :

Installation et déploiement d'une solution de monitoring (Zenoss)

Mathieu Lederman
Réseaux et Télécommunications 2e année

A l'attention de :

– M. Thierry Val, tuteur pédagogique
– M. James Regis, maître de stage

2009/2010
Promotiom 14

Remerciements

Je tiens à remercier mon maître de stage M. James Regis, ingénieur Système Réseaux, pour m'avoir accordé toute sa confiance ; pour le temps qu'il m'a consacré tout au long de ce stage ; pour avoir répondu à toutes mes questions et pour m'avoir enseigné de nouvelles méthodes de travail.

Je souhaite également remercier M. Morgan Barbier, doctorant dans l'équipe de Cryptologie, pour son aide et ses conseils avisés en matière de gestion du temps de travail ainsi que pour m'avoir intégré dans des activités physiques au sein du campus de l'École.

Je remercie mon tuteur pédagogique, M. Thierry Val, ainsi que les professeurs de l'IUT. Je remercie plus particulièrement Mme. Chantal Labat pour le travail formidable qu'elle réalise en tant que responsable des stages. Je remercie enfin Mme. Danièle Caballero pour le temps qu'elle a consacré à la préparation de la soutenance de stage et pour ses précieux conseils concernant la rédaction de ce rapport.

Table des matières

Résumé

Étudiant en seconde année d'un DUT Réseaux et Télécommunications, je réalise un stage en entreprise au sein du laboratoire d'informatique LIX. L'équipe d'ingénieurs Système Réseaux, à laquelle j'ai été affecté, assure la maintenance du réseau informatique. L'une des idées du service est de déployer un outil visant à superviser l'ensemble du laboratoire.

Mon projet s'articule autour de Zenoss et a pour objectif d'installer et de déployer cette solution de supervision. Le travail qui m'a été confié consiste à évaluer les différentes fonctionnalités et possibilités de ce logiciel.

Pour réaliser ce travail, j'ai analysé la documentation technique « Zenoss Core Networks and System Monitoring » et j'ai mené des recherches auprès de la communauté très active de Zenoss. Ce logiciel apparaît comme étant un outil de supervision efficace, qui propose de nombreuses fonctionnalités au travers d'une interface graphique conviviale. On peut entre autre citer *la découverte automatique de réseau* ou bien encore *la localisation de réseau* avec Google Maps.

La possibilité d'y intégrer des plug-ins et son utilisation à la fois simple et facile font de ce logiciel un outil de supervision apprécié. Chacun est libre de créer ses propres plug-ins pour une personnalisation plus approfondie de la surveillance de ses réseaux. Zenoss constitue une solution fort intéressante et correspond aux attentes de l'équipe.

0.1 Abstract

I am a student in the two-year course in Networks and Telecommunications and I did my professionnal training in LIX computing laboratory. This company particularly deals with network communications and the department in which I worked is responsible for ensuring the management of the lab network.

My project centres on Zenoss software. I was asked to install and configure a monitoring solution for servers with this software. I was in charge of this project to test the effectiveness of this solution.

In order to carry out this work, I had to analyse various documents such as the book « Zenoss Core Network and System Monitoring » and I found much from the very active Zenoss community. This software is an effective monitoring tool which offers many features with a user-friendly graphical interface. It is appreciated for its ability to integrate plug-ins and is both simple and easy to use. Everyone can create their own plug-ins for further customization of network monitoring.

Zenoss is a very interesting solution and matches to the purposes of the team. This work was an enriching experience because it has enabled me to improve my knowledge in networks monitoring and it was beneficial on a professional level in terms of relationships.

Introduction

La formation DUT Réseaux et Télécommunications s'achève par un stage en entreprise en fin de seconde année. Ce stage est nécessaire afin de valider l'obtention du diplôme et a pour objectif de préparer l'étudiant à son insertion dans la vie professionnelle. Il permet d'exploiter et d'approfondir les connaissances apprises tout au long de la formation.

J'effectue mon stage au sein du laboratoire d'informatique LIX, situé au cœur du centre de recherche de l'École polytechnique de Palaiseau. Les principales activités du LIX se regroupent en trois grands domaines : l'algorithmique, les réseaux et les méthodes formelles. Je suis assigné à l'équipe d'ingénieurs Système Réseaux, chargée d'assurer la gestion et la sécurité du parc informatique du laboratoire et qui recherche continuellement des solutions dans le but d'améliorer le réseau. M James Regis, membre de cette équipe, est mon maître de stage.

L'équipe souhaite pouvoir superviser l'intégralité du parc informatique et de toutes ses composantes (serveurs, clients, imprimantes,...). Le but serait de surveiller l'ensemble du réseau de manière rapide et centralisée afin de pouvoir réagir le plus efficacement possible en cas de problèmes. Mon travail consiste donc à installer et déployer une solution de monitoring des serveurs au travers d'un logiciel adapté : Zenoss. Ce projet m'a été confié afin de tester l'efficacité de cette solution et pour ainsi déterminer si un jour elle est susceptible d'être appliquée à l'ensemble du réseau informatique du laboratoire.

En premier lieu, le rapport s'ouvrira sur une présentation détaillée de l'entreprise ainsi que du sujet de stage. Une partie du rapport portera ensuite sur l'installation du logiciel Zenoss avant d'expliquer son utilisation simple dans un réseau local. Le paramètrage de base du logiciel sera traité et il précédera quelques réglages plus complexes permettant une maitrise plus avancée du logiciel.

1 L'entreprise

1.1 Présentation

Le laboratoire d'informatique LIX se situe au cœur du centre de recherche de l'École polytechnique de Palaiseau. Il est composé d'une centaine de membres (environ une dizaine de nationalités différentes), dont une grande majorité de doctorants et une quarantaine de permanents.

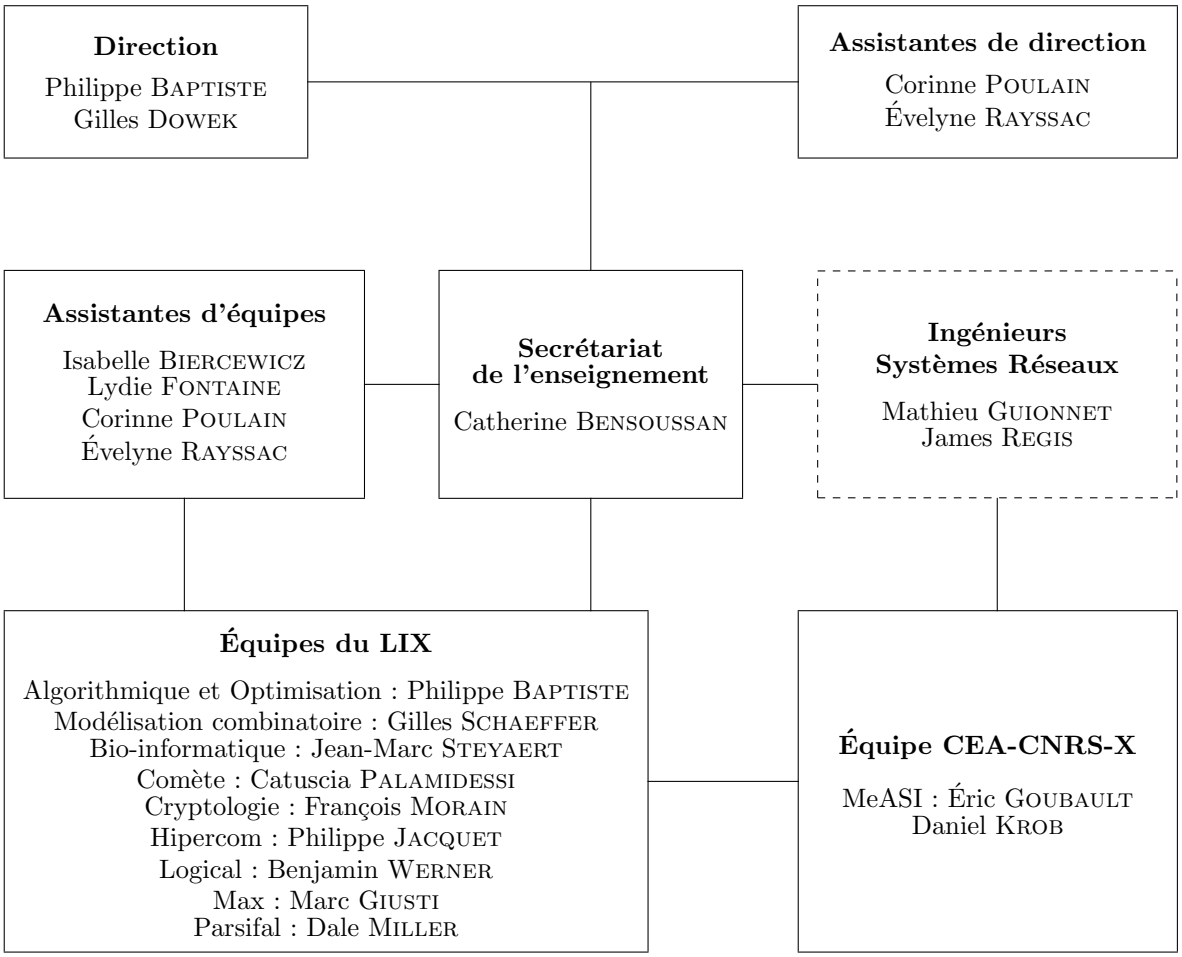


Fig. 1.1 – Organigramme du LIX

1.2 Fonctionnement

Les activités du laboratoire sont axées sur 3 grands domaines de recherche : les réseaux, l'algorithmique et les méthodes formelles. Elles sont réparties sur une dizaine d'équipes :

- **Cryptologie**
- **Modélisation Algébrique (MAX)**
- **Modélisation Combinatoire**
- **Hipercom**
- **Comète**
- **Parsifal**
- **Algorithmique et Optimisation**
- **Bio-informatique**
- **Logical**
- **MeASI**

Le laboratoire est principalement financé par l'École polytechnique, des organismes tels que le Centre national de la recherche scientifique (CNRS) ou l'Institut national de recherche en informatique et en automatique (INRIA), et ses différents partenaires.
Chaque équipe dispose de fonds qui lui sont distribués et qui évoluent en fonction de sa productivité. Cette dernière est principalement déterminée par le nombre et la qualité des articles publiés par l'équipe.

1.3 Missions

Le laboratoire s'intéresse tout particulièrement au domaine des communications en réseau, afin de se doter de compétences globales relatives au traitement du signal, au chiffrement et au routage des communications, à la distribution et à la mobilité des calculs, à la sécurité et à l'ingénierie des protocoles.

Au cœur de ces recherches dont le but est de mettre au point des systèmes de communication fiables et efficaces, l'accent est mis sur les réseaux mobiles qui deviendront une composante incontournable des futurs systèmes embarqués.

Le LIX est fortement impliqué dans les enseignements de l'École polytechnique, et aussi dans plusieurs MASTER de la région parisienne. Il a des relations contractuelles avec des organismes publics (Ministère de l'Enseignement Supérieur et de la Recherche, Direction Générale de l'Armement, Agence Française pour l'Innovation...) ou des organisations internationales.

1.4 Partenariats

Les principaux organismes travaillant en partenariat avec les équipes du laboratoire sont :

- CNRS
- INRIA-FUTURS : *Alien, Comète, Hipercom, Logical, Tanc, Parsifal*
- CEA-LIST : *Équipe de recherche commune MeASI*
- Université Paris Sud : *LRI, LIMSI, IEF*
- SUPELEC : *LSS, Département Signaux Systèmes Electroniques*
- ENS Cachan : *LSV*
- Thales Research and Technology

Au cours de ces dernières années, le laboratoire s'est attaché à développer de nombreuses collaborations avec le monde industriel. On peut citer parmi ses grands partenaires internationaux Microsoft Research, Hitachi Labs ou encore la NASA.

1.5 Le service d'accueil

Le laboratoire nécessite une perpétuelle maintenance et optimisation de son réseau informatique de façon à ne pas entraver les projets menés par les différentes équipes. L'équipe d'ingénieurs Systèmes Réseaux est composée de M. Mathieu Guionnet et de M. James Regis.

Elle assure la maintenance et la sécurité du parc informatique du LIX, tout en faisant face aux divers problèmes informatiques que les membres du laboratoire peuvent être amenés à rencontrer. Le service intervient rapidement en cas de problèmes et c'est pourquoi la grande majorité des opérations est réalisée à distance, permettant ainsi un précieux gain de temps. En parallèle l'équipe recherche continuellement des solutions en vue d'optimiser et de faire evoluer le réseau du laboratoire.

De par ses responsabilités le service entretient d'étroites relations avec les autres équipes et prévient les membres du laboratoire si besoin est. Le service est chargé de l'implémentation, du remplacement ou de la suppression des équipements (imprimantes, serveurs, câbles, disques durs...) constituant le réseau. En règle générale, les postes de travail et les serveurs utilisent des distributions Linux telles que CentOS ou Fedora Core.

1.6 Sujet du stage

Les ingénieurs de l'équipe veulent pouvoir surveiller l'ensemble du réseau informatique du LIX, rapidement et de manière centralisée. Mon projet consiste à installer et déployer une solution de monitoring de serveurs ; autrement dit l'objectif de ce travail est la surveillance d'un réseau local au travers d'un logiciel adapté : Zenoss.

Cette solution dispose de plusieurs fonctionnalités intéressantes et je suis chargé d'exploiter les possibilités du logiciel. Je vais également réaliser un petit tutorial d'installation et de configuration basique de Zenoss. Je pense finaliser ce projet en effectuant une étude comparative entre Zenoss et d'autres solutions open-source comme Nagios ou Munin.

1.7 Outils utilisés

Toutes les stations employées dans le cadre de mon projet sont équipées du système d'exploitation Linux/ CentOS 5.4 et du noyau Linux 2.6.18-164.15.1.el5. J'effectue l'ensemble de mes opérations à distance, me connectant de façon sécurisée via le protocole SSH.

J'ai puisé bon nombre de mes informations dans le livre « Zenoss Core Network and System Monitoring » et je me suis renseigné auprès de la communauté de Zenoss. Egalement je me suis servi de plusieurs tutorials concernant cette solution de monitoring, éparpillés un peu partout sur Internet. J'ai utilisé des plug-ins (disponibles sous formes de packages à installer) récupérables sur le site officiel du logiciel. Finalement j'ai appris les bases du langage LaTeX qui m'ont permis de rédiger ce rapport.

2 Utilisation basique de Zenoss

2.1 Fiches de présentation

2.1.1 Le logiciel Zenoss

Zenoss est un outil de supervision réseau écrit en Python et basé sur un serveur d'applications Zope. Il est distribué sous plusieurs versions dont une qui est open-source : la version Zenoss Core.[1] Zenoss s'appuie sur des technologies telles que MySQL, RRDtool ou encore Net-SNMP. Les principales fonctions de ce logiciel sont l'administration et la supervision du réseau, la gestion des disponibilités, des performances, des évènements, des alertes. La découverte automatique du réseau permet d'établir les relations entre les composants et de tracer la topologie de l'environnement. Les communications entre les clients et le serveur reposent essentiellement sur les protocoles SNMP, Telnet/SSH et WMI (Windows Management Instrumentation) et les données sont enregistrées dans une base de données.

Nom du logiciel	Zenoss
Type de logiciel	Supervision réseau
Licence	GPL / version commerciale
Environnements	Linux, Windows
Version actuelle	2.5.2
Communauté	Zenoss
Développeurs	Zenoss Inc.
Fondation	2006
Langage de développement	Python
Site web	www.zenoss.org

Fig. 2.1 – Tableau d'informations

Zenoss est un système composé de 4 grandes couches principales. Chacune d'entre elles assurent des fonctions bien précises.

- la couche utilisateur
- la couche de données
- la couche de traitement
- la couche de collection

[1] Version utilisée dans le cadre du projet.

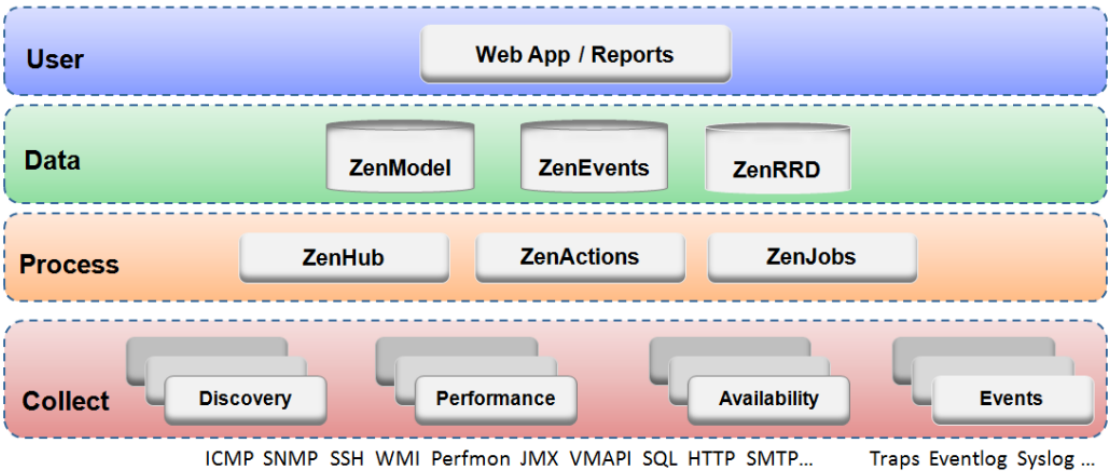


FIG. 2.2 – Architecture du système, *extrait de Zenoss Administration*

La couche utilisateur - Construite autour de l'environnement Zope, elle se manifeste comme un portail web. Elle interagit avec la couche de données et est chargée de convertir les informations pour les afficher dans l'interface utilisateur.

La couche de données - Les informations de configuration et de collection sont stockées au niveau de cette couche, dans 3 bases de données distinctes :

- ZenRRD : stockage des données cycliques, chronologiques et des tracés de graphiques (utilisation de RDDTool)
- ZenModel : configuration de base de Zenoss (composants, groupes, état des services) dans une base Zope
- ZenEvents : stockage des événements (ex : *les syslog, les SNMP trap*) dans une base de données MySQL

La couche de traitement - Elle assure la gestion des communications entre la couche de collection et la couche des données. Elle traite également les actions périodiques ainsi que celles initiées par l'utilisateur (les ZenActions et les ZenJobs).

La couche de collection - Elle est constituée de services qui collectent les données pour ensuite les transmettre à la couche de données. Ces services sont fournis par de nombreux « daemons » qui effectuent des modélisations, de la surveillance et de la gestion d'événements.

2.1.2 Le réseau local

Le projet est réalisé au sein d'un réseau local afin d'avoir une parfaite autonomie et de posséder tous les droits (utilisateur : root) sur les machines supervisées. On va travailler sur 4 postes en particulier :

- *Server-Zenoss* : machine physique supervisée sur laquelle est installé Zenoss.
- *Support-Clients-Zenoss* : machine physique supervisée qui abrite 2 clients virtuels.
- *ClientZenoss1* : machine virtuelle supervisée.
- *ClientZenoss2* : machine virtuelle supervisée.

L'utilisation de l'hyperviseur KVM nous permet d'obtenir des machines virtuelles libres sous Linux, qui fonctionnent sur une architecture x86. *(c.f. Annexes. Installation d'une machine virtuelle)*. Un serveur DHCP est installé sur le réseau, ainsi chaque machine se voit attribuer automatiquement une adresse IP.

Réseau local	192.168.112.0
Server-Zenoss	192.168.112.224
Support-Clients-Zenoss	192.168.112.253
ClientZenoss1	192.168.112.236
ClientZenoss2	192.168.112.129

FIG. 2.3 – Liste des adresses IP

Le raisonnement est le suivant : il faut installer Zenoss sur une machine physique et déployer la solution de monitoring sur l'ensemble des autres machines à superviser. La connexion sur les machines du réseau local se fait à distance via le protocole SSH et pour des raisons de simplicité, tous les outils nécessaires sont récupérés sur Internet à partir d'une machine du laboratoire (nom : *sixfeet*) pour ensuite être copiés sur les postes adéquats.

2.2 Déploiement de la solution

2.2.1 Installation du serveur Zenoss

La station de supervision, appelée aussi *manager*, réalise la lecture d'informations relatives à un agent en l'interrogeant via des requêtes. On peut ainsi visualiser de manière centralisée la configuration, les statistiques des agents supervisés.

On décide d'utiliser une machine physique (que l'on renomme *Server-Zenoss* pour plus de clarté) comme serveur sur lequel sera installé le logiciel. Bien que ce soit sur cette machine que Zenoss va être installé, cela ne change en rien le fait qu'on puisse superviser cette station de travail par la suite.

Sur la machine *sixfeet*, on télécharge le dernier paquetage RPM du logiciel pour Red Hat Enterprise Linux, récupérable depuis `http://www.zenoss.com/download/`. Ensuite on effectue les commandes suivantes :

```
[sixfeet]
  $ scp -r zenoss-2.5.2.el5.x86_64.rpm root@192.168.112.224:/opt
  ## copie du paquetage Zenoss de sixfeet vers Server-Zenoss ##
[Server-Zenoss]
  $ su - root                                        ## mode administrateur ##
  $ yum -y install mysql mysql-server
  $ yum -y install net-snmp net-snmp-utils           ## packages dépendants de Zenoss ##
  $ yum -y install python python-dev
  $ rpm -ivh zenoss-2.5.2.el5.x86_64.rpm             ## installation du paquetage RPM ##
  $ service snmpd start                              ## démarrage du service SNMP ##
  $ service mysqld start                             ## démarrage de MySQL ##
  $ /etc/init.d/zenoss start                         ## lancement de Zenoss ##
```

Désormais Zenoss est installé sur la machine *Server-Zenoss*. On ouvre un navigateur Internet afin de s'assurer que l'installation s'est bien déroulée et on tape l'url suivante dans la barre d'adresses : `http://192.168.112.224:8080`.

```
192.168.112.224     adresse IP de Server-Zenoss
:8080               port d'écoute de Zenoss
```

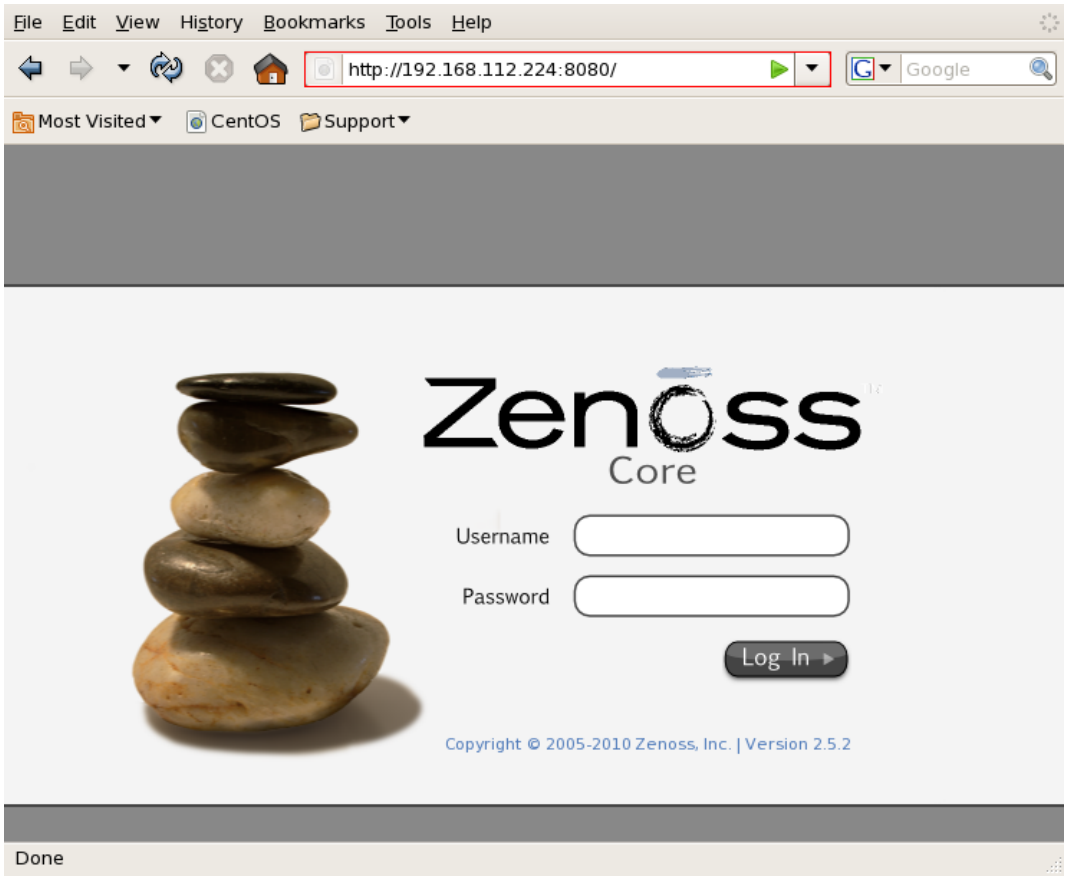


FIG. 2.4 – Fenêtre d'authentification de Zenoss

Par défault lors de l'amorce de la machine, Zenoss ne se lance pas automatiquement et il faut démarrer manuellement les daemons du logiciel. On peut remédier à ce petit détail en indiquant que l'on souhaite que Zenoss s'exécute au démarrage de la machine :

```
[Server-Zenoss]
  $ /sbin/chkconfig zenoss
  $ /sbin/chkconfig --level 345 zenoss on
```

Cependant dans notre cas, cela importe peu puisque les machines sont toujours allumées. Zenoss a besoin d'avoir accès à certains ports dans le but de communiquer avec les équipements qu'on souhaite superviser. Le serveur doit accepter les connexions sur les ports 8080 (HTTP), 514 (syslog) et 22 (SSH) tandis que les équipements supervisés doivent autoriser les ports 161, 162 (SNMP) et 22 (SSH). On doit agir au niveau des iptables pour modifier les règles de filtrage du firewall de chaque machine. On simplifie la situation en arrêtant le service iptables, ce qui implique que tout traffic est autorisé dans le réseau.

```
[nom_de_la_machine]
  $ service iptables stop              ## arrêt du service iptables ##
  ## iptables -L permet de visualiser les règles actives de filtrage ##
```

2.2.2 Mise en place d'un agent SNMP

Afin de pouvoir superviser à distance les équipements du réseau local, on doit installer le protocole SNMP sur chacun de ces équipements appelés alors : agents SNMP. Les agents sont chargés de rester à l'écoute sur le port UDP[2] 161 des éventuelles requêtes envoyées par la station de supervision et d'exécuter les ordres émanant de ces requêtes (ex : *modification des paramètres*). Un agent peut éventuellement émettre des alarmes de sa propre initiative et il utilise pour cela le port UDP 162.

On va paramétrer l'agent SNMP de chacune des 4 machines de notre réseau. La procédure à suivre est appliquée pour une machine et il faut la répéter pour chaque équipement que l'on souhaite superviser dans le réseau local.

```
[nom_de_la_machine]
  $ yum -y install net-snmp*          ## installation des packages SNMP ##
  $ vim /etc/snmp/snmpd.conf          ## modification du fichier de conf. SNMP ##

      ## sécurisation des noms de communautés ##
      ##        sec.name   source             community ##
      com2sec   Local      localhost          test
      com2sec   Reseau     192.168.112.0/24   lix

      ## déclaration des groupes d'accès aux objets de la MIB ##
      ##        group.name   sec.model   sec.name ##
      group     RWGroup      v1          Local
      group     RWGroup      v2c         Local
      group     SecGroup     v1          Reseau
      group     SecGroup     v2c         Reseau
```

[2]Communications en temps réel, plus judicieux d'utiliser le mode non-connecté UDP.

```
## création des vues autorisées aux groupes ##
##        view.name    incl/excl  subtree   mask(optional) ##
view      systemview   included   .1
view      all          included   .1        80

## accès aux vues en fonction des groupes ##
##      group.name  context  sec.model  sec.level  prefix  read  write  notif ##
access  RWGroup     ""       any        noauth     exact   all   all    none
access  SecGroup    ""       any        noauth     exact   all   all    none

## Informations - contacter l'admin ##
syslocation Labo LIX (Aile 0) - Bureau 1012, Ecole polytechnique
syscontact Mathieu L. <lederman@lix.polytechnique.fr>

trapcommunity  lix
trapsink       192.168.112.0
```

Le fichier de configuration modifié ci-dessus, permet à la communauté **lix** d'écrire ou de lire l'intégralité de la MIB de l'agent depuis n'importe quel poste du réseau local (192.168.112.0). Quant à la commuanuté **test**, elle a les mêmes droits mais uniquement depuis le poste local (localhost).

Chaque machine ayant la copie conforme de ce fichier, elles sont toutes dans la communauté **lix** et peuvent communiquer entre elles via le protocole SNMP. Pour cela il suffit juste de démarrer le service (sauf sur le poste *Server-Zenoss* puisqu'il y est déjà lancé).

```
[nom_de_la_machine]
  $ service snmpd start                          ## démarrage du service SNMP ##
```

L'installation des agents, leur configuration et la communication avec les différents équipements de la communauté se vérifient rapidement.

```
[nom_de_la_machine]
  $ service snmpd status                  ## affiche le statut du service SNMP ##
  snmpd (pid  23806) is running...
```

Syntaxe de la commande snmpwalk :

```
$ snmpwalk -v version -c communauté adresse_ip_de_l'agent chemin_dans_la_MIB
```

Attention : Il faut bien faire correspondre le nom de communauté et l'adresse IP. Par exemple si on indique **test** comme nom de communauté, alors l'adresse IP de l'agent sera forcément 127.0.0.1 (ou localhost). En revanche si l'on indique **lix** comme nom de communauté, alors on placera l'adresse IP (192.168.112.xxx) de l'agent qu'on souhaite interroger.

Cette commande permet de vérifier que les agents SNMP sont capables d'émettre et de répondre à des requêtes. Dans le cas où l'on souhaiterait savoir comment contacter l'administrateur système du poste *Server-Zenoss* à partir du poste *ClientZenoss2*, on exécute la commande suivante :

```
[ClientZenoss2]
  $ snmpwalk -v 2c -c lix 192.168.112.224 .1.3.6.1.2.1.1.4
  SNMPv2-MIB::sysContact.0 = STRING: Mathieu L. <lederman@lix.polytechnique.fr>
```

Avec le logiciel Wireshark, on peut notamment analyser les différentes trames qui passent sur le réseau local. La capture d'écran suivante affiche les trames SNMP de la requête émise précédemment à partir du poste *ClientZenoss2* et la réponse à cette requête.

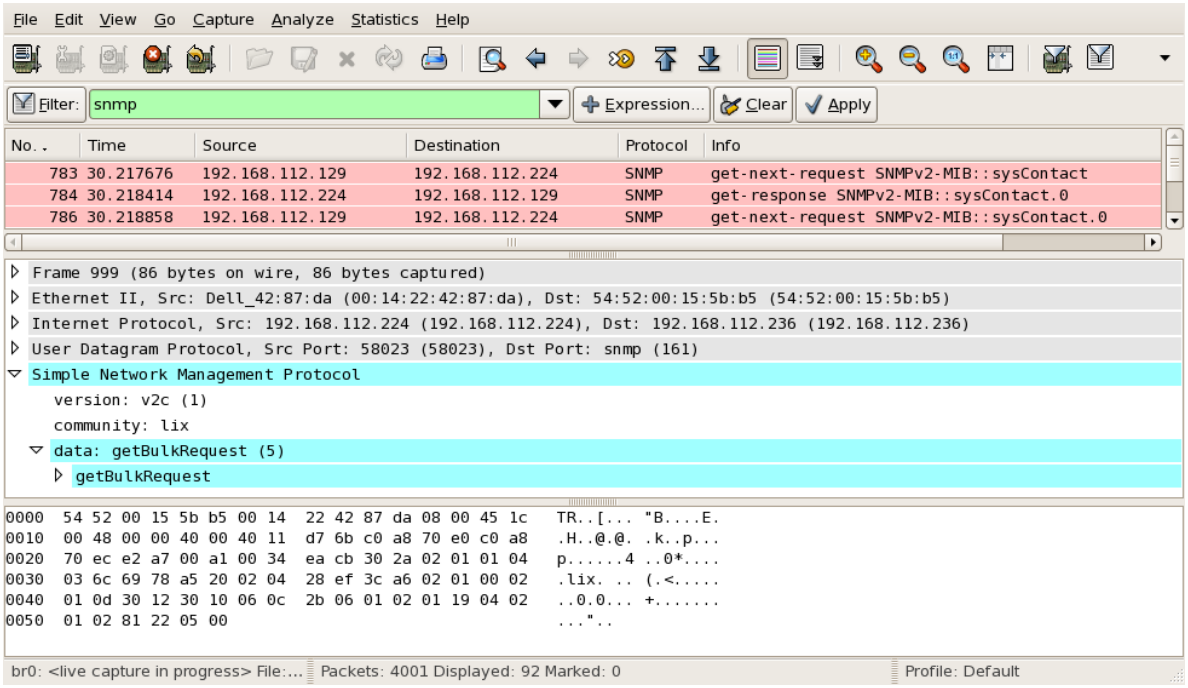


Fig. 2.5 – Capture d'écran du traffic SNMP

2.3 Configuration classique

Des identifiants sont utilisés par défaut pour s'authentifer, ce qui fait donc que n'importe quelle personne un minimum renseignée est capable de se connecter sur Zenoss. Pour des raisons de sécurité, on change donc ces identifiants afin d'en être le seul détenteur.

2.3.1 Concepts de base

Zenoss utilise des dossiers de regroupement afin de classer hiérarchiquement les équipements du réseau. Effectivement le logiciel nous aide à établir des relations entre les divers éléments qui composent notre réseau afin d'en faciliter la supervision. On peut classer un équipement de différentes façons : classes, localisation géographique, systèmes, groupes, réseaux, rapports...

Chaque élément supervisé hérite de la configuration de son dossier parent, qui hérite lui-même de la configuration de son dossier parent, et ainsi de suite. Il suffit donc de modifier la configuration d'une classe pour que tout ce qui se trouve dans cette classe en soit affecté. Si l'on souhaite qu'un élément (ou une sous-classe) ait une configuration

personnalisée alors il suffit de se déplacer dans l'arborescence jusqu'à cet élément (ou cette sous-classe) et de configurer selon les besoins.

```
Arborescence standard traduisant l'organisation des classes

        Devices
              Discovered
              KVM
              Network
                    Router
                          Cisco
                          Firewall
                          RSM
                          TerminalServer
                    Switch
              Ping
              Power
                    UPS
                          APC
              Printer
                    InkJet
                    Laser
              Server
                    Cmd
                    Darwin
                    Linux
                    Remote
                    Scan
                    Solaris
```

L'arborescence ci-dessus représente l'organisation basique des classes de Zenoss ; elle est bien évidemment modifiable selon les besoins de l'utilisateur. On est libres de définir l'organisation de notre propre environnement et on peut pour cela ajouter, modifier, déplacer ou supprimer une classe, une sous-classe ou un équipement.

Exemples concrets :

Toutes les machines placées dans **Server/Linux** héritent donc par défaut de la configuration de cette sous-classe.

On peut très bien déplacer la classe **Printer**, et donc par la même occasion toutes ses sous-classes, dans la classe **Network**.

2.3.2 Equipements supervisés

Zenoss permet de superviser tout type d'équipements : station de travail, serveur, imprimante, routeur, switch, firewall... ainsi que toutes leurs composantes.

Il y a deux méthodes pour ajouter un équipement supervisé. La première méthode consiste à ajouter manuellement l'équipement en définissant ses paramètres. La seconde méthode, plus simple, est de lancer une découverte automatique du réseau, ce qui a pour objectif de recenser tous les équipements trouvés sur le réseau.

Ajout manuel d'une station supervisée

Dans un premier temps on va ajouter manuellement la machine *Server-Zenoss*. Pour cela on va dans le menu **Management/Add Device** dans lequel on doit rentrer certains paramètres concernant la machine que l'on va ajouter.

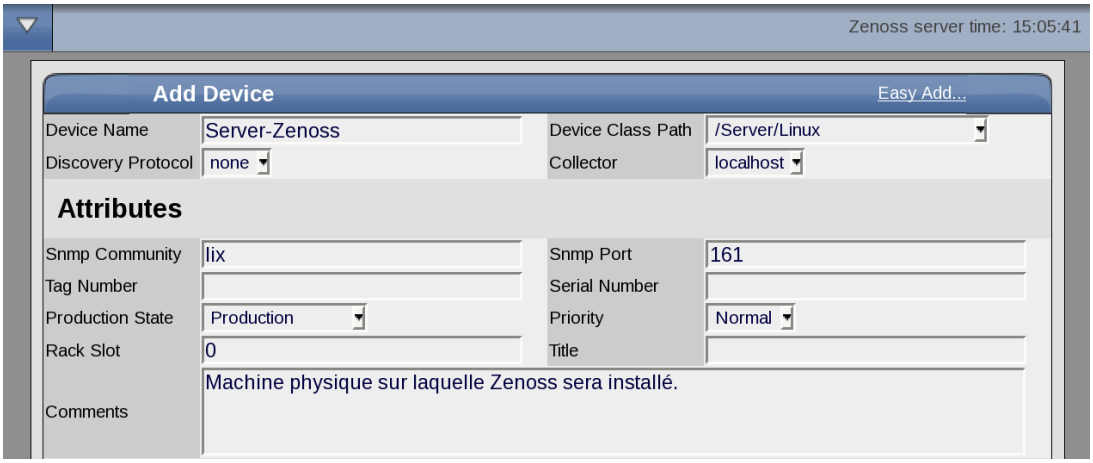


FIG. 2.6 – Ajout d'une machine supervisée

Paramètres importants :

Device Name : nom que portera la machine dans Zenoss et grâce auquel on pourra la reconnaître. Chaque machine doit porter un nom unique. Notre choix : Server-Zenoss

Device Class Path : classe dans laquelle sera placée la machine. On peut placer plusieurs machines dans une même classe. Notre choix : /Server/Linux

Discovery Protocol : paramètre concernant la découverte de la machine. Les options **auto** (découverte automatique de la machine) et **none** sont disponibles. Notre choix : la seconde option

Snmp community : nom de la communauté SNMP dans laquelle la machine sera intégrée. Notre choix : lix

Snmp port : port d'écoute du protocole SNMP sur la machine. Notre choix : 161

On valide la modification de ces paramètres, ce qui aura pour conséquence d'ajouter la machine *Server-Zenoss* dans la classe **/server/Linux**. Le protocole SNMP pour cette machine est paramétré de telle sorte à ce que la station fasse partie de la communauté **lix** et qu'elle écoute sur le port 161. Toutes ces informations correspondent bien avec celles établies un peu plus tôt *(c.f. Mise en place d'un agent SNMP)*.

Découverte automatique du réseau

Zenoss est également doté d'une fonctionnalité très intéressante, qui permet de découvrir automatiquement tous les équipements possédant une adresse IP au sein du réseau local.

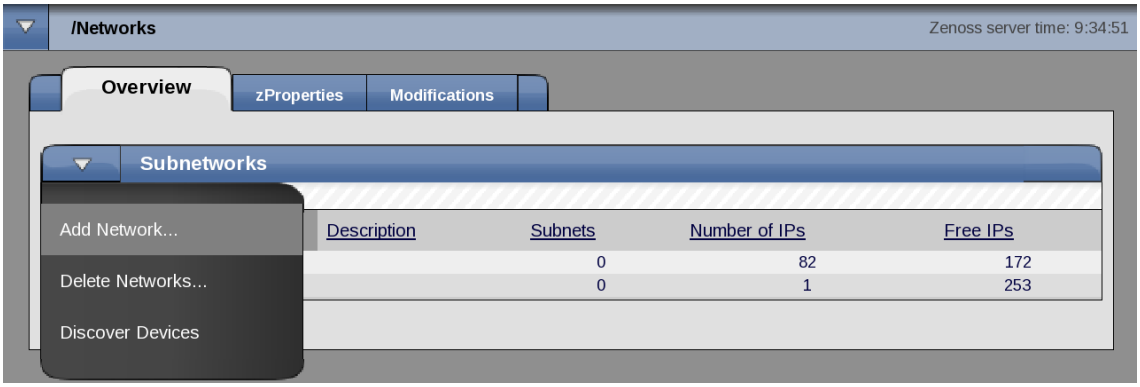


FIG. 2.7 – Ajout d'un réseau

Il faut aller dans le menu **Browse By/Networks** afin d'ajouter un nouveau réseau. On entre alors l'adresse et le masque de sous-réseau de l'environnement concerné. Dans notre cas, il faut rentrer 192.168.112.0/24. Ensuite on coche le réseau que l'on vient tout juste d'ajouter et on lance la découverte automatique des équipements du réseau en cliquant sur *Discover Devices*.

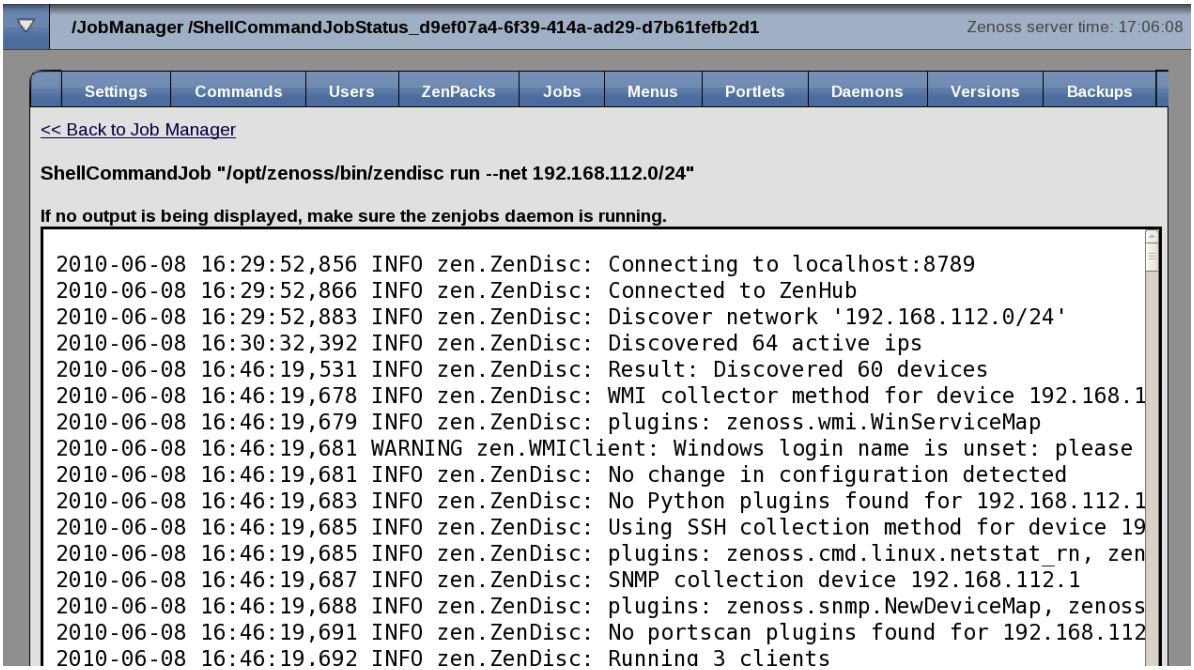


FIG. 2.8 – Découverte automatique des stations actives sur le réseau

La durée de l'opération est variable (entre 5 et 25 minutes parfois) en fonction de l'importance du nombre d'équipements et de la disponibilité du réseau. Zenoss se base sur la commande ping pour déterminer les adresses IP actives. Cette opération est parfois longue puisque Zenoss teste une par une toutes les adresses du réseau. Il utilise la commande : `/opt/zenoss/bin/zendisc run --net 192.168.112.0/24`.

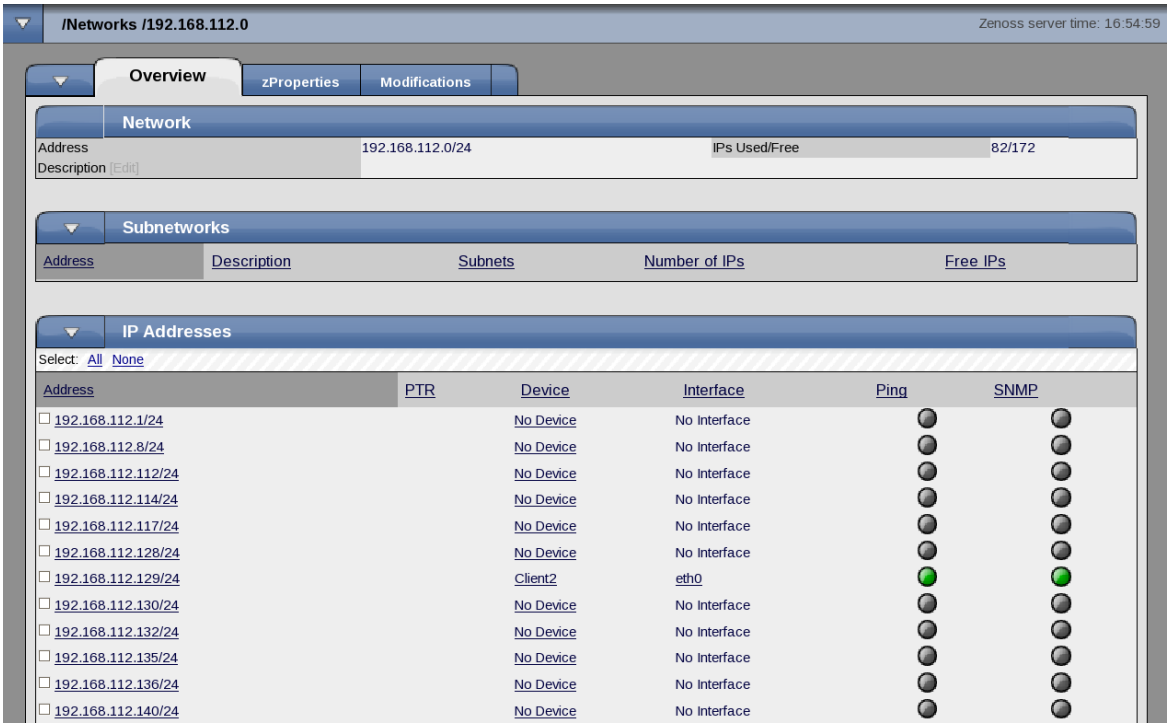


Fig. 2.9 – Début de liste des stations découvertes automatiquement

Les machines découvertes par cette méthode sont alors placées dans la classe **Discovered**. On peut toutefois les déplacer dans une classe plus appropriée. C'est exactement ce que l'on fait pour les machines *Support-Client-Zenoss*, *ClientZenoss1* et *ClientZenoss2* que l'on déplace dans la classe **/Server/Linux** où figure déjà la station *Server-Zenoss*.

De la même façon que l'on a modifié les paramètres importants de *Server-Zenoss*, on modifie ceux des 3 autres machines. Pour cela on clique soit sur leur nom ou soit sur leur adresse IP et on va dans le menu d'édition afin de procéder aux changements.

2.3.3 Regrouper des éléments supervisés

En plus de se ranger dans des classes, les équipements peuvent être triés selon leur :
Localisation - permet de classer les équipements supervisés en fonction de l'endroit où ils sont situés géographiquement.
Système et Groupe - permet de regrouper un ensemble d'équipements pouvant provenir de classes différentes.
Réseau - permet de classer les équipements supervisés en fonction du réseau auquel ils appartiennent.

On peut afficher les éléments en fonction de la méthode avec laquelle ils sont rangés. Ainsi l'affichage s'en trouve d'autant plus compréhensible et cela permet une supervision plus rapide du réseau. En effet, il est par exemple plus facile et plus aisé de surveiller un réseau de machines plutôt que de les superviser une par une.

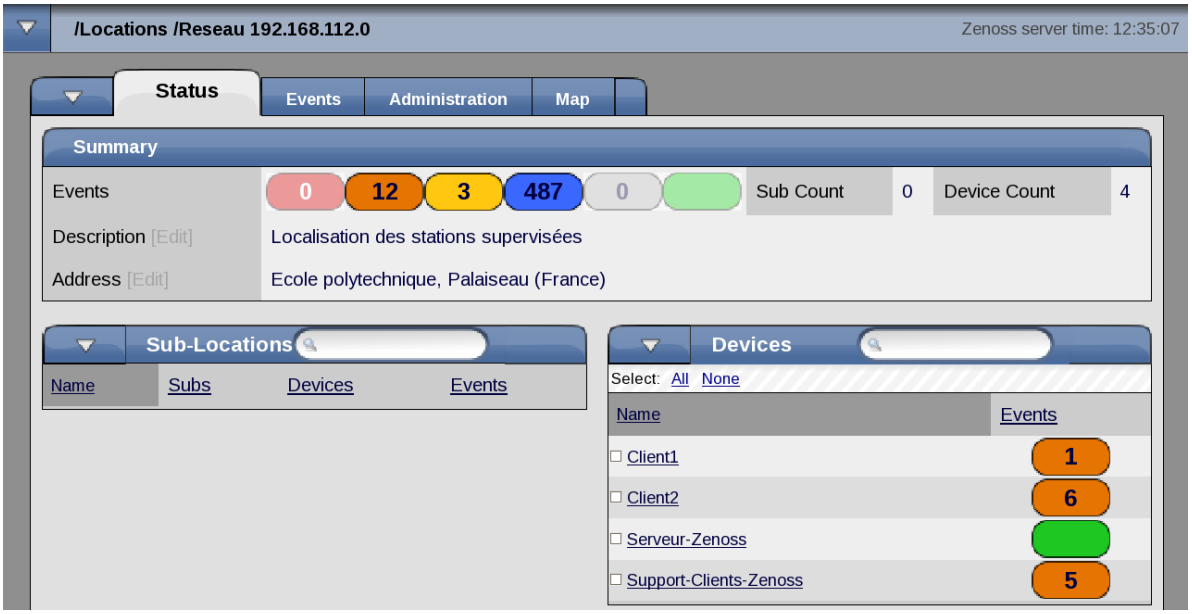


FIG. 2.10 – Classification par localisation

On décide de regrouper les 4 machines supervisées en utilisant les méthodes citées précédemment. De la même façon que l'on a ajouté un groupe réseau, on crée donc un groupe pour la localisation géographique et un groupe normal. Afin de simplifier les choses on donne le même nom à tous les groupes : 192.168.112.0 (l'adresse IP du réseau). Pour chaque groupe on peut choisir de donner une brève description expliquant par exemple le rôle du groupe ou le type de machines qu'il rassemble, et concernant plus particulièrement le groupe pour localiser géographiquement les stations supervisées, il faut éditer le champs **Address** et lui en fournir une traçable par satellite. On indique l'adresse suivante pour le groupe 192.168.112.0 : Ecole polytechnique, Palaiseau (France). Il faut maintenant intégrer les machines dans ces groupes et pour cela on retourne compléter certains paramètres de chaque machine.

Location Path : localisation de la machine. Notre choix : /192.168.112.0

Systems : choix du groupe système dans lequel sera intégrée la machine. Notre choix : /192.168.112.0

Groups : choix du groupe normal dans lequel sera intégrée la machine. Notre choix : /192.168.112.0

Pour résumer, les stations sont désormais intégrées dans les différents groupes ce qui va faciliter leur supervision.

2.3.4 Localisation par Google Maps

Maintenant que l'on a entré une adresse identifiable par satellite dans le groupe de localisation, il faut mettre en place l'application Google Maps dans Zenoss. On récupère pour cela une clé API Google Maps en se rendant sur `http://www.google.com/apis/google` et en suivant les instructions. Une fois la clé obtenue, on copie sa valeur dans le champs **Google Maps API Key** situé tout en bas du menu **Management/Settings** et on valide les modifications. En retournant sur le tableau de bord principal de Zenoss, on remarque qu'une carte géographique est apparue[3]. Elle indique l'emplacement de nos stations ainsi que leur status.

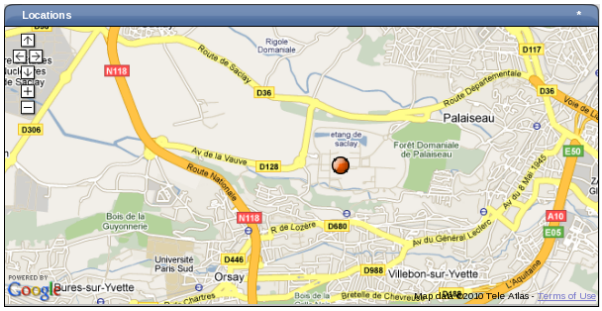


FIG. 2.11 – Carte régionale

FIG. 2.12 – Carte mondiale

On peut s'amuser à modifier la localisation de certaines de nos machines. On indique par exemple que la station *Support-Clients-Zenoss* se trouve actuellement à Sydney en Australie, que la station *ClientZenoss1* est à Buenos Aires en Argentine ou bien encore que la station *ClientZenoss2* est à Naples en Italie. La carte géographique se réactualise pour afficher la nouvelle position des stations.

FIG. 2.13 – Modification de la position des stations

[3]Si ce n'est pas le cas, il faut alors cliquer sur le bouton **Add Portlet** et choisir Google Maps.

2.3.5 Topologie du réseau

Un autre point fort de Zenoss est qu'il établit des relations entre les différents équipements actifs du réseau. Il peut par conséquent dessiner l'architecture du réseau à partir des éléments qu'il supervise. Il faut aller dans le menu **Main Views/Network Map**, indiquer le nom du réseau que l'on souhaite modéliser et valider.

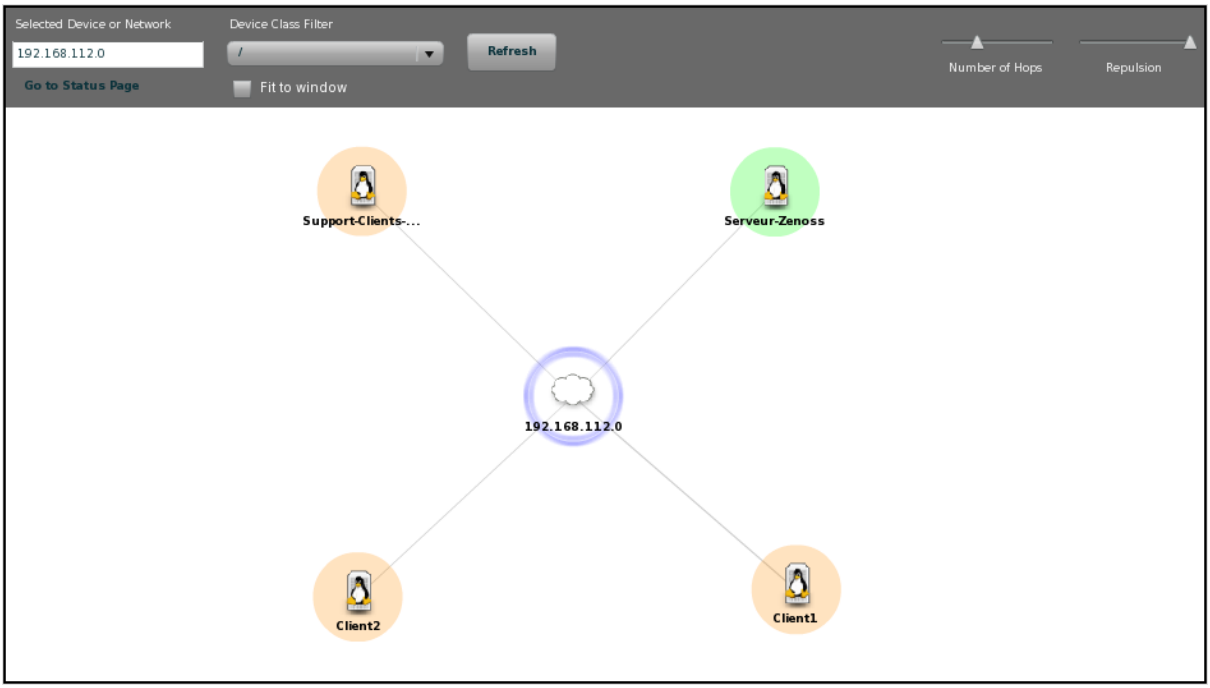


FIG. 2.14 – Topologie du réseau

2.4 Machine supervisée

2.4.1 Etat de la machine

Lorsque l'on clique sur le nom ou l'adresse IP d'une machine, on arrive sur une page qui affiche une vue d'ensemble de la machine supervisée. Cette page fournit de nombreuses informations concernant la machine et son état.

Device Status :

Device : nom de la machine.
IP : son adresse IP.
Status : statut de la machine.
Availibility : sa disponibilité.
Uptime : moment où le système a été allumé.
State : son état.

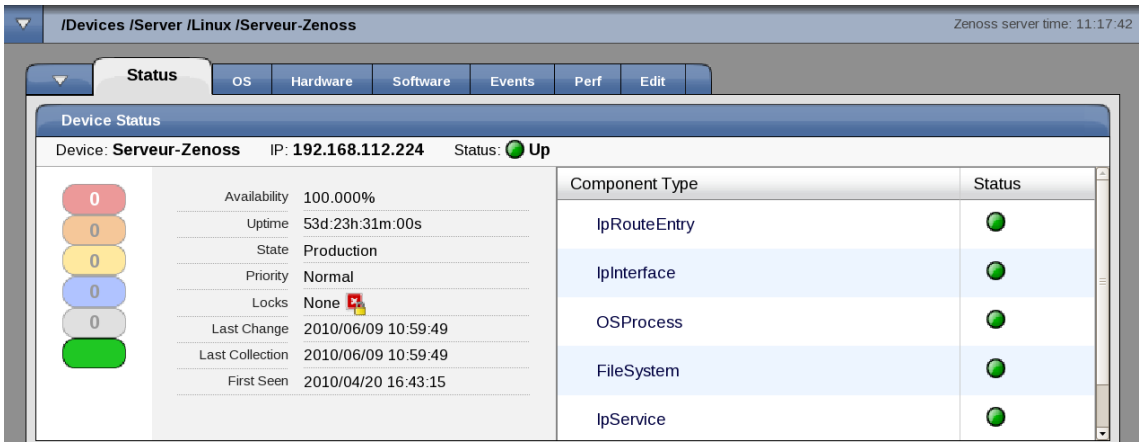


FIG. 2.15 – Statut de la machine

Priority : priorité de supersivion de la machine.
Locks : verrouillage ou non de la machine.
Last Change : date et heure des derniers changements effectués sur la machine.
Last Collection : date et heure de la dernière modélisation.
First seen : date et heure de la première modélisation.
Component Type : nom et statut des composants de la machine.

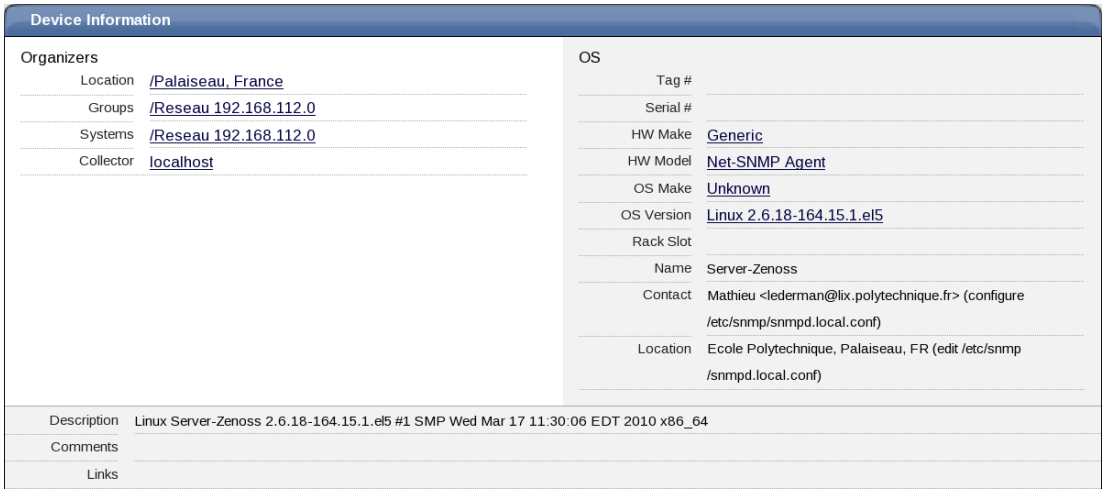


FIG. 2.16 – Informations sur la machine

Device Information :

Organizers : différents groupes dont la machine fait partie.
OS : données concernant la structure de la machine.

Dans la vue d'ensemble d'une machine supervisée est également listé le nombre d'événements relatifs à cette machine. Ils sont rangés par ordre d'importance :

Couleur	Degré d'importance
Rouge	Critique
Orange	Erreur
Jaune	Avertissement
Bleu	Information
Vert	Débogage

2.4.2 Administration

Zenoss analyse automatiquement toutes les machines présentes dans notre inventaire et remodélise celles où des modifications ont été apporté. A propos du verrouillage de la configuration d'une machine, plusieurs choix sont disponibles. Des événements peuvent être envoyés lorsqu'une action est bloquée par un verrou. La station peut être verrouillée contre la suppression et les mises à jour de sa configuration, seulement contre la suppression de sa configuration ou alors tout simplement être déverrouillée.

On peut aussi renommer une station ou bien réinitialiser son adresse IP. Il peut arriver que l'adresse IP d'une station change auquel cas on doit indiquer la nouvelle adresse. Enfin il est possible de forcer les changements de la configuration d'une station (de telle sorte à ce qu'ils prennent effet immédiat sans besoin d'attendre une prochaine modélisation de la station) ou encore de supprimer la station.

2.4.3 Modélisation

Le logiciel modélise les machines via les protocoles SNMP, SSH, telnet ou par scan des ports. Il rassemble des informations grâce à des plug-ins collecteurs. Chaque classe possède par défaut une série de plug-ins collecteurs utilisés par Zenoss pour modéliser les machines assignées à la classe en question. Il est possible d'ajouter ou de supprimer des plug-ins collecteurs, pour une machine ou pour une classe.

par protocole SNMP

Par défaut Zenoss utilise le protocole SNMP pour superviser les machines. On a vérifié à partir d'un terminal que la mise en place de ce protocole s'est bien déroulée, on peut également s'en assurer au travers du logiciel. Pour tester le protocole sur une machine supervisée, on doit aller sur la vue d'ensemble de cette machine et aller dans le menu **Run Commands/snmpwalk**. Une nouvelle fenêtre apparaît et on peut voir le résultat de la commande.

Attention : Dans Zenoss on vérifie que les machines ont le nom de communauté **lix** (comme dans le fichier de configuration de l'agent SNMP) et qu'elles écoutent sur le port 161. Il faut s'assurer également que les iptables ne bloquent pas le traffic SNMP.

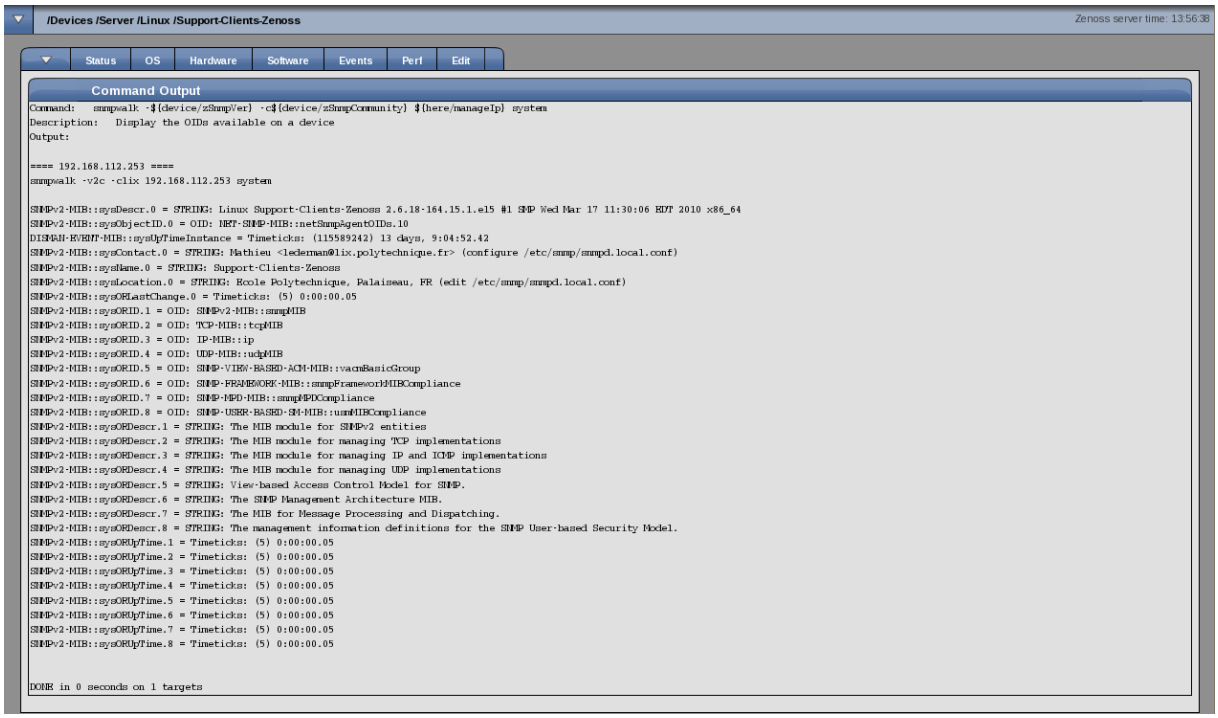


FIG. 2.17 – Commande snmpwalk

par protocole SSH

Si un équipement ne supporte pas le protocole SNMP ou si l'on souhaite superviser une machine derrière un firewall, le protocole SSH fournit une alternative au protocole SNMP. Cependant il nécessite plusieurs conditions pour fonctionner : les machines supervisées doivent être équipées de Zenoss Plug-ins et d'un serveur SSH installé. Elles doivent également être sous une plateforme Linux, Darwin ou FreeBSD.

Pour faciliter la supervision de nos stations, Zenoss nous fournit une classe **/Server/Cmd** déjà configurée avec les plug-ins nécessaires pour superviser via le protocole SSH. Il faut déplacer dans cette classe les machines que l'on souhaite superviser via SSH. Dans notre cas nous n'avons aucun système Mac OS X à superviser donc on peut très bien supprimer tous les plug-ins contenant le mot « darwin ».

Model Device

Zenoss modélise automatiquement et périodiquement (toutes les 6 heures) chaque station de notre inventaire. Cependant on peut forcer manuellement le logiciel à modéliser une station. A partir de la vue d'ensemble de la machine en question il faut aller dans **Manage/Model Device**.

Scan des ports

Parfois la seule méthode que l'on a pour modéliser une machine est d'en scanner les ports. Elle a pour objectif d'essayer de deviner quels services sont actifs sur la machine en se connectant à plusieurs ports. Le scan des ports peut émettre des alertes de sécurité sur le réseau. Zenoss crée une classe **/Server/Scan** dans laquelle il faut placer les machines que l'on souhaite superviser de cette manière.

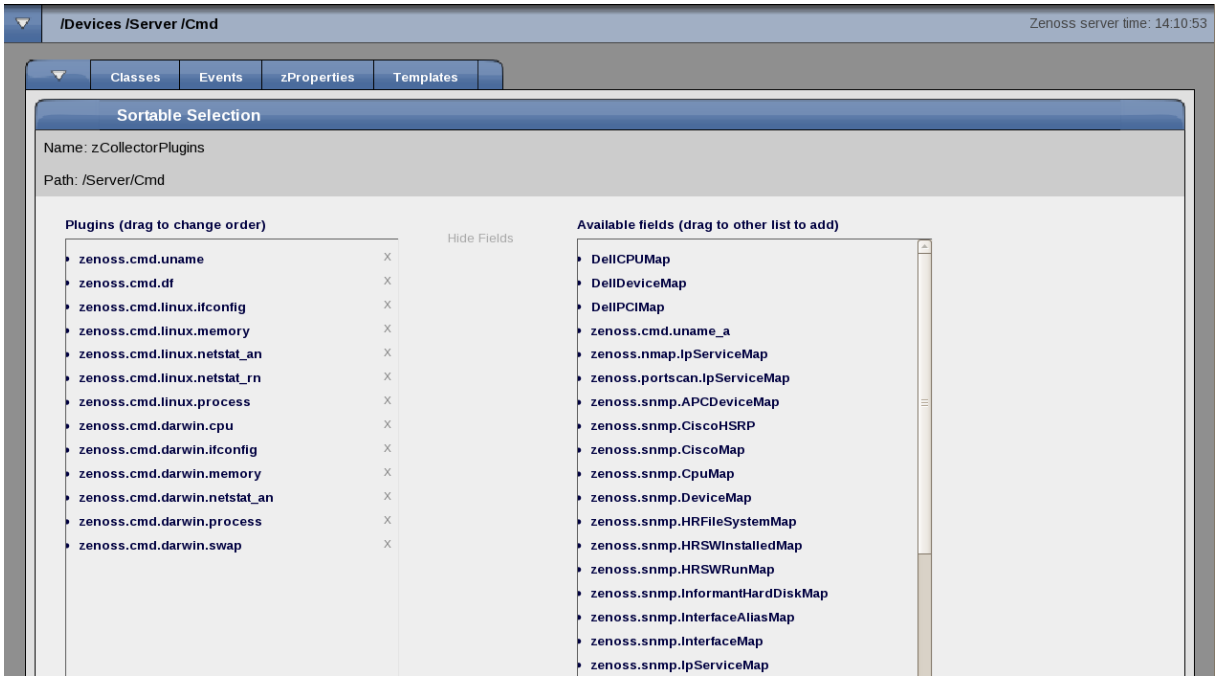


FIG. 2.18 – Liste des plug-ins collecteurs d'informations

3 Supervision plus approfondie

3.1 Composants d'une station

Zenoss permet de surveiller l'ensemble des équipements d'un réseau informatique ainsi que leurs composants. On peut superviser le statut de l'équipement, ses interfaces réseau, ses services actifs, ses routes, ses points de montage, son OS, ses événements en fonction de nos besoins.

3.1.1 Table du système d'exploitation

Lorsqu'on sélectionne une station depuis l'inventaire, la table relative au système d'exploitation de cette machine devient disponible. Les composants qui y sont listés dépendent de la machine, du protocole de modélisation et des plug-ins collecteurs utilisés pour la machine. Ce n'est pas parce que Zenoss a découvert une composante qu'il doit obligatoirement la superviser. On peut décider d'ajouter, de supprimer ou bien de garder une composante sans pour autant la superviser. Encore une fois cela dépend des envies et des besoins de l'utilisateur.

Interfaces

Une partie de la modélisation d'une machine consiste à en découvrir les interfaces réseau actives et ensuite Zenoss commence automatiquement à les superviser.

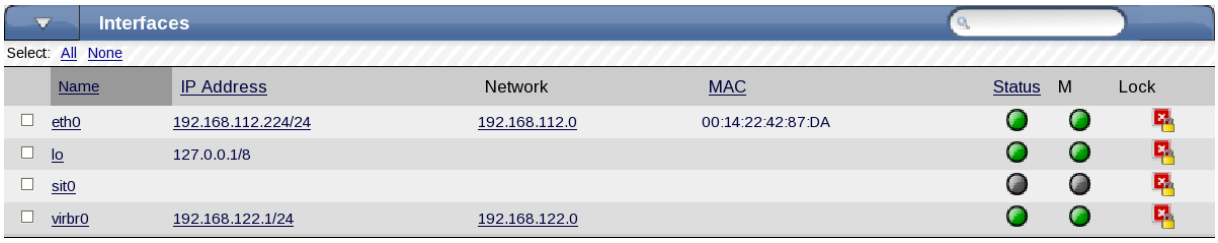


FIG. 3.1 – Liste des interfaces réseau de *Server-Zenoss*

Name : nom de l'interface réseau.
IP Address : son adresse IP.
Network : brin réseau dans lequel elle est située.

MAC : son adresse MAC associée.
Status : statut opérationnel, disponibilité de l'interface.
M : statut administratif, supervision ou non de l'interface.

On peut très facilement vérifier l'exactitude de ces informations en comparant les données collectées par Zenoss et le résultat de la commande `ipconfig /all` dans un terminal sur la machine *Server-Zenoss*.

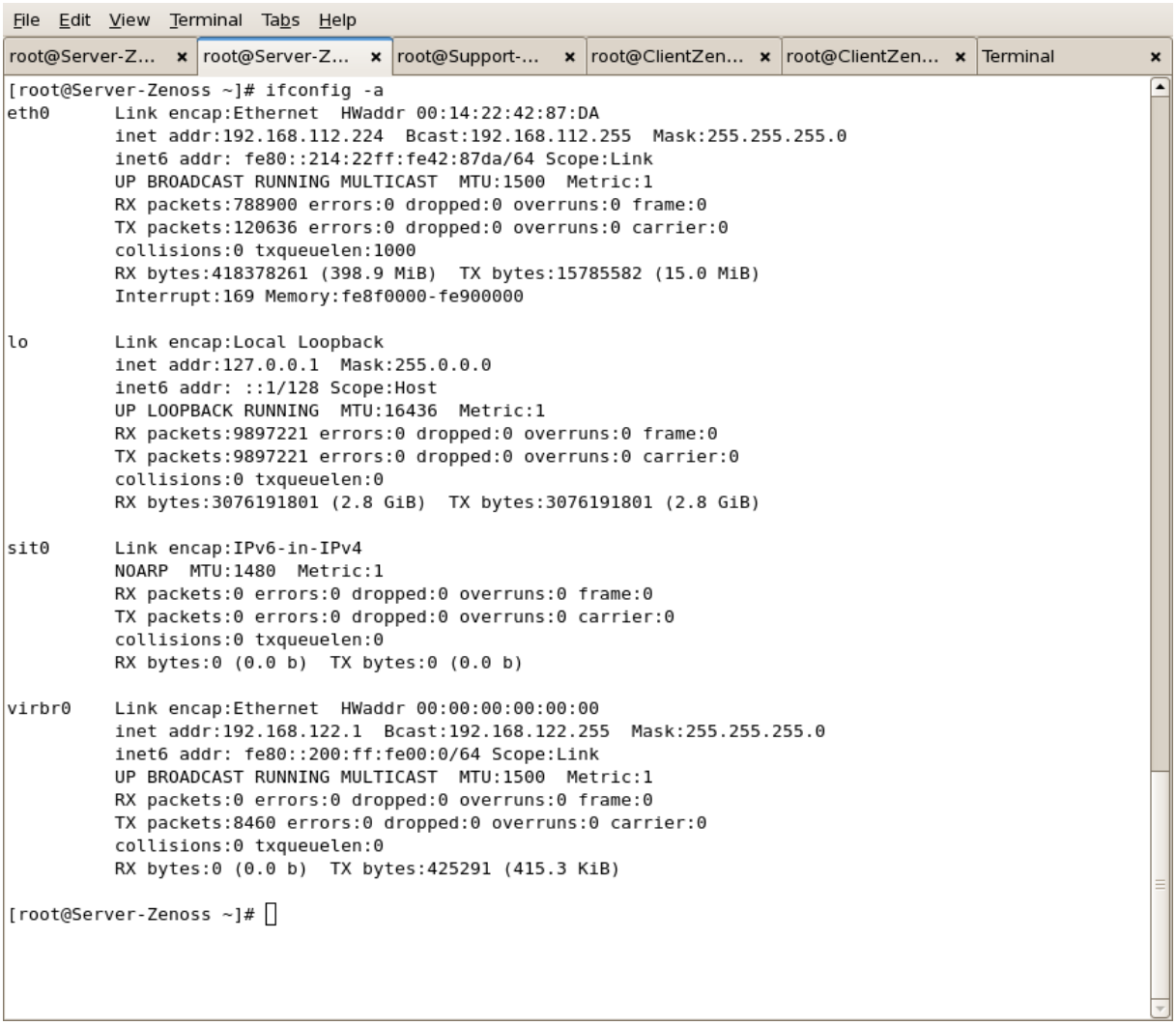


FIG. 3.2 – Résultat de la commande ifconfig

La liste des interfaces réseau découvertes par Zenoss correspond parfaitement à celle des interfaces réseau trouvées sur la machine.

Processus

Le logiciel ne semble pas capable de rassembler des données en ce qui concerne certains processus et en particulier ceux qui ont une faible durée de vie, qui évoluent à des

intervalles de temps irréguliers. Pour Zenoss chaque application en cours d'exécution est représentée par un processus.

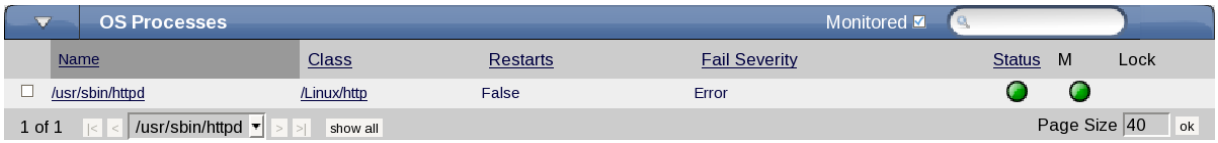


FIG. 3.3 – Liste des processus supervisés sur *Server-Zenoss*

Name : nom du processus.
Class : classe dans laquelle est rangé le processus.
Restarts : émission ou non d'un événement lors du redémarrage du processus.
Fail Severity : degré d'importance de l'événement si redémarrage.
Status : statut opérationnel, disponibilité du processus.
M : statut administratif, supervision ou non du processus.

Services

Les services sont différents des processus et Zenoss nous fournit une liste de ceux disponibles et pouvant être supervisés sur les machines. Les services se lancent automatiquement lors du démarrage du système d'exploitation et ils ne nécessitent souvent aucun contrôle de la part des utilisateurs. Les services sont rangés dans des dossiers prédéfinis mais on peut personnaliser l'arborescence comme bon nous semble.

FIG. 3.4 – Liste des services supervisés sur *Server-Zenoss*

Name : nom du service.
Proto : protocole utilisé par le service.
Port : port d'écoute.
Description : rapide description du service.
Status : statut opérationnel, disponibilité du service.
M : statut administratif, supervision ou non du service.

Fichiers systèmes

Egalement Zenoss modélise la hiérarchie des fichiers systèmes et nous fournit des données très intéressantes sur le volume, sur l'espace de chaque point de montage. On est libres de

définir soi même un point de montage : on le paramètre, Zenoss se charge de le détecter automatiquement et lors de la prochaine modélisation de la station il sera ajouté dans la liste et modélisé.

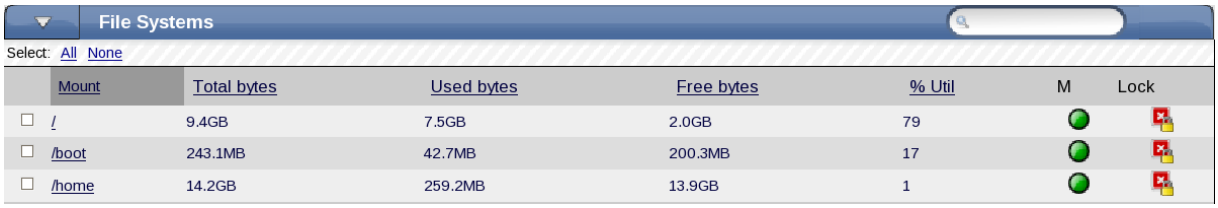


FIG. 3.5 – Liste des points de montage supervisés sur *Server-Zenoss*

Mount : point de montage du fichier système.
Total Bytes : espace total disponible.
Used Bytes : espace utilisé.
Free Bytes : espace non utilisé.
% Util : pourcentage de l'espace utilisé.
M : statut administratif, supervision ou non du point de montage.

Routes

La communication entre les réseaux informatiques se fait au travers de routes. Pour chaque station Zenoss découvre automatiquement la table de routage qui lui est associée.

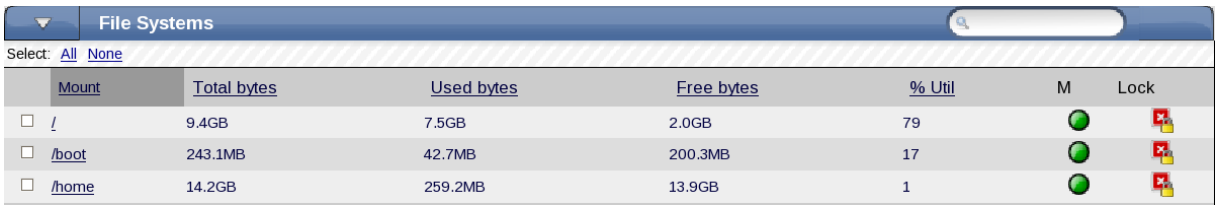


FIG. 3.6 – Table de routage de *Server-Zenoss*

Comme pour les interfaces, il est possible de vérifier la correspondance des données entre celles fournies par Zenoss et celles obtenues grâce à la commande `routes -n` lancée dans un terminal de la station supervisée.

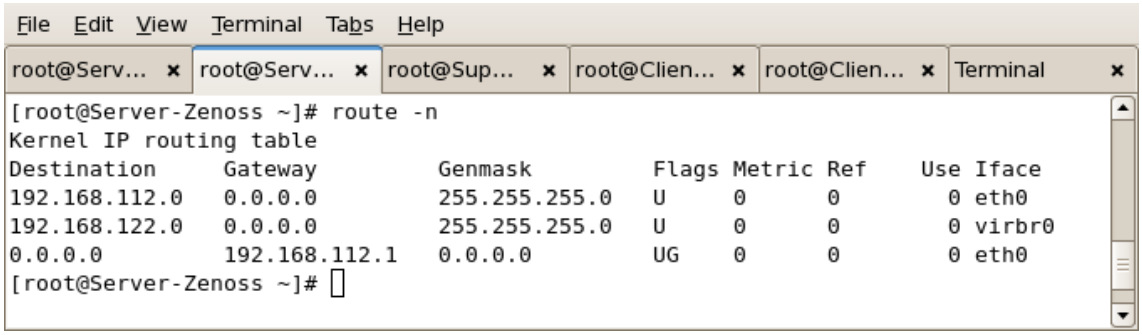


FIG. 3.7 – Résultat de la commande route -n

On constate que la liste des routes découvertes par Zenoss est identique à la liste des routes affichée par la commande.

3.1.2 Graphes de performances

En utilisant RRDTool Zenoss créé des graphes de performance des équipements supervisés. Il relève continuellement les données à des intervalles de temps réguliers. En se rendant dans la rubrique **Perf** d'une machine, on accède à une nouvelle page sur laquelle sont affichés plusieurs graphes pour la charge de la machine, l'utilisation de la CPU et l'utilisation de la mémoire.

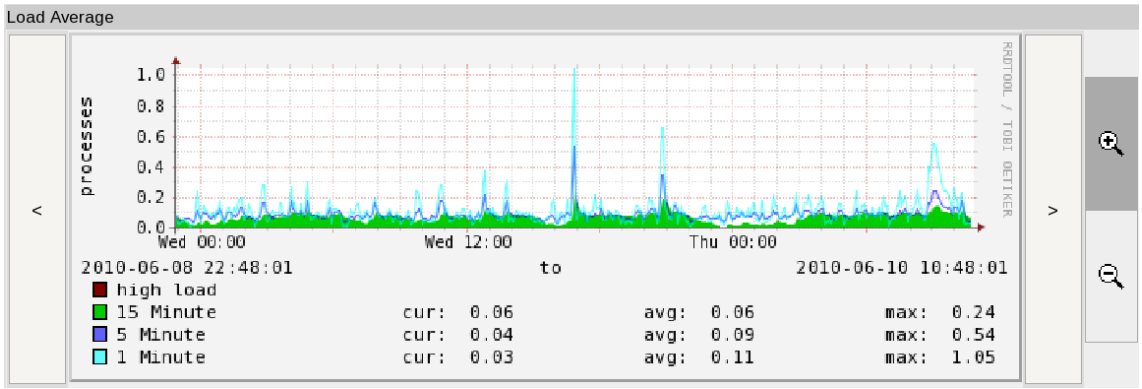


FIG. 3.8 – Charge moyenne de *Support-Clients-Zenoss*

Ce graphe représente la charge moyenne de la station *Support-Client-Zenoss* sur les dernières heures.

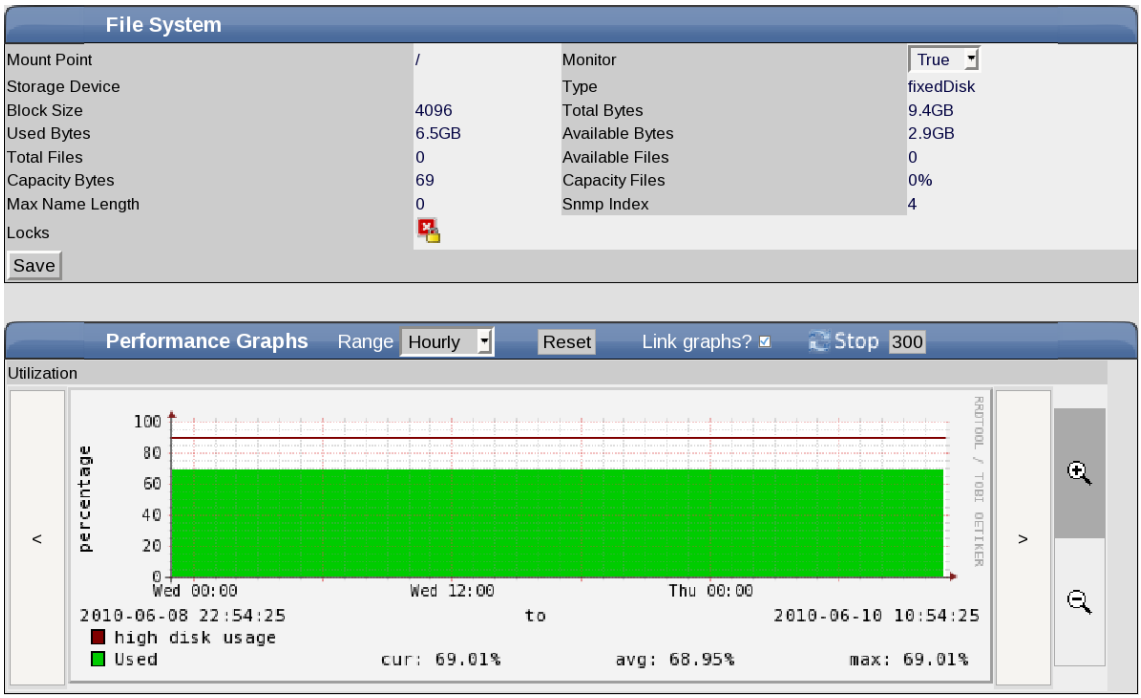


FIG. 3.9 – Point de montage / de *Support-Clients-Zenoss*

Ce graphe reflète l'utilisation du système à partir du point de montage / de la station *Support-Client-Zenoss* sur les dernières heures. Si l'utilisation du fichier système à partir de ce point de montage dépasse les 90% alors un événement d'avertissement sera généré.

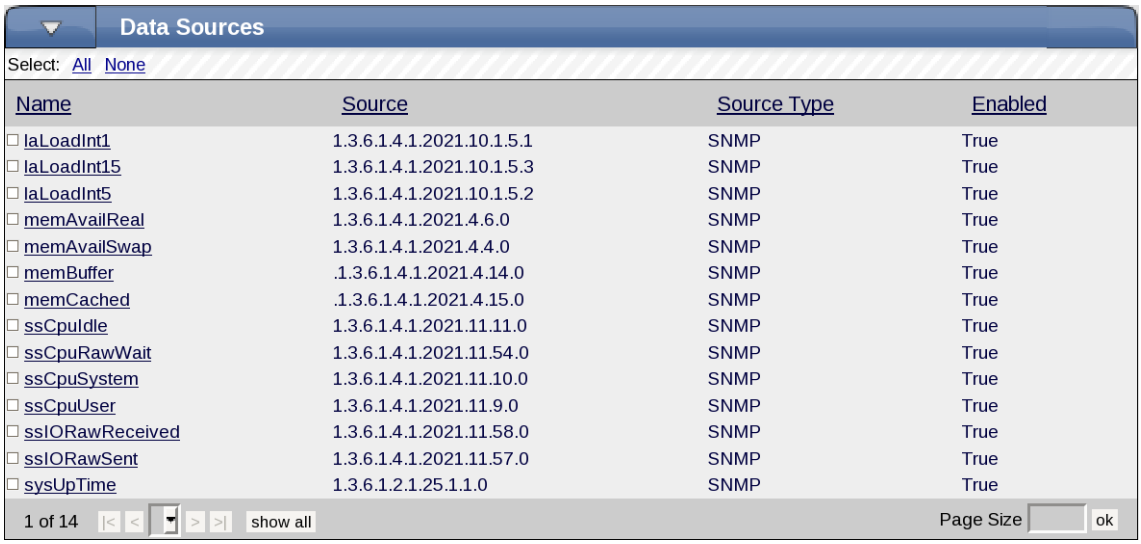
Des réglages peuvent être effectués concernant l'affichage des graphes. L'option **range** permet d'afficher les graphes selon une certaine plage fréquentielle : heure, jour, semaine, mois, année. Le bouton **Reset** permet de remettre la vue par défaut et la case **Link Graphs** permet de lier tous les graphes de manière à ce que leur lecture reste synchronisée.

3.1.3 Modèles de performances

Les modèles de performances indiquent à Zenoss quelles sont les sources de données à collecter et comment tracer des graphes de ces données. Ils utilisent les données pour établir des seuils de supervision qui une fois dépassés peuvent générer des événements. Ils peuvent s'appliquer à une classe de machines ou à une simple station.

Sources de données

Une source de données, comme son nom l'indique, est un endroit dans lequel Zenoss va aller chercher afin de trouver ce dont il a besoin. La table **Data Sources** affiche les caractéristiques suivantes pour chacune des sources de données : le nom, la source (le chemin d'accès), le type de la source et le status de la source.



Data Sources

Select: All None

Name	Source	Source Type	Enabled
laLoadInt1	1.3.6.1.4.1.2021.10.1.5.1	SNMP	True
laLoadInt15	1.3.6.1.4.1.2021.10.1.5.3	SNMP	True
laLoadInt5	1.3.6.1.4.1.2021.10.1.5.2	SNMP	True
memAvailReal	1.3.6.1.4.1.2021.4.6.0	SNMP	True
memAvailSwap	1.3.6.1.4.1.2021.4.4.0	SNMP	True
memBuffer	.1.3.6.1.4.1.2021.4.14.0	SNMP	True
memCached	.1.3.6.1.4.1.2021.4.15.0	SNMP	True
ssCpuIdle	1.3.6.1.4.1.2021.11.11.0	SNMP	True
ssCpuRawWait	1.3.6.1.4.1.2021.11.54.0	SNMP	True
ssCpuSystem	1.3.6.1.4.1.2021.11.10.0	SNMP	True
ssCpuUser	1.3.6.1.4.1.2021.11.9.0	SNMP	True
ssIORawReceived	1.3.6.1.4.1.2021.11.58.0	SNMP	True
ssIORawSent	1.3.6.1.4.1.2021.11.57.0	SNMP	True
sysUpTime	1.3.6.1.2.1.25.1.1.0	SNMP	True

1 of 14 show all Page Size ok

FIG. 3.10 – Liste des sources de données de la classe **/Server/Linux**

Exemple :

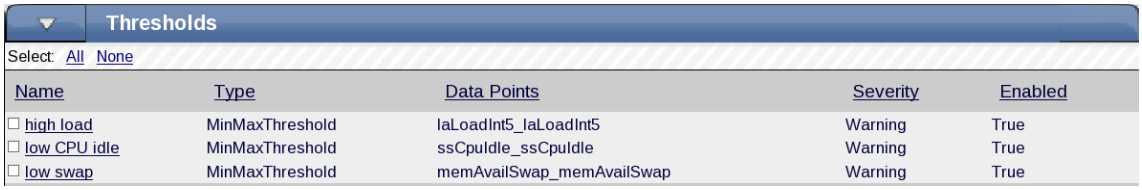
Pour obtenir la date de la dernière fois que le système d'une machine a démarré, Zenoss va fouiller dans l'Object IDentifier de chemin d'accès 1.3.6.1.2.1.25.1.1.0. On peut vérifier qu'au bout de ce chemin, il y a bien la valeur de **sysUpTime**.

```
snmpwalk -v 2c -c lix Server-Zenoss 1.3.6.1.2.1.25.1.1.0
```

Le système de la machine *Server-Zenoss* semble avoir été démarré il y a 55 jours, résultat qui paraît tout à fait logique.

Seuils de supervision

Un seuil de supervision permet de fixer une certaine valeur à un composant. Si cette valeur vient à être dépassée alors le seuil peut envoyer des événements plus ou moins importants selon sa configuration. La table **Thresholds** permet de lister les seuils : le nom, le type, la source de données associée, le degré d'importance accordé à l'événement et le statut administratif du seuil.



Thresholds

Select: All None

Name	Type	Data Points	Severity	Enabled
high load	MinMaxThreshold	laLoadInt5_laLoadInt5	Warning	True
low CPU idle	MinMaxThreshold	ssCpuIdle_ssCpuIdle	Warning	True
low swap	MinMaxThreshold	memAvailSwap_memAvailSwap	Warning	True

FIG. 3.11 – Liste des seuils de supervision de la classe **/Server/Linux**

Exemple :

Pour superviser de façon plus élaborée l'utilisation de la CPU, on peut créer un seuil de supervision de type **MinMaxThreshold** c'est-à-dire que c'est un seuil constitué d'une valeur minimale et d'une valeur maximale. Quelques paramètres à modifier pour un exemple plus concret :

Data Points : ssCpuRawIdle_ssCpuRawIdle
Min Value : 5
Severity : warning
Escalate : 3

La configuration de ce seuil signifie que si la valeur de la CPU inoccupée tombe sous la barre des 5% alors Zenoss générera un événement de type avertissement. Si cette valeur tombe sous la barre des 5% trois fois consécutives alors Zenoss augmentera le degré d'importance de l'événement jusqu'au niveau erreur.

3.2 Gestion des événements

Lorsqu'un problème survient sur le réseau, on souhaite avoir des informations en ce qui le concerne. On cherchera à savoir la localisation du problème ou son origine (la cause qui l'a déclenché). Zenoss nous tient informés des problèmes en générant des événements quand une station devient indisponible ou qu'elle dépasse une certaine valeur d'un seuil de supervision par exemple.

3.2.1 Surveillance des messages syslog

Le logiciel offre la possibilité de superviser les syslog des stations Linux du réseau. Il utilise le daemon *zensyslog* pour transformer les messages syslog en événements. Pour que Zenoss collecte les syslog d'une machine, il faut autoriser celle-ci à les envoyer au serveur et lui en indiquer l'adresse.

```
[nom_de_la_machine]
  $ vim /etc/syslog.conf

      ## modifier ou ajouter les lignes suivantes ##
      mail.debug        @192.168.112.224
      *.debug           @192.168.112.224

  $ service syslog restart

[Server-Zenoss]
  $ su - zenoss            ## utilisateur Zenoss ##
  $ zensyslog restart      ## redémarrage du daemon zensyslog ##
```

On peut tester la remontée des syslog en utilisant l'outil en ligne de commande **logger** qui permet d'envoyer un message syslog test.

```
logger -p mail.error "Ceci est un message test"
```

Théoriquement Zenoss devrait afficher un événement de priorité erreur dont le message est « Ceci est un message test ». Pour s'assurer que la commande a bien fonctionné, on ouvre la console d'événements et on vérifie que le message syslog soit bien présent dans la liste. La commande a eu l'effet escompté puisque l'événement est dans la liste.

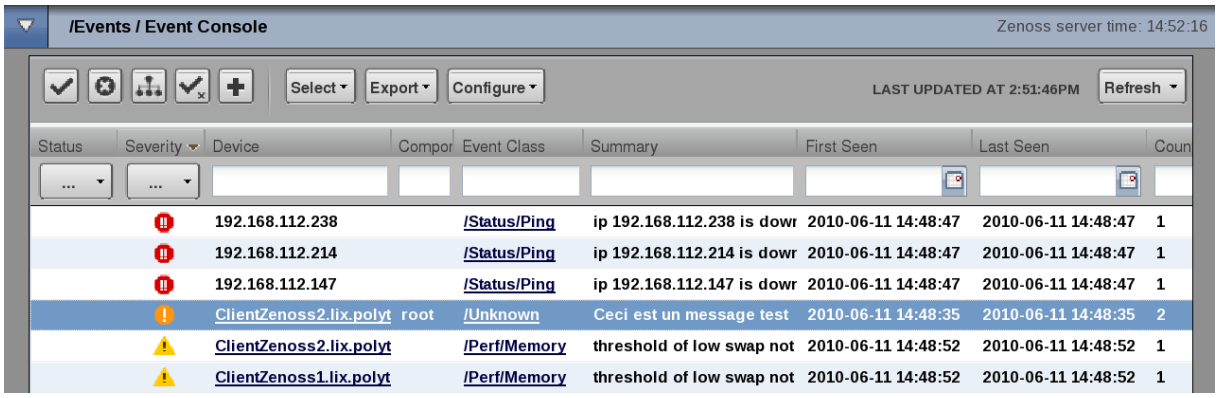


FIG. 3.12 – Test de la remontée des messages syslog

3.2.2 Console d'événements

Cette console permet de visualiser tous les événements actuels présents dans le système. Les événements peuvent êtres rangés par importance (information, erreur...) ou par statut (nouveau, connu ou supprimé). Des règles de filtrage sont également applicables pour n'afficher que certains événements. La console est supervisée en temps réel et est actualisée toutes les 60 secondes, cette valeur peut d'ailleurs être modifiée.

Journal d'un événement

A partir de la console on peut accéder à n'importe quel journal d'un événement. Il suffit juste pour cela de double-cliquer sur l'événement et une multitude d'informations complémentaires viennent s'afficher.

Le journal rend compte de toutes les données relatives à l'événement : on sait vraiment tout ce que l'on souhaite savoir sur l'événement. Pour l'événement de la figure précédente, on s'aperçoit qu'il s'agit d'un email et qu'il a été généré par l'agent zenpop3.

Par exemple, le champs **message** nous révèle le contenu du mail et celui **ipAdress** nous indique l'adresse IP d'où provient l'email. L'adresse IP en question est 129.104.11.2 et en cherchant le nom de la machine associé à cette adresse IP (une commande `traceroute adresse_ip` fait parfaitement l'affaire), on découvre que la machine s'appelle **argos** et qu'il s'agit de la machine sur laquelle est implémenté le serveur mail du laboratoire ! Conclusion : le mail nous vient tout droit du serveur POP3 du LIX.

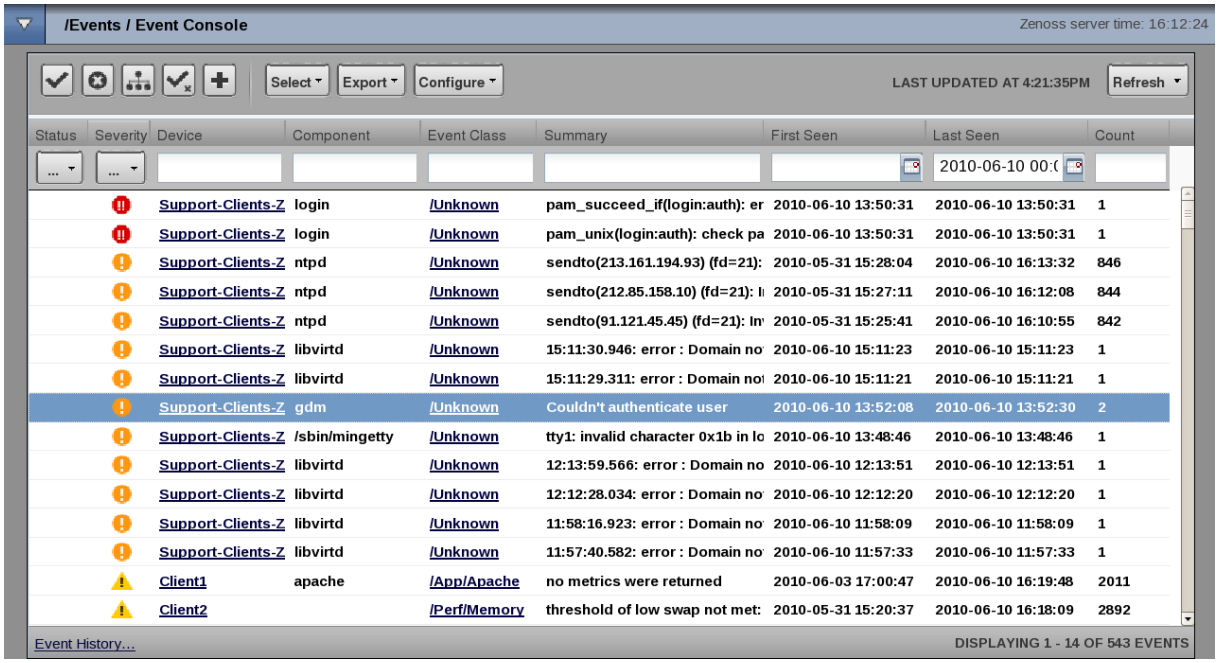


Fig. 3.13 – Console d'événements

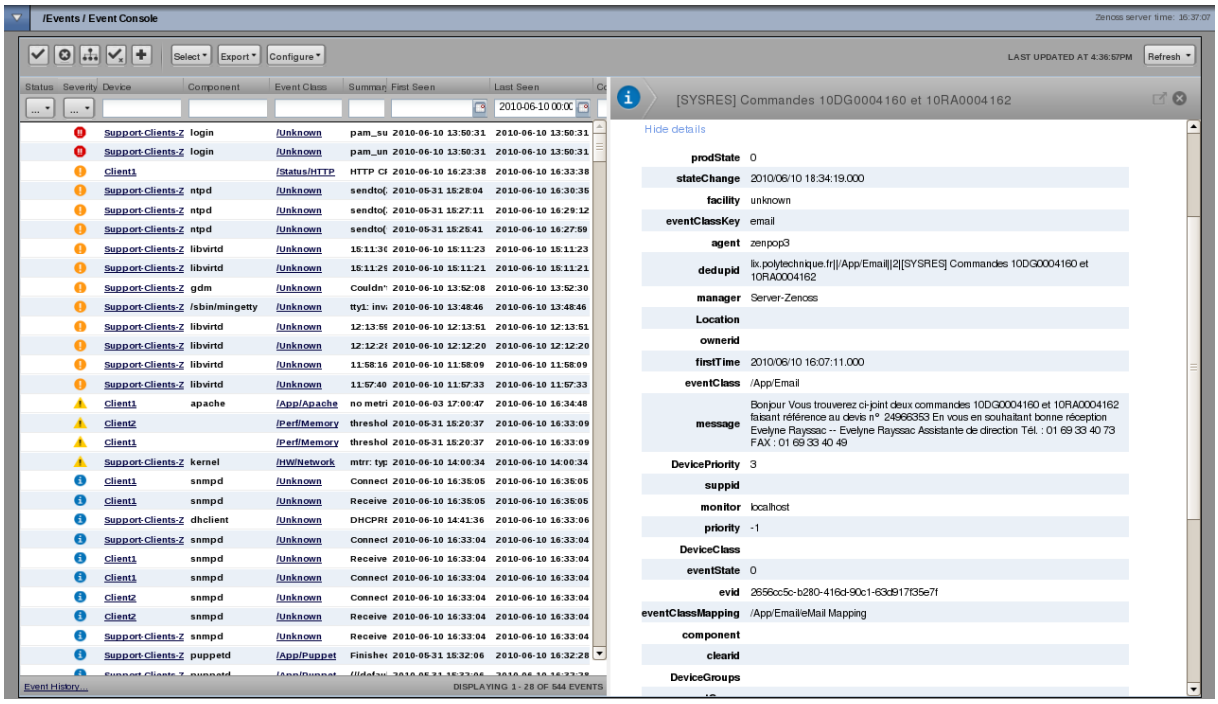


Fig. 3.14 – Journal d'un événement d'information

3.2.3 Classe d'événements

Les événements sont générés en tant que réponses à des activités de surveillance sur le réseau. Zenoss les range dans une classe d'événement précise par correspondance avec mot-clé et en fonction d'événemts similaires déjà rangés par le logiciel. Le principe et les avantages sont les mêmes que pour les équipements, cela permet principalement une meilleure gestion.

Souvent lorsque Zenoss reçoit un événement qu'il ne peut classer, il se contente de le ranger dans la classe **Unknown**. C'est par exemple le cas des syslog vus précédemment. On va donc créer une classe que l'on appelle **Syslog** et qui aura pour vocation de ranger tous les événements de messages syslog. Une fois la classe créée, on filtre la console d'événements pour afficher seulement les messages syslog. On les sélectionne tous et on les fait correspondre à la classe **Syslog** par l'option **Maps selected events to an event class**.

FIG. 3.15 – Journal d'un événement d'information

Un nouvel événement est comparé à ceux déjà classés. Si il est similaire à l'un d'eux ou qu'ils ont des mots-clefs en relation, alors il sera placé dans la même classe. On décide d'ajouter manuellement un événement (avec seulement les 3 paramètres suivants).

Message : `[syslog] Ceci est un AUTRE message test !`

Component : root

Event Class Key : syslog

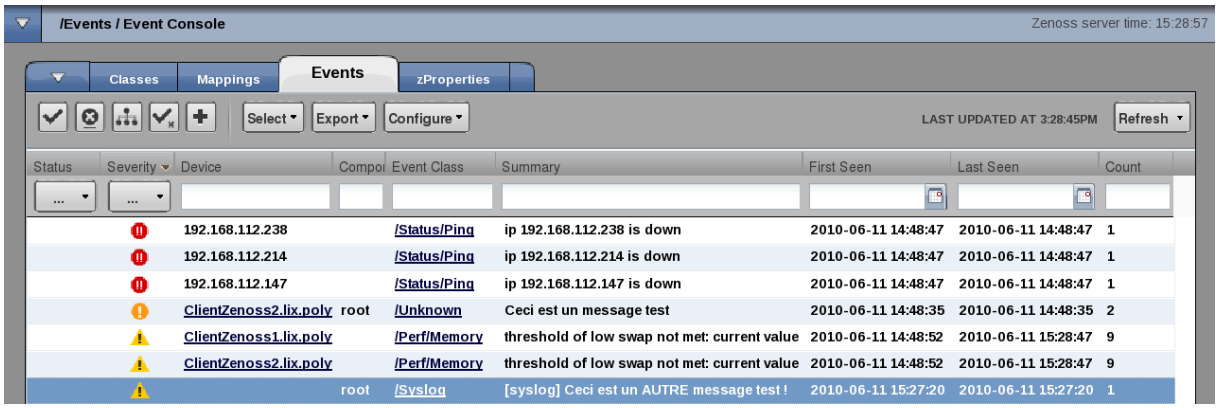


FIG. 3.16 – Ajout d'un événement

On remarque qu'effectivement l'événement a été rangé dans la classe **Syslog** et il en sera de même pour tous les message syslog relevés par Zenoss.

3.3 Serveur Web Apache

3.3.1 Déploiement

On décide de créer un serveur Web Apache sur la station *ClientZenoss1* pour ensuite le superviser avec Zenoss. Pour cela il suffit d'effectuer les commandes suivantes :

```
[ClientZenoss1]
  $ yum -y install httpd
  $ vim /etc/httpd/conf/httpd.conf

      ## vérifier les les lignes suivantes ##
      ServerRoot "/etc/httpd"
      Listen 80
      ServerAdmin lederman@lix.polytechnique.fr
      DocumentRoot "/var/www/html"
      <Directory "/var/www/html">
          Order allow,deny
          Allow from 192.168.112.0/24
      </Directory>
      DirectoryIndex index.html index.htm index.php
```

C'est dans le fichier `httpd.conf` que l'on écrit la configuration du serveur Apache. On se contente de le configurer basiquement car cela peut devenir très vite complexe. Si les quelques lignes ci-dessus sont présentes dans le fichier, alors n'importe quelle machine du réseau 192.168.112.0 est autorisée à accéder au serveur Apache. Il suffit pour cela d'ouvrir un navigateur Internet et de rentrer `http://192.168.112.236` dans la barre d'adresses.

Après avoir rapidement écrit une page d'accueil **index.html** placée dans le dossier `/var/www/html` on peut envisager de superviser cette page du serveur.

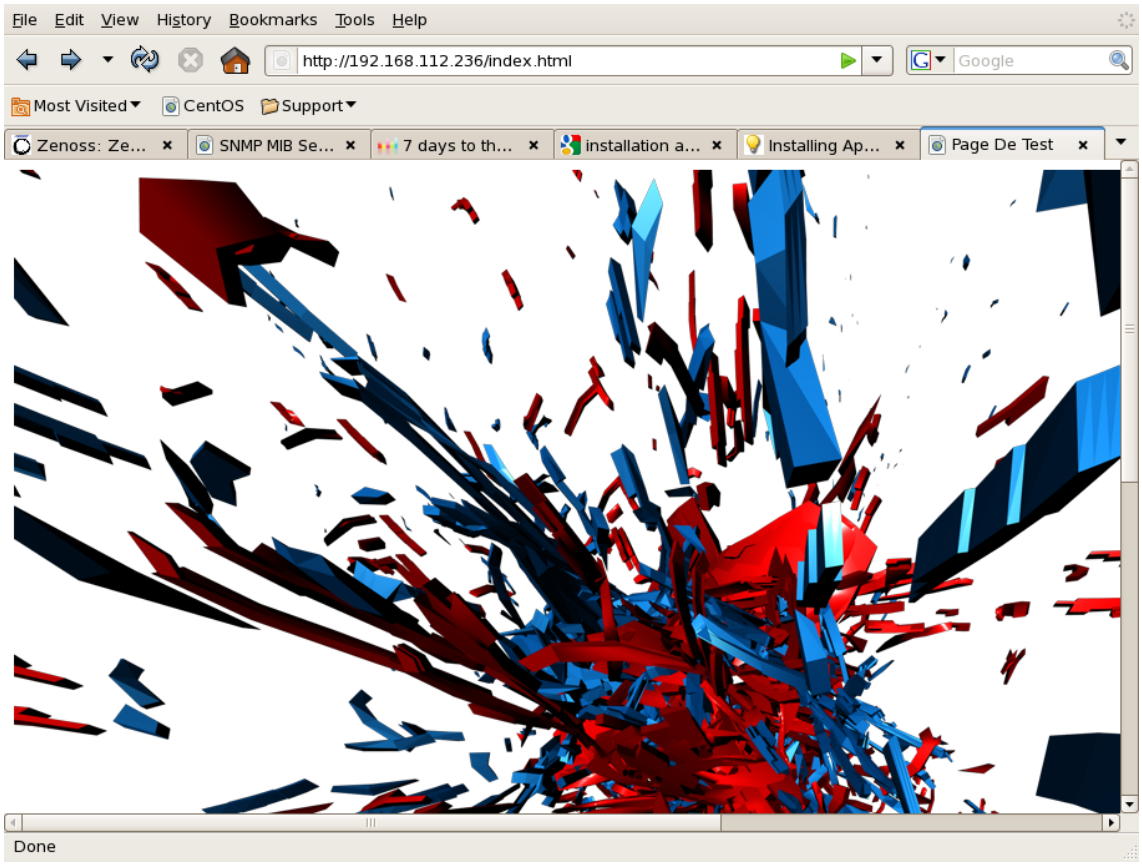


FIG. 3.17 – Page d'accueil du serveur Apache

3.3.2 Surveillance par Zenoss

Le ZenPack **ApacheMonitor** va nous permettre de superviser directement dans Zenoss, certaines données émises depuis le serveur web Apache. Il utilise pour cela le module Apache **mod_status**. Pour réaliser des graphes, on peut par exemple collecter le nombre de requêtes par seconde sur le serveur ou son utilisation de la CPU. On doit éditer le fichier de configuration du serveur web.

```
[ClientZenoss1]
  $ vim /etc/httpd/conf/httpd.conf

      ## décommenter les lignes suivantes ##
      LoadModule status_module /usr/lib/apache/1.3/mod_status.so
      ExtendedStatus on
      <Location /server-status>
            SetHandler server-status
            Order deny,allow
            Deny from all
            Allow 192.168.112.0/24
      </Location>

  $ service httpd restart
```

Une fois que le serveur Apache a été configuré, on doit vérifier qu'il fonctionne correctement. Il faut entrer l'url suivante dans le navigateur Internet :
`http://192.168.112.236/server-status?auto`
Le résultat affiché dans la page correspond à quelque chose comme cela :

```
Total Accesses: 261455
Total kBytes: 85992
CPULoad: .0321946
Uptime: 480888
ReqPerSec: .543692
BytesPerSec: 183.111
BytesPerReq: 336.791
BusyWorkers: 1
IdleWorkers: 19
```

Le serveur Apache est désormais configuré comme il se doit et donc on peut ajouter sa supervision par la machine *ClientZenoss1* dans Zenoss. Il suffit simplement de lier la modélisation du serveur Apache à la station. On procède de la manière suivante :

- On se rend sur la vue d'ensemble de la machine *ClientZenoss1*.
- On va dans le menu **More/Templates**.
- On va ensuite dans le menu **Bind Templates...**.
- En maintenant appuyer la touche **Ctrl**, on clique sur Apache puis on valide.

Il est maintenant possible de collecter des informations du serveur Apache depuis cette machine. En naviguant parmi les graphes disponibles pour la station, on constate la présence de nouveaux graphes qui afficheront des valeurs relatives au serveur Apache.

Retour d'expériences

Tout au long de mes deux ans de formation, j'ai appris énormément de connaissances dans le monde du Réseau par lequel je me sens désormais inévitablement attiré. Le déploiement de réseaux informatiques, l'implémentation de services et leur configuration sur les équipements constituent l'un de mes principaux centres d'intérêts. Je me suis familiarisé avec la manipulation de protocoles sur les systèmes d'exploitation Windows (XP, Seven, Server 2000-2008) et Linux. Cette formation m'a donné l'occasion de mettre en place de nombreuses architectures réseaux pour des besoins bien précis.

Pendant la durée de mon stage en entreprise, j'ai assuré le déploiement et la configuration de Zenoss dans un réseau informatique. L'objectif premier était de surveiller l'ensemble des composants du réseau de manière rapide et centralisée, grâce à cet outil de supervision. Je me suis penché sur les fonctionnalités proposées par le logiciel pour tester son efficacité. De mon point de vue, il s'agit d'une solution puissante qui est relativement facile à prendre en main. Entièrement personnalisable, on peut surveiller tout ce que l'on souhaite concernant les équipements actifs sur le réseau. Zenoss constitue une très bonne alternative de supervision réseau.

En formation j'ai découvert le protocole SNMP : son utilité, son principe de fonctionnement, sa configuration... et je l'ai réinvesti au cours des deux derniers mois puisque Zenoss se base sur ce protocole pour effectuer la supervision des équipements du réseau. Le stage m'a permis de développer mon niveau de langue par la lecture de documentations techniques et de tutorials écrits en anglais. J'ai également acquis des nouvelles méthodes de travail telles que la manipulation du langage LaTeX et je me suis retrouvé plongé dans le monde Unix/Linux sur des machines utilisant des distributions Fedora Core ou CentOS. J'ai découvert une nouvelle méthode de virtualisation des machines et j'ai appris des connaissances de toute sorte (benchmark d'un serveur Apache, utilisation de Puppet).

Je suis pleinement satisfait du sujet qui m'a été confié et qui correspondait à mes attentes. Déployer une solution de supervision dans un réseau permet de surveiller tout ce qui se passe dessus et d'être averti en temps réel si le besoin s'en fait ressentir. En utilisant Zenoss, j'ai pu visualiser à distance les différentes pannes survenues dans mon réseau local et cela m'a permis de remédier aux problèmes rapidement. Je n'ai pas assez exploité la communauté s'articulant autour de Zenoss Core et les ZenPacks (extensions du logiciel) qui auraient pu me permettre une supervision toujours plus poussée du réseau. En entreprise, la supervision du réseau devient primordiale aujourd'hui. Mon travail pourra être repris et continué par la suite.

En dehors de l'aspect technologique, je me suis parfaitement adapté à la vie du laboratoire. Bien intégré dans l'équipe, j'ai noué de bonnes relations pédagogiques avec mon maître de stage et avec plusieurs membres du laboratoire. L'ambiance était décontractée sans trop l'être ; la bonne humeur et la sympathie de chacun donnaient envie de venir travailler.

Ce stage m'a été bénéfique, aussi bien sur le plan intellectuel que sur le plan relationnel.

Il m'a permis de faire une véritable approche avec le monde professionnel. J'étais responsable d'un travail, on a compté sur moi pour mener à bien un projet ; donc il m'a aussi aidé à avoir encore plus confiance en moi.

Sans doute parce que le fonctionnement d'un laboratoire de recherche diffère de celui d'une entreprise standard et que j'en ai pris conscience au cours de ces 10 semaines, je serai intéressé pour devenir chercheur dans le domaine des Réseaux essentiellement. Je compte poursuivre activement mes études en faisant un IUP STRI, prendre des cours particuliers et j'aimerai aussi me consacrer à l'obtention de certifications en tout genre.

A Machines virtuelles

A.1 Définition

Une machine virtuelle est un environnement entièrement isolé, capable d'exécuter des systèmes d'exploitation et des applications. Les utilisateurs ont l'illusion de disposer d'un ordinateur complet. La machine virtuelle se comporte exactement comme une machine physique et contient ses propres composantes : CPU, mémoire RAM, interfaces réseau.

L'utilisation des ressources d'un réseau informatique se fait de façon optimale grâce à une répartition des machines virtuelles sur les postes physiques en fonction des charges respectives. On peut noter aussi une meilleure sécurisation du réseau et cela permet d'économiser sur le matériel au niveau de la consommation électrique et de l'entretien.

A.2 Principe d'un hyperviseur

On utilise l'hyperviseur KVM (Kernel Virtual Machine) pour nos machines virtuelles. Il permet de configurer le noyau Linux pour que celui-ci soit en mesure de recevoir plusieurs systèmes d'exploitations virtualisés. KVM est un outil complètement libre, performant et facile à installer et utiliser.

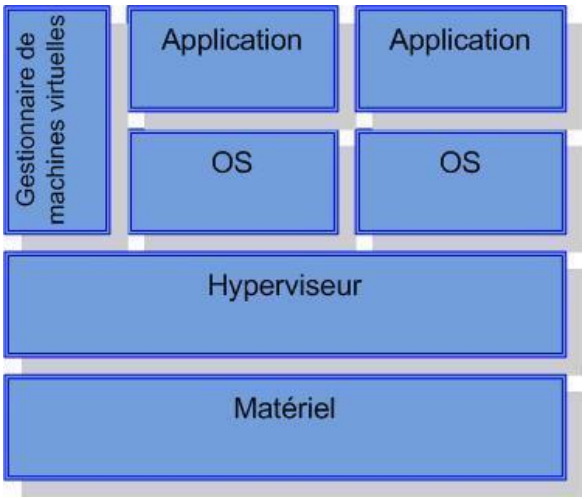


Fig. A.1 – Modèle en couches d'un hyperviseur

A.3 Installation de KVM

On souhaite déployer des machines virtuelles sur la machine *Support-Clients-Zenoss*, c'est donc sur ce poste que l'on va installer l'hyperviseur. Il faut procéder de la manière suivante dans un terminal :

```
[Support-Clients-Zenoss]
  $ yum install kvm kmod-kvm          ## installation des packages KVM ##
  $ modprobe kvm-intel                ## ajout de KVM dans le noyau Linux ##
  $ /sbin/lsmod | grep kvm            ## vérifie que le module est bien chargé ##
  $ chown root:kvm /dev/kvm           ## définition des permissions
  $ chmod 0660 /dev/kvm                  pour /dev/kvm ##

Il faut ensuite redémarrer le système pour s'assurer que le module KVM
soit bel et bien chargé et que les permissions soient bien définies.
```

Attention : Les commandes que l'on vient d'exécuter ne s'appliquent que pour une machine ayant une architecture de processeur x86 (Intel).

A.4 Utilisation de virt-manager

Le gestionnaire virt-manager est un outil graphique permettant de gérer les machines virtuelles des hyperviseurs Xen et/ou KVM. On va donc s'en servir dans le but de créer notre toute première machine virtuelle.

```
[Support-Clients-Zenoss]
  $ virt-manager                      ## ouverture du gestionnaire ##
```

Il faut procéder par étapes afin de s'assurer d'avoir une machine virtuelle opérationnelle en peu de temps. L'installation d'une station prend environ 2 minutes.

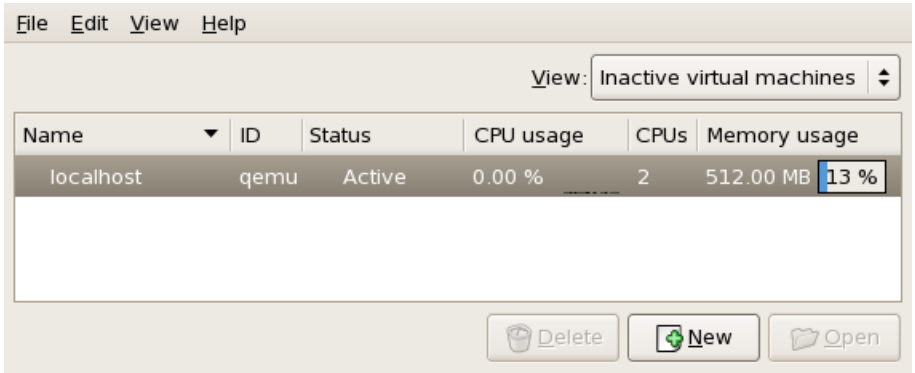


Fig. A.2 – Virtual Machine Manager

Etapes à suivre pour installer une machine virtuelle :

- Sélectionner la machine localhost.
- Cliquer sur **New** pour lancer le guide d'installation d'une nouvelle machine puis cliquer sur **Forward**.
- Entrer le nom de la future machine puis cliquer sur **Forward**.
- Choisir l'option **Fully Virtualized**, l'architecture CPU x86_64 et l'hyperviseur KVM puis cliquer sur **Forward**.
- Choisir l'option **Network boot (PXE)**, laisser les options concernant l'OS sur leur valeur par défaut (Generic) puis cliquer sur **Forward**.
- Choisir l'option **Block device (partition)** et indiquer l'emplacement du volume logique sur lequel sera la machine virtuelle puis cliquer sur **Forward**.
- Choisir l'option **Virtual Network**, laisser le réseau sur sa valeur par défaut (default) puis cliquer sur **Forward**.
- Indiquer les paramètres concernant la mémoire et la CPU de la machine virtuelle puis cliquer sur **Forward**.
- Cliquer enfin sur **Finish** pour lancer l'installation et patienter quelques minutes.

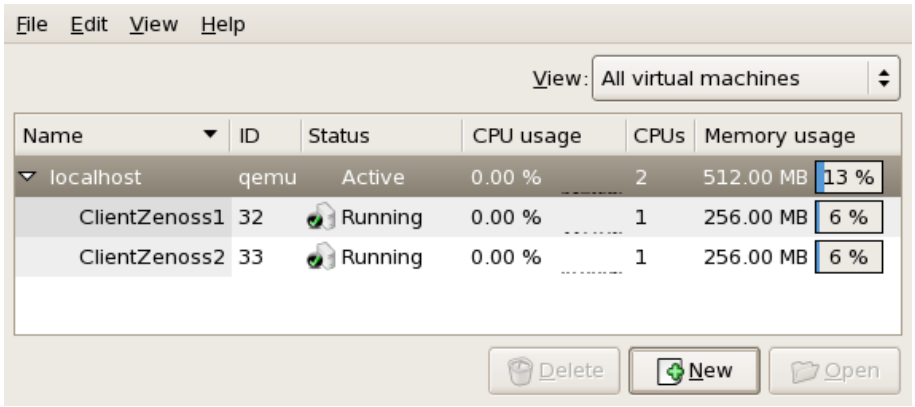


FIG. A.3 – Liste des machines virtuelles

En sélectionnant une machine, on obtient quelques informations sur son disque dur et ses performances au niveau de l'utilisation de la CPU et de la mémoire. Le gestionnaire permet aussi d'ouvrir une console propre à la machine virtuelle.

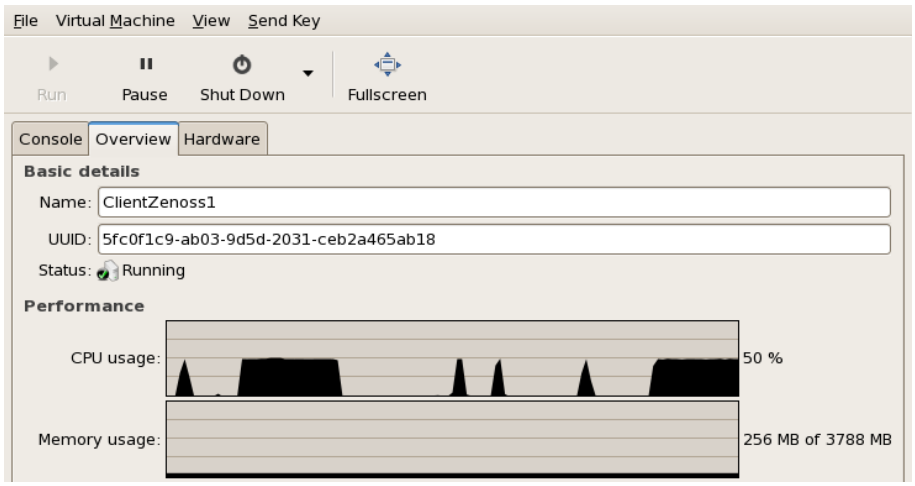


FIG. A.4 – Utilisation de la CPU sur *ClientZenoss1*

On remarque que le graphique ci-dessus reproduit fidèlement l'utilisation de la CPU après avoir réalisé des tests de montée en charge. Cependant on constate que l'outil est très limité puisque les tracés ne sont pas définis sur une plage horaire précise, on ressent le manque cruel de l'unité de temps sur l'axe des abscisses.

Rappel :

L'un des paramètres à définir lors de l'installation d'une machine virtuelle est le volume logique sur lequel elle sera installée. Pour créer un volume logique **ClientZenoss1** de **15 gigaoctets** dans le volume groupe **VolGroup00** la commande est la suivante :

```
lvcreate -n ClientZenoss1 -L 15G VolGroup00
```

Pour visualiser respectivement les volumes physiques, groupes et logiques d'une machine, les commandes respectives pvscan, vgscan et lvscan sont à exécuter dans un terminal de la machine en question.

B Puppet : administration centralisée

B.1 Présentation

Puppet est un outil permettant de définir et de déployer des configurations. Une configuration est un ensemble de paramètres qui peut se constituer de fichiers de configuration, de packages, de services... Basé sur une architecture clients-serveur, Puppet permet de centraliser l'administration système.

Il faut installer et configurer un client Puppet sur chaque station que l'on souhaite automatiser. Ce client communiquera avec le serveur, dans lequel sont stockés les manifests qui contiennent les spécifications. La communication entre les clients et le serveur, appelé « puppetmaster » se fait par utilisation de certificats dans une couche ssl.

B.2 Déploiement de l'outil

On souhaite installer et configurer Puppet dans le réseau local. Le serveur sera placé sur la station *Server-Zenoss* et les 3 autres stations seront équipées d'un client.

Dans un premier temps il faut établir (si ce n'est déjà fait) les relations entre le nom d'hôte et l'adresse IP des autres machines du réseau. Ensuite il faut pour chaque client : installer les packages, modifier les fichiers de configuration et demander un certificat ssl au puppetmaster.

```
[nom_de_la_machine]
  $ vim /etc/hosts

      127.0.0.1     localhost.localdomain  localhost
      ::1          localhost6.localdomain6  localhost6

      192.168.112.224  Server-Zenoss.lix.polytechnique.fr  Server-Zenoss
      192.168.112.253  Support-Clients-Zenoss.lix.polytechnique.fr  Support-Clients-Zenoss
      192.168.112.236  ClientZenoss1.lix.polytechnique.fr  ClientZenoss1
      192.168.112.129  ClientZenoss2.lix.polytechnique.fr  ClientZenoss2

  $ vim /etc/sysconfig/network

      NETWORKING=yes
      NETWORKING_IPV6=no
      HOSTNAME=xxx.lix.polytechnique.fr
                        ## xxx <==> nom de la station ##
```

```
[Server-Zenoss]
  $ yum -y install puppet-server
  $ yum -y install ruby-rdoc
  $ vim /etc/puppet/manifests/AdresseServer

      $server = "Server-Zenoss.lix.polytechnique.fr"
      node default {
      }

  $ vim /etc/puppet/manifests/site.pp
  $ vim /etc/puppet/puppetd.conf

      [puppet]
              vardir = /var/lib/puppet
              logdir = /var/log/puppet
              rundir = /var/run/puppet
              ssldir = /var/lib/puppet/ssl
              factpath = $vardir/lib/facter
              pluginsync = true

      [puppetmasterd]
              classfile = $vardir/classes.txt
              localconfig = $vardir/localconfig
              templatedir = /etc/puppet/templates

  $ vim /etc/sysconfig/puppet

      ## il faut modifier les lignes suivantes ##
      PUPPET_SERVER = 192.168.112.224
      PUPPET_PORT = 8140

  $ service puppetmaster start

[clients_Puppet]
  $ yum -y install puppet
  $ yum -y install ruby-rdoc
  $ vim /etc/puppet/puppetd.conf

      [puppet]
              vardir = /var/lib/puppet
              logdir = /var/log/puppet
              rundir = /var/run/puppet
              ssldir = /var/lib/puppet/ssl

      [puppetd]
              server = Server-Zenoss.lix.polytechnique.fr
              classfile = $vardir/classes.txt
              localconfig = $vardir/localconfig

  $ puppetd --test --waitforcert 60

[Server-Zenoss]
  $ puppetca --list   ## liste des certificats en attente de signature ##
  $ puppetca --sign nom_du_certificat   ##signe le certificat en question ##

[clients_Puppet]
  $ puppetd --test
```

B.3 Test d'utilisation

Avec Puppet les possibilités de configuration sont immenses. De façon automatisée et centralisée on peut modifier la configuration de plusieurs centaines de stations, qui seront chargées de comparer leur configuration avec celle du puppetmaster et de se mettre à jour en fonction. On peut ainsi déployer des fichiers, exécuter des services ou encore mettre en place une certaine configuration. Le but de ce petit exemple est de distribuer sur le réseau un fichier test quelconque.

```
[clients_Puppet]
  $ more /tmp/fichier_test_1
  ## le résultat de cette commande nous indique
     que ce fichier n'existe pas ##

[Server-Zenoss]
  $ vim /etc/puppet/manifests/site.pp

      class test {
           file { "/tmp/fichier_test_1":
                  owner => "root",
                  group => "root",
                  mode => 644,
                  content => "--- Ceci est un test ! ---" }

           file { "/etc/puppet": ensure => directory }
      }

      node 'default' { include test }

  $ service puppetmaster restart

[clients_Puppet]
  $ puppetd --test
  $ more /tmp/fichier_test_1
  --- Ceci est un test ! ---
```

Dans le fichier de la configuration du puppetmaster, on développe une classe que l'on nomme **test** et qui contient deux instructions précises. La première est de définir les permissions concernant le fichier `/tmp/fichier_test_1` et le contenu de ce fichier. Si le fichier n'existe pas, il sera automatiquement créé. L'autre instruction est de s'assurer que `/etc/puppet` est un répertoire. Ensuite on indique le nom des stations auxquelles on veut appliquer la classe **test** ; en l'occurence ici toutes les machines du réseau seront affectées par cette classe.

On exécute la commande pour que le client compare sa configuration avec celle du serveur, il constate qu'il est concerné par la classe **test** dont il va suivre les instructions. Les stations ont compris que le fichier en question n'existe pas et donc elles le créent. Une commande de lecture appliquée au fichier affiche désormais son contenu. On vient de distribuer automatiquement le fichier `/tmp/fichier_test_1` sur l'ensemble des machines.