Mise en place d'une supervision centralisée à l'INPL

Nancy-Université
INPL

Licence ASRALL 2009/2010
Université Nancy 2 - IUT Nancy-Charlemagne

Tuteur : Sébastien JEAN
Parrain : Bernard MANGEOL

Julien VAUBOURG
<julien@vaubourg.com>

1er juillet 2010

Mise en place d'une supervision centralisée à l'INPL

INPL - CRI-SRT
2 avenue de la Forêt de Haye
BP 3 F-54501
VANDOEUVRE

Licence ASRALL 2009/2010
Université Nancy 2 - IUT Nancy-Charlemagne
2 ter Boulevard Charlemagne
CS 5227 - 54052 NANCY Cedex

Tuteur : Sébastien JEAN
Parrain : Bernard MANGEOL

Julien VAUBOURG
<julien@vaubourg.com>

Merci.

- **Mon tuteur, Sébastien**, pour le temps qu'il m'a accordé et nos nombreuses discussions
- **Benoît**, pour son aide et sa bonne humeur
- **Pierre**, pour son énergie et sa persévérance Windowsienne
- **Sébastien**, pour sa compagnie et pour m'avoir supporté durant trois mois
- **Arnaud**, **Mohammad**, **Christian**, **Valéry**, **Sylvain**, et tous ceux qui contribuent à la bonne humeur du service

Table des matières

1. Préambule

1.1 L'INPL et le CRI

L'INPL[1] est une université, essentiellement composée d'écoles d'ingénieurs.

Parmi elles :

l'EEIGM Ecole Européenne d'Ingénieurs en Génie des Matériaux
l'ENSAIA Ecole Nationale Supérieure d'Agronomie et des Industries Alimentaires
l'ENSEM Ecole Nationale Supérieure d'Electricité et de Mécanique
l'ENSG Ecole Nationale Supérieure de Géologie
l'ENSGSI Ecole Nationale Supérieure en Génie des Systèmes Industriels
l'ENSIC Ecole Nationale Supérieure des Industries Chimiques
l'ENSMN Ecole Nationale Supérieure des Mines de Nancy

Pour la beauté des chiffres :

- Environ 4 sites géographiques
- 4000 étudiants dont 450 étrangers
- 560 enseignants et chercheurs permanents
- 600 administratifs et techniciens

L'apparition d'un CRI[2] date de moins de deux ans. Elle vise à rassembler les services informatiques de toutes ces écoles en une seule entité. Parmi les pôles du CRI, on distingue le service CRI-SRT[3], au sein duquel s'est déroulé mon stage. Sa mission principale est de fournir des services informatiques fonctionnels à l'université, comme les hébergements web, la messagerie, la téléphonie, etc.

Ses missions sont en train d'être redéfinies. En effet, peu de temps après notre arrivée, nous avons assisté à l'arrivée de Eric Sand, le nouveau directeur du système d'informations. Sa venue a pour principale mission d'aider le CRI à s'instaurer complètement, et d'organiser cette nouvelle façon de faire. Il a aussi pour mission de faire démarquer les services de l'université face aux autres universités, en vue de la Grande Université, promise par Valérie Pécresse il y a deux ans dans le cadre de la loi L.R.U.[4], alors ministre de l'éducation supérieure.

Parmi les membres de l'équipe que j'ai eu l'occasion de côtoyer au quotidien :

- **Sébastien, mon tuteur**, administrateur systèmes et réseaux. *Aficianado* du Libre,

[1] INPL : Institut National Polytechnique de Lorraine
[2] CRI : Coordination/Centre des Ressources Informatiques
[3] SRT : Systèmes Réseaux et Télécommunications
[4] L.R.U. : Liberté et Responsabilité des Universités

et pour cause il était en licence ASRALL[5] l'année précédente. Je risquerais presque de le vexer en disant qu'il est très sympa.
- **Benoît**, le directeur du CRI-SRT. *Macboy*, unixien d'expérience, *perliste* et adepte d'Emacs. Il était également le tuteur de mon collègue stagiaire, Sébastien.
- **Pierre**, administrateur systèmes et réseaux. Il a la particularité d'être, aussi, très bon sur Windows.
- **Arnaud**, ingénieur d'études. Très compétent sur Zimbra, il a validé son diplôme d'ingénieur d'études durant notre stage.
- **Mohammad**, ingénieur d'études et grand défenseur du modèle privateur. Comme il aime le dire, il a déjà le fait le tour de France.
- Et enfin **Sébastien**, stagiaire de même parcours, avec qui j'ai partagé avec plaisir les trois mètres carrés du bureau qui nous était dédié durant ces trois mois.

2. Problématiques

2.1 Une interface unique

La problématique de mon sujet de stage s'inscrit directement dans celle de constituer un vrai CRI, en effaçant le passé fragmenté des services informatiques.

Le CRI recense déjà à mon arrivée pas moins de trois interfaces officielles différentes de supervision, héritées des anciens services :

- Un Nagios (logiciel de supervision qui sera abordé dans un prochain chapitre) pour l'ENSEM
- Un autre Nagios pour l'ENSIC
- Et un dernier moins spécialisé qui regroupe des machines d'un peu partout dans l'INPL

Ces trois Nagios réunis, nous arrivons à un total d'environ deux-cent-cinquante machines supervisées, pour un peu moins d'un millier de services.

Mon premier objectif est donc défini : proposer au CRI-SRT une interface unique de surveillance des machines pour l'ensemble du parc de l'INPL, tous sites confondus. Ceci implique donc de prendre en compte l'existant, en récupérant, en important et en fusionnant les configurations déjà mises en place.

2.2 Une authentification centralisée

L'accès à l'interface sera obligatoirement privé.

Ainsi, il faudra gérer les comptes utilisateurs, sans avoir à les configurer directement dans l'interface. On souhaite que celle-ci valide les authentifications par le biais de l'annuaire LDAP[1] de l'infrastructure, et que les informations personnelles comme les adresses de

[5] ASRALL : Administration de Réseaux, Systèmes et Applications à base de Logiciels Libres

[1] LDAP : Lightweight Directory Access Protocol est à l'origine un protocole permettant l'interrogation et la modification des services d'annuaire.

courriels (qui servent pour les alertes de notification) soient aussi puisées dans cette source d'informations.

2.3 Des vues métier

Le CRI-SRT souhaite proposer ses services de la façon la plus ciblée, conviviale et pratique possible. Ainsi, on souhaite pouvoir proposer aux autres services (par exemple, le service de proximité), la possibilité de consulter en temps réel l'état des machines et des services par lesquels ils sont concernés.

Dans la même logique mais à plus grande échelle, on souhaite proposer une interface publique qui synthétise l'état global des grands services. En un coup d'oeil, n'importe quel étudiant doit pouvoir savoir si la messagerie, par exemple, est parfaitement fonctionnelle, connait des perturbations ou n'est tout simplement plus en fonctionnement. Outre la transparence que cela apporte sur le travail du service, ceci permet de décharger l'équipe en cas de problème : si un utilisateur rencontre des problèmes et qu'en allant consulter l'état du service il se rend compte qu'il est dans le rouge, il saura immédiatement que ce n'est pas sa faute, et que l'équipe est déjà au courant. Inutile donc de surcharger les boîtes aux lettres ou les téléphones du service.

A l'inverse, si un utilisateur qui rencontre des problèmes constate que le service n'en a pas, cela l'invitera à chercher plus en profondeur de son côté, ne pouvant plus accuser le service. C'est encore du temps de gagné, et un confort réel pour les utilisateurs finaux.

3. Nagios / Centreon

3.1 Etat des lieux

Pour reprendre afin de mieux intégrer, il faut commencer par observer.

Et avant d'observer, il faut commencer par apprendre (bien qu'on apprenne beaucoup en observant). A mon arrivée, je ne connaissais pas Nagios. Celui-ci n'ayant été que rapidement abordé - parmi d'autres applications - durant une simple séance de travail de ma licence. Difficile de juger sans connaître, ainsi mes premiers jours de stage se sont déroulés à rester absorbé par un ouvrage de Jean Gabès, modestement intitulé « *Nagios 3 pour la supervision et la métrologie* », aux éditions Eyrolles. Un ouvrage de quatre-cent-quatre-vingt-deux pages, dévoré de la préface de Cédric Temple jusqu'à la quatrième de couverture plastifiée.

Une fois au fait des mécanismes les plus obscures de Nagios, un premier point sur l'existant s'est imposé.

Ainsi, c'est un document d'une dizaine de page qui a vu le jour, proposé au directeur du CRI en guise de premier bilan synthétique.

Parmi les remarques, on peut détacher :

- **Une hétérogénéité au niveau de la déclaration des machines :** Parfois associées à des adresses IP[1], parfois à des adresses DNS[2]. Outre le manque de cohérence, ce second cas permet de poser la question des conséquences d'une indisponibilité des serveurs de noms de domaine. Toutes les machines définies par leurs nom de domaine ne répondront plus, et les serveurs de supervision seront noyés dans cette masse d'informations.
- **Des services inutiles :** Chaque machine est considérée en vie ou non selon sa capacité à répondre au `ping`. Ainsi, vérifier ce même `ping`, en tant que service cette fois-ci, est polluant.
- **Des hôtes non surveillés :** Certains services sont associés à des machines qui ne sont pas elles-mêmes surveillées. Si une de ces machines ne répond plus, aucun de ses services ne peut répondre. Nagios, plutôt que de n'alerter que sur l'état de la machine, polluera les boîtes aux lettres d'alertes des services associés, qui ne peuvent nécessairement pas fonctionner.
- **Indépendance de Nagios vis-à-vis du réseau :** Que se passe-t-il si la branche sur laquelle est assise le serveur de supervision est sciée ? Autrement dit, comment Nagios pourra alerter les administrateurs en cas de coupure de réseau dans le secteur où il se trouve ? La meilleure réponse à cela est probablement un lien avec un modem GSM[3] afin de pouvoir bénéficier d'un réseau parallèle.
- **Trop d'alertes :** Après prise de température au sein de l'équipe, le bilan est que la plupart des administrateurs filtrent les notifications de Nagios dans leur boîte aux lettres, au profit de l'interface elle-même ou d'une extension Firefox.

L'intégralité du document est disponible en annexe A. La dernière partie de celui-ci concerne Centreon, sujet d'une section suivante.

3.2 Nagios

3.2.1 La supervision

3.2.1.1 Un service réactif

Les systèmes d'information varient tous, par leur nature ou par leur taille. Une chose est certaine, c'est que la loi de Murphy n'a pas de pitié : tous les systèmes sont faillibles et finiront un jour ou l'autre par montrer une limite, une faille, un problème. La prévention de ces problèmes est le premier travail des administrateurs. Premier en terme de logique, mais pas en terme de temps. On estime en effet généralement à 80%[4] le temps consacré par ceux-ci à la résolution des problèmes. Il est donc impératif de diminuer cette charge de travail, qui n'apporte aucune plus value aux services fournis.

La méthode de supervision la plus ancestrale est probablement l'utilisateur. Celui-ci n'hésitera pas à faire remonter l'information à l'administrateur. Il ne sera généralement ni très tendre ni très précis. Et c'est justement cette imprécision qui fera perdre beaucoup de

[1] IP : Internet Protocol

[2] DNS : Domain Name Server

[3] GSM : Global System for Mobile communications, le réseau de communication majoritaire des téléphones portables.

[4] Jean Gabès, dans « *Nagios 3 pour la supervision et la métrologie* »

temps, et qui dirigera parfois l'administrateur sur la mauvaise piste.

Le rôle de la supervision est donc de pouvoir prévenir les administrateurs avant que les utilisateurs n'aient le temps de s'en rende compte. Soit le problème se règle vite, et l'utilisateur qui retente n'a pas besoin d'appeler le service, soit le problème est important, auquel cas les utilisateurs peuvent être prévenus avant qu'ils ne s'énervent. Dans le pire des cas, il vaut toujours mieux répondre un « *je sais, nous nous en occupons* » qu'un « *ha bon ?* ». Grâce aux vues publiques, un utilisateur peut lui-même avoir accès à une partie de la supervision, et ainsi déterminer facilement si l'état d'un service peut justifier ses difficultés.

Enfin, certains services ne sont utilisés que dans de rares cas. Mais c'est souvent ces mêmes services qui doivent être impérativement opérationnels lorsqu'on souhaite s'en servir. C'est le cas, par exemple, des batteries de cartes RAID[5]. Pour ces services, même l'utilisateur ne sert plus de système de supervision, il est primordial d'avoir un indicateur permanent.

3.2.1.2 Intervenir avant l'effet domino

En réagissant rapidement à un problème, on peut gagner deux fois plus de temps qu'on ne le croit. En effet, en prenant l'exemple d'un disque plein, on se rend compte que si l'administrateur ne réagit pas très vite, une application critique peut-être paralysée, et elle-même entraîner d'autres dysfonctionnements pour d'autres services. Ce qui n'était à la base qu'un problème de ménage devient très vite une panne généralisée.

Admettons à présent que cette panne soit découverte un matin à huit heures, par le premier utilisateur matinal qui vient râler, son *mug* de café à la main. Le système de supervision natif - l'utilisateur - lui indique que la messagerie ne fonctionne plus. Difficile de déterminer rapidement que si il ne fonctionne pas, c'est parce que le */var* du serveur de l'annuaire est plein, que l'annuaire est donc tombé, faisant ainsi aussi tomber la messagerie qui en dépend.

Avec un système de supervision efficace (sans le *mug* et l'haleine matinale), l'administrateur peut constater rapidement que différentes choses sont en rouge. Il constatera très vite d'après les dates de dernier changement de statut que le premier problème survenu est un problème d'espace disque sur le serveur de l'annuaire. Il est donc à même de réagir, sans perdre de temps avec une batterie de tests.

Avec un système de supervision, il aurait pu également recevoir une alerte dans sa boîte aux lettres, ou sur son téléphone portable si le service est vraiment critique. Il aurait pu ainsi éventuellement lancer un grand « *je sais* » au travers de la vitre de sa voiture au premier plaignant, en train justement de remplir son *mug* à la machine à café du parking.

3.2.1.3 Mieux vaut prévenir que guérir

Ce vieil adage nous renvoie à la possibilité des systèmes de supervision d'avertir avant d'alerter. Un problème de disque plein typiquement, aurait été annoncé par un état d'avertissement, avant qu'il ne produise une notification d'alerte. Le problème aurait donc pu être

[5]RAID : « *En informatique, le mot RAID désigne une technologie permettant de stocker des données sur de multiples disques durs afin d'améliorer, en fonction du type de RAID choisi, la tolérance aux pannes et/ou les performances de l'ensemble.* » Wikipédia

résolu avant même qu'il ne se produise.

Certains types de problèmes dépassent le cadre informatique. Ils sont malgré tout primordiaux : batterie d'un groupe électrogène, fonctionnement de la climatisation, humidité, etc. Si ces services ne peuvent pas être surveillés tous les matins par un employé, ils peuvent être contrôlés en permanence par un système de supervision, par différents moyens.

3.2.2 Nagios, un logiciel qui ne fait pas grand chose

3.2.2.1 Un ordonnanceur

Nagios est un de ces systèmes de supervision, libre. Il respecte le principe KISS[6] d'UNIX. A savoir qu'il ne fait pas grand chose mais qu'il le fait bien. C'est en réalité un ordonnanceur : il a pour seul charge d'exécuter, de récupérer et éventuellement avertir. Nagios ne fait aucune vérification lui-même, il délègue.

Une sonde est un script ou un programme qui se charge d'aller faire une vérification particulière. C'est elle que Nagios va appeler lorsque son horloge lui dira qu'il est temps d'aller faire une vérification. Elle retournera deux choses : une sortie texte, contenant une courte description de l'état ainsi que des données de métrologie[7], et un code de retour UNIX variant entre zéro et deux. Zéro c'est un retour vert, un c'est orange et deux c'est donc rouge.

En respectant ce concept simple, il est aisé de développer soit-même n'importe quelle sonde qui sera compatible Nagios, et dans n'importe quel langage. Avec Nagios, il est possible de surveiller le nombre de manchots volants sur la banquise, et de se faire envoyer une alerte dès qu'on en a un, si peu qu'on trouve un moyen informatique de les compter[8].

3.2.2.2 NRPE

NRPE[9] est un intermédiaire entre Nagios et les sondes. Si il est possible pour le serveur Nagios de vérifier qu'un port est toujours en écoute sur une machine distante, il est plus difficile de connaître la charge de la machine. Des solutions comme SNMP[10] existent mais sont parfois lourdes d'utilisation et trop peu flexibles. Toujours est-il que nous nous intéressons à NRPE parce que c'est le choix qui a été fait durant ce stage.

Le fonctionnement est simple : plutôt que d'appeler directement une commande particulière depuis Nagios, on appelle la commande de NRPE. Son premier argument correspond au nom de la sonde distante à exécuter. Les autres seront simplement transmis à cette sonde.

Sur la machine distante, on trouve un serveur NRPE (un simple démon), ainsi qu'un répertoire de sondes, les mêmes que pour Nagios mais locales. Une fois contacté, NRPE se

[6] KISS : « *Keep it Simple, Stupid est un principe philosophique qui désigne le fait que la simplicité dans la conception devrait être le but recherché et que toute complexité non-nécessaire devrait être évitée.* » Wikipédia

[7] La métrologie est la science de la mesure au sens le plus large. Dans ce contexte, elle désigne les données qui serviront à établir des graphiques.

[8] `/dev/manchots/volants` ?

[9] NRPE : Nagios Remote Process Executor (exécuteur distant de processus Nagios)

[10] SNMP : « *Simple Network Management Protocol, protocole simple de gestion de réseau en français, est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, superviser et de diagnostiquer des problèmes réseaux, matériels à distance..* » Wikipédia

charge donc d'exécuter la sonde demandée à distance et de renvoyer le retour de la sonde sans le modifier.

En étant directement localisé sur la machine distante, NRPE permet de faire des tests plus poussés : tester si un port est toujours en écoute n'est pas toujours une certitude que le processus répond toujours bien. NRPE peut donc vérifier que le processus est toujours lancé, mieux il peut tester directement le service (cas d'un serveur de base de données, sur lequel il pourra régulièrement se connecter sur une base test pour vérifier que ça fonctionne). Avantage significatif de NRPE : il crée un tube dans lequel toutes les vérifications sont permises, avec un seul port à ouvrir sur la machine.

Il fonctionne de façon sécurisée, puisqu'il permet de restreindre les adresses des machines autorisées à le contacter et fait transiter ses informations en TLS[11].

3.2.3 Une référence déjà choisie

Parmi les grands de la supervision libre, on recense actuellement :

- Nagios
- Zabbix
- Zenoss
- OpenNMS

Ainsi que tous les clones de Nagios.

L'analyse des différentes solutions de supervision fût courte. En effet, je me suis très vite aperçu qu'aucune solution de supervision n'égale Nagios dans sa réputation, sa communauté (près de deux milles sondes proposées) et sa flexibilité.

Nagios étant déjà en place pour l'intégralité des solutions de supervision actuelles de l'INPL, il aurait fallu un argument de poids pour décider d'une autre application. Celui-ci n'ayant pas été relevé, c'est donc Nagios qui assurera la supervision de l'ensemble du parc informatique.

3.2.4 Actualité politique

Tout n'est pas rose dans l'univers Nagios. Il est important de souligner que des tensions sont présentes dans la communauté. L'histoire remonte à quelques années, où l'auteur de Nagios décide d'en déposer le nom, et de créer son entreprise *Nagios Enterprise*. Depuis, tout se dégrade lentement entre cet auteur et la communauté impressionnante du projet.

Tant bien que, agacé par l'absence de réactions de l'auteur face aux propositions de la communauté, un clone (*fork*) a vu le jour : *Icinga*. Là où l'histoire manque de tact, et bien que la GPLv2[12] soit respectée, c'est que ce clone a été commandé par une société allemande,

[11] TLS : « *Le Transport Layer Security est un protocole cryptographique utilisé pour la sécurisation de couches de transport telles que HTTP. Il offre des mécanismes d'authentification, de protection en intégrité, et de confidentialité. TLS est le successeur de SSL (Secure Sockets Layer).* » Wikipédia.

[12] Licence de Nagios, éditée par le groupe GNU, qui impose notamment qu'un projet libre reste libre.

qui fait du support Nagios : le domaine d'activité naturel de Nagios Enterprise. Cet opportunisme a certes irrité l'auteur de Nagios, mais aussi sa communauté qui est restée plutôt fidèle.

Malheureusement, la communauté, qui a permis ce clone, a été plus encore ignorée par l'auteur. Pire, celui-ci a réclamé qu'on lui cède la propriété du nom de domaine *nagios-fr.org*, sous prétexte du dépôt du nom et que la communauté se soit permise de parler du clone honteux de façon trop peu offensive. Le dénouement de cette affaire est que la communauté lui cédera ce nom de domaine, lorsqu'elle aura fini sa migration vers *monitoring-fr.org*, sa nouvelle adresse. En retirant le nom de son logiciel à sa communauté française, Nagios pourrait bien avoir perdu un hexagone de soutien.

Entre temps, Nagios Enterprise a édité *Nagios XI*, au format privateur. L'entreprise a déclaré, en réponse à une lettre ouverte de la communauté, que la version libre resterait toujours soutenue, et que si Nagios XI est privateur, c'est parce qu'il utilise des modules privateurs. L'auteur ne remettra par contre pas en cause le dépôt de marque du nom du logiciel, considère que le code source n'a pas à être revu, et indique qu'il restera le chef, qu'il a toujours été depuis onze ans.

Pourquoi Nagios resterait parmi mes possibilités pour mon choix de solution de supervision, si ses jours son comptés ? Parce qu'il reste la solution la plus performante, et que si son avenir est compromis sous la forme actuelle, l'importance de sa communauté assure qu'un Nagios libre restera toujours en vie, qu'il s'appelle Nagios ou pas. Pardon, *Nagios (R)*.

Jean Gabès, l'auteur du livre que j'ai absorbé au début de ce stage, a récemment sorti un Nagios tout en python, qui affiche des performances très intéressantes. Il considère que le C était une très bonne idée il y a onze ans, mais que le python est plus souple et plus adapté. D'après ses dires il compte autant que faire ce peut rester lié au projet initial - malgré quelques refus - mais n'exclut pas l'idée d'un projet parallèle. Jean Gabès sera présent aux rencontres mondiales du Logiciel Libre 2011, et j'espère bien l'y rencontrer.

3.3 Centreon

3.3.1 De la bonne répartition de la supervision

Le parc informatique de l'INPL est grand. Si grand qu'il est difficilement concevable de prévoir de réunir l'intégralité de la supervision sur une même machine. Si le nombre de services actuellement supervisées pourrait le permettre, ce ne sera rapidement plus le cas, une fois qu'il sera aisé d'ajouter des machines ou des services à superviser. Il faut donc penser à déléguer les vérifications à des machines tierces, tout en gardant l'objectif de l'interface unique.

Après réunion, décision a été prise de répartir la supervision par sites géographiques. En effet, l'INPL est divisée en une dizaine de sites géographiques. Il est donc prévu qu'il y ait autant de serveurs, en plus du central qui supervisera du même coup les services communs, plus difficilement classables. Un autre choix aurait pu être de classer par type de service. Idéalement, la politique géographique permet de mieux répartir la charge et la disponibilité au niveau réseaux, évitant ainsi les goulots d'étranglement.

Un goulot d'étranglement il en reste un : la fameuse machine centrale. Nagios stocke ses données sous forme de fichiers textes, et ceux-ci sont particulièrement mal adaptés quand la quantité d'informations devient trop importante. Ainsi, un stockage en base de données serait plus confortable, autant pour la fluidité de parcours des données que pour la charge de la machine.

Un projet connu et reconnu dans le domaine de la supervision répond à ces problématiques : Centreon.

3.3.2 Une surcouche de Nagios

Centreon est outil de configuration de Nagios. Par le biais de technologies comme NDO[13], il se sert de MySQL pour stocker la configuration de Nagios. Au delà de celle-ci, il y stocke sa propre configuration. Finis les fichiers textes et les définitions patiemment écrites ou copiées-collées dans `vi`. Centreon propose une interface web conviviale (cf. figure 3.1) capable de reproduire fidèlement toutes les possibilités de Nagios, tout en en rajoutant lui-même (comme la génération de services automatique).

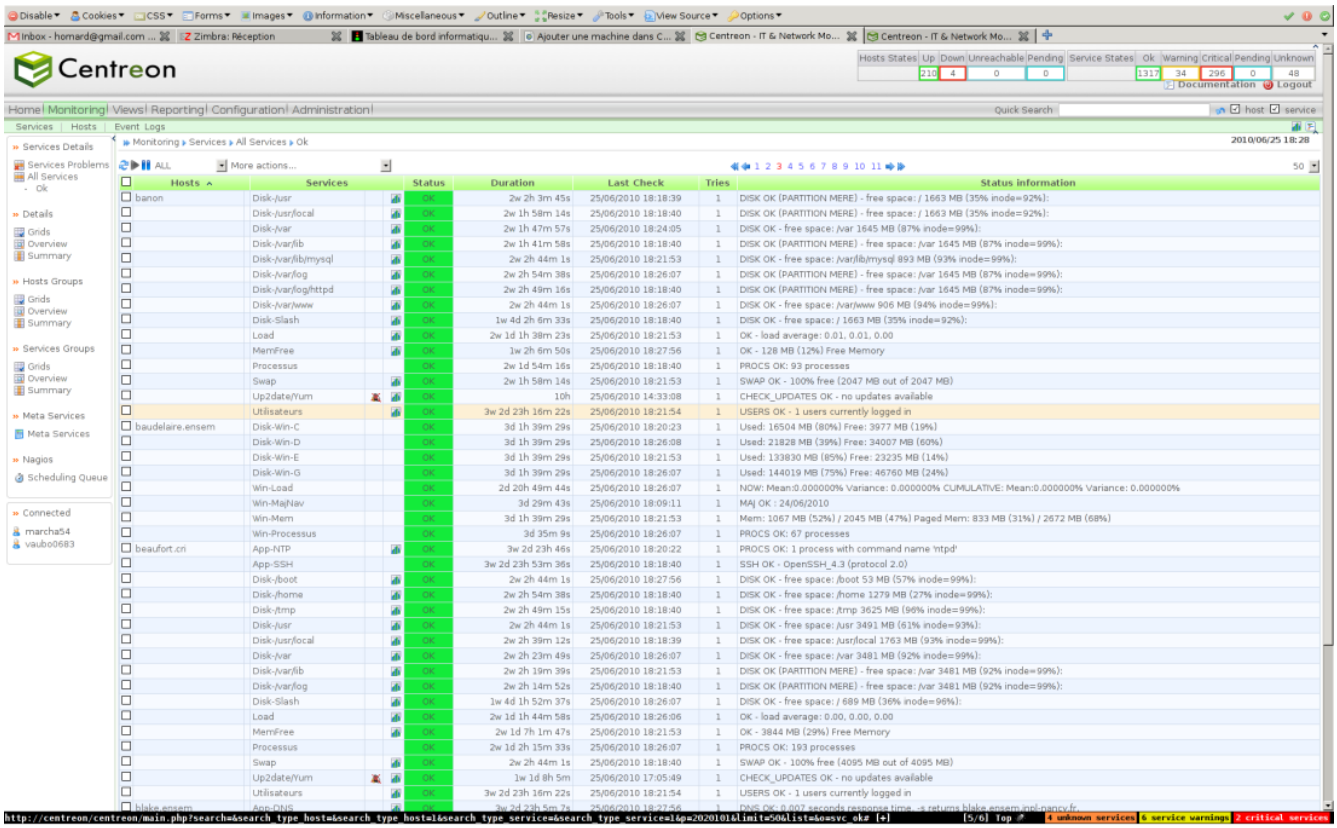


FIG. 3.1 – Page de supervision globale de Centreon

Outre l'aspect convivial, c'est aussi l'assurance de ne plus faire des erreurs de syntaxe inévitables, bien que ce soit au désespoir des *aficionados* de l'écran noir. Ecrite en PHP/MySQL, cette interface permet une utilisation bien plus fexible que celle de Nagios,

[13]NDO est un plugin pour Nagios permettant de centraliser les données dans une base données MySQL.

depuis toujours écrite en CGI C, langage dont la capacité d'adaptation au web a montré ses limites depuis longtemps. Voir figure 3.2.

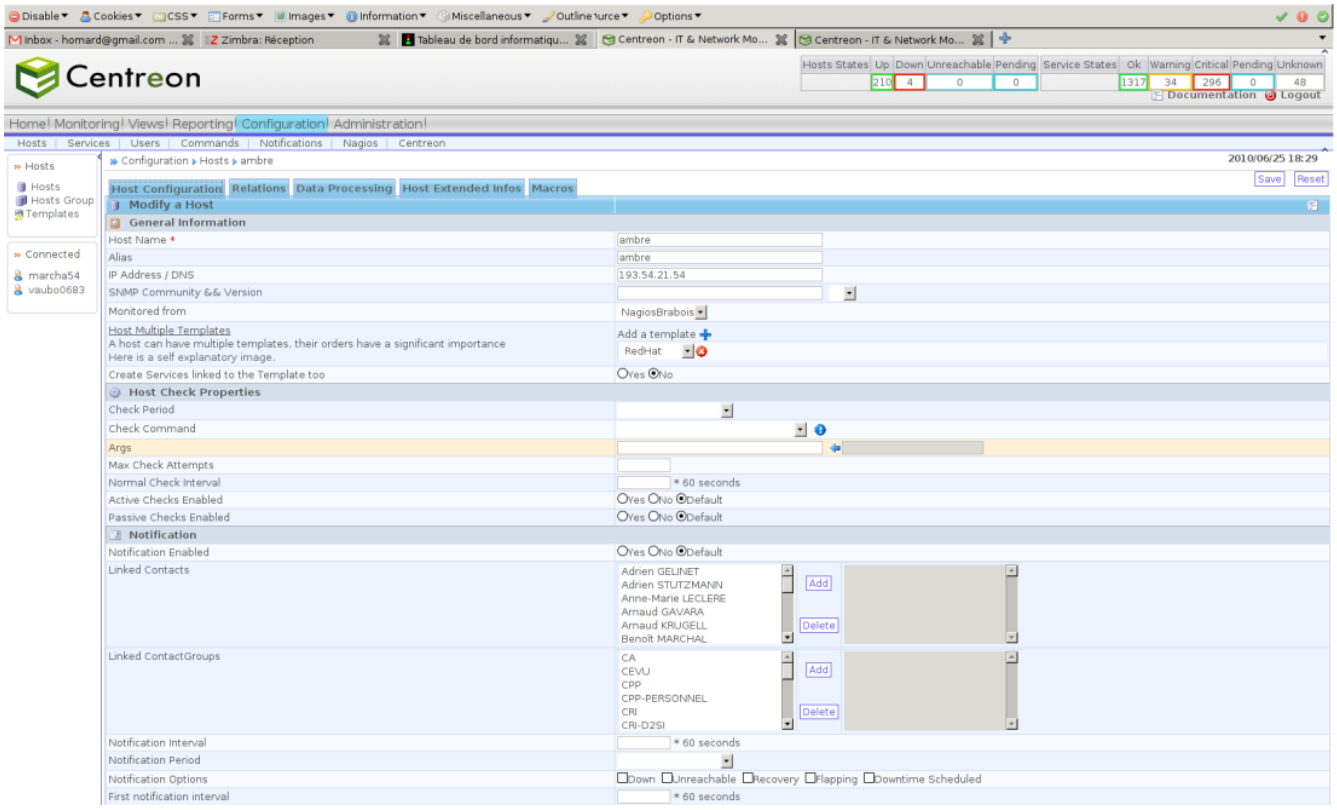


FIG. 3.2 – Ajout d'une machine dans Centreon

3.3.3 Des graphiques à la demande

Outre la supervision par le biais de Nagios, Centreon propose la gestion des données de métrologie. C'est un programme en tâche de fond (*CentStorage*) qui se charge de gérer la récupération et le stockage des données. Celles-ci sont injectées dans des fichiers au format RRD[14] que l'interface web consulte pour la génération des images de graphiques. Les données de métrologie peuvent être stockées en parallèle dans la base de données, permettant ainsi la régénération des fichiers RRD. Voir figure 3.3.

3.3.4 La gestion des satellites

Dans le jargon Centreon, un Nagios satellite est désigné sous le terme de *poller*. Centreon est en effet capable de gérer plus d'un Nagios à la fois. Une fois les Nagios distants installés et un minimum configurés pour bien communiquer, toute leur administration se fait depuis le Centreon. En déclarant une machine à superviser, depuis l'interface Centreon, il est possible de sélectionner le Nagios qui sera chargé de s'en occuper. Il sera plus tard possible de le changer à n'importe quel instant. Lorsque l'utilisateur demandera à Centreon de charger

[14]RRD : Round-Robin Database, c'est un système de sauvegarde de valeurs agrégées dans le but de produire des graphiques. Son principal avantage est que ses fichiers gardent une taille fixe

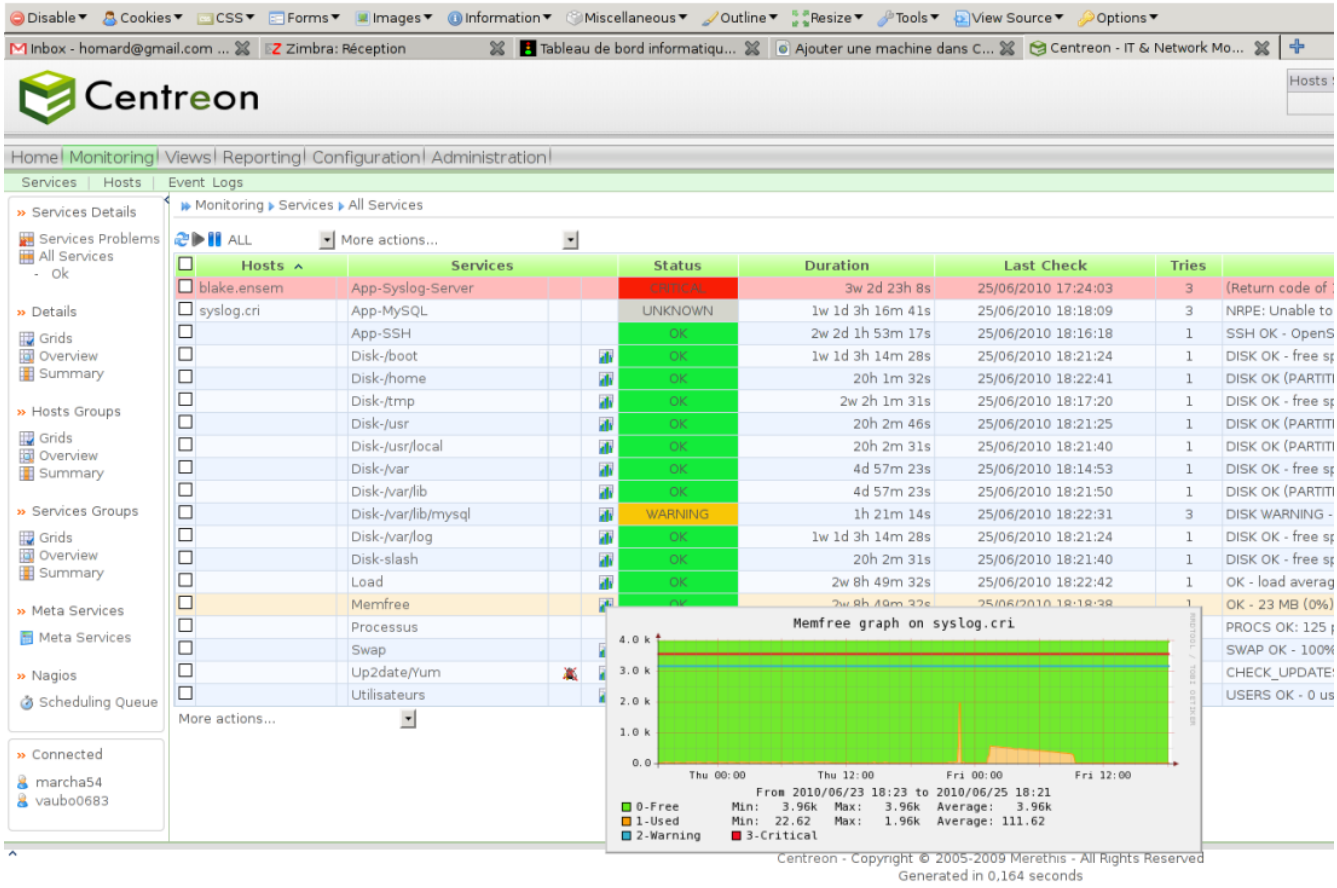


FIG. 3.3 – Des graphiques partout dans Centreon

la configuration, il enverra par SSH[15] tous les fichiers nécessaires aux serveurs satellites, et redémarrera leur service. Par le biais de NDO et d'un protocole spécifique à Nagios (NSCA), toutes les données récoltées de ces satellites sont récupérées puis stockées sur le Centreon, qui les rend alors disponibles, de façon transparente, depuis son interface (voir le schéma explicatif figure 3.4, qui sera expliqué en détails lors de la présentation).

4. Le grand changement

4.1 Une importation douloureuse

Ce qui devait n'être qu'un simple exercice de style s'est en fait révélé être un gouffre de temps. Centreon propose d'importer une configuration de Nagios, par de simples copiés-collés des fichiers de configuration, qu'il importe ainsi dans sa base de données.

Si l'offre est alléchante sur l'écran, elle le devient beaucoup moins après quelques essais... peu concluants. En effet, même en respectant scrupuleusement l'ordre des dépendances entre fichiers, Centreon n'importe pas la moitié des définitions qui lui sont données.

Sur la toile ou sur les plate-formes de documentation, seul un rapport de *bug* général-

[15]SSH : Protocol de communication sécurisé

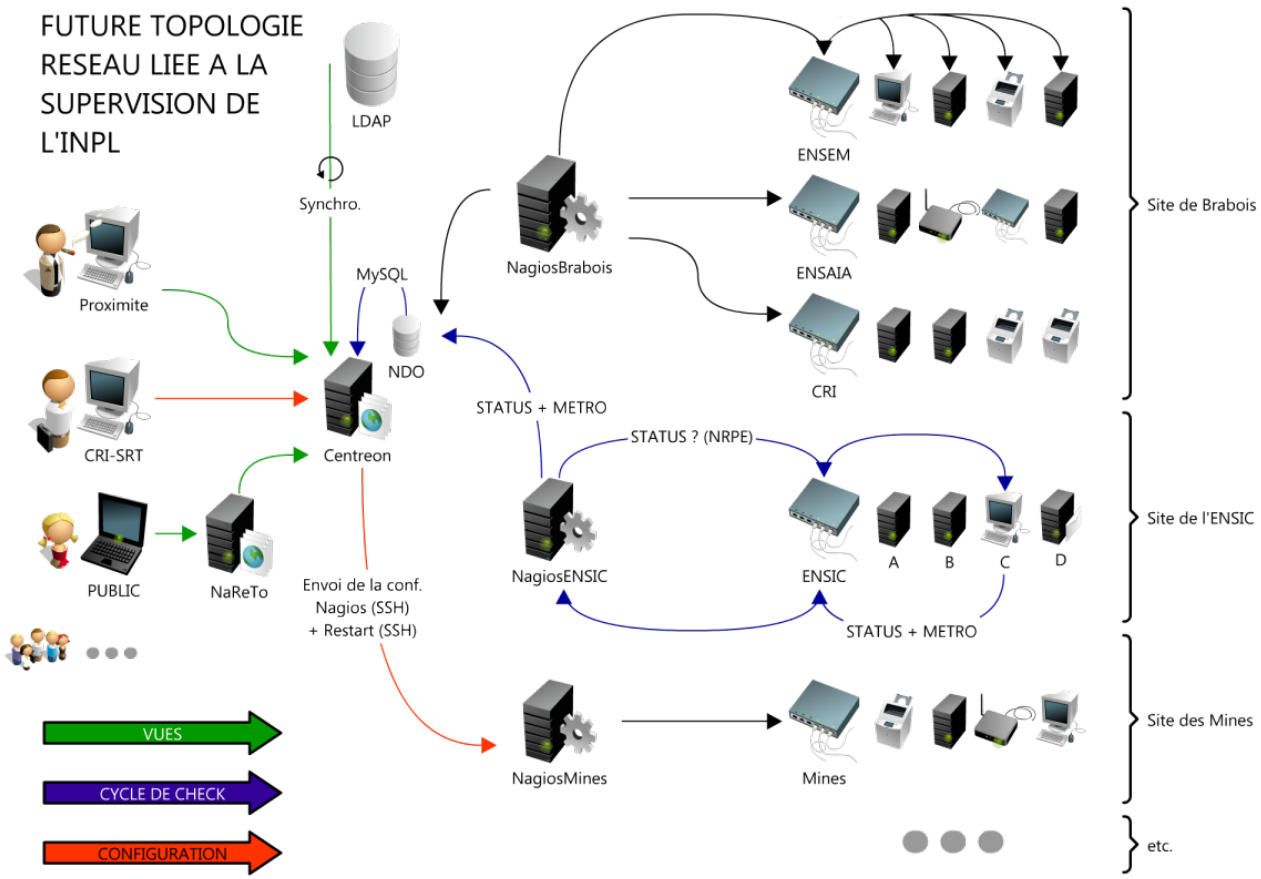


FIG. 3.4 – Schéma de la topologie réseaux de la future supervision

iste existe, regroupant quelques un des problèmes les plus visibles. Sur beaucoup de forums, des gens se plaignent de cet outil et l'évoquent comme excuse pour ne pas passer à Centreon.

Durant de nombreuses semaines, je me suis ainsi attelé à péniblement cibler les problèmes d'importation, leur trouver une logique, et réussir à les outrepasser.

Cette croisade s'est faite au travers d'un script (disponible en annexe B) destiné à adapter une suite de fichiers de configuration Nagios à l'import dans Centreon. Le script, rédigé en ruby, prend en paramètre la série de fichiers CFG du Nagios à convertir. De ceux-ci, il extrait les informations pour les nettoyer, les trier, les adapter.

Parmi ses actions :

- **Toute les définitions** (machines, services, commandes, les différents modèles, groupes, etc.) de l'ensemble des fichiers **sont extraites et triées**. Les fichiers finaux portent le nom du type de définitions qu'ils contiennent. Cette réorganisation est primordiale, puisqu'il est impossible sans elle d'injecter la configuration à Centreon dans l'ordre dans lequel il le préconise.
- En les réinjectant dans des fichiers différents, **les définitions sont reformatées** : ainsi, plus de commentaires, plus d'espaces ou de tabulations manquantes ou en trop qui pourraient permettre à Centreon d'ignorer une partie de la configuration (cas

réellement rencontré durant les essais).
- Si Nagios n'impose pas de description aux services, ça n'est pas le cas de Centreon. En l'absence d'un tel champ, il ignorera simplement sa définition. **Le script complète donc automatiquement toutes les définitions** de services, en leur ajoutant une description correspondant - dans l'ordre de préférence - au nom du modèle duquel elles héritent, au nom de la définition, ou bien, à défaut, à un nombre aléatoire. Pour ce qui est des modèles de services, c'est un attribut de nom qui est obligatoire, et donc à générer si besoin est.
- Nagios permet de lier un service à un groupe de service de deux façons différentes : passivement ou activement (du point de vue du service). Dans le premier cas, c'est le groupe qui définira ses membres, par le biais d'un attribut qui accueille une liste de machines. Dans le second cas, ce sont les services qui se déclarent eux-mêmes comme appartenant au groupe. Malheureusement, Centreon ne gère que la méthode passive. Le script a donc la charge de **capter tous les membres d'un groupe pour injecter leurs liens d'appartenance** directement dans la définition de celui-ci. Les groupes de machines sont modifiés de la même façon.
- La définition d'un contact, tout comme un service ou une machine, peut être basée sur un modèle. Nous parlons de Nagios, et non de Centreon, puisque celui-ci ne les gère pas. Ainsi, dans le cas des contacts, le script **retrouve le modèle et injecte directement sa définition dans la définition du contact**.

En accompagnement, un script bash (disponible en annexe C) permet de faciliter l'interaction entre l'interface de Centreon et cette génération de fichiers de configuration.

Bien que très laborieuse, cette reprise de l'existant a finalement réussi à aboutir. Mais c'était sans compter sur la lourdeur des mesures de sécurité.

4.2 La lourdeur de la sécurité

Une machine de supervision est une machine qui doit avoir un pied partout et qui se mêle de tout. Ainsi, elle est donc relativement difficilement intégrable dans un environnement complexe de jeux de droits et d'autorisations, avec lesquels il faut sans cesse négocier.

On peut regrouper les barrières en cinq catégories :

- **Les ACL[1] des VLAN[2] :** Le plan d'adressage IP lié aux commutateurs restrictifs du réseau demandent sans cesse à autoriser la communication entre réseaux, afin de pouvoir faire communiquer le Nagios responsable de la supervision d'une machine avec celui-ci. Les ACL sont gérées par un service informatique à plus bas niveau, le CIRIL, et doivent être chaque fois validées.
- **Les `iptables`[3] des machines supervisées :** Bien que le réseau fasse lui-même office de filtre, certaines machines s'imposent un pare-feu plus ou moins restrictif. Ainsi, il faut repérer (sans aucun moyen de le faire de façon automatisée) les machines concernées et leur ajouter les règles nécessaires et spécifiques à chaque cas. Il faut bien entendu avoir un accès à toutes ces machines, ce que je n'avais pas.

[1] ACL : Access Control List
[2] Partie segmentée et isolée d'un réseau.
[3] Commande principale du logiciel NetFilter, il s'agit par extension du pare-feu intégré au noyau Linux.

- **Les restrictions applicatives :** Certaines vérifications portent sur des services comme MySQL. Pour vérifier correctement ces services, il est d'usage de créer une table vide avec un utilisateur capable de ne faire qu'un `SELECT` dessus. Mais cet utilisateur est rattaché à un hôte, et si il était bien autorisé pour une ancienne machine de supervision, ça n'est plus le cas pour cette nouvelle. Il faut donc repérer ces machines et ces services pour les configurer à la main.
- **`xinetd` :** Certaines machines utilisent `xinetd` pour le contrôle des ports de la machine. Il faut donc repérer les cas concernés, et autoriser textuellement (dans */etc/hosts.allow*) la machine de supervision à se connecter à la machine, notamment pour le port du NRPE.
- Enfin, **les *nrpe.cfg* :** Lorsqu'un service est vérifié en NRPE, celui-ci est accompagné du nom d'une commande. Ce nom de commande doit correspondre à un nom dans le fichier *nrpe.cfg* de la machine supervisée pour s'exécuter correctement. En mutualisant les configurations, nous mutualisons les services et donc les commandes. Ainsi, parmi les commandes qui se recoupent, un seul nom a été conservé, et ne correspond donc plus sur les machines liées au nom perdant. Il faut donc harmoniser tous les fichiers de configuration de NRPE.

4.3 La grande harmonie

4.3.1 Un *melting pot*

Rassembler plusieurs configurations différentes, c'est rassembler plusieurs philosophies différentes. Chacun des anciens Nagios était administré par une personne différente, et tous n'ont pas adopté les mêmes stratégies.

Sur Centreon, trois stratégies peuvent être distinguées :

Stratégie des modèles de services

Lorsqu'on ajoute une machine, aucun service ne lui ait attribué. Par contre on va créer ses services un par un en se basant sur des modèles. Ainsi, il n'y a rien à faire pour créer le service, à part adapter quand il y a besoin (par exemple, adapter les seuils).

Avantage Souple et simple, il y a juste a créer les services nécessaires sans les paramétrer et les adapter si besoin.

Inconvénients Impossible d'ajouter/supprimer/configurer un service a posteriori pour un lot de machines. Il faut créer les services un par un, avec le risque d'en oublier.

Stratégie des services associés à un modèle de machine

Centreon permet, lorsqu'on ajoute une machine en l'associant à un modèle de machine, de créer automatique la série de services qui lui sont associés.

Avantages Il n'y a rien à faire en plus d'ajouter la machine, et il est impossible d'oublier

un service. On peut ensuite adapter manuellement les différents seuils, et éventuellement supprimer les services qui ne sont pas utiles pour la machine.

Inconvénients Impossible d'ajouter/supprimer/configurer un service a posteriori pour un lot de machines.

Stratégie des groupes de machines

On associe la machine à des groupes en l'ajoutant. Ainsi, chaque groupe étant relié à une série de services qui le concerne, la machine en hérite automatiquement.

Avantages Pratique et cohérent. Possibilité d'ajouter/supprimer/configurer des services a posteriori pour un lot de machines. Possibilité de classer facilement les machines par groupe, et donc par caractéristique.

Inconvénients Manque de souplesse, puisqu'en adhérant à un groupe, elle héritera de tous ses services sans exception. Il sera impossible d'adapter les seuils de ces services pour la machine.

La bonne stratégie ne saute pas aux yeux. Certains Nagios en place tendaient vers la première stratégie, d'autres vers la troisième (la seconde n'étant pas possible avec Nagios). Le choix a finalement du se faire entre efficacité et finesse. Benoît a tranché cette question, considérant qu'il y avait trop de machines et de services pour risquer de devoir ajouter ou supprimer des services à la main, à des centaines de machines. La contrepartie de ce choix, est d'avoir des services qui ne sont pas pertinent pour certaines machines, et des seuils qui ne sont parfois pas adaptés pour toutes. C'est donc la solution des groupes de machines qui a été retenue.

En fonction de cette décision, je me suis mis à la tâche : concevoir une politique de groupe pratique, cohérente et évolutive.

Ajouter une machine dans Centreon en reviendra donc surtout à l'associer avec les bons groupes. Certains groupes sont incontournables comme *Systeme_Linux* et d'autres plus spécialisés comme *Application_Smbd*. La seule difficulté sera donc de ne pas en oublier. D'autant plus qu'il est impossible de faire des groupes de groupes. Une machine Red Hat 5 devra donc être associée au groupe *Systeme_Linux* et *Systeme_Linux-RedHat5*. Des vérifications pouvant être faîtes pour tous les GNU/Linux (ex. : vérification de la partition */var*), ou pour tous les Red Hat (ex : vérification des mises à jour, d'où la nécessité de différencier la version, pour Red Hat). Plus tard, ajouter une vérification sur tous les GNU/Linux à la fois se fera aisément puisqu'en associant un service directement au groupe, tous ses membres en bénéficieront.

Une fois cette stratégie décidée, il a fallu faire disparaître l'ancienne : les multiples modèles de services, devenus inutiles.

4.3.2 Changement de stratégie

4.3.2.1 Une stratégie de groupes

Trouver une solution efficace et pertinente pour rassembler les machines en groupes eux-mêmes reliés aux bons services n'a pas été simple.

Outre le fait que je me suis confronté à énormément de noms de services, de processus et de machines que je ne connaissais absolument pas, il a fallu trouver une solution pour automatiser. Ceci s'est fait par le biais d'un premier script (disponible en annexe D). Celui-ci se charge de rassembler les machines qui dépendent d'un même service, en un groupe de machines lui-même dépendant de ce service. Pour pouvoir ensuite s'y retrouver, le nom de ce nouveau service porte le nom du service associé. Si deux groupes de machines fraîchement créés contiennent les mêmes machines, alors les groupes sont fusionnés en un seul, dépendant de l'ensemble des services. Au terme de l'exécution du script, des groupes de machines sont créés, avec leurs services associés.

Après le travail de la machine, s'ensuit toujours le travail de l'homme. Cette façon de procéder a permise une intelligence artificielle, mais nécessite un passage en revue des groupes de machines pour considérer ceux qui peuvent encore être fusionnés. Il faut également déterminer le nom et la description de ces nouveaux groupes, aux noms peu commodes.

Pour ce faire, il a fallu copier des machines d'un groupe de machines dans un autre (ex : cas de tous les *Systeme_Linux*-OS qu'il faut copier dans *Systeme_Linux*), et parfois fusionner des groupes de machines, en déplaçant toutes les machines du premier vers le second.

Des outils bien pratiques ont été créés pour l'occasion : *cpHostgroup.pl* et *mvHostgroup.pl* (annexes F et E). Il suffit de renseigner le nom des groupes de machines source et destinataire, et les machines naviguent de l'un à l'autre. Idéal pour concevoir l'arborescence des groupes : à ce stade, une ossature fiable est créée.

A des fins de vérification, et pour pallier à ma méconnaissance des nombreuses machines, Benoît s'est chargé de vérifier la pertinence des groupes d'hôtes à partir de trois documents générés depuis la base de données.

Exemple de génération de documents pour une vérification papier :

```
for i in $(mysql centreon2 -e "SELECT hg_name FROM hostgroup
   ORDER BY hg_name")
do
  mysql centreon2 -H -e "SELECT host_name '$i'
                FROM host, hostgroup_relation
                WHERE host_host_id=host_id
                AND hostgroup_hg_id=(SELECT hg_id
                    FROM hostgroup
                    WHERE hg_name='$i'
                    ORDER BY host_name)";
done > ben_hostgroups.html
```

4.3.2.2 Suppression des modèles

Reste que tous ces services associés à des groupes de machines sont eux-mêmes dépendant de modèles. Ces modèles ne leur servent à rien, puisque dans la plupart des cas il n'y a qu'un service par modèle. C'est une perte de temps que de passer de l'un à l'autre pour consulter la configuration. Par le biais d'un script (disponible en annexe G), tous les services ont directement récupéré les informations de leurs modèles (y compris les associations avec les contacts et groupes de contacts).

Dernière étape pour ne plus parler des modèles de services : les supprimer. Ce qui a été fait par le biais d'un autre script (disponible en annexe H), capable de repérer les modèles de services inutilisés pour les supprimer. A noter que ceci aurait pu être fait dans une seule grande requête SQL, mais force est de constater que la puissance de MySQL égale celle d'une fourmi sans pattes : « *Currently, you cannot update a table and select from the same table in a subquery.* ». MySQL ne peut pas modifier et consulter une table en même temps, ce qui est possible dans à peu près tous les autres systèmes de gestion de bases de données et qui m'a fait perdre beaucoup de temps.

4.3.3 Du ménage dans les commandes

4.3.3.1 Suppression des intermédiaires et des inutiles

Chaque service est associé à une commande de vérification. Au sens Nagios (et donc Centreon) du terme, une commande correspond à une définition qui elle-même correspond à une sonde. Faire du ménage dans les commandes, c'est donc avant tout faire le ménage dans les définitions inutiles. Ici encore, c'est une histoire de stratégie. Dans une partie de la configuration importée, la stratégie était de créer une commande par service.

Exemple : si un service consiste à vérifier la présence d'un processus *PA* sur la machine distante, le service *A* dépend d'un modèle *A'* qui appelle une commande *A''* qui elle-même dépend du fichier de sonde *check_nrpe* à qui elle passe *check-PA* en paramètre. Et ça n'est pas fini : une fois que *check_nrpe* contacte la machine distante avec *check-PA*, le NRPE consulte son fichier de configuration pour retrouver la définition de *check-PA* et donc le fichier de sonde associé. Toutes ces étapes sont donc dupliquées pour chaque service. Retracer tout le chemin d'une commande en récoltant au gré des étapes des options qui s'ajoutent, s'échangent ou se renomment, est fastidieux.

L'étape des modèles a déjà été réglée plus haut. Au niveau de la commande invoquée par le service, l'idée est de faire en sorte que le service *A* appelle directement la définition de la commande *check_nrpe* en lui passant en paramètre *check_processus* et *PA*. Il n'existe plus de définition de commande *A''*, et une seule ligne du fichier de configuration du NRPE distant pourra couvrir les besoins de tous les services de vérification de processus.

Plusieurs grandes requêtes SQL plus tard et de patients copié-collés, l'idée est mise en pratique. Tous les services qui utilisent NRPE contactent maintenant directement l'agent en utilisant des définitions génériques, associées à tous les paramètres à la fois.

En conséquence, beaucoup de définitions de commandes ont été à supprimer. Un script (disponible en annexe I) s'en ai chargé.

La suppression des anciennes commandes et l'ajout des commandes génériques dans les fichiers de configuration des NRPE distants nécessitent de mettre à jour l'ensemble des machines.

4.3.3.2 Génération d'un *nrpe.cfg*

Le fichier *nrpe.cfg* correspond au fichier de configuration de NRPE sur les machines distantes. Ce fichier contient la définition de toutes les commandes que les machines autorisées à communiquer avec lui ont le droit d'invoquer. Le nom de la commande est subjectif, c'est sa définition qui définira la sonde à utiliser.

Aucune politique n'était en place pour ces fichiers : chaque machine possédait un fichier plus ou moins unique, plus ou moins complet, et plus ou moins pollué. Avec mon tuteur et Benoît, nous avons pris la décision de concevoir deux uniques fichiers de configuration NRPE (GNU/Linux et Windows) qui contiennent toutes les définitions de commandes. Ainsi, même si tous ne servent pas pour toutes les machines, il est possible de se mettre à les utiliser n'importe quand sans devoir les ajouter sur les machines distantes. La collection de sondes doit donc, elle aussi, être uniforme sur toutes les machines.

Afin de pouvoir proposer un fichier de configuration réduit à ce qui est nécessaire, et pour retrouver ce qui, au final, est utile, c'est un petit script (disponible en annexe N) qui a été utilisé pour fournir la première piste.

Son rôle consiste à parcourir tous les services définis dans Centreon, et à récupérer les arguments passés à ceux qui utilisent une commande *check_nrpe*. Le premier paramètre correspond toujours à la commande distante à lancer, et les autres aux options. Ainsi, il est aisé de générer un fichier de configuration au bon format avec une liste exhaustive des commandes appelées par NRPE.

Le seul point d'ombre dans cette génération, c'est la partie droite de la définition : impossible de deviner le nom des sondes associés, ni même le nom de leurs options. C'est donc à la main, et au grès de nombreuses récupérations d'anciens fichiers de configuration, que j'ai pu reconstituer le fichier final.

Certaines commandes ayant besoin d'être appelées en `sudo` pour avoir plus de droits, il ne faut pas oublier de les préciser. Bien entendu, il faut également que le fichier de configuration de `sudo` accepte ces prises de droit, sur chacune des machines supervisées.

4.3.4 Tri et sélection des sondes

Un problème s'est posé pour ce qui est des fichiers binaires : des machines sont en trente-deux bits, et d'autres en soixante-quatre bits. De plus, certaines sondes sont dépendantes de librairies dont la version doit parfaitement correspondre (ex : openssl). Impossible donc de déployer les mêmes fichiers pour toutes les machines.

Les gestionnaires de paquets auraient pu être une solution. Non pas en utilisant les dépôts officiels, puisque certaines distributions ne les proposent pas tous, mais en créant un dépôt privé. Ainsi, on aurait pu imaginer un `.deb` et un `.rpm` (l'un pouvant se transformer en l'autre) qui contiennent directement les bons binaires pour la bonne architecture, et qui

auraient relancés les processus NRPE en fin d'installation. Ceci aurait permis aussi de pouvoir rajouter une sonde sur toutes les machines GNU/Linux en mettant simplement à jour un dépôt et en demandant une mise à jour des machines. Pour ne pas avoir à compiler les binaires pour chacune des architectures, il aurait été envisageable d'embarquer les sources et que le gestionnaire de paquets les compile automatique à l'installation. Ceci aurait permis de régler un autre problème : les dépendances des scripts perl, qui auraient pu être gérées directement en tant que dépendance du paquet.

J'ai fait cette proposition. Malheureusement, la mise en perspective de monter des machines de dépôt APT et RPM, de configurer toutes les machines pour qu'elles piochent dedans, d'ouvrir tous les droits nécessaires pour que cela fonctionne, et de gérer les différences entre Red Hat 4 (qui utilise `up2date` au lieu de `yum`) et les systèmes basés sur `yum`, ont rendus mon auditoire plutôt frileux. Je pense que si le stage avait été plus long, ils auraient envisagé cette solution, qui résout le problème fondamental de l'ajout d'une sonde sur toutes les machines (ainsi que la mise à jour correspondante du fichier de configuration NRPE). A terme, l'idée serait probablement d'utiliser une solution de déploiement de configuration de grande ampleur, comme le logiciel libre `puppet`.

La solution retenue a donc été la plus rustique : la compilation sur les machines distantes. A partir du fichier de configuration de NRPE généré, j'ai donc pu me mettre en quête des sources de toutes les sondes nécessaires, pour les rassembler en un seul dossier, qui devra être déployé.

4.3.5 Déploiement des machines distantes

4.3.5.1 Cas des GNU/Linux

Une fois les commandes épurées, sélectionnées et repérées, il reste à harmoniser les machines distantes. Pour que NRPE réponde correctement dessus, il faut que les fichiers de configuration contiennent bien toutes les définitions nécessaires, et que toutes les machines possèdent toutes les sondes.

N'ayant pas accès à la plupart des machines, c'est avec mon tuteur que nous avons réglé ce détail. Ainsi, pour chacune des machines GNU/Linux, nous avons envoyé le nouveau fichier de configuration NRPE ainsi que l'ensemble des fichiers de vérification, et relancé les processus NRPE. Plus précisément, et pour que les Nagios actuels continuent de fonctionner malgré les changements de définitions, les définitions ont été ajoutées aux configurations existantes. Au niveau de Centreon, toutes les commandes appelées par NRPE ont été préfixées par *cent_* afin d'être certains de ne pas créer de collisions.

A l'occasion d'un premier déploiement de configurations lié aux autorisations sur les machines, j'ai eu l'occasion d'écrire un script de déploiement (disponible en annexe K), qui sera utile par la suite à mon tuteur.

Pour ce qui est de la compilation des sondes envoyées, de l'installation de librairies perl ou la configuration du `sudo`, c'est *cluster ssh* qui a été utilisé. Ce logiciel, très utilisé par mon tuteur et Benoît, permet d'ouvrir simultanément des terminaux distants par dizaines et d'écrire dans tous à la fois. Cette solution est pratique, mais ne m'a pas convaincue. Elle correspond typiquement aux désillusions de l'informatique dans un cadre professionnel, qui

ne donne pas le temps de mettre en place des solutions sérieuses et satisfaisantes.

Afin de faciliter la vie à mon tuteur avec qui j'ai fait ce déploiement, j'ai écrit un script (disponible en annexe L) qui retourne une ligne de commande `cssh`, avec toutes les machines supervisées par Centreon qui font partie du domaine d'une école donnée, et qui répondent au `ping` sur le port vingt-deux.

4.3.5.2 Cas des Windows

Le monde n'étant pas parfait, il n'y a pas que des machines libres de supervisées. Il a donc fallu prendre les Windows en compte. Ceux-ci nous ont posés plus de problèmes (...), pour différentes raisons. Si le SSH est une évidence dans le monde libre, il l'est moins dans ce milieu. Comment déployer facilement et automatique des fichiers sur tout un lot de machines Windows, sans authentifications ? Différentes solutions ont été proposées, dont le passage par un serveur FTP[4]. Le problème étant qu'il faut aussi relancer les processus NRPE.

Après mûre réflexion, la solution était devant nos yeux : la meilleure façon d'administrer du Windows est encore de passer par du GNU/Linux[5]. Avec la précieuse aide de Pierre, qui a passé beaucoup de temps à m'aider dans cette tâche, nous avons conçu un script en bash (disponible en annexe M, sans les scripts batch), relié à une collection de fichiers Windows exécutables écrits pour l'occasion, destinés à effectuer les différentes opérations sur le serveur.

Ce script a très vite eu une seconde fonction : selon les cas, NRPE était installé sur le disque C : ou D :. Le fichier de configuration n'était donc pas toujours situé au même endroit. Plutôt que de simplement tester sa localisation pour adapter automatiquement le script, Pierre a souhaité que celui-ci soit capable de désinstaller le NRPE lorsqu'il était sur le D : pour le réinstaller sur le C :. Les mêmes manipulations sont ensuite effectuées pour tous, en ayant au passage harmonisé les configurations.

L'algorithme conçu avec Pierre consiste, pour chaque machine, à monter les partitions Windows localement sur la machine bastion depuis laquelle le script est lancé. Puis, chercher dans ces répertoires dans quelle partition se trouve le fichier de configuration, pour en déduire le disque d'installation de NRPE. Si le dossier correspondant est celui du point de montage du D :, NRPE est arrêté puis désinstallé, les fichiers sont déplacés et éventuellement renommés et NRPE est réinstallé. Les actions qui ne peuvent être faîtes directement avec des commandes UNIX sont faîtes à partir de scripts exécutés sur la machine distante avec le logiciel GNU/Linux `winexe`. Les sondes étant également déplacées, le script doit adapter le nom du disque et des dossiers des anciennes définitions de commandes dans le fichier de configuration (dans lequel les chemins sont absolus). Différents autres soucis se sont posés au cours de cette migration. Encore une fois, il a fallu demander toutes les autorisations nécessaires, pour les nombreux ports devant être accessibles depuis la machine d'exécution du script.

[4]FTP : File Transfer Protocol

[5]Je prie mes lecteurs adeptes du modèle privateur de bien vouloir m'excuser pour cette réflexion, reflet de mon expérience, mais peut-être pas de celles d'un expert Microsoft.

4.3.6 Harmonisation des noms et des adresses

4.3.6.1 Noms de domaines vers IP

Concernant la problématique des machines désignées par leurs adresses IP ou leurs nom de domaine, une nouvelle problématique s'est posée. Il faut harmoniser c'est certains, mais dans quel sens ?

Avec uniquement des noms de domaine, il est inutile de retoucher la configuration de Centreon en cas de déplacement d'une machine : le serveur de nom de domaines lui donnera toujours la bonne adresse. Par contre, si ce dernier a un problème, plus aucune machine ne peut répondre. Uniquement avec les adresses IP cet inconvénient ne se pose pas, mais cet avantage non plus.

Après concertation avec mon tuteur et Benoît, nous avons pris la décision de ne s'appuyer que sur les adresses IP. J'étais partisan de cette solution, considérant qu'une supervision ne doit pas être dépendante du serveur de noms de domaine, et que celui-ci peut ne pas toujours pointer sur même machine (cas des répartiteurs de charges), rendant ainsi bancale la supervision.

Afin de transformer tous les noms de domaines en adresses IP, un énième script (disponible en annexe **??**) a été composé. Celui-ci parcours toutes les définition de machines définies par un nom de domaine, lance une requête DNS pour en récupérer l'adresse IP et remplace l'un par l'autre dans la base de donnée.

4.3.6.2 Noms des machines

Elles sont parfois en majuscules, parfois renommée en fonction du seul service qui leur est attribué, parfois abrégées, parfois associées à leur domaine, parfois pas... Pour obtenir une configuration propre, il faut de nouveau définir une stratégie. Toujours en concertation avec mes responsables, nous avons pris la décision d'utiliser le nom de domaine des machines en tant que noms dans Centreon. Pour éviter d'obtenir des noms trop longs sur l'interface, les *.inpl-nancy.fr* sont supprimés.

Les noms de domaine des machines étant bien conçus, la plupart des noms se retrouvent donc sur le schéma *machine.ecole*, correspondant au nom de domaine *machine.ecole.inpl-nancy.fr*. Un des problèmes récurrents étant de savoir sur quel domaine se trouve une machine, le Centreon peut maintenant directement apporter la réponse, en fournissant le nom de machine qui lui a été attribuée. Si une machine n'appartient pas au domaine *.inpl-nancy.fr.* son nom de domaine complet est utilisé.

Mes lecteurs concentrés et avertis ont certainement déjà froncé un sourcil. En effet, difficile de déterminer automatiquement le nom de domaine d'une machine à partir de son adresse IP. Le cas d'un serveur web qui héberge plusieurs dizaines de noms de domaines est flagrant. Alors comment choisir de façon automatisée ? C'est un peu comme administrer une machine Windows, on demande à plus performant : Benoît, qui connait les machines.

J'ai donc écrit un script (disponible en annexe O), conçu non pas pour automatiser mais pour assister. En parcourant toutes les IP et noms actuels, il cherche à déterminer si il peut faire le travail tout seul. Concrètement, il fait une requête DNS inverse sur l'adresse l'IP. Si

celle-ci lui renvoie plusieurs résultats, il cherche si un parmi eux contient le nom actuel de la machine. Si c'est le cas, il supprime éventuellement le suffixe INPL et met à jour tout seul. Si ça n'est pas le cas, il propose à l'utilisateur le premier nom de machine trouvé, avec son nom actuel. L'utilisateur peut alors valider pour accepter le nom proposé, ou en renseigner un différent. Il arrive parfois, dans le cas des sondes de température par exemple, qu'une machine ne possède aucun nom de domaine. Le script demandera donc à l'utilisateur d'en renseigner un jusqu'à ce qu'il l'ai fait.

Benoît s'est volontiers prêté à ce nouveau petit jeu, harmonisant du même coup la politique des noms de machine.

4.3.6.3 Indicateur de système d'exploitation

Parce que l'interface de Centreon n'est pas si bien faîte que ça, on ne voit pas du premier coup d'oeil, dans la liste des machines, celles qui appartiennent à tel ou tel système d'exploitation. Pour palier à ce problème et obtenir une liste agréable et pratique, j'ai décidé de jouer sur les modèles de machines, qui eux sont visibles partout.

Ainsi, en ajoutant une machine, il est de bon ton de choisir le modèle correspondant au nom du système installé. Ceci pouvant permettre aussi de faire varier différents paramètres de vérification ou de notification selon le système.

Un script (disponible en annexe P) a été spécialement écrit pour ça, il permet, à partir d'un nom de groupe de machines, d'associer toutes ses machines au modèle donné.

4.4 Intégration de nouveautés

4.4.1 Une authentification centralisée

4.4.1.1 Les solutions de l'INPL

Comme toutes les universités, l'INPL possède un annuaire *OpenLDAP*. Dans les faits elle n'a pas le choix, puisque c'est une obligation gouvernementale.

Du côté des Windows, et c'est une nouveauté qui a été décidée durant l'une des réunions de pôle auxquelles j'ai assisté avec mon collègue stagiaire, la connexion se fait par une *Active Directory*. Peu de temps avant notre départ, un *Kerberos* venait de donner des résultats concluants, permettant de gérer l'authentification de façon unique, vers le OpenLDAP comme vers l'Active Directory. Pour s'authentifier d'un peu partout en ne renseignant leurs identifiants qu'une seule fois, les utilisateurs utilisent de façon transparente une solution de CAS.

L'interface de Centreon proposant une authentification de mot de passe et une gestion des utilisateurs, il est souhaitable qu'elle se base sur un des annuaires afin d'être en permanence à jour.

4.4.1.2 Le cas Centreon

Si Centreon propose parmi ses menus une configuration d'annuaire LDAP, il ne faut pas s'y fier : c'est une blague.

Autrement dit, après tests de la dite fonctionnalité, Centreon ne sait pas faire grand chose avec LDAP. Il sait s'y connecter. Mais n'est capable, ensuite, que de permettre l'importation des contacts, et de recopier toutes les données, en dur, dans sa base de données. A l'exception d'une chose toutefois : le mot de passe, qu'il fait vérifier, à chaque connexion, par l'annuaire.

Une fois le premier import effectué, il n'y a plus aucune synchronisation entre la base de donnée et l'annuaire. Si un utilisateur est ajouté ou supprimé de l'annuaire, rien ne sera répercuté sur Centreon. Le combat est le même si un utilisateur change son adresse de courriel : il recevra toujours ses notifications Nagios sur son ancienne adresse. Pire, il n'y a aucun moyen de mettre à jour la base Centreon en réimportant régulièrement l'annuaire LDAP : celui-ci ignorera l'import si un utilisateur existe déjà (sans le mettre à jour) et ne supprimera aucun utilisateur si il n'existe plus dans l'annuaire.

Difficile de trouver une utilité à une telle fonctionnalité, puisque si une entreprise à opté pour une solution d'annuaire, ça n'est pas pour dupliquer ses contacts un peu partout. Après enquête, une fonctionnalité de CAS est prévue pour Centreon, mais il n'y a encore aucun morceau de code de mis en ligne, et aucune date indiquée.

Développer cette fonctionnalité aurait été la meilleure des solutions, me permettant de contribuer au projet. Encore une fois, la durée du stage et le temps que me prend l'intégration ne me le permettent pas. Quant à modifier le Centreon de l'INPL pour que cela fonctionne, cela signifie qu'il n'est plus envisageable de faire simplement les mises à jour.

4.4.1.3 Développement d'un palliatif

La solution que j'ai proposée est de développer un script qui se charge de synchroniser, de façon externe, la base de données de Centreon avec l'annuaire. Ainsi toutes les nuits, ce script s'exécute pour mettre à jour la base de données. Il ajoute un utilisateur si il n'est pas présent dans l'annuaire, et le supprime si il est dans la base mais n'est plus dans l'annuaire. Si des informations du contact ont changées, il les met à jour. En précisant un `dn` pour chaque utilisateur, on exploite la seule fonctionnalité intéressante de Centreon vis-à-vis du LDAP, qui est la récupération de mot de passe. Ainsi, inutile de le recopier, et donc de donner des droits d'administrateur LDAP au script.

Afin de profiter au mieux de l'annuaire, les groupes sont aussi importés. Pour chacun des utilisateurs de l'annuaire, les groupes auxquels il appartient sont créés en tant que groupes de contact dans Centreon. Ainsi, si après un premier import on attribut le groupe de contact CRI-SRT comme groupe destinataire des courriels de notification, et qu'une personne est ajoutée dedans dans l'annuaire LDAP, il sera automatiquement parmi les destinataires, après synchronisation.

Dans l'optique de créer les vues métier, ces mêmes groupes sont créés au niveau ACL (pour Centreon, les ACL correspondent aux autorisations des utilisateurs non administra-

teurs vis-à-vis de son interface). Les membres de ces groupes peuvent ensuite avoir un affichage personnalisé de l'interface à leur connexion.

Le script crée les relations entre les utilisateurs et les groupes, et relance les Nagios distants si besoin est.

Le script (disponible en annexe Q) est lancé tous les soirs à 21h00 juste avant les sauvegardes, sur la machine du Centreon.

4.4.2 Question de point de vue

4.4.2.1 Les vues métier

Le lecteur attentif aura compris en lisant la fin du paragraphe précédent, que le problème est déjà quasiment réglé.

Puisque les groupes LDAP ont été importés dans Centreon, il y a fort à parier qu'on y retrouve les groupes nécessaires à l'élaboration des vues métier. En effet, un des objectifs étant de faire une vue pour le service de proximité, le groupe CRI-PROXIMITE tombe à pic.

Au niveau de Centreon, il suffit donc de créer une ressource d'accès, reliée uniquement aux machines des salles de cours, aux imprimantes et aux bornes d'accès wifi. Au niveau menus de Centreon, on crée une ressource qui n'a accès qu'à la partie supervision, sans la configuration ou l'administration. Enfin, on relie ces deux ressources au groupe d'accès CRI-PROXIMITE créé par la synchronisation LDAP. Si un utilisateur du service de proximité s'identifie avec ses identifiants LDAP, il n'aura accès qu'à un Centreon épuré de tout ce qui ne le concerne pas.

Par défaut, lors de la synchronisation LDAP, le script place tous les utilisateurs du groupe CRI-SRT dans la catégorie administrateurs, avec aucune restriction.

4.4.2.2 La vue publique

La vue publique est plus délicate à concevoir. En effet, elle sera accessible par tout le monde, et donc en grande partie des non-informaticiens. L'objectif est que chaque étudiant, professeur, ou personnel concerné par les services informatiques de l'INPL puisse consulter l'état des différents services offerts.

Inutile donc de préciser à un étudiant de géologie que le LDAP ne répond plus ou que Zimbra doit être redémarré, pour justifier que sa messagerie ne fonctionne plus. L'objectif est de n'obtenir que des voyants à trois états pour un service global. Si le voyant pour la messagerie est rouge, c'est normal qu'elle ne réponde plus, si il est orange c'est normal qu'elle soit lente, et si il est vert ça n'est pas normal en cas de problème pour l'utilisateur final, qui sera donc probablement isolé.

Afin d'arriver à tels voyants, il faut donc rassembler tout un ensemble d'états, écrire des scénarios et prendre des décisions de façon automatisée. Prenons le cas de la messagerie.

Pour afficher vert il faut que :

- Les services IMAPS, POPS et HTTPS du répartiteur de charges (etorki) répondent
- Les services IMAP(S) et POP(S) de deux machines Zimbra (feta et rollot) fonctionnent (l'une envoie, l'autre reçoit)
- Le LDAP de l'ancien serveur LDAP (athena) toujours en fonctionnement réponde sur le port 3500, ou bien que son équivalent (entdb) réponde sur ce même port
- Le LDAP du nouveau serveur (entdb) réponde sur le port standard, ou bien que sont équivalent (rouy.cri) réponde sur ce même port
- Les deux serveurs SMTP (epoisse et chaource) répondent
- Le service `amavisd` réponde sur l'une ou l'autre des deux machines précédentes, mais si une seule répond c'est un état orange
- Idem que la condition ci-dessus mais avec le service `postfix`
- Le service DNS du serveur epoisse ou son équivalent (athena) réponde

Nous avons donc, au passage, deux cas où la condition n'est pas binaire et peut passer à orange. Si parmi toutes ces conditions, il y a un rouge, le voyant du service global passe au rouge. Idem pour le orange, sinon tout est vert.

Centreon ne permet pas ce genre de vue globalisée.

NaReTo L'outil le plus prometteur dans ce domaine semblait être celui de *Linagora* : *NaReTo*. Il propose beaucoup de fonctionnalités, et notamment des rapports de supervision. Il permet également de créer des voyants hiérarchiques. A savoir, ce qu'on souhaite, mais avec la possibilité de cliquer pour obtenir une vue plus détaillée des services, chose non souhaitée dans notre cas.

Mais surtout, NaReTo se base sur Nagios uniquement, ce que je n'avais pas mesuré le jour où j'ai proposé ce produit comme solution. En installant NaReTo sur la machine Centreon, celui-ci n'a de vue que sur les machines supervisées par cette machine, c'est à dire uniquement les services communs, en faisant abstraction des Nagios satellites.

De plus, NaReTo ne semblait pas pouvoir respecter des scénarios complexes comme celui imaginé ci-avant, il aurait donc fallu faire des concessions, ou inventer de faux services sur lequel il aurait pu se baser, qui répondent vert lorsqu'un des deux autres services associés ne répond plus, et rouge s'il s'agit des deux, par exemple.

Pas de compromis ni de bricolage Trop de contraintes pour trop peu de résultats, NaReTo, semble avoir mal été évalué. Pire, avec son ignorance des satellites, il devient inutile.

Cette étape fut la dernière du stage. La reprise laborieuse de l'existant dans Centreon m'ayant pris énormément de temps, cette découverte n'était pas propice à mes délais. Décidé à ne pas perdre de temps à faire marcher un produit qui ne rendra pas tous les services souhaités, je propose de changer de stratégie et de développer moi-même l'outil des vues publiques. Mon tuteur et Benoît acceptent, malgré les délais relativement courts.

L'interface est écrite en PHP : c'est le seul langage supporté sur le serveur web qui l'hébergera, et il est mieux adapté que perl ou d'autres langages CGI pour l'écriture d'une interface de ce type. Ma connaissance du langage et mon apprentissage, tout au long du stage, de la base de données de Centreon, m'ont permis de très vite rattraper mon retard.

L'interface pioche directement ses informations dans la base de données de Centreon, à laquelle elle se connecte à distance. Ainsi, elle peut savoir en temps réel l'état de n'importe quelle machine ou de n'importe quel service. Les scénarios sont stockés dans un fichier de configuration qu'elle consulte, et qui utilise donc directement les noms des services et des machines de Centreon.

Pour qu'un scénario puisse être conçu, il faut des *OU* et des *ET*. Sur une même ligne du fichier de configuration, chaque service séparé par un espace est donc automatiquement considéré comme une condition *OU*. Si il y a un retour à la ligne, alors c'est un *ET*. On peut donc aisément rejoindre des lignes de *OU* par des *ET*.

Un service qui est en état orange fonctionne toujours pour l'utilisateur final. C'est en partant de ce principe que je considère que le orange doit être interprété comme du vert. Alors, si sur une même ligne, au moins un service est vert, la ligne est considérée comme verte. Si tous les services sont rouges, alors la ligne est rouge.

Afin d'avoir plus de souplesse, chaque ligne peut finir par un ou deux paramètres. Le premier indique le nombre de services, sur la ligne, qui doivent être critiques (rouges) pour que la ligne devienne rouge. Ce cas peut se produire pour les tests de bornes wifi par exemple : si quatre bornes sur dix sont rouges, alors on considère que la ligne est orange. Le second paramètre change la donne du *OU* : il indique le nombre de services qui doivent être critiques pour que la ligne devienne rouge. Ainsi, si huit bornes sont rouges, on considère la ligne comme rouge. Si un seul paramètre est indiqué, il s'agit du nombre de services critiques pour que la ligne devienne rouge. Dans le cas des bornes wifi par exemple, il peut être utile de passer des valeurs en pourcentage : le fichier gère cette possibilité.

Exemple de petit scénario d'un service :

```
ENT ! Acces au portail ID=ent
        ent1!App-ENT ent2!App-ENT ent3!App-ENT ent4!App-ENT !2!3
        entdb!App-HTTP
```

Le service global s'appelle ENT. Il possède une description (facultative), et un ID obligatoire. Ce dernier sera vu un peu plus loin. Le point d'exclamation (!) sert de séparateur de champs, le concept étant repris du passage d'arguments aux commandes de Nagios. `ent1 !App-ENT` signifie qu'on s'intéresse à l'état du service `App-ENT` de la machine `ent1`. Si `ent1` avait été désignée toute seul, sans service associé et donc de point d'exclamation, cela signifie qu'on se serait intéressé à l'état de la machine (répond ou ne répond pas).

Dans cet exemple, pour que le service global soit orange, il faut que deux conditions de la première ligne soient rouges, et que la seconde soit verte. Pour que le service passe au rouge, il faut soit qu'au moins trois services de la première ligne soient rouges, soit que la seconde le soit, soit que les deux le soient.

Si l'ensemble renvoie vert, l'utilisateur final verra :

Le `!2` de l'exemple aurait pu être écrit `!50%`. Mais les pourcentages n'ont pas d'utilité avec peu de machines. Il est pourtant difficile de s'imaginer indiquer tous les noms d'une centaine de bornes wifi, sur une même ligne. De ce constat est apparu la notion d'intégration de groupes de machines. Le regroupement des machines dans Centreon étant intelligemment conçu (...), il existe évidemment un groupe *Equipement_Bornes-Wifi*.

Il est donc agréable de faire :

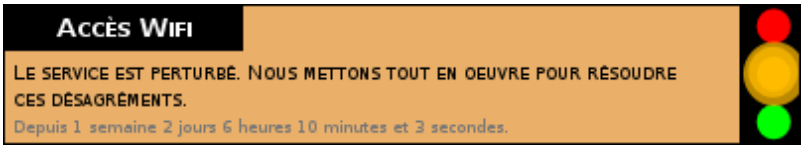
```
Wifi ID=wifi
  AP-WDS-Master1
  AP-WDS-Master2
        HG:Equipement_Bornes-Wifi !20%!70%
```

Le préfixe `HG :` (pour *HostGroup*) permet d'intégrer directement toutes les machines qui appartiennent au groupe désigné. Puisqu'aucun service n'a été précisé, on s'intéresse dans ce cas à l'état des bornes wifi. A noter que dans cet exemple, aucune description n'a été indiquée.

Il aurait été possible d'écrire :

```
Wifi ID=wifi
  AP-WDS-Master1
  AP-WDS-Master2
        HG:Equipement_Bornes-Wifi!App_SNMP-Encombrement !20%!70%
```

Ainsi, si 20% des bornes wifi indiquent un encombrement critique, on considère que l'accès wifi est orange :



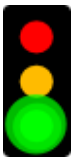
En revenant sur l'exemple de la problématique de la messagerie, voici comment le scénario décrit plus haut est modélisé :

```
Messagerie ! Boites et acheminement des courriels ID=messagerie
        # Mailbox
        etorki.cri!App-IMAPS
        etorki.cri!App-POPS
        etorki.cri!App-HTTPS
        feta.cri!App-Zimbra-IMAP rollot.cri!App-Zimbra-IMAP !1
        feta.cri!App-Zimbra-IMAPS rollot.cri!App-Zimbra-IMAPS !1
        feta.cri!App-Zimbra-POP rollot.cri!App-Zimbra-POP !1
        feta.cri!App-Zimbra-POPS rollot.cri!App-Zimbra-POPS !1
        # Ancien LDAP
        athena!App-LDAP-3500 entdb!App-LDAP-3500
        # Nouveau LDAP
        entdb!App-LDAP rouy.cri!App-LDAP-390
        # SMTP
        epoisse.cri
```

```
        chaource.cri
        epoisse.cri!App-Zimbra-Amavisd chaource.cri!App-Zimbra-
            Amavisd !1!2
        epoisse.cri!App-Zimbra-Amavis-Postfix chaource.cri!App-
            Zimbra-Amavis-Postfix !1!2
        # DNS
        epoisse.cri!App-DNS athena!App-DNS
```

L'ID a été ajouté pour implémenter une fonctionnalité réclamée par un collègue. S'occupant de la messagerie[6], il souhaite remplacer l'icône d'état de celle-ci directement par la mienne. Jusqu'ici l'icône était mise à jour de façon manuelle et donc peu réactive. L'interface doit donc être capable de ne renvoyer que l'image d'état d'un service en particulier.

Ainsi, en ajoutant `?id=messagerie` à l'adresse de la page, celle-ci se transforme en PNG (`image/png`) pour afficher l'icône correspondante :



Le collègue étant un personnage exigeant, il souhaite pouvoir y accéder en horizontal, chose faisable en ajoutant `&rotate` :

Les images ont été créées en vectoriel avec *Inkscape*.

Un exemple complet de page de configuration est disponible en annexe R, avec tous ses entêtes qui font figure de manuel embarqué. Ces derniers sont écrits en anglais, afin de pouvoir redistribuer ce script dans la communauté.

Extrait des entêtes de manuel :

```
# SYNTAX :
#
#  PAGE_TITLE = pageTitle
#
#  LABEL [! DESCRIPTION] ID=id
#  {host|HG:hostgroup}[!service] [[OR] {host|HG:hostgroup}[!service] ...] [![nbCritForWarn
    [%]!]nbCritForCrit[%]]
#  [[AND] {host|HG:hostgroup}[!service] [[OR] {host|HG:hostgroup}[!service] ...] [![
    nbCritForWarn[%]!]nbCritForCrit[%]]
#       ...]
```

Bien que l'appel à l'image ne demande de déterminer l'état du service que pour la messagerie, chaque chargement de page de son site, entraînera deux lectures de fichier (la configuration et l'image) et tout un lot de requêtes SQL. Concernant la page principale de la vue publique, elle est rafraîchie automatiquement toutes les cinq minutes : si quelqu'un l'oublie dans un onglet et que tout le monde fait de même, le serveur peut très vite exploser, ou au moins

[6] http://messagerie.inpl-nancy.fr

dégrader la réactivité du Centreon, déjà bien chargé par son travail.

Un cache est donc nécessaire. Le temps entre deux rafraîchissements est paramétrable dans le script, il sera donc logiquement aussi utilisé pour déterminer la durée du cache. Si un utilisateur arrive sur la page et que le cache date de moins de temps que la durée entre deux rafraîchissements, alors toutes les informations sont extraites de celui-ci et directement affichées sur la page. Le serveur Centreon n'est même pas contacté. Si ça n'est pas le cas, alors les requêtes SQL sont lancées et le fichier de cache est mis à jour. Ainsi, si mille utilisateurs se connectent sur la page (ou appellent indirectement l'icône depuis la page de la messagerie) pendant la période entre deux rafraîchissements, un seul entraînera des contacts avec le serveur de Centreon, tous les autres ne feront que consulter un fichier plat.

Exemple de fichier de cache généré :

```
1277468027
Tableau de bord informatique
mess|Messagerie|Etat du webmail et du SMTP|2|1276781786
morti|Mortimer||1|1276781786
wifi|Acces Wifi||0|1276089522
```

La valeur de tête correspond à la date de dernière mise à jour du cache. La seconde valeur correspond au titre principal de la page.

Les autres lignes correspondent aux services globaux.

Les tubes (|) font office de séparateurs de champs, avec de gauche à droite :

- L'ID
- Le label du service
- La description du service (facultative)
- L'état (0 pour vert, 1 pour orange, 2 pour rouge)
- La date du dernier changement d'état enregistré

La date du dernier état sert pour afficher aux utilisateurs depuis combien de temps l'état est resté constant (promis, le nombre de jours depuis lequel la messagerie est indisponible, sur cet exemple, est purement fictif) :

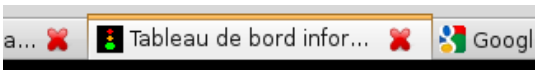
Cette fonctionnalité sert surtout au CRI-SRT à prouver combien leurs services sont performants, en affichant des scores toujours meilleurs (du moins on espère fortement).

Si le cache contient le titre principal de la page, c'est parce qu'il peut être indiqué directement dans le fichier de configuration, grâce à une ligne commençant par `PAGE_TITLE`. Si cette fonctionnalité a été implémentée, c'est parce que le résultat de la vue publique, pourrait donner envie de se baser sur ce modèle pour créer d'autres types de vues (des vues

métier par exemple). Ainsi, si le fichier de configuration par défaut s'appelle *confs/conf_public*, il est possible de le dupliquer en *confs/conf_toto*. Il suffira alors d'ajouter `?toto` à l'adresse de la page pour charger le fichier de configuration correspondant. Avec une restriction Apache basée sur LDAP, il devient envisageable de créer des vues privées très facilement.

Pour trouver pourquoi un service global affiche un état particulier, sans passer par Centreon, il est possible de passer en mode *debug* pour faire afficher l'état de tous les services vérifiés. Ce mode est à activer dans le script, puis à faire afficher en ajoutant `?debug`.

Enfin, et pour finir sur une fonctionnalité discrète mais pratique :



Lorsqu'un service devient orange sur la page, le *favicon* passe au orange, et si un service devient rouge, il passe au rouge. Puisque la page se recharge toute seule, par défaut, toutes les cinq minutes, il suffit de laisser la page traîner dans ses onglets pour savoir, sans même la consulter, si il se passe quelque chose d'anormal ou non.

Une copie d'écran de la page complète est disponible en annexe S.

Le script complet, commenté en anglais et sans sa feuille de styles, est disponible en annexe T.

4.5 De l'art des graphiques

4.5.1 NagiosGrapher et Centreon

Les seuls graphiques disponibles sur les anciens Nagios étaient produits par *NagiosGrapher*, une extension de Nagios dédiée à cette tâche. Il est capable d'utiliser les retours de métrologie des sondes ou le contenu du fichier *service-perfdata* qui se rempli continuellement de ces données, sur la machine du Nagios.

Centreon fonctionne lui aussi avec ce dernier. Il se connecte régulièrement en SSH et vide le fichier pour tout injecter dans sa base de données (grâce à *CentStorage*). Selon la configuration choisie par l'administrateur, il peut l'injecter en parallèle dans des bases RRD ou uniquement dans celles-ci. Cette dernière solution apporte un confort au niveau performances et espace disque, mais condamne à jamais les bases RRD à ne plus jamais être régénérées. Par défaut, il utilise les deux méthodes de stockage.

Sa philosophie pour la génération des courbes est la suivante : dans les données de métrologie, si il repère une valeur au format `nom=valeur`, il crée une courbe `nom` à laquelle il injecte la valeur `valeur`. Concernant le rendu, si rien n'est paramétré, il utilise le style par défaut, qui peut être changé (une simple ligne bleue, par défaut). Pour définir un autre style, il suffit, dans ses menus, de créer une nouvelle courbe nommée `nom`, avec des propriétés de style personnelles. Le style de courbe n'est rattaché à rien, il sera utilisé pour n'importe quelle courbe de n'importe quel graphique qui portera le nom `nom`.

Ce mécanisme est à la fois confortable et contraignant. En effet, à l'usage il est plutôt agréable de voir toutes ses valeurs `used` de tous ses graphiques s'afficher automatiquement en fond semi-transparent orange orné d'une ligne épaisse d'un orange plus foncé. Mais le monde n'est pas parfait (Windows n'a pourtant rien à voir avec ces graphiques), et le fait est que beaucoup de sondes ne renvoient pas de données de métrologie pertinentes pour cette façon de procéder. Avec une solution comme NagiosGrapher, rien n'est automatisé mais l'utilisateur a beaucoup plus de possibilités. Il pouvait donc s'en sortir avec des retours de métrologie comme *98790 ;745 ;976 ;64*. Si Centreon trouve des valeurs sans nom dans ce genre, il les ignore.

4.5.2 Un intermédiaire pour corriger

Modifier les sondes pour obtenir des données de métrologies mieux formatées serait un exercice périlleux et fatal pour les mises à jour. De plus, il faudrait les remplacer sur toutes les machines sur lesquelles elles ont été déployées. Vient alors l'idée de l'intermédiaire.

Plutôt que d'appeler la commande *check_nrpe* dans ces services qui posent problèmes, appelons la commande *check_nrpe_graph*, sans rien changer aux paramètres. Cette commande ne sera en réalité qu'un intermédiaire entre Centreon et NRPE. Sa première action sera d'appeler la commande distante par NRPE et de récupérer sa valeur de retour ainsi que son code de retour.

Ensuite, selon le type de vérification qu'il aura fait, il extraira les différentes valeurs des données de métrologie pour les retourner au bon format, en ajoutant des noms de champs.

Au passage, il fera bien plus :

- Les valeurs sont couramment renvoyées en méga-octets. Or, pour le format RRD dans lequel la valeur sera injecté, il s'agira d'une unité basique. Lorsqu'une valeur sera supérieure à mille-vingt-quatre mégas (donc plus d'un giga), RRD fera apparaître sur son ordonnée « *1k* ». Normal, un giga n'est rien d'autre qu'un kilo-méga. Pour pallier problème, **toutes les valeurs sont repassés en octets**, et c'est RRD qui gère tout seul l'unité.
- Nagios n'impose aucune norme sur le format des retours de métrologie. Sur trois sondes différentes (disque dur, *swap* et mémoire), qui devraient pourtant se ressembler, les données de métrologie sont renvoyées sous la forme `utilisé ; seuilUtiliséPourWarning ; seuilUtiliséPourCritique ; total`. Presque simple, en réalité...
  - Pour le disque dur, le schéma est bien respecté.
  - Pour le *swap*, **il faut juste tout inverser**. En effet, les valeurs sont renvoyées en terme de libre et non d'utilisé. Pour les trois valeurs il faut donc les déduire du total.
  - Pour la mémoire, c'est aussi inversé. Mais l'astuce réside dans les seuils qui sont renvoyés en terme de pourcentage libre. **Il faut donc les recalculer** en fonction du total et les déduire à celui-ci pour obtenir la bonne valeur.

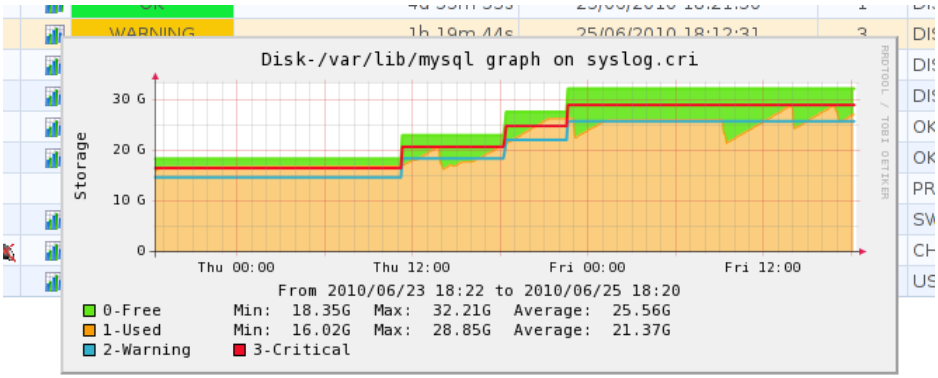
- Pour des sondes comme celle des niveaux des disques durs, il arrive qu'on spécifie une valeur qui n'est pas celle renvoyée. Par exemple, en demandant le taux d'espace disque

utilisé sur */var/lib/mysql*, si ce dernier n'est pas une partition, la sonde reverra une valeur attribuée à */var* si c'est la première partition mère rencontrée. Cette façon de faire peut être trompeuse, mais aussi pratique, puisqu'elle permet de généraliser des vérifications sans avoir une erreur si les partitions ne correspondent pas. Par contre il est primordial de garder à l'esprit que la valeur retournée n'est pas forcément exactement celle que l'on attendait. Ainsi, afin de prévenir l'utilisateur, le script **ajoute des `(PARTITION MERE)` dans le texte de retour**, lorsque c'est nécessaire.
– Il **renomme les sources** de façon à ce qu'elles s'affichent dans l'ordre, ce qui va avoir de l'importance dans la prochaine explication.

Centreon affiche les valeurs récupérées dans l'ordre alphabétique de leur nom. Dans certains cas, l'ordre peut être important : en affichant la valeur totale d'utilisation en fond vert et en affichant la valeur d'utilisation en fond orange, on obtient artificiellement une vue de la valeur libre.

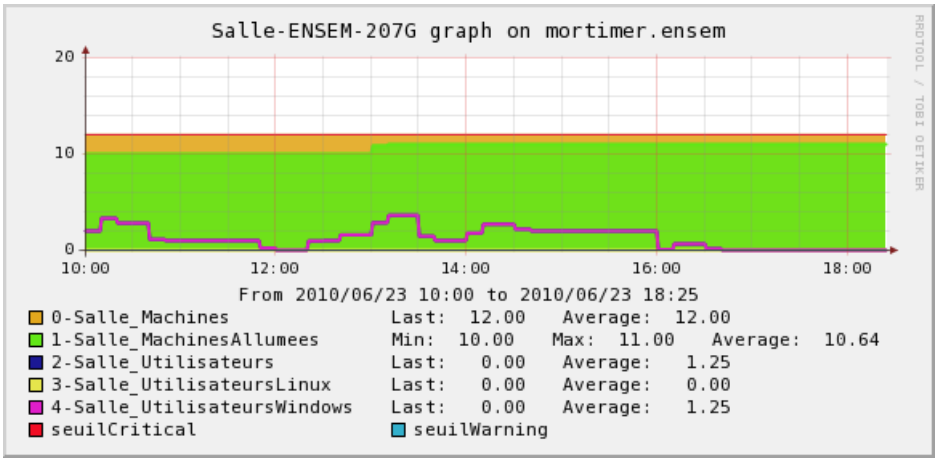
Le script en disponible en annexe U.

Démonstration pour un test de disque dur :



Cet exemple montre une activité plutôt mouvementée. En un seul coup d'oeil, on peut observer les augmentations successives de quotas, les seuils en pourcentages qui augmentent du même coup, et l'espace utilisé qui oscille dans les limites de notifications, jusqu'à fleureter avec la barre critique. En vert, on obtient une idée de l'espace libre, qui n'est autre que la valeur totale. Les unités sont correctement gérées. Tout ceci est malheureusement impossible, de base, avec Centreon et les sondes de la communauté Nagios.

Démonstration de ce qu'il est possible de faire avec des sondes personnalisées (ici, un nombre de machines allumées ou éteintes dans une salle, avec le nombre d'utilisateurs Linux/Windows - sonde de Benoît, adaptée pour Centreon) :



Beaucoup de brebis égarées, dans cette salle.

5. Conclusion

5.1 Etat des lieux

Comme pour un appartement, l'état des lieux se fait aussi bien au début qu'à la fin, il est donc logique que nous en fassions un de nouveau.

A mon départ, les anciens Nagios n'étaient pas encore arrêtés. Le Centreon, lui, comptabilisait encore trop de faux services critiques. Durant tout mon stage, une course effrénée aux services critiques a eu lieu. A cause de toutes les barrières de sécurité que j'ai évoqué durant ce rapport, il est très difficile d'intégrer une nouvelle machine aussi curieuse. Aussi, le temps nécessaire pour demander toutes les autorisations des VLAN, passer sur toutes les machines pour configurer les pare-feu, et régler tous les autres problèmes qui se posent m'aura vaincu.

Etant très dépendant de Benoît et mon tuteur pour ces tâches, il ne fait aucun doute qu'il pourront continuer sans moi, comme ils l'ont déjà énormément fait quand ils se sont rendus compte du temps que cela nécessitait. Ca n'est donc qu'une question de temps pour que Centreon remplace définitivement les anciens systèmes de supervision. Les différentes politiques adoptées et son organisation devraient lui permettre de rester propre et fonctionnel pendant longtemps. Ajouter une nouvelle machine ou un nouveau service est aisé.

Tous les satellites ne sont pas en place. Aussi, il a été très important de documenter la procédure. Plusieurs pages de wiki ont donc été remplies par mes soins (voir figure 5.1), et permettront de passer facilement le relais.

L'authentification LDAP fonctionne parfaitement et se synchronise toute seule.

A partir des exemples déjà créés, les vues métier sont extrêmement simple à mettre en place. Pour ce qui est de la vue publique, qui est un aboutissement de tout mon travail, la flexibilité et les possibilité de son fichier de configuration devrait permettre de l'affiner petit à petit pour qu'elle soit la plus performante possible. Il est question de l'intégrer directement

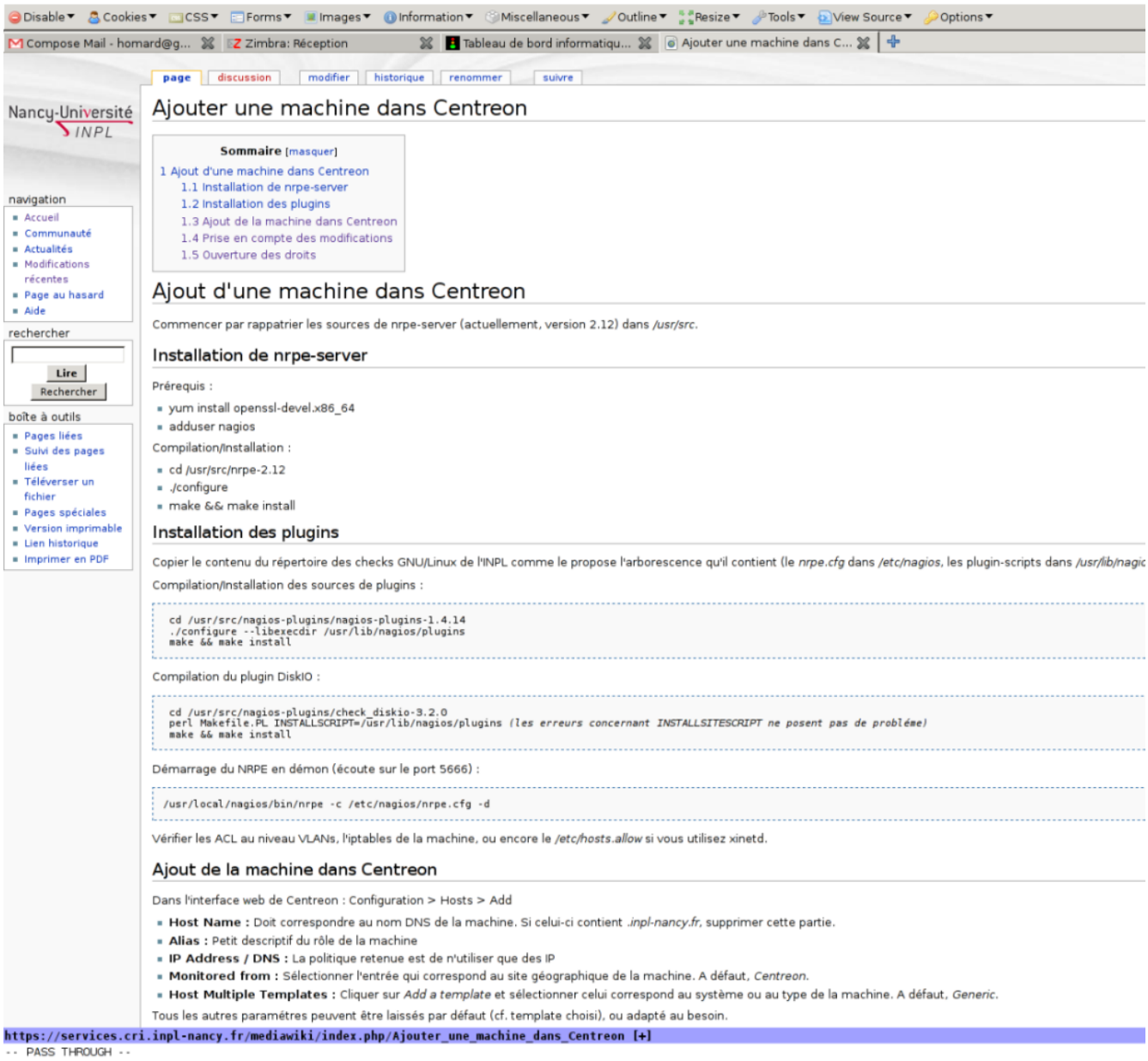


FIG. 5.1 – Exemple de page de documentation

sur l'espace numérique de travail de l'INPL.

Benoît souhaitait prolonger mon stage pour que je puisse m'attaquer à d'autres problématiques, comme l'ajout de sonde pour des services qui ne sont pas encore vérifiés, mais le budget a été refusé.

5.2 Apports personnels

Ce stage aura été pour moi l'occasion de découvrir le travail dans le domaine publique. Il aura aussi été l'occasion de découvrir le travail au sein d'une grosse équipe, plus grosse que celles que j'ai pu connaître.

Il m'aura permis d'apprendre le fonctionnement interne de Nagios, et la structure de

Centreon. De façon générale, il m'aura permis de me poser énormément de questions quant aux problématiques liées à la supervision sur un gros parc informatique hétérogène.

Durant ce stage, j'ai développé majoritairement en perl, mais aussi en ruby, en PHP, en bash et même en awk et un peu en batch. Sans compter bien sûr les nombreuses heures à tenter des requêtes SQL de plusieurs lignes dans la base de données du Centreon. Ca aura donc été l'occasion de me confronter aux problématiques d'automatisation des administrateurs, et éventuellement de contribuer à la communauté Centreon.

Pour enfin conclure, ce stage aura aussi été l'occasion de me faire embaucher - grâce à Christian, un collègue fort sympathique que je remercie - par une des écoles de l'INPL pour développer un pilote en C++ destiné à faire l'interface entre une sonde de température G800 et un logiciel Matlab, dans le cadre d'un projet de recherche sur la déformation par gravité du verre.

6. Annexes

A. Etat des lieux sur la supervision à l'INPL

La supervision à l'INPL

Notes concernant Nagios, Centreon, et les projets périphériques

---

Stage ASRALL 2010

Julien VAUBOURG <julien@vaubourg.com>

14 avril 2010

Table des matières

1 Préambule

Ce document a pour but de synthétiser les différentes observations faîtes sur le système de supervision de l'INPL Les idées évoquées ne sont que des idées de pistes pour améliorer le mécanisme global et sont soumises à débat.

2 Nagios

2.1 Préambule

Nagios est **la référence libre de la supervision**. A ce titre, de **nombreuses sondes** sont mises à disposition par la communauté. **Sa qualité ne fait plus douter**, et avec Centreon elle s'est dotée d'un puissant allié. On peut noter que sa popularité entraîne **la profusion de logiciels compatibles** qui apportent encore de nouvelles possibilités.

Enfin, afin de **gérer au mieux l'existant**, il faudrait une raison sans parade pour décider de renier tout le travail déjà effectué. Cette raison n'a pas été mise sous la lumière, ainsi **Nagios n'a aucune raison d'être remplacé**.

Avant de complexifier la supervision déjà en place et avant de mettre en place de nouveaux services générateurs de notifications, il m'a semblé important de **faire un point sur les problèmes qui peuvent déjà être notés de ce qui est en place**. Parmi ces problèmes, celui de la gestion des notifications et leur pertinence vis-à-vis des administrateurs, est un problème qui sort de l'aspect technique mais qui représente pourtant la finalité concrète de la supervision.

2.2 Problèmes constatés

2.2.1 Nom des machines

Hétérogénéité au niveau de la désignation des machines des hôtes : adresse IP ou **nom de domaine**. Les hôtes utilisant des noms de domaine (ex : chenas) pour pointer les machines sont dépendants du service DNS. Si celui-ci ne répond plus, Nagios considérera que l'hôte ne répond plus, et le mettra en cause injustement.

2.2.2 Service `ping`

La présence de tous les hôtes est vérifiée par un test de `ping`. Sur la plupart des serveurs, un service `ping` est également monitoré. **La vérification est donc double.** En considérant le nombre de machines concernées, ce nombre de requêtes inutiles est non-négligeable.

2.2.3 Hôtes non-surveillés

Certains serveurs (ex : athenas) proposent des services qui sont monitorés en temps que tel, mais **ne sont pas elles-mêmes monitorées en temps que *hosts***. On peut remarquer que si aucun service ne répond, c'est probablement que c'est la machine qui ne répond plus, et s'en contenter. Mais puisque Nagios ne connait pas l'hôte auquel relier ces services (et ne sait donc pas qu'il est en carafe), il enverra une alerte pour chacun de ces services plutôt que de comprendre qu'il faut juste envoyer une alerte pour l'hôte (les services sous-jacents sont donc considérés `UNREACHABLE`, ce qui ne doit pas donner lieu à une levée de notifications).

2.2.4 Limites `PROCS` et `LOAD`

Des alertes sur ces services sont levées plusieurs fois par jour pour les mêmes machines. Ils reviennent rapidement en `OK`, sans aucune intervention. Soit la limite est trop basse, soit il y a un emballement régulier des serveurs. Dans tous les cas, **les `WARNING` entraînés polluent les alertes**.

2.2.5 Notifications `UNKNOWN`

Un statut `UNKNOWN` est souvent lié à une autre ressource qui n'est pas correcte. Cette ressource est probablement monitorée, et lèvera une alerte d'elle-même si elle ne se rétablit pas très vite. Ainsi, reporter **ce type de notification qui ne donne aucune information** est rarement utile et pollue les alertes.

2.2.6 Intitulés des messages

Améliorer la clarté des emails permettrait de **mieux les interpréter** (ex : `PROCS` désigne les processus mais pourrait désigner les processeurs), et de les rendre plus agréables.

2.2.7 Espace libre des disques durs

Beaucoup de `WARNING` sont levés sur ce type d'alerte, inutilement. La raison semble être que le même test avec les mêmes limites est effectué sur tous les types de machines supervisées. Or, un espace disque d'utilisateur plein n'a pas la même importance qu'un `/var` saturé.

2.3 Disponibilité de Nagios

Que se passe-t-il si Nagios lui-même, son service de mail, ou un commutateur réseau le reliant ne répond plus ? **Plus aucune alerte ne peut être émise**, ni pour lui, ni pour le reste du réseau.

Pistes :
- Haute disponibilité (cf. plus bas)
- Utilisation d'un réseau indépendant (**GSM**)

2.4 Du côté des administrateurs

Beaucoup d'alertes sont **filtrées automatiquement** par les administrateurs (et sont donc inutiles, puisque même en étant pertinente, c'est la considération de l'alerte par l'utilisateur qui lui donne son sens). Pour les autres, elles polluent les vrais alertes.

Pistes :
- Augmenter certaines limites
- Reconsidérer l'intérêt de la notification pour certains services (qui seront toujours visibles dans la console)
- Segmenter les utilisateurs pour qu'ils ne reçoivent que les alertes qui les concernent (séparation Windows - GNU/Linux ?)
- Ajouter de la finesse dans les périodes en prenant en compte les astreintes (problème de configuration dynamique en fonction des calendriers) et les opérations redondantes (ex : redémarrage hebdomadaire du LDAP)
- Moyens d'alertes alternatifs (**SMS**, écrans, afficheurs, etc.)
- Regroupement des équipements pour envoyer des alertes pour un lot de machines (*checks* de cluster - ex : bornes wifi)

3 Dimensionnement de la machine de supervision

3.1 Estimations

Part son fonctionnement même, Nagios consomme ses ressources à l'inverse de la plupart des applications : il consomme beaucoup de mémoire CPU pour peu de mémoire vive. Ceci est dû au lancement des sondes, sa principale activité, qui augmente le nombre de processus.

On considère qu'un serveur moyenne gamme peut superviser sans problème de **15 000 à 20 000 services**[1] . Actuellement, un millier de services sont supervisés (en ne prenant en compte qu'un Nagios). En cherchant sur un moteur de recherche populaire la définition d'un serveur moyenne gamme, hormis Sun[2] qui considère ce type de serveur comme un serveur à seize coeurs avec 256Go de mémoire, la moyenne se trouve autour d'un bi-coeur à deux gigas de mémoire. En relation avec le postulat évoqué ci-dessus, le serveur actuel est donc deux fois plus puissant qu'un tel serveur. On pourrait donc en conclure qu'il est capable de superviser **dans les 30 000 services**.

La performance d'un Nagios se teste par la valeur de sa latence. Celle-ci peut s'obtenir grâce à `nagiostats`. Elle est calculée à partir du temps moyen mis par un *check* pour s'exécuter, par rapport au moment où son exécution a été réclamée. Tant que la latence est inférieure à un, on considère que le serveur est bien proportionné (et que sa

[1] Réf. : Jean Gabès, *Nagios 3*, Eyrolles
[2] http://fr.sun.com/practice/systems/sparcenterprise/midrange.jsp

configuration est bonne).

Pour tester un serveur pour Nagios, il suffit de lancer un nombre d'occurrences croissant d'un *check* de `ping`. En observant l'évolution de la latence, on peut avoir une idée du nombre de services **simultanés** maximum supportés par la machine.

3.2 Améliorer les performances

3.2.1 Scripts contre programmes

Le test est fait avec un **script bash et un programme C** qui ne font que vérifier la présence ou non d'un fichier (avec `test` pour le premier et `fopen` pour le second) : pour 10 000 occurrences du test, la version bash met **35 secondes alors que la version C n'en met que huit**.

3.2.2 Perl, une exception

Nagios propose un module qui porte le nom de `ePN`. Il s'agit d'un **interpréteur Perl intégré**. Il a pour charge de compiler les scripts et de les mettre en cache sous forme de *bytecode*. Ainsi, les prochaines exécutions se feront sans l'étape de compilation - habituellement faîte à la volée et responsable des performances moindres des langages de script. Inconvénient toutefois, si un script Perl est modifié, **il faut relancer Nagios** pour faire regénérer le cache. Les scripts doivent être testés comme compatibles, avec une version indépendante de l'interpréteur, et il est possible d'activer/désactiver cette méthode de traitement des scripts Perl par Nagios à l'aide d'un simple commentaire.

L'interpréteur Perl de Nagios permettrait de **gagner 30% de performances**.

3.2.3 La virtualisation

D'après un test réalisé par Jean Gabès[3], un Nagios sur un système virtualisé perd **jusqu'à 40% de performances** par rapport à un hébergement sur un serveur physique.

3.2.4 LIT (*Large Installation Tweaks*) 1/3 : Les variables d'environnement

Pour obtenir des informations spécifiques, une sonde a deux moyens : les variables d'environnement et les arguments qui lui sont passées. **En désactivant ces premières** (paramètre Nagios) au profit de ces secondes, **le gain de performances serait de 15%**. Il faut toutefois parfaitement vérifier que les sondes sont adaptées à ce mode de fonctionnement.

3.2.5 LIT 2/3 : Nettoyage de la mémoire

Lorsqu'une sonde vient de faire son travail, Nagios s'occupe lui-même de nettoyer sa mémoire. Sur des systèmes récents, ce processus est quasiment inutile. En faisant confiance au système, et donc en désactivant ce mode, **Nagios gagnerait 90% de performances**.

3.2.6 LIT 3/3 : *Fork* de Nagios

Afin de ne pas mettre en carafe Nagios lorsqu'une sonde s'emballe, **celui-ci se *fork* pour lancer les *checks***. En considérant que les sondes sont parfaitement fiables et en faisant confiance au système pour détecter les processus zombies, **le gain de performances serait de 190%** ! En combinant les trois LIT, on obtient un gain de performances **aux alentours de 225%**.

3.2.7 Fichiers temporaires

Certains fichiers utilisés par Nagios sont volatiles et n'ont aucune importance en cas de redémarrage. Ainsi, **ce genre de fichier peut être écrit dans la mémoire vive** (`/dev/shm`) plutôt que sur le disque dur. Le gain de performances serait de **3% et les risques absolument nuls**.

[3] *Nagios 3*, Eyrolles

4 Centreon

4.1 Les modèles

Les modèles sont une notion que Nagios ne connait pas. Ils servent à gérer des services identiques pour plusieurs hôtes, mais qui différent légèrement d'un hôte à l'autre. Une solution dans Nagios aurait été les *macro-variables*. Centreon résout le problème d'une façon plus simple avec les modèles **en générant tous les services pour tous les hôtes**, puis en laissant la main sur chacun qui devient paramétrable et qui permet donc de rajouter les différences. Si c'est indéniablement plus rapide et pratique, il est intéressant de noter que que **Centreon n'est pas capable de supprimer tous ces services** générés lorsqu'on supprime le modèle.

4.2 Lancement des *checks*

Plus besoin d'aller se connecter sur un noeud pour forcer son *check*, Centreon propose de **les lancer manuellement** si besoin est depuis l'interface.

4.3 *Acknowledge*

Parmi les commandes passives de Nagios existe la commande `ACKNOWLEDGE_SVC_PROBLEM`, qui **permet de spécifier que le problème a été pris en compte**. Ceci a pour incidence que l'état reste visible dans la console, mais que les notifications ne sont plus envoyées.

4.4 Désactivation des notifications

En un clic, un service n'enverra **plus de notifications**.

4.5 Maintenances

Si une maintenance est programmée, Centreon propose de lui indiquer : **aucune des machines concernées, et donc des services associés, n'enverra de notification dans ce labs de temps**.

4.6 MySQL

L'interface traditionnelle de Nagios tire ses informations d'un fichier *status.dat*. Chaque consultation demande de parcourir l'intégralité du fichier. Il est courant que **celui-ci pèse près de dix méga-octets**. Centreon, grâce à NDO, utilise une base MySQL. Ainsi, les consultations sont **très fortement allégées** en terme de ressources demandées. Cerise sur le gâteau, Centreon dispose incidemment de toute **la puissance de SQL** pour sélectionner et filtrer, rapidement et efficacement.

4.7 Exports

Les données peuvent être exportées en **XML ou CVS**.

4.8 Métrologie

Point fort de Centreon : il embarque la métrologie (CentStorage). Il conserve ces données **sous forme de RRD, et dans la base MySQL**. Pour ces deux formats, il est possible de lui indiquer une **durée de rétention**.

4.9 SNMP

Second point fort de Centreon : la gestion du SNMP. L'interface propose de **compiler quasiment toutes les MIB constructeur existantes**. Il intègre tout ce qu'il faut pour faire des *checks* SNMP facilement, et **sans installation fastidieuse**.

4.10 Répartition des charges

Avec CentCore qu'il intègre, Centreon permet de facilité la répartition des charges. Ainsi, il sera possible de n'installer que Nagios sur des **serveurs satellites**, qui lui remonteront les informations qu'il synthétisera. **Centreon génère les fichiers de configuration** de tous les Nagios satellites.

4.11 NagVis

En utilisant MySQL avec NDO, **Centreon apporte quasiment tout ce qu'il faut à NagVis** pour fonctionner. Si celui-ci est amené à être utilisé, il ne reste quasiment plus rien à installer.

4.12 ACL

Un jeu d'ACL très poussé permet de créer des utilisateurs à **différents profils sur l'interface**.

4.13 Fichiers de configuration

En utilisant une base de données, des systèmes de modèles dont Nagios n'a pas connaissance, et en générant une configuration Nagios automatique, Centreon prend la main sur celle-ci de façon définitive. **Aucune factorisation** n'est effectuée sur le fichier, qui devient alors inutilisable pour une reprise « à la main » de la configuration. On peut noter enfin que cette dépendance de l'outil de gestion entraîne un léger retard sur les versions de Nagios puisque les mises à jour de l'un et l'autre ne sont pas toujours synchronisées.

5 Service de proximité

Les salles TP sont amenées à être supervisées. Pour autant, ce ne sont pas les même administrateurs qui sont concernés. Ainsi, il peut être souhaitable de monter **un second serveur dédié au service de proximité**. En s'assurant que ce serveur n'est pas dépendant des mêmes branches du réseaux que le Nagios principal, il serait envisageable que l'un surveille l'autre et avertisse les personnes appropriées. Cette solution **résout en partie la question de l'inaccessibilité du serveur Nagios**.

6 NagVis

Parmi les moyens de communication des notifications alternatifs, il y a **l'écran dans la salle des administrateurs**. En couplant cette possibilité avec NagVis, qui permet de générer des cartes de disponibilités, il deviendrait inutile de rester les yeux rivés sur son écran dans l'angoisse d'une alerte. D'un seul coup d'oeil, un problème peut être détecté et localisé.

La carte NagVis peut ensuite être associé à des **photos de baies**. Une carte générale de l'INPL proposerait des points colorés pour chaque bâtiment, à partir desquels on pourrait avoir à un plan du bâtiment avec des points colorés pour chaque salle qui héberge des machines, à partir desquels on pourrait avoir accès à ces photos qui pointeront alors directement les machines fautives.

7 Distribution FAN

Promise à un grand avenir dans le monde de la supervision, elle permet d'**installer aisément et rapidement un Nagios, un Centreon, un Nagvis ainsi qu'un Noreto** (outils de monitoring) et un serveur de mails. Elle est basée sur une CentOS et met vingt minutes pour s'installer et être prête à configurer.

B. Adaptation d'une conf. Nagios pour Centreon

```ruby
#!/usr/bin/ruby -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

# Formatage d'une configuration Nagios dans le but de la rendre compatible avec les imports Centreon.
# Prend en arguments les fichiers .cfg a prendre en compte, dans n'importe quel ordre.
# Cree les nouveaux fichiers .cfg dans le repertoire de la variable dirTo (en bas).

# Ce que le script fait :
# - Tri les definitions selon leur type, et les range dans des fichiers de meme nom, en differenciant les templates (permet d'importer les differents
# elements dans l'ordre que Centreon preconise)
# - Reformate parfaitement la conf (indentation parfaite, espaces en trop, commentaires inutiles, lignes superflues, etc), pour eviter les
# bugs lorsque Centreon importe
# - Ajoute les attributs name/service_description aux services, obligatoires pour Centreon
# - Recupere les differents attributs *groups pour donner les membres d'un groupe directement au groupe lui-meme (attribut members),
# Centreon n'etant capable de faire les relations que dans ce sens
# - Injecte la configuration des templates de contacts directement dans la definition des contacts (Centreon ne connait pas les tpl de contact)

# Ce que le script ne fait pas :
# - Verifier les doublons d'attributs si il y a plusieurs use pour un contact
# - Gerer les servicegroups correctement

# Procedure d'importation dans Centreon :
# - Administration > Nagios > Load
#
# 1) Cocher Yes pour <Delete all configuration for the chosen type of files>
# - Cocher <Template based method file>
# - Cocher <Notification Command>
# - Copier-coller le contenu notifyCommands.cfg dans le <Manual Filling>
# - Cocher Yes pour <Run Nagios debug (-v)>
# - Cliquer sur <Load> et verifier que le nombre de commandes qu'il dit avoir pris en compte
# correspond bien au nombre de commandes a importer (grep -c 'define command' notifyCommands.cfg)
#
# 2) Recommender, en cochant No a <Delete all configuration for the chosen type of files>
# - Et en cochant <Check Command>, le reste ne change pas
# - Copier-coller le contenu des fichiers qui restent (un par un, ou a la suite dans le champ) dans cet ordre :
# <commands timeperiods contacts contactgroups tplHosts hosts hostgroups tplServices services servicegroups serviceescalations>
# - Toujours verifier le nombre d'insertion : Centreon ne dit rien quand il ignore une definition

# Un script bash accompagne ce script pour aider a importer les fichiers.


##########################
######## FUNCTIONS ######
##########################

# Creation du dossier qui recuperera les fichiers convertis
def createDirTo(dirTo)
  Dir.mkdir dirTo unless File.exist? dirTo

```

```
51    unless File.directory? dirTo
52      puts "ERREUR: #{dirTo} n'est pas un dossier ou n'a pas pu etre cree."
53      exit 1
54    end
55  end
56
57  # Purge du dossier qui recois les fichiers convertis
58  def emptyDir(dirTo)
59
60    # Si le dossier n'est pas vide
61    if Dir.entries(dirTo).length > 2
62      printf "Purger #{dirTo}/ (Y/n) ? "
63
64      # Pour chaque fichier, le supprimer, si l'utilisateur donne sont accord (oOyY)
65      Dir.foreach(dirTo) do |f|
66        f = "#{dirTo}/#{f}"
67        File.delete(f) if File.file? f
68      end if $stdin.gets.chomp =~ /^[yo]?$/i
69    end
70  end
71
72  # Reformatage des fichiers actuels
73  # (indentation correcte, tabulations partout, suppression des lignes inutiles)
74  # Centreon bug a l'import si le fichier n'est pas parfait.
75  def reformat(conf)
76
77    # Suppression de tous les commentaires
78    conf.gsub!(/[[:space:]]*[#;].*$/, "")
79
80    # Suppression de tous les espaces/tab de debutde ligne
81    conf.gsub!(/^[[:space:]]+/, "")
82
83    # Suppression de tous les espaces/tab de fin de ligne
84    conf.gsub!(/[[:space:]]+$/, "")
85
86    # Remplacement de toutes les tabulations par des espaces
87    conf.gsub!(/\t/, " ")
88
89    # Remplacement des blocs d'espaces par une seule espace
90    conf.gsub!(/[ ]+/, " ")
91
92    # Suppression des lignes vides
93    conf.gsub!(/^ ?\n/m, "")
94
95    # Indentation des definitions
96    conf.gsub!(/^((?!define|\}).*)$/, "\t\\1")
97
98    # Ajout d'une ligne vierge entre chaque def.
99    conf.gsub!(/^\}$/, "}\n")
100
101   # Ajout d'une espace avant les accolades
102   conf.gsub!(/([^ ])\{/, "\\1 {")
103
104   return conf
105 end
106
107 # Extraction des membres des groupes.
108 # Si une entite contient un attribut de groupe, on le stocke dans un tableau pour ce
      groupe et on
109 # lui retire son attribut. Si un groupe contient un attribut members, on extrait
      aussi ses membres.
110 # A terme, le but est de recreer proprement les definitions des groupes, qui
      declarent eux-meme
111 # leurs membres (Centreon ne supporte les relations que dans ce sens, sinon il ne les
       prend pas en compte a l'import).
112 def saveMembers(definition, type)
113   group = {}
114
115   # Extraction du nom de l'entite (attribut name pour les services, type_name sinon)
116   name = definition.match(/^\t#{type != "service" && type != "serviceescalation" ?
        "#{type}_" : ""}name (.*)/)[1]
117
118   # Nous avons affaire a un groupe : il faudra alors isoler les members qu'il definit
        , eventuellement
```

```
119     if(type =~ /group/)
120       if(definition =~ /^\tmembers (.*)/)
121
122         # Pour chaque membre, on l'ajoute dans un hash du nom du groupe
123         $1.split(",").each do |member|
124           member.strip!
125           group[name] = [] unless group[name]
126           group[name] << member
127         end
128
129         # On vire ensuite la definition du members du groupe, qu'on rajoutera plus tard
                proprement
130         definition.gsub!(/^\tmembers ([^\n]+)\n/m, "")
131       end
132
133     # Ce n'est pas un groupe : il faut donc regarder si il se reclame d'un groupe
134     else
135       if(definition =~ /^\t#{type}groups (.*)/)
136
137         # Pour chaque groupe qu'il declare, on ajoute l'entite au hash portant le nom
              du groupe correspondant
138         $1.split(",").each do |groupName|
139           groupName.strip!
140           group[groupName] = [] unless group[groupName]
141           group[groupName] << name
142         end
143
144         # On vire la definition des groupes de l'entite, qui sera reportee directement
              dans le groupe
145         definition.gsub!(/^\t#{type}groups ([^\n]+)\n/m, "")
146       end
147     end
148
149     # Integration des nouveaux groupes avec leurs membres dans le hash global,
150     # trie en fonction du type des groupes (ex: groups['contactgroup']['criSrt'])
151     group.each do |g,mbrs|
152       groupType = type + (type !~ /group/ ? "group" : "")
153       $groups[groupType][g] = [] unless $groups[groupType][g]
154       $groups[groupType][g] |= mbrs
155     end
156
157     # Retour de la definition de l'entite, sans les *groups et members
158     return definition
159   end
160
161   # Parse un lot de definition pour trier les differentes entites par fichiers
162   # (ex: les contactgroup dans contactgroups.cfg)
163   # L'interet est d'ensuite pouvoir importer la conf dans Centreon, en rentrant bien
        les elements
164   # dans le bon ordre pour qu'il prenne tout en compte.
165   def injectFiles(conf, dirTo)
166
167     # Pour chaque definition (extraction facile puisque la conf a ete correctement
          reformatee)
168     conf.scan(/^(define ([^ ]+) \{\n[^\}]+\})/m) do |definition, type|
169
170       # Une definition est un template si il y a un register0.
171       # Les templates sont ranges dans des fichiers a part.
172       template = definition.match("register 0")
173
174       # Centreon fait une distinction que Nagios ne connais pas : le commandes qui
            servent a notifier
175       # et celles qui servent pour les checks. Arbitrairement, on considere que si il y
              a notify dans le nom
176       # de la commande, si c'est une commande de notification, sinon de check
177       notify = (type == "command" && definition.match("notify"))
178
179       # Si c'est un template de definition de contact, c'est un cas que Centreon ne
            supporte pas du tout.
180       # On isole donc sa definition dans un hash global pour l'injecter ensuite dans
            les definition qui dependent
181       # de ce template.
182       if type == "contact" && template
183         name = definition.match("\tname (.*)")[1]
```

```
184         $tplContacts[name] = definition.gsub(/\t(register 0|name [^\n]+)\n/m, "").match
                (/\{\n(.*)\n\}/m)[1]
185
186       # Tout les autres definitions
187       else
188         # Dans le cas d'un service, il faut s'assurer qu'il ne manque aucun element,
                essentiel a Centreon
189         # mais facultatif pour Nagios
190         if(type == "service" || type == "serviceescalation")
191           name = definition.match(/\tname (.*)/)
192           desc = definition.match(/\tservice_description (.*)/)
193           tpl = definition.match(/\tuse (.*)/)
194
195           name = name[1] if name
196           desc = desc[1] if desc
197
198           # On a extrait le use (template) pour creer un nom de service intelligent, si
                  celui-ci n'en a pas.
199           # Quand les services n'ont pas de nom dans Nagios, c'est souvent que le tpl
                qu'ils utilisent en porte un
200           # suffisament explicite.
201           tpl = tpl =~ / / ? tpl[1].split(' ').select { |w| w.capitalize! || w } * ' '
                : tpl[1] if tpl
202
203           # Si on ne peut pas faire autrement, un nom et une description seront generes
                  avec un chiffre aleatoire
204           randId = rand 10000
205           insertIndex = definition.index("}") - 1
206
207           # Modification ou ajout du name, selon les cas.
208           # Si il existe deja, on remplace ses espaces par des tirets, pour que
                Centreon accepte de les importer correctment
209           definition.sub!(/\tname .*/, "\tname #{ name.gsub(' ', '-') }") if name
210           definition.insert(insertIndex, "\n\tname #{ (desc ? desc : (tpl ? "use #{tpl
                }" : "NAME #{randId}")).gsub(' ', '-') }") unless name
211
212           insertIndex = definition.index("}") - 1
213
214           # Idem pour les description : si le service a deja un nom, on le recopie en
                  description, sinon on utilise le nom du tpl,
215           # ou bien encore on genere un nom aleatoire
216           definition.sub!(/\tservice_description .*/, "\tservice_description #{ desc.
                gsub(' ', '-') }") if desc
217           definition.insert(insertIndex, "\n\tservice_description #{ (name ? name : (
                tpl ? "use #{tpl}" : "DESC #{randId}")).gsub(' ', '-') }") unless desc
218         end
219
220         # Extraction des membres du groupes ou des appartenances aux groupes
221         definition = saveMembers(definition, type) if type =~ /^(host|contact|service)(
                group|escalation)?$/ && !template
222
223         # Creation du nom du fichier qui accueillera cette definition
224         newFileName = "#{dirTo}/#{template ? "tpl" + type.capitalize : type}s.cfg"
                unless notify
225         newFileName = "#{dirTo}/notifyCommands.cfg" if notify
226
227         # Ajout de la definition au fichier qui va bien
228         File.open(newFileName, "a") { |f| f << "\n" << definition << "\n" }
229       end
230     end
231   end
232
233   # Injection des membres des groupes, a posteriori
234   def injectMembers(dirTo)
235
236     # Pour chaque groupe pour lequel des membres ont ete collectes
237     $groups.each do |groupType, groups|
238
239       # Selon le type de groupe, on trouve le fichier qui contient les definitions
240       fileName = "#{dirTo}/#{groupType}s.cfg"
241
242       if File.exist? fileName
243         conf = IO.read fileName
244         definitions = ""
```

```
245
246         # Puisque le fichier est parfaitement formater, on recupere chaque defininition
                de groupe
247         # en coupant aux lignes blanches
248         conf.split(/^$/).each do |definition|
249
250           # Recuperation du nom du groupe
251           group = definition.match(/^\t#{groupType}_name (.*)/)[1]
252
253           # Si ce groupe a des membres qui ont ete trouves, on lui injecte son attribut
                members
254           # qui contiendra la liste de ses membres
255           if groups[group]
256             insertIndex = definition.index("}") - 1
257             definition.insert(insertIndex, "\n\tmembers #{groups[group] * ','}")
258           end
259
260           # Constitution de la nouvelle liste de groupes, completes
261           definitions += definition
262         end
263
264         # Mise a jour du fichier
265         File.open(fileName, "w") { |f| f << definitions }
266       end
267     end
268   end
269
270   # Centreon ne supporte pas les templates de contact : durant le parsage,
271   # on a donc collecte tous les tps de contact, qui ne se retrouve plus dans la conf.
272   # L'idee est donc maintenant d'injecter la conf. des templates directement dans les
        contacts
273   # qui en dependent.
274   def extendTplContacts(dirTo)
275     fileName = "#{dirTo}/contacts.cfg"
276     definitions = ""
277
278     if File.exist? fileName
279       conf = IO.read fileName
280
281       # Pour chaque contact trouve dans le fichier des contacts
282       conf.split(/^$/).each do |definition|
283
284         # Si le contact depend d'un template, on remplace l'appel du tpl par la
                definition
285         # qu'il contient
286         if definition =~ /^\tuse (.*) ?/
287           definition.gsub!($&, $tplContacts[$1]) if $tplContacts[$1]
288         end
289
290         # Constitution de la nouvelle liste de contacts, completes
291         definitions += definition
292       end
293     end
294
295     # Mise a jour du fichier
296     File.open(fileName, "w") { |f| f << definitions }
297   end
298
299
300   ############################
301   ######## PROGRAMME ######
302   ############################
303
304   # Repertoire qui contiendra les fichiers cfg convertis
305   dirTo = "./toCentreon"
306
307   # Creation et purge du dossier, avant remplissage
308   createDirTo(dirTo)
309   emptyDir(dirTo)
310
311   # Creation des hashs globaux
312   $groups = { "hostgroup" => {}, "contactgroup" => {}, "servicegroup" => {} }
313   $tplContacts = {}
314
```

```
315 # Pour chaque fichier cfg passe en argument
316 ARGV.each do |fileName|
317   unless File.readable? fileName
318     puts "ERREUR: Le fichier #{fileName} ne peut etre lu."
319   else
320     # Recuperation de la conf du fichier, en la reformatant a la volee
321     conf = reformat(IO.read(fileName))
322
323     # Decoupage et corrections, classement en fichiers de type
324     injectFiles(conf, dirTo)
325   end
326 end
327
328 # Injection a posteriori des membres des groupes
329 injectMembers(dirTo)
330
331 # Remplacement des appels de template des contacts par la definition du tpl
332 extendTplContacts(dirTo)
333
334 # Hop!
335 puts "Fichiers correctement generes dans #{dirTo}/."
```

C. Aide à l'import dans Centreon

```
1  #!/bin/bash
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Aide a l'importation d'une configuration Nagios existante dans un Centreon, en
       utilisant
6  # le script ruby nagios2centreon.rb.
7  # Le script place les definitions a copier-coller dans la rubrique Load de Centreon,
       directement dans le presse-papier (clic milieu).
8  # Voir la section Procedure d'importation dans l'entete du fichier nagios2centreon.rb
9
10 # Prend en argument le dossier des .cfg a convertir et demande a creer le dossier
       toCentreon dans ce dossier (si le script
11 # n'a pas ete change).
12
13 # Depends : xclip
14
15 # Script de preparation des CFG
16 nagios2centreon=/home/ju/SCRIPTS/nagios2centreon.rb
17
18 if [ ! -x "$nagios2centreon" ]; then
19   echo "ERREUR: nagios2centreon est introuvable ou non-executable" >&2
20   exit 1
21 fi
22
23 # Ordre d'affichage des definitions pour une correcte importation dans Centreon
24 orderCfg='commands timeperiods contacts contactgroups tplHosts hosts hostdependencys
       hostgroups tplServices services servicegroups serviceescalations'
25
26 if [ -z "$1" -o ! -r "$1" ]; then
27   echo "ERREUR: Le dossier de confs est absent ou non-consultable" >&2
28   echo "Usage: $0 dossier_de_confs/" >&2
29   exit 1
30 fi
31
32 cd $1
33
34 # Creation du repertoire toCentreon et conversion des fichiers .cfg
35 $nagios2centreon *.cfg
36 cd toCentreon
37
38 # Remplacement des emails de contact (periodes de tests)
39 sed -i 's/email .*/email julien.vaubourg@inpl-nancy.fr/' contacts.cfg
40
41 # Copie des commandes de notification dans le presse-papier
```

```
42 cat notifyCommands.cfg | xclip
43 nbDefine=$(grep -c '^define ' notifyCommands.cfg)
44
45 echo -e "\n$nbDefine commandes de notification dans le presse-papier."
46 echo "Appuyer sur Entree pour continuer (penser a vider le <Manual Filling> avant de
       continuer)."
47 read ok
48
49 # Affichage du nombre de definitions pour chaque fichier
50 for i in $orderCfg; do
51   if [ -f $i.cfg ]; then
52     nbDefine=$(grep -c '^define ' $i.cfg)
53     echo $nbDefine $i
54   fi
55 done
56
57 # Copie le reste de la configuration dans le presse-papier
58 for i in $orderCfg; do
59   [ -f $i.cfg ] && cat $i.cfg;
60 done | xclip
61
62 echo -e "\nCommandes de checks et autres definitions dans le presse-papier."
63 echo "Appuyer sur Entree pour quitter."
64 read ok
```

D. Conversion de modèles en groupes

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Rassemble les hosts qui dependent d'un meme service au sein d'un hostgroup lui-meme
       dependant
6  # de ce service. Fusionne ensuite les hostgroups identiques, cree les groupes et les
       relations
7  # et defait les anciennes.
8
9  use strict;
10 use warnings;
11 use DBI;
12
13 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
       { 'RaiseError' => 1 });
14
15 my (%hostsByHostgroup, %servicesByHostgroup);
16
17 # Selection de tous les couples host/services pour lesquels services
18 # se retrouve partage entre plusieurs hosts
19 my $sql = $mysql->prepare("SELECT service_service_id service, host_host_id host
20       FROM host_service_relation
21       WHERE service_service_id IN (SELECT service_service_id
22               FROM host_service_relation
23               GROUP BY service_service_id
24               HAVING COUNT(*) > 1)
25       AND host_host_id IS NOT NULL");
26 $sql->execute;
27
28 # Pour chacun de ces couples
29 while(my $rel = $sql->fetchrow_hashref) {
30
31   # Les hostgroups crees porteront le nom des services et contiendront les hosts
       concernes par ce service
32   $hostsByHostgroup{$rel->{'service'}} = [] unless @hostsByHostgroup{$rel->{'service'
       }};
33   push @{$hostsByHostgroup{$rel->{'service'}}}, $rel->{'host'};
34
35   # Association du service concerne au hostgroup fraichement cree
36   $servicesByHostgroup{$rel->{'service'}} = [$rel->{'service'}] unless
       @servicesByHostgroup{$rel->{'service'}};
```

```
37  }
38
39  # Pour chacun des hostgroups
40  for my $hg_id (keys %hostsByHostgroup) {
41    my $hosts = $hostsByHostgroup{$hg_id};
42
43    # Pour chacun des hostgroups
44    for my $hg_id_cmp (keys %hostsByHostgroup) {
45      my $hosts_cmp = $hostsByHostgroup{$hg_id_cmp};
46
47      # Si le hostgroup courant est dependant des memes services que le second
            hostgroup courant,
48      # alors un troisieme hostgroup. Il aura pour nom la concatenation des deux
            premiers, contiendra
49      # l'ensemble des hosts reunis et sera donc dependant des deux services
50      if($hg_id != $hg_id_cmp && join('', sort @$hosts) eq join('', sort @$hosts_cmp))
            {
51
52        # Pour chacun des service du second, on l'ajoute au premier, qui sera
              transforme
53        for(my $i = 0; $i < @{$servicesByHostgroup{$hg_id_cmp}}; $i++) {
54          push @{$servicesByHostgroup{$hg_id}}, $servicesByHostgroup{$hg_id_cmp}[$i];
55          delete $servicesByHostgroup{$hg_id_cmp}[$i];
56        }
57      }
58    }
59  }
60
61  # Pour chaque hostgroup
62  for my $hg_id (keys %hostsByHostgroup) {
63
64    # Un hostgroup fusionne a entraine un hosgroup vide
65    next if @{$servicesByHostgroup{$hg_id}} == 0;
66
67    $mysql->do("LOCK TABLES
68      service READ,
69      hostgroup WRITE,
70      hostgroup_relation WRITE,
71      host_service_relation WRITE");
72
73    $mysql->do("SET AUTOCOMMIT = 0");
74    my $hg_name = 'AUTO';
75
76    # Pour chacun des hostgroups, on retrouve le nom des services associes et on
77    # cree son nom a partir de la concatenation des leurs
78    for (@{$servicesByHostgroup{$hg_id}}) {
79      my $sql = $mysql->prepare("SELECT service_description name
80        FROM service
81        WHERE service_id=$_");
82      $sql->execute;
83
84      my $service_name = $sql->fetchrow_hashref->{'name'};
85      $service_name =~ s/(-service|use-)//g;
86      $hg_name .= "___$service_name";
87    }
88
89    # Creation du hostgroup
90    $mysql->do("INSERT INTO hostgroup (
91      hg_name,
92      hg_alias,
93      hg_activate
94    ) VALUES (
95      '$hg_name',
96      'AUTO',
97      '1'
98    )");
99
100   # Recuperation de l'ID genere pour ce nouveau groupe, afin de pouvoir cree
101   # ses relations
102   my $sql = $mysql->prepare("SELECT LAST_INSERT_ID() id");
103   $sql->execute;
104   my $real_hg_id = $sql->fetchrow_hashref->{'id'};
105
106   # Creation des liaisons entre les hosts et le hostgroup
```

```
107     for (@{$hostsByHostgroup{$hg_id}}) {
108       $mysql->do("INSERT INTO hostgroup_relation (
109         hostgroup_hg_id,
110         host_host_id
111       ) VALUES (
112         $real_hg_id,
113         $_
114       )");
115     }
116
117     # Creation des liaisons entre les services et le hostgroup
118     for (@{$servicesByHostgroup{$hg_id}}) {
119       $mysql->do("INSERT INTO host_service_relation (
120         hostgroup_hg_id,
121         service_service_id
122       ) VALUES (
123         $real_hg_id,
124         $_
125       )");
126     }
127
128     # Suppression des relations entre les hosts eux-memes et ces services
129     # dont ils vont maintenant heriter depuis leur hostgroup
130     for my $h (@{$hostsByHostgroup{$hg_id}}) {
131       for my $s (@{$servicesByHostgroup{$hg_id}}) {
132         $mysql->do("DELETE FROM host_service_relation
133           WHERE host_host_id=$h
134           AND service_service_id=$s");
135       }
136     }
137
138     $mysql->do("COMMIT");
139     $mysql->do("SET AUTOCOMMIT = 1");
140     $mysql->do("UNLOCK TABLES");
141   }
142
143   $sql->finish;
144   $mysql->disconnect;
```

E. Déplacement de machines d'un groupe à l'autre

```
1   #!/usr/bin/perl -w
2   # Auteur <julien@vaubourg.com> pour le CRI de INPL
3   # 2010 - Centreon 2.0.2
4
5   # Deplace les hosts d'un hostgroup vers un autre hostgroup
6
7   use strict;
8   use warnings;
9   use DBI;
10
11  my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
        { 'RaiseError' => 1 });
12  my $h;
13
14  # Hostgroup a partir duquel il faut copier les hosts
15  print "Groupe de depart : ";
16  my $from = <>;
17  chomp($from);
18
19  # Hostgroup dans lequel les copier
20  print "Groupe de destination : ";
21  my $to = <>;
22  chomp($to);
23
24  # Verification de l'existence du hostgroup de destination
```

```
25 my $sql = $mysql->prepare("SELECT TRUE FROM hostgroup WHERE hg_name=".$mysql->quote(
       $to));
26 $sql->execute;
27
28 if($sql->rows == 0) {
29   print STDERR "ERREUR: Le groupe d'hotes ".$mysql->quote($to)." n'existe pas.\n";
30   exit 1;
31 }
32
33 # Selection de tous hosts du hostgroup source qui ne sont pas dans le groupe de
       destination
34 $sql = $mysql->prepare("SELECT host_host_id id FROM hostgroup_relation
35       WHERE hostgroup_hg_id=(SELECT hg_id
36             FROM hostgroup
37             WHERE hg_name=".$mysql->quote($from).")
38       AND host_host_id NOT IN (SELECT host_host_id
39               FROM hostgroup_relation
40               WHERE hostgroup_hg_id=(SELECT hg_id
41                     FROM hostgroup
42                     WHERE hg_name=".$mysql->quote($to)."))");
43
44 $sql->execute;
45
46 my $nbHosts = $sql->rows;
47
48 # Deviation des liaisons entre les hosts et leur ancien hostgroup pour les faire
       pointer vers leur
49 # nouveau hostgroup
50 $mysql->do("UPDATE hostgroup_relation
51     SET hostgroup_hg_id=(SELECT hg_id FROM hostgroup WHERE hg_name=".$mysql->quote(
           $to).")
52     WHERE hostgroup_hg_id=(SELECT hg_id FROM hostgroup WHERE hg_name=".$mysql->quote(
           $from).")
53     AND host_host_id=".$h->{'id'}
54   ) while($h = $sql->fetchrow_hashref);
55
56 print "\nINFO: Les $nbHosts hosts appartenant a $from (et qui n'etaient pas deja dans
       $to) ont ete deplaces vers $to.\n";
57
58 # Confirmation de la suppression du hostgroup a present vide, ou uniquement rempli
       des hosts qui etaient
59 # deja dans le groupe de destination
60 print "\nSupprimer le groupe $from (Y/n) ? ";
61 my $del = <>;
62 chomp($del);
63
64 # Suppression
65 if($del =~ /^[yo]?$/i) {
66   $mysql->do("DELETE FROM hostgroup_relation WHERE hostgroup_hg_id=(SELECT hg_id FROM
         hostgroup WHERE hg_name=".$mysql->quote($from).")");
67   $mysql->do("DELETE FROM hostgroup WHERE hg_name=".$mysql->quote($from));
68   print "INFO: Le groupe d'hotes $from a ete supprime.\n";
69 }
70
71 $mysql->disconnect;
```

F. Copie de machines d'un groupe à l'autre

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Copie les hosts d'un hostgroup dans un autre
6
7  use strict;
8  use warnings;
9  use DBI;
10
11 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
```

```
        { 'RaiseError' => 1 });
my $h;

# Hostgroup des machines a copier
print "Groupe de depart : ";
my $from = <>;
chomp($from);

# Hostgroup de destination de ces machines
print "Groupe de destination : ";
my $to = <>;
chomp($to);

# Verification de l'existence du hostgroup de destination
my $sql = $mysql->prepare("SELECT TRUE FROM hostgroup WHERE hg_name=".$mysql->quote($to));
$sql->execute;

if($sql->rows == 0) {
  print STDERR "ERREUR: Le groupe d'hotes ".$mysql->quote($to)." n'existe pas.\n";
  exit 1;
}

# Selection des hosts qui sont dans le groupe de depart mais pas encore dans le groupe de destination
$sql = $mysql->prepare("SELECT host_host_id id FROM hostgroup_relation
        WHERE hostgroup_hg_id=(SELECT hg_id
              FROM hostgroup
              WHERE hg_name=".$mysql->quote($from).")
        AND host_host_id NOT IN (SELECT host_host_id
                FROM hostgroup_relation
                WHERE hostgroup_hg_id=(SELECT hg_id
                      FROM hostgroup
                      WHERE hg_name=".$mysql->quote($to)."))");

$sql->execute;

my $nbHosts = $sql->rows;

# Insertion des hosts a copier dans le groupe de destination
$mysql->do("INSERT INTO hostgroup_relation (
    hostgroup_hg_id,
    host_host_id
  ) VALUES (
    (SELECT hg_id FROM hostgroup WHERE hg_name=".$mysql->quote($to)."),
    ".$h->{'id'}."
  )") while($h = $sql->fetchrow_hashref);

print "\nINFO: Les $nbHosts hosts appartenant a $from (et qui n'etaient pas deja dans $to) ont ete copies dans $to.\n";

$mysql->disconnect;
```

G. Suppression des modèles de services optionnels

```
#!/usr/bin/perl -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

use strict;
use warnings;
use DBI;

my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
        { 'RaiseError' => 1 });

```

```
11 # Selection des templates de services
12 my $sqlTpl = $mysql->prepare("SELECT * FROM service WHERE service_register='0'");
13 $sqlTpl->execute;
14
15 # Pour chaque tpl de service
16 while(my $tpl = $sqlTpl->fetchrow_hashref) {
17
18   # Copies des informations du template directement dans le service
19   while(my ($k, $v) = each %$tpl) {
20     $mysql->do("UPDATE service
21         SET $k='$v'
22         WHERE service_template_model_stm_id=".$tpl->{'service_id'}." AND $k IS NULL")
               if $v;
23   }
24
25   # Selection des services dependants de ce tpl
26   my $sqlServices = $mysql->prepare("SELECT service_id
27             FROM service
28             WHERE service_template_model_stm_id=".$tpl->{'service_id'});
29   $sqlServices->execute;
30
31   # Pour chaque service ayant pour template le tpl courant
32   while(my $service = $sqlServices->fetchrow_hashref) {
33
34     # Recuperation des contacts relies au tpl
35     my $sqlTplContacts = $mysql->prepare("SELECT contact_id
36               FROM contact_service_relation
37               WHERE service_service_id=".$tpl->{'service_id'});
38     $sqlTplContacts->execute;
39
40     # Pour chaque contact associe au tpl courant, on l'associe directement au service
           courant
41     while(my $contact = $sqlTplContacts->fetchrow_hashref) {
42       $mysql->do("INSERT INTO contact_service_relation(
43         service_service_id,
44         contact_id
45       ) VALUES (
46         ".$service->{'service_id'}.",
47         ".$contact->{'contact_id'}."
48       )");
49     }
50
51     # Recuperation des groupes de contacts relies au tpl
52     my $sqlTplContactgroups = $mysql->prepare("SELECT contactgroup_cg_id
53                 FROM contactgroup_service_relation
54                 WHERE service_service_id=".$tpl->{'service_id'});
55     $sqlTplContactgroups->execute;
56
57     # Pour chaque contactgroup associe au tpl courant, on l'associe directement au
           service courant
58     while(my $contactgroup = $sqlTplContactgroups->fetchrow_hashref) {
59       $mysql->do("INSERT INTO contactgroup_service_relation(
60         service_service_id,
61         contactgroup_cg_id
62       ) VALUES (
63         ".$service->{'service_id'}.",
64         ".$contactgroup->{'contactgroup_cg_id'}."
65       )");
66     }
67
68     $sqlTplContacts->finish;
69     $sqlTplContactgroups->finish;
70   }
71
72   # A present que toutes les informations du tpl ont ete directement associees aux
         services correspondant, on denoue ces services du tpl
73   $mysql->do("UPDATE service SET service_template_model_stm_id=1 WHERE
         service_template_model_stm_id=".$tpl->{'service_id'});
74 }
75
76 $sqlTpl->finish;
77 $mysql->disconnect;
```

H. Suppression des modèles de services inutilisés

```
#!/usr/bin/perl -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

# Supprime les templates de services inutilises

use strict;
use warnings;
use DBI;

my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
    { 'RaiseError' => 1 });

my $sql = $mysql->prepare("SELECT service_id id FROM service WHERE service_id NOT IN
    (SELECT service_template_model_stm_id FROM service) AND service_register='0';");
$sql->execute;

while(my $s = $sql->fetchrow_hashref) {
  my $id = $s->{'id'};
  $mysql->do("DELETE FROM service WHERE service_id=$id");
}

$sql->finish;
$mysql->disconnect;
```

I. Suppression des définitions de commandes inutilisées

```
#!/usr/bin/perl -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

# Suppression de toutes les commandes non utilises

use strict;
use warnings;
use DBI;

# COMMANDES NRPE
# select command_command_id_arg from service where command_command_id = (select
    command_id from command where command_name='check_nrpe_arg')

# COMMANDES NON UTILISEES
# select command_name from command where command_id NOT IN (select command_command_id
    from service);

# SERVICES NON UTILISES
# select service_description from service where service_id not in (select
    service_service_id from host_service_relation) and service_register='1';

# SERVICES TPL NON UTILISES
# select service_description from service where service_id not in (select
    service_template_model_stm_id from service) and service_register='0';

my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
    { 'RaiseError' => 1 });

```

```
25 my $sql = $mysql->prepare("SELECT command_name
26       FROM command LEFT JOIN
27         (SELECT s.command_command_id s_id, h.command_command_id h_id
28         FROM service s, host h) sh
29       ON command_id IN (sh.s_id, sh.h_id)
30       WHERE s_id IS NULL AND h_id IS NULL AND command_type=2");
31 $sql->execute;
32
33 while(my $r = $sql->fetchrow_hashref) {
34   my $command = $r->{'cn'};
35   my $nb = $mysql->do("DELETE FROM command WHERE command_name='$command';") unless
         $command =~ /salles/;
36   print "$nb commandes supprimees\n";
37 }
38
39 $sql->finish;
40 $mysql->disconnect;
```

J. Génération de *nrpe.cfg* automatiquement

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Tente de creer un nrpe.cfg a partir des informations du Centreon
6
7  use strict;
8  use warnings;
9  use DBI;
10
11 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
       { 'RaiseError' => 1 });
12
13 # Selection de tous les parametres passes a une commande check_nrpe
14 my $sql = $mysql->prepare("SELECT command_command_id_arg arg
15         FROM service
16         WHERE command_command_id IN (SELECT command_id
17                 FROM command
18                 WHERE command_name
19                 LIKE 'check_nrpe%')");
20 $sql->execute;
21
22 my %cmds;
23
24 # Pour chaque ligne de parametres de NRPE
25 while(my $r = $sql->fetchrow_hashref) {
26   my $arg = $r->{'arg'};
27
28   # Centreon remplace les slashs par des #S#
29   $arg =~ s/#S#/\//g;
30
31   # Separation des arguments
32   my @args = split('!', $arg);
33
34   # Le nom de la commande distante est en premier
35   shift @args;
36
37   # Les arguments ensuite
38   my $c = shift @args;
39
40   # Tentative d'ecriture de nrpe.cfg
41   $cmds{$c.@args} = "command[$c] = /usr/lib/nagios/plugins/$c ".join('     ', @args).
       "\n";
42 }
43
44 # Affichage par ordre alphabetique
45 for (sort keys %cmds) {
46   print $cmds{$_};
47 }
```

```
48
49 $sql->finish;
50 $mysql->disconnect;
```

K. Déploiement massif de fichiers

```
1  #!/bin/bash
2  # Author <julien@vaubourg.com> 2010
3
4  # Copier massivement des fichiers sur un lot de machines
5  # myPuppet.sh -h pour plus de renseignements
6
7  help() {
8          echo -e "\n\tCopier un fichier sur une liste de machines distances via scp."
9
10         echo -e "\n\tOPTIONS OBLIGATOIRES"
11         echo -e "\t\t-m <file> : Liste des machines concernees (une machine par ligne
               )"
12         echo -e "\t\t-f <file> : Fichier source"
13         echo -e "\t\t-t <path> : Chemin de destination"
14
15         echo -e "\n\tOPTIONS"
16         echo -e "\t\t-h        : Afficher cette aide"
17
18         echo -e "\n\tEXEMPLE"
19         echo -e "\t\t./myPuppet.sh -m machines.lst -f fichiers/nrpe -t /etc/xinetd.d/
               "
20 }
21
22 while getopts m:f:t:h option
23 do
24         case $option in
25                 m)
26       machinesFile=$OPTARG
27                 ;;
28
29     f)
30       fileFrom=$OPTARG
31     ;;
32
33     t)
34       pathTo=$OPTARG
35     ;;
36
37     h)
38       help
39       exit 0
40     ;;
41
42     *)
43       help
44       exit 1
45   esac
46 done
47
48 if [ -z "$machinesFile" -o -z "$fileFrom" -o -z "$pathTo" ]; then
49   echo "ERREUR: Toutes les options sont obligatoires." >&2
50   help
51   exit 1
52 fi
53
54 if [ ! -r $machinesFile -o ! -r $fileFrom ]; then
55   echo "ERREUR: Un des deux fichiers (machines ou fichier a envoyer) n'existe pas ou
         ne peut etre lu." >&2
56   exit 1
57 fi
58
59 echo -n "Copier le fichier distant sur la machine distante en .old, si il existe (Y/n
       ) ? "
```

```
60 read copy
61
62 file=${fileFrom##*/}
63
64 while read machine; do
65   echo "Copie de $file dans $machine:$pathTo..."
66
67   if [ "$copy" = "y" -o "$copy" = "Y" -o -z "$copy" ]; then
68     ssh root@$machine "cp $pathFile/$file{,.old}"
69   fi
70
71   scp $fileFrom root@$machine:$pathTo/$file
72 done < $machinesFile
73
74 echo -e "\nAux toilettes puppet."
75 exit 0
```

L. Sélection de machines appartenant à un DNS donné

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Selectionne tous les hosts qui contiennent un mot dans leur nom de domaine
6  # et qui repondent sur le port 22
7
8  use strict;
9  use warnings;
10 use DBI;
11 use Net::Ping;
12 use List::Util 'first';
13
14 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
       { 'RaiseError' => 1 });
15
16 my $sql = $mysql->prepare("SELECT host_address addr
17         FROM host
18         WHERE host_register='1'")
19 ;
20 $sql->execute;
21
22 # Mot a filtrer
23 print "Ecole a filtrer : ";
24 my $school = <>;
25 chomp($school);
26
27 my @hosts;
28 my $i = 0;
29
30 # Debut de la barre de chargement
31 print "\n[";
32
33 # Pour chacun des hosts
34 while(my $h = $sql->fetchrow_hashref) {
35   my $addr = $h->{'addr'};
36
37   # DNS inverse pour recuperer le nom de domaine
38   my $dns = `host $addr | awk '\$NL ~ /pointer/ { print substr(\$NF, 0, length(\$NF)
         -1) }' | tail -n1`;
39
40   # L'host a un nom de domaine
41   if($dns ne '') {
42     chomp($dns);
43
44     # Progression de la barre de chargement
45     print '|' if ++$i%2;
```

```
46
47      # Si le nom de domaine contient le mot
48      if($dns =~ /$school/i) {
49
50        # On delegue a une machine tierce qui en a les droits, la charge de verifier
51        # si la machine repond au ping sur le port SSH
52        `ssh nagiosBrabois "perl -e '\\\
53          use Net::Ping;\\\
54          my \\\$ssh = Net::Ping->new(\\"tcp\\", 3);\\\
55          \\\$ssh->{port_num} = 22;\\\
56          exit substr(\\\$ssh->ping(\\"$dns\\"), 0, 1);'"`;
57
58        # Si c'est le cas, on l'ajoute a la liste
59        if($?) {
60          push(@hosts, $dns) unless first { $_ eq $dns } @hosts;
61        }
62      }
63    }
64  }
65
66  # Sortie pour en syntaxe CSSH
67  print "]\n\ncssh ";
68
69  print "root\@$_ " for (@hosts);
70  print "\n\n";
71  print "$_\n" for (@hosts);
72  print "\n";
73
74  $sql->finish;
75  $mysql->disconnect;
```

M. Déploiements NRPE sur Windows

```
1   #!/bin/bash
2   # Auteur <julien@vaubourg.com> 2010
3   # complete par Pierre MAFFEIS
4
5   # A executer dans le repertoire parent de $scripts
6   #
7   # Dans le dossier $scripts :
8   # - machines (un nom de machine par ligne)
9   # - winexe
10  # - nrpe.delta.cfg
11  # - cent_checks/
12
13
14  # Repertoire principal
15  rep=scripts
16
17  # Compte administrateur
18  admin=root
19
20  # Mot de passe des Windows
21  echo -n "Mot de passe Admin : "
22  read -s mdp
23
24  # Pour chaque machine windows
25  while read host; do
26          # Sort le nom court de la machine
27          host_nbt=`echo $host|cut -d . -f 1`
28
29          echo "Mount de c: et d: en local"
30          mount -t cifs -o "username=$admin%$mdp" '//'$host'/c$' $rep/c
31          mount -t cifs -o "username=$admin%$mdp" '//'$host'/d$' $rep/d
32
33          echo "Recherche de nrpe.cfg sur les deux partitions"
34          path_nrpe=$(find $rep -name nrpe.cfg | head -n1)
35
36          if [ ! -z "$path_nrpe" ]; then
```

```
37          echo "nrpe.cfg a ete trouve : "$path_nrpe
38
39                  # Suppression de nrpe.cfg pour ne garder que le path
40                  path_nrpe=${path_nrpe%/*}
41
42                  echo -n "Arret de NRPE...."
43                  cat $rep/stop_nrpe.bat|$rep/winexe --uninstall --runas=$host_nbt/
                        $admin%$mdp -U $host_nbt/$admin%$mdp //$host cmd >> /dev/null &&
                        echo OK || echo NOK
44
45                  if [ "$path_nrpe" != "$rep/c/NRPE" ]; then
46
47                          # Suppression de $rep, transformation de d/ en d:\\ et
                                transformation des /
48                          # en \\ (double pour sed)
49                          path_nrpe_win=${path_nrpe#*/}
50                          path_nrpe_win=${path_nrpe_win/\//:\\\\}
51                          path_nrpe_win=${path_nrpe_win//\//\\\\}
52
53                          # Remet un fichier uninstall par defaut
54                          cp -f $rep/uninstall_nrpe_template.bat $rep/uninstall_nrpe.
                                bat
55
56                          # Adapte le fichier uninstall
57                          sed "s/ancien_dossier/$path_nrpe_win/gi" -i $rep/
                                uninstall_nrpe.bat
58                          sed "s/lecteur/${path_nrpe_win%:*}/gi" -i $rep/uninstall_nrpe
                                .bat
59
60                          echo "Copie outils de suppression"
61                          cp $rep/unlocker.exe $rep/c/
62
63                          echo -n "Desintallation et deplacement de NRPE...."
64                          #cat $rep/uninstall_nrpe.bat|$rep/winexe --uninstall --runas=
                                $host_nbt/$admin%$mdp -U $host_nbt/$admin%$mdp //$host
                                cmd >> /dev/null && echo OK || echo NOK
65                          cat $rep/uninstall_nrpe.bat|$rep/winexe --uninstall --runas=
                                $host_nbt/$admin%$mdp -U $host_nbt/$admin%$mdp //$host
                                cmd
66
67                          # Suppression de l'outils de suppression
68                          #rm $rep/c/unlocker.exe
69
70                          # echo "Changement de repertoire NRPE"
71                          #mv "$path_nrpe" $rep/c/NRPE
72
73                          # Mise a jour des chemins des anciens checks
74                          sed "s/$path_nrpe_win/c:\\\\NRPE/gi" -i $rep/c/NRPE/nrpe.cfg
75
76                          echo -n "Installation de NRPE...."
77                          cat $rep/install_nrpe.bat|$rep/winexe --uninstall --runas=
                                $host_nbt/$admin%$mdp -U $host_nbt/$admin%$mdp //$host
                                cmd >> /dev/null && echo OK || echo NOK
78                  fi
79
80                  echo "Mise a jour du nrpe.cfg pour les nouveaux checks"
81                  cat $rep/nrpe.delta.cfg >> $rep/c/NRPE/nrpe.cfg
82
83                  echo "Copie des nouveaux checks"
84                  cp -r $rep/cent_checks $rep/c/NRPE
85
86                  echo -n "Demarrage de NRPE...."
87                  cat $rep/start_nrpe.bat|$rep/winexe --uninstall --runas=$host_nbt/
                        $admin%$mdp -U $host_nbt/$admin%$mdp //$host cmd >> /dev/null &&
                        echo OK || echo NOK
88
89          # nrpe.cfg n'existe pas : pas de nrpe d'installe ??
90          else
91                  echo "$host : Aucune nrpe.cfg sur c: et d:\n" >&2
92          fi
93
94          # Umount de c: et d:
95          umount $rep/c
96          umount $rep/d
```

```
97
98          # Une petite pause
99          sleep 6
100
101 done < $rep/machines
```

N. Conversion de toutes les machines en DNS vers IP

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Transforme les adresses DNS en adresses IP
6
7  use strict;
8  use warnings;
9  use DBI;
10
11 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
       { 'RaiseError' => 1 });
12
13 my $sql = $mysql->prepare("SELECT host_address addr FROM host WHERE host_address NOT
       REGEXP '^[0-9.]+\$'");
14 $sql->execute;
15
16 while(my $h = $sql->fetchrow_hashref) {
17   my $addr = $h->{'addr'};
18   my $ip = `host $addr | awk '\$NL ~ /address/ { print \$NF }'`;
19   chomp $ip;
20
21   if($ip =~ /^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/) {
22     $mysql->do("UPDATE host SET host_address='$ip' WHERE host_address='$addr'");
23     print "INFO: $addr change en $ip\n";
24   } else {
25     print STDERR "ERREUR: $addr\n";
26   }
27 }
28
29 $sql->finish;
30 $mysql->disconnect;
```

O. Changement interactif des noms de machines en fonction de leur DNS

```
1  #!/usr/bin/perl -w
2  # Auteur <julien@vaubourg.com> pour le CRI de INPL
3  # 2010 - Centreon 2.0.2
4
5  # Remplace les noms d'hosts par leur nom DNS, parfois abrege
6
7  use strict;
8  use warnings;
9  use DBI;
10
11 my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
       { 'RaiseError' => 1 });
12
13 # Selection de tous les hosts qui ne sont pas des tpl
14 my $sql = $mysql->prepare("SELECT host_name name, host_address ip
15         FROM host
```

```
16          WHERE host_register='1'");
17
18  $sql->execute;
19
20  # Pour chaque host
21  while(my $h = $sql->fetchrow_hashref) {
22    my $name = $h->{'name'};
23    my $ip = $h->{'ip'};
24
25    # Recuperation de tous les noms de domaine retournes par une requete DNS invers sur
          l'IP
26    my $hostCmd = `host $ip 2> /dev/null | awk '\$NL ~ /domain name pointer/ { print
          substr(\$NF, 0, length(\$NF)-1) }'`;
27    my @addrs = split /\n/, $hostCmd;
28    my $addr = 0;
29
30    # Pour chacune de ces adresses, on selectionne celle qui porte le nom actuel de la
          machine
31    for (@addrs) {
32      $addr = $_ if $_ =~ /^$name/i;
33    }
34
35    # Si un des DNS retournes ne porte le nom de l'host, il faut une intelligence
          humaine
36    unless($addr) {
37
38      # Suppression du suffixe .inpl-nancy.fr, si besoin est,
39      # sinon le DNS restera complet
40      if($addrs[0]) {
41        $addrs[0] =~ s/\.inpl-nancy\.fr//;
42
43      # Aucun DNS n'a ete trouve pour la machine : c'est le cas pour les sondes
44      # de temperature qui n'ont pas de nom de domaine, par ex
45      } else {
46        $addrs[0] = '!ERREUR!';
47      }
48
49      # Tant qu'un nom n'a pas ete trouve, on propose le premier trouve
50      do {
51        print "\nDOUTE: Nouveau nom pour <$name> (defaut: <", $addrs[0], "> - $ip) : ";
52
53        $addr = <>;
54        chomp($addr);
55
56        # Si un nom a ete propose, on l'enregistre
57        # Sinon, cela signifie qu'on garde celui propose
58        $addr = $addrs[0] if $addr eq "";
59      } while($addr eq '!ERREUR!');
60
61    # Si un des DNS contient le nom de l'host, on se contente de supprimer
          eventuellement
62    # le suffixe INPL et on met a jour
63    } else {
64      $addr =~ s/\.inpl-nancy\.fr//;
65    }
66
67    # Mise a jour du nom d'host
68    if($name ne $addr) {
69      print "CHANGE: <$name> en <$addr>\n";
70      $mysql->do("UPDATE host SET host_name=".$mysql->quote($addr)." WHERE host_name='
          $name'");
71    } else {
72      print "OK: $addr\n";
73    }
74  }
75
76  $sql->finish;
77  $mysql->disconnect;
```

P. Attribution d'un modèle de machine à

toutes les machines d'un groupe

```perl
#!/usr/bin/perl -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

# Attribut un template d'host a tous les hosts d'un groupe

use strict;
use warnings;
use DBI;

my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'tmp', 'tmp',
    { 'RaiseError' => 1 });
my $h;

# Group source
print "Hostgroup des hosts de depart : ";
my $from = <>;
chomp($from);

# Tpl de destination
print "Tpl host de destination : ";
my $to = <>;
chomp($to);

# Verification de l'existence du tpl de destination
my $sql = $mysql->prepare("SELECT TRUE FROM host WHERE host_name=".$mysql->quote($to)
    ." AND host_register='0'");
$sql->execute;

if($sql->rows == 0) {
  print STDERR "ERREUR: Le TPL d'hote ".$mysql->quote($to)." n'existe pas.\n";
  exit 1;
}

# Selection des hosts du hostgroup qui n'ont pas encore le tpl d'attribue
$sql = $mysql->prepare("SELECT host_host_id id
      FROM hostgroup_relation
      WHERE hostgroup_hg_id=(SELECT hg_id
            FROM hostgroup
            WHERE hg_name=".$mysql->quote($from).")
      AND host_host_id NOT IN (SELECT host_host_id
            FROM host_template_relation
            WHERE host_tpl_id=(SELECT host_id
                FROM host
                WHERE host_name=".$mysql->quote($to)."
                AND host_register='0'))");
$sql->execute;
my $nbHosts = $sql->rows;

# Association des hosts trouves avec le tpl donne
$mysql->do("INSERT INTO host_template_relation (
    host_tpl_id,
    host_host_id
  ) VALUES (
    (SELECT host_id FROM host WHERE host_name=".$mysql->quote($to)." AND
        host_register='0'),
    ".$h->{'id'}."
  )") while($h = $sql->fetchrow_hashref);

print "\nINFO: Les $nbHosts hosts appartenant a $from ont ete noues avec le tpl $to.\
    n";

$mysql->disconnect;
```

Q. Synchronisation LDAP des utilisateurs et

groupes de la base Centreon

```
#!/usr/bin/perl -w
# Auteur <julien@vaubourg.com> pour le CRI de INPL
# 2010 - Centreon 2.0.2

# Synchronisation de la base de donnee MySQL Centreon des utilisateurs en fonction du LDAP.
# Ajoute, met a jour ou supprime les utilisateurs et leurs relations en fonction du LDAP.
#
# Gere egalement des groupes d'ACL (vues) de Centreon en fonction des groupes LDAP de l'utilisateur
# (ajoute les groupes et lie les utilisateurs avec, ne les supprime pas et ne delie pas les utilisateurs,
# sauf en cas de suppression de celui-ci, puisque c'est un cas rarissime)
#
# Des groupes de type contactgroup sont crees en parallele et geres de la meme facon
# (ils sont dependant des groupes ACL : si un utilisateur n'appartient plus au contactgroup dans Centreon mais
# appartient toujours au groupe ACL, il ne sera pas rajoute, permettant ainsi une plus grande fluidite)
#
# Les groupes sont reconnus en fonction de leur nom, et les utilisateurs en fonction de leur dn
#
# Relance les Nagios s'il en est besoin.

### Droits detailles de l'utilisateur MySQL synchro (mdp: synchro, 1ere ligne) qui sert pour la connexion en local :
#
# GRANT ALL ON centreon2.contact TO synchro@localhost IDENTIFIED BY 'synchro';
# GRANT LOCK TABLES ON centreon2.* TO synchro@localhost;
# GRANT SELECT ON centreon2.general_opt TO synchro@localhost;
# GRANT SELECT ON centreon2.command TO synchro@localhost;
# GRANT SELECT ON centreon2.timeperiod TO synchro@localhost;
# GRANT SELECT ON centreon2.nagios_server TO synchro@localhost;
# GRANT ALL ON centreon2.contact_hostcommands_relation TO synchro@localhost;
# GRANT ALL ON centreon2.contact_servicecommands_relation TO synchro@localhost;
# GRANT ALL ON centreon2.acl_groups TO synchro@localhost;
# GRANT ALL ON centreon2.acl_group_contacts_relations TO synchro@localhost;
# GRANT ALL ON centreon2.contactgroup TO synchro@localhost;
# GRANT ALL ON centreon2.contactgroup_contact_relation TO synchro@localhost;
# GRANT SELECT,DELETE ON centreon2.contact_host_relation TO synchro@localhost;
# GRANT SELECT,DELETE ON centreon2.contact_param TO synchro@localhost;
# GRANT SELECT,DELETE ON centreon2.contact_service_relation TO synchro@localhost;
#
###

use strict;
use warnings;
use List::Util 'first';
use Net::LDAP;
use DBI;


##
## Divers
##

# Nom de la commande de notification qui sera attribuee par defaut
my $notifyCommand_email = 'host-notify-by-email';

# Mot de passe MySQL pour l'utilisateur synchro
my $pwdMySQLSynchro = '*******';

# Nom de l'attribut qui defini les groupes pour un utilisateurs
my $attrGroup = 'INPLatrGroupe';

# Groupes (attrGroup) du LDAP pour lesquels les utilisateurs seront utilises
my @topGroups = qw( CRI );

# Groupe pour lequel tous les utilisateurs ont automatiquement les pleins pouvoirs
```

```
64  my $admGroup = 'CRI-SRT';
65
66  # Definition de la date pour les commentaires de utilisateurs ajoutes/modifies
67  my ($s, $min, $h, $d, $m, $y) = localtime time;
68  my $date = sprintf('%02d/%02d/%d a %02dh%02d', $d, $m+1, 1900+$y, $h, $min);
69
70  # Mises a jour qui necessitent de relancer les nagios :
71  # - qui ne concernent pas juste les ACL
72  # - qui ne sont pas des ajouts, puisque ca signifie qu'elles n'ont pas encore d'
        utilite
73  my $updates = 0;
74
75  # C'est parti
76  print "\nSYNCHRONISATION LDAP DU $date\n";
77
78
79  ##
80  ## Recuperation des utilisateurs de Centreon
81  ##
82
83  # Connexion MySQL
84  my $mysql = DBI->connect('DBI:mysql:database=centreon2;host=localhost', 'synchro',
        $pwdMySQLSynchro, { 'RaiseError' => 1 });
85
86  # Dump des utilisateurs inscrits dans Centreon qui s'authentifient avec le LDAP
87  # qui servent pour les vues de Centreon (ACL) comme pour les notifications des Nagios
88  my $sql = $mysql->prepare("SELECT
89    contact_id,
90    contact_ldap_dn,
91    contact_name,
92    contact_alias,
93    contact_email
94  FROM contact WHERE contact_ldap_dn IS NOT NULL");
95
96  $sql->execute;
97  my %contactsSQL;
98
99  # Creation d'un hash de hash de tous les utilisateurs Centreon qui ont une
        authentification LDAP,
100 # en donnant la correspondance de noms des champs entre le MySQL et le LDAP
101 while(my $c = $sql->fetchrow_hashref) {
102   $contactsSQL{$c->{'contact_ldap_dn'}} = {
103     id => $c->{'contact_id'},
104     uid => $c->{'contact_alias'},
105     displayName => $c->{'contact_name'},
106     mail => $c->{'contact_email'}
107   };
108 }
109
110
111 ##
112 ## Parcours des utlisateurs du LDAP et mise en conformite du MySQL
113 ##
114
115 # Extraction des parameters LDAP depuis l'interface de Centreon (Administration >
        Options > LDAP)
116 $sql = $mysql->prepare("SELECT
117   ldap_host,
118   ldap_port,
119   ldap_base_dn
120 FROM general_opt");
121
122 $sql->execute;
123 my $ldap_params = $sql->fetchrow_hashref;
124
125 # Connexion au LDAP
126 my $ldap = Net::LDAP->new(
127   $ldap_params->{'ldap_host'},
128   port => $ldap_params->{'ldap_port'},
129   version => 3
130 ) or die "ERREUR: Impossible de contacter l'annuaire LDAP ($@)";
131
132 # Liaison
133 my $msg = $ldap->bind;
```

```
134 $msg->code && die "ERREUR: ".$msg->error;
135
136 # Parcours des utilisateurs, groupe par groupe (permet de selectionner les branches d
      'utiliateurs
137 # interessants et eventuellement diviser les requetes pour en recevoir moins a la
      fois
138 foreach my $topGroup (@topGroups) {
139
140   # Recherche
141   $msg = $ldap->search(
142     base   => $ldap_params->{'ldap_base_dn'},
143     scope  => 'sub',
144     filter => "(&(objectClass=person)($attrGroup=$topGroup))",
145     attrs  => [ 'uid', 'displayName', 'mail', $attrGroup ]
146   );
147
148   $msg->code && die "ERREUR: ".$msg->error;
149   my @dnInLDAP;
150
151   # Pour chaque utilisateur trouve dans le LDAP, on l'ajoute au MySQL si il n'existe
        pas, ou bien on modifie ses
152   # infos si elles ont changees depuis le LDAP. On le supprime si il n'existe plus
        dans le LDAP, et on ajoute
153   # ses eventuels nouveaux groupes si ils n'existent pas deja. Sinon, on l'ajoute
        juste dans le groupe.
154   foreach my $person ($msg->all_entries) {
155
156     # Une absence d'email est un crime dans Centreon
157     my $mail = $person->get_value('mail') ? $person->get_value('mail') : 'no@mail';
158
159     # Recuperation de la liste des groupes auquels appartient l'utilisateur
160     my $personGroups = $person->get_value($attrGroup, asref => 1);
161
162     # L'utilisateur LDAP courant n'existe pas : on l'ajoute au MySQL
163     unless($contactsSQL{$person->dn}) {
164       print "INFO: Ajout de ".$person->dn."\n";
165
166       # Fermeture des tables avant d'ecrire, ca ne coute rien
167       $mysql->do("LOCK TABLES
168         contact WRITE,
169         contact_hostcommands_relation WRITE,
170         contact_servicecommands_relation WRITE,
171         command READ,
172         timeperiod READ
173       ");
174
175       $mysql->do("SET AUTOCOMMIT = 0");
176
177       # Injection avec les parametres par defaut
178       # (notifications 24/7, interface en FR, acces au Centreon sans etre admin,
            notification en cas de down warning ou critical)
179       $mysql->do("INSERT INTO contact (
180         timeperiod_tp_id,
181         timeperiod_tp_id2,
182         contact_alias,
183         contact_name,
184         contact_passwd,
185         contact_lang,
186         contact_host_notification_options,
187         contact_service_notification_options,
188         contact_email,
189         contact_pager,
190         contact_comment,
191         contact_oreon,
192         contact_admin,
193         contact_type_msg,
194         contact_activate,
195         contact_auth_type,
196         contact_ldap_dn,
197         contact_acl_group_list,
198         contact_autologin_key,
199         contact_charset
200       ) VALUES (
201         (SELECT tp_id FROM timeperiod WHERE tp_name='24x7'), timeperiod_tp_id,
```

```
202           ".$mysql->quote($person->get_value('uid')).", ".$mysql->quote($person->
                  get_value('displayName')).",
203           NULL, 'fr_FR','d','w,c',
204           ".$mysql->quote($mail).", NULL,
205           '[Ajout] Synchronisation LDAP du $date',
206           '1','".((first { $_ eq $admGroup } @$personGroups) ? 1 : 0)."',NULL,'1','ldap
                  ',
207           '".$person->dn."',
208           NULL,NULL,NULL
209         );");
210
211         # Liaison de l'utilisateur avec une commande de notification pour les alertes
                de services
212         $mysql->do("INSERT INTO contact_hostcommands_relation (
213           contact_contact_id,
214           command_command_id
215         ) VALUES (
216           (SELECT contact_id FROM contact WHERE contact_ldap_dn='".$person->dn."' LIMIT
                  1),
217           (SELECT command_id FROM command WHERE command_name=".$mysql->quote(
                  $notifyCommand_email)." AND command_type=1 LIMIT 1)
218         )");
219
220         # Liaison de l'utilisateur avec une commande de notification pour les alertes d
                'hotes
221         $mysql->do("INSERT INTO contact_servicecommands_relation (
222           contact_contact_id,
223           command_command_id
224         ) VALUES (
225           (SELECT contact_id FROM contact WHERE contact_ldap_dn='".$person->dn."' LIMIT
                  1),
226           (SELECT command_id FROM command WHERE command_name=".$mysql->quote(
                  $notifyCommand_email)." AND command_type=1 LIMIT 1)
227         )");
228
229         # On remballe
230         $mysql->do("COMMIT");
231         $mysql->do("SET AUTOCOMMIT = 1");
232         $mysql->do("UNLOCK TABLES");
233
234       # L'utilisateur a des informations differentes du LDAP au MySQL : on met a jour
              ce dernier
235       } elsif($contactsSQL{$person->dn}->{'uid'} ne $person->get_value('uid') ||
236         $contactsSQL{$person->dn}->{'displayName'} ne $person->get_value('displayName')
                ||
237         $contactsSQL{$person->dn}->{'mail'} ne $mail) {
238
239         print "INFO: Modification de ".$person->dn."\n";
240
241         # Mise a jour, avec une trace de debug dans le champ de commentaire de l'
                utilisateur
242         $mysql->do("UPDATE contact SET
243           contact_name=".$mysql->quote($person->get_value('uid')).",
244           contact_alias=".$mysql->quote($person->get_value('displayName')).",
245           contact_email=".$mysql->quote($mail).",
246           contact_comment=CONCAT(contact_comment, '\n[Maj] Synchronisation LDAP du
                  $date')
247         WHERE contact_ldap_dn='".$person->dn."'");
248
249         $updates++;
250       }
251
252       # Parcours de ses groupes : si un de ses groupes n'existe pas, il sera cree, en
              ACL (vues) et en contactgroup (notifications)
253       # Dans tous les cas, l'utilisateur est ajoute dans le groupe ACL si il n'y est
              pas deja.
254       # Le contactgroup est dependant du groupe ACL (on ne rajoute pas l'utilisateur
              dans le contactgroup si il n'y est plus)
255       foreach (@$personGroups) {
256         my $isInGroup = 1;
257         my $isNewGroup = 0;
258
259         # Le groupe ACL existe-t-il deja ?
260         $sql = $mysql->prepare("SELECT COUNT(*) exist FROM acl_groups WHERE
```

```
            acl_group_name=".$mysql->quote($_));
261       $sql->execute;
262
263       # Ajout du groupe si ca n'est pas le cas
264       unless($sql->fetchrow_hashref->{'exist'}) {
265
266         # Un administrateur ne peut pas etre affecte a un groupe d'ACL (il a deja
              tous les droits)
267         if(! first { $_ eq $admGroup } @$personGroups) {
268           print "INFO: Ajout du groupe $_ (ACL)\n";
269
270           # Injection
271           $mysql->do("INSERT INTO acl_groups (
272             acl_group_name,
273             acl_group_alias,
274             acl_group_activate
275           ) VALUES (
276             ".$mysql->quote($_).",
277             ".$mysql->quote("$_ (LDAP)").",
278             1
279           )");
280         }
281
282         $sql = $mysql->prepare("SELECT COUNT(*) exist FROM contactgroup WHERE cg_name
              =".$mysql->quote($_));
283         $sql->execute;
284
285         # Le groupe jumeau existe-t-il en contactgroup ? (theoriquement non)
286         unless($sql->fetchrow_hashref->{'exist'}) {
287           print "INFO: Ajout du groupe $_ (ContactGroup)\n";
288
289           $mysql->do("INSERT INTO contactgroup (
290             cg_name,
291             cg_alias,
292             cg_comment,
293             cg_activate
294           ) VALUES (
295             ".$mysql->quote($_).",
296             ".$mysql->quote("$_ (LDAP)").",
297             '[Ajout] Synchronisation LDAP du $date',
298             '1'
299           )");
300         }
301
302         # Puisque le groupe vient d'etre cree, inutile de verifier si l'utilisateur
              est deja dedans
303         # Et si c'est un nouveau groupe, inutile de relancer les Nagios pour l'ajout
              d'utilisateur
304         $isNewGroup = 1;
305
306       # Si le groupe existe deja, verification de l'appartenance de notre utilisateur
            a ce groupe
307       } else {
308
309         $sql = $mysql->prepare("SELECT COUNT(*) exist FROM
              acl_group_contacts_relations
310           WHERE contact_contact_id=(SELECT contact_id FROM contact WHERE
                contact_ldap_dn='".$person->dn."' LIMIT 1)
311           AND acl_group_id=(SELECT acl_group_id FROM acl_groups WHERE acl_group_name=
                ".$mysql->quote($_)." LIMIT 1)");
312
313         $sql->execute;
314         $isInGroup = $sql->fetchrow_hashref->{'exist'};
315       }
316
317       # Ajout de l'utilisateur dans le groupe si il n'y est pas deja
318       if(!$isInGroup || $isNewGroup) {
319
320         # Un administrateur ne peut pas etre affecte a un groupe d'ACL (il a deja
              tous les droits)
321         if(! first { $_ eq $admGroup } @$personGroups) {
322           print "INFO: Ajout de ".$person->dn." dans $_ (ACL)\n";
323
324           # Injection de la relation utilisateur-groupe ACL
```

```
325             $mysql->do("INSERT INTO acl_group_contacts_relations (
326               contact_contact_id,
327               acl_group_id
328             ) VALUES (
329               (SELECT contact_id FROM contact WHERE contact_ldap_dn='".$person->dn."'
                      LIMIT 1),
330               (SELECT acl_group_id FROM acl_groups WHERE acl_group_name=".$mysql->quote
                      ($_)." LIMIT 1)
331             )");
332           }
333
334           # L'utilisateur est-il deja rattache au contactgroup ? (theoriquement non)
335           $sql = $mysql->prepare("SELECT COUNT(*) exist FROM
                  contactgroup_contact_relation
336             WHERE contact_contact_id=(SELECT contact_id FROM contact WHERE
                    contact_ldap_dn='".$person->dn."' LIMIT 1)
337             AND contactgroup_cg_id=(SELECT cg_id FROM contactgroup WHERE cg_name=".
                    $mysql->quote($_)." LIMIT 1)");
338
339           $sql->execute;
340
341           # Injection de la relation utilisateur-contactgroup
342           unless($sql->fetchrow_hashref->{'exist'}) {
343             print "INFO: Ajout de ".$person->dn." dans $_ (ContactGroup)\n";
344
345             $mysql->do("INSERT INTO contactgroup_contact_relation (
346               contact_contact_id,
347               contactgroup_cg_id
348             ) VALUES (
349               (SELECT contact_id FROM contact WHERE contact_ldap_dn='".$person->dn."'
                      LIMIT 1),
350               (SELECT cg_id FROM contactgroup WHERE cg_name=".$mysql->quote($_)." LIMIT
                      1)
351             )");
352           }
353
354           # Inutile de prendre l'ajout en consideration pour le reload des nagios si c'
                  est un nouveau groupe
355           # (il n'est encore rattache a rien, donc invisible pour les nagios)
356           # NOTE: Si le groupe a ete cree pour l'utilisateur d'avant, on considere donc
                   quand meme qu'il est potentiellement utilise
357           $updates++ unless $isNewGroup;
358         }
359       }
360
361       # Tenue d'un tableau de dn (identifiants unique des utilisateurs) du LDAP, pour
              savoir ensuite
362       # qui est dans le MySQL mais n'est plus dans le LDAP
363       push(@dnInLDAP, $person->dn);
364     }
365
366     # Suppression des utilisateurs qui sont dans MySQL en ayant un dn LDAP attribue,
          alors qu'il n'existent plus dans le LDAP
367     foreach my $dn (keys %contactsSQL) {
368
369       # Si l'utilisateur n'existe pas dans les dn du LDAP
370       unless(first {$_ eq $dn} @dnInLDAP) {
371         print "INFO: Suppression de $dn\n";
372
373         # Suppression de la table de contacts, de celles servant pour les commandes de
                notifications rattachees et suppression
374         # des appartenances aux groupes
375         $mysql->do("DELETE FROM contact WHERE contact_id=".$contactsSQL{$dn}->{'id'});
376         $mysql->do("DELETE FROM contact_hostcommands_relation WHERE contact_contact_id=
                ".$contactsSQL{$dn}->{'id'});
377         $mysql->do("DELETE FROM contact_servicecommands_relation WHERE
                contact_contact_id=".$contactsSQL{$dn}->{'id'});
378         $mysql->do("DELETE FROM contact_host_relation WHERE contact_id=".$contactsSQL{
                $dn}->{'id'});
379         $mysql->do("DELETE FROM contact_service_relation WHERE contact_id=".
                $contactsSQL{$dn}->{'id'});
380         $mysql->do("DELETE FROM acl_group_contacts_relations WHERE contact_contact_id="
                .$contactsSQL{$dn}->{'id'});
381         $mysql->do("DELETE FROM contactgroup_contact_relation WHERE contact_contact_id=
```

```
            ".$contactsSQL{$dn}->{'id'});
382         $mysql->do("DELETE FROM contact_param WHERE cp_contact_id=".$contactsSQL{$dn
                }->{'id'});
383
384         $updates++;
385       }
386     }
387
388   }
389
390   # Si il y a eu des modifications du MySQL, on relance tous les Nagios (reload)
391   # (pour optimiser il faudra ne relancer que lorsque ca concerne des utilisateurs qui
        ne servent pas juste pour les ACL)
392   if($updates) {
393
394     # Recuperation des parametres des differentes Nagios
395     $sql = $mysql->prepare("SELECT id, name, localhost, init_script FROM nagios_server"
          );
396     $sql->execute;
397
398     # Pour chaque Nagios, recharger sa configuration
399     while(my $n = $sql->fetchrow_hashref) {
400       print "INFO: Rechargement de ".$n->{'name'}."\n";
401
402       # Nagios du Centreon
403       if($n->{'localhost'}) {
404         my $init = $n->{'init_script'};
405         `sudo $init reload`;
406
407       # Nagios satellite
408       } else {
409         my $nsId = $n->{'id'};
410         `echo 'RELOAD:$nsId' >> /var/lib/centreon/centcore.cmd`
411       }
412     }
413
414   # Rien a faire
415   } else {
416     print "INFO: Aucun rechargement de Nagios.\n";
417   }
418
419   # Vous pouvez eteindre votre poste de television et reprendre une activite normale
420   $ldap->unbind;
421   $sql->finish;
422   $mysql->disconnect;
```

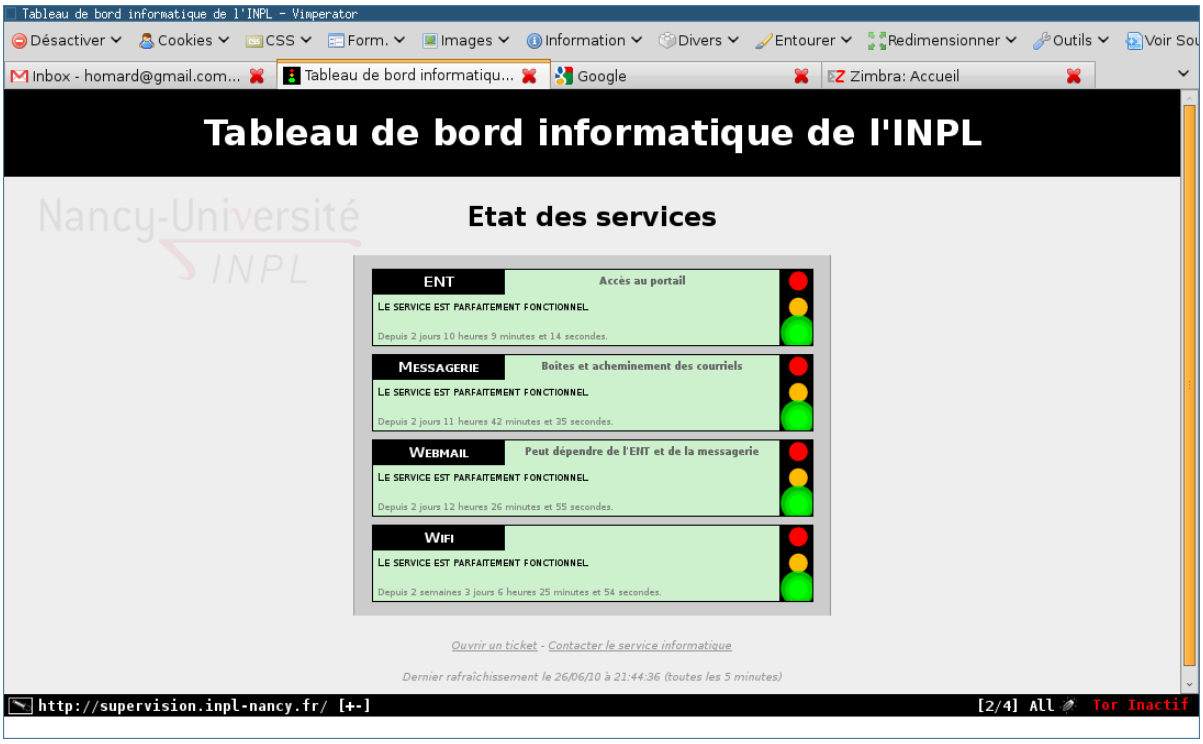
R. Exemple de fichier de configuration de la vue publique

```
1   # Copyright (C) 2010 Julien VAUBOURG <julien@vaubourg.com>
2   #
3   # SYNTAX :
4   #
5   # PAGE_TITLE = pageTitle
6   #
7   # LABEL [! DESCRIPTION] ID=id
8   # {host|HG:hostgroup}[!service] [[OR] {host|HG:hostgroup}[!service] ...] [![
        nbCritForWarn[%]!]nbCritForCrit[%]]
9   # [[AND] {host|HG:hostgroup}[!service] [[OR] {host|HG:hostgroup}[!service] ...] [![
        nbCritForWarn[%]!]nbCritForCrit[%]]
10  #     ...]
11  #
12  # LINES EXAMPLE :
13  #
14  # host1!my-service1 host2!my-service1 host2!my-service2 host3!my-service1 !2!3
15  # => If 2 services are critical, this line returns warning, and if there is 3
        critical it returns critical ; else ok.
```

```
16 # => Warning are considered ok (the service works)
17 # => Here, !2 could have been written !50%
18 #
19 # host2!my-service1 HG:hostgroup1!my-service5 host1!my-service3 !2
20 # => If 2 services are critical, this line returns critical, else ok
21 # => HG:hostgroup1!my-service5 will replaced by : hostA!my-service5 hostB!my-service5
       hostC!my-service5
22 #
23 # host2!my-service2 host2!my-service4
24 # => With no thresholds at the end, this line returns critical is one of its services
       is critical
25 #
26 # COMPLETE EX. :
27 #
28 # # My sample service
29 # My Global Service
30 #   host1!my-service1 host2!my-service1 host2!my-service3 !2!3
31 #   host2!my-service1 HG:hostgroup1!my-service5 host1!my-service3 !2
32 #
33 # => If one of these lines returns critical, this bloc is critical
34 # => Idem for warning
35 #   => Lines beginning with sharp are comments and can be somewhere (but no at the
       end of a line)
36 #
37 # TIPS :
38 #
39 # This :
40 #   host1!my-service1 host1!my-service2 !1
41 # Is equivalent to writing :
42 #   host1!my-service1
43 #   host1!my-service2
44 #
45 # INFORMATIONS :
46 #
47 # - With a line beginning by PAGE_TITLE=, you can define the main page title.
48 # - If this conf is named confs/conf_toto, it will used on URL/?toto.
49 # - With the ID=xxx, you can access to its image directly specifying [?&]id=xxx in
       the URL.
50 #   And with &rotate in more, you can have the traffic lights horizontally.
51 # - For debugging, you can activate the $allowDebugMode variable at the top of the
       PHP script,
52 # and add [?&]debug to the main page URL.
53
54
55 PAGE_TITLE=Tableau de bord informatique de l'INPL
56
57 ENT ! Acces au portail ID=ent
58   ent1!App-ENT ent2!App-ENT
59   entdb!App-HTTP
60
61 Messagerie ! Boites et acheminement des courriels ID=messagerie
62   # Mailbox
63   etorki.cri!App-IMAPS
64   etorki.cri!App-POPS
65   etorki.cri!App-HTTPS
66   feta.cri!App-Zimbra-IMAP rollot.cri!App-Zimbra-IMAP !1
67   feta.cri!App-Zimbra-IMAPS rollot.cri!App-Zimbra-IMAPS !1
68   feta.cri!App-Zimbra-POP rollot.cri!App-Zimbra-POP !1
69   feta.cri!App-Zimbra-POPS rollot.cri!App-Zimbra-POPS !1
70   # Ancien LDAP
71   athena!App-LDAP-3500 entdb!App-LDAP-3500
72   # Nouveau LDAP
73   entdb!App-LDAP rouy.cri!App-LDAP-390
74   # SMTP
75   epoisse.cri
76   chaource.cri
77   epoisse.cri!App-Zimbra-Amavisd chaource.cri!App-Zimbra-Amavisd !1!2
78   epoisse.cri!App-Zimbra-Amavis-Postfix chaource.cri!App-Zimbra-Amavis-Postfix !1!2
79   # DNS
80   epoisse.cri!App-DNS athena!App-DNS
81
82 Webmail ! Peut dependre de l'ENT et de la messagerie ID=webmail
83   entdb!App-HTTPS-/cas/login
84
```

```
85  Wifi ID=wifi
86    HG: Equipement_Bornes-Wifi !20%!70%
```

S. Interface web de la vue publique



T. Génération de services globaux pour Centreon

```php
<?php

/* Copyright (C) 2010 Julien VAUBOURG <julien@vaubourg.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 */

error_reporting(E_ALL);
ini_set('display_errors', 1);


define('OK', 0);
define('WARNING', 1);
```

```php
define('CRITICAL', 2);

$states = array('ok', 'warning', 'critical');


// Every $refresh seconds, the page will refreshed
$refresh = 5 * 60;

// Allow debug mode, activated by add ?debug GET parameter to the URL
$allowDebugMode = false;


/*******************************
 ********** FUNCTIONS **********
 *******************************/

/*
 * Read the cache file corresponding to the conf file used
 * and return a globalState array version, completed
 * (see its definition in the main program for consult the structure)
 *
 * Cache file syntax (with line numbers) :
 * 0: timestampOfTheLastCachRefresh
 * 1: titlePage
 * [n: ID|label|description|status|timestampOfTheLastUpdate
 *  ...]
 */
function readCache($file) {
  $lastRefresh = false;
  $globalStates = array();

  // There is an existing cache file for the conf file used
  if(file_exists($file)) {

    // Opening cache file
    $fcache = fopen($file, 'r')
      or die("ERROR: The cache file can't be read");

    // The first line is the unix timestamp of the last cache refresh
    $line = fgets($fcache, 1024);
    $lastRefresh = trim($line);

    // The second line is the page title
    $line = fgets($fcache, 1024);
    $page_title = trim($line);

    $l = 2;

    // For each global service of the cache file
    while($line = fgets($fcache, 1024)) {
      $l++;

      // See comments' function for consult the lines structure
      list($id, $label, $description, $state, $time) =
        explode('|', trim($line));

      (empty($id) || empty($label) || empty($state) && $state != 0 || empty($time))
        && die("ERROR (cache l.$l) : Wrong syntax or missing field");

      // Filling the main array
      $globalStates[$id]['label'] = $label;
      $globalStates[$id]['description'] = $description;
      $globalStates[$id]['state'] = $state;
      $globalStates[$id]['newState'] = OK;
      $globalStates[$id]['time'] = $time;
      $globalStates[$id]['newTime'] = false;
    }

    fclose($fcache);

  // There is no cache file
  } else {
    $page_title = NULL;
    $lastRefresh = -1;
```

```
100       $globalState = array();
101     }
102
103     return array($lastRefresh, $page_title, $globalStates);
104   }
105
106   /*
107    * Callback function, for delete pipes in the cache fields
108    * (pipes are used as delimiters)
109    */
110   function dropPipes(&$value, $key) {
111     $value = str_replace('|', NULL, $value);
112   }
113
114   /*
115    * Update the main array globalStates, and the cache file
116    */
117   function updateCacheAndValues($file, $lastRefresh, $page_title, &$globalStates) {
118
119     // New refresh date
120     $cache = time()."\n$page_title\n";
121
122     // For each global service
123     while($id = key($globalStates)) {
124       $infos =& $globalStates[$id];
125
126       // The update time is refreshed only if the state was changed, and if the old
              date is
127       // really the oldest (wich should alway be the case)
128       if($infos['state'] != $infos['newState'] && $infos['time'] < $infos['newTime']) {
129         $infos['state'] = $infos['newState'];
130         $infos['time'] = $infos['newTime'];
131       }
132
133       // Delete pipe chars, used as delimiters
134       array_walk($infos, 'dropPipes');
135
136       // New cache line, updated
137       $cache .= sprintf("%s|%s|%s|%s|%s\n",
138           $id,
139           $infos['label'],
140           $infos['description'],
141           $infos['state'],
142           $infos['time']
143         );
144
145       next($globalStates);
146     }
147
148     file_put_contents($file, $cache)
149       || die("ERROR: The cache file can't be written");
150   }
151
152   /*
153    * Extends hostgroups, prefixed by HG: :
154    *  If the line contains HG:my-hg!my-srvce, and my-hg has 3 hosts, it'll be replaced
          by :
155    *  my-host1!my-srvce my-host2!my-srvce my-host3!my-srvce
156    */
157   function extendHG($line, $l) {
158
159     // For each hostgroup definition found
160     while(preg_match('/HG:([^!\s]+)!?(\S+)?/', $line, $hg)) {
161       $hosts = array();
162
163       // This hosgroup definition has an associated service ?
164       (count($hg) > 2) ?
165         list($hg_def, $hg_name, $hg_service) = $hg :
166         list($hg_def, $hg_name) = $hg;
167
168       // Select all hosts that it belong to the hostgroup
169       $req = mysql_query(
170         sprintf("SELECT host_name
171             FROM centreon2.hostgroup_relation, centreon2.host
```

```
172             WHERE host_host_id = host_id
173             AND hostgroup_hg_id=(SELECT hg_id
174                   FROM centreon2.hostgroup
175                   WHERE hg_name='%s'
176                   LIMIT 1)",
177           mysql_real_escape_string($hg_name)
178         ));
179
180       // This hosgroup really exist ?
181       mysql_num_rows($req)
182         || die("ERROR (conf 1.$l) : Hostgroup <em>$hg_name</em> does not exist or is
              empty");
183
184       // Each host, eventually associated to a service, is concatenated
185       while($host = mysql_fetch_assoc($req))
186         $hosts[] = $host['host_name'].(isset($hg_service) ? "!$hg_service" : NULL);
187
188       // The hostgroup definition is replaced by its members
189       $line = str_replace($hg_def, implode(' ', $hosts), $line);
190     }
191
192     // Line definition, without hostgroup definitions
193     return $line;
194   }
195
196   /*
197    * Replace the percentages by absolute numbers
198    */
199   function replacePercent($percent, $nbServices) {
200     if(preg_match('/(\d+)%$/', $percent, $nb))
201       $percent = $nbServices / 100 * $nb[1];
202
203     return $percent;
204   }
205
206   /*
207    * Return a relative date
208    * Adapted from :
209    *  http://blog.angechierchia.com/php-mysql/afficher-une-date-relative-en-php
210    */
211   function getRelativeTime($time) {
212     $timeDiff = time() - $time;
213
214     if($timeDiff <= 0)
215       return "moins d'une seconde";
216
217     $timeDiff = abs($timeDiff);
218
219     $times = array(
220       31104000 => 'an{s}',        // 12 * 30 * 24 * 60 * 60 secondes
221       2592000  => 'mois',         // 30 * 24 * 60 * 60 secondes
222       604800   => 'semaine{s}',   // 7 * 24 * 60 * 60 secondes
223       86400    => 'jour{s}',      // 24 * 60 * 60 secondes
224       3600     => 'heure{s}',     // 60 * 60 secondes
225       60       => 'minute{s}');
226
227     $strTime = NULL;
228
229     // Until that the rest can't being converted
230     while($timeDiff >= 60) {
231       foreach($times AS $seconds => $unit) {
232         $delta = floor($timeDiff / $seconds);
233
234         if($delta >= 1) {
235           $unit = str_replace('{s}', ($delta == 1 ? NULL : 's'), $unit);
236           $strTime .= "$delta $unit ";
237           $timeDiff -= $delta * $seconds;
238         }
239       }
240     }
241
242     // If there is still seconds
243     ($timeDiff > 0)
244       && $strTime .= "et $timeDiff secondes";
```

```
245
246     return trim($strTime);
247   }
248
249
250
251   /*******************************
252    ************* MAIN *************
253    *******************************/
254
255   // ID of the global service that is parsed
256   $currentDefID = false;
257
258   // Debug mode activated (this requires to have $allowDebugMode on true)
259   if(isset($_GET['debug'])) {
260     $debug = true;
261     unset($_GET['debug']);
262   } else
263     $debug = false;
264
265   // With ?id=<ID> in the URL, you can have only the traffic lights for only one
        service
266   if(isset($_GET['id'])) {
267     $serviceID = strtolower($_GET['id']);
268     unset($_GET['id']);
269   } else
270     $serviceID = false;
271
272   // With ?id=<ID>&rotate, you can have the traffic lights horizontally
273   if(isset($_GET['rotate'])) {
274     $rotate = true;
275     unset($_GET['rotate']);
276   } else
277     $rotate = false;
278
279   // If this page is called with ?toto, the conf file confs/conf_toto will be used
280   // By default, it'll be confs/conf_public
281   $confID = empty($_GET) ? 'public' : key($_GET);
282
283   // Opening conf file
284   $fconf = fopen("./confs/conf_$confID", 'r')
285     or die("ERROR: The conf file can't be read");
286
287   // Global states hash structure :
288   // 'ID' => [
289   //   'label' => AAAA
290   //   'state' => X
291   //   'description' => AAAA
292   //   'time' => XXXX
293   //   'newTime' => XXXX
294   // ]
295   list(
296     $lastRefresh, // Last time the cache has been updated
297     $page_title, // Main page title
298     $globalStates // See above
299   ) = readCache("./caches/cache_$confID");
300
301   $l = 0;
302
303   // Cache's infos are outdated
304   if(time() - $lastRefresh > $refresh) {
305
306     // The supervision user has the select right on all tables of nagios and centreon2
307     mysql_connect('centreon.cri.inpl-nancy.fr', 'supervision', 'supervision')
308       || die ("ERROR: Mysql connect");
309
310     // For each line of the conf file
311     while($line = fgets($fconf, 1024)) {
312       $l++;
313
314       // A line beginning by PAGE_TITLE is the line that define the page title
315       if(preg_match('/^\s*PAGE_TITLE\s*=\s*(.*)/', $line, $title)) {
316         $page_title = htmlentities(utf8_decode($title[1]));
317
```

```
318       // A white line is a separator between both global service definition
319       } elseif(preg_match('/^\s*$/', $line))
320         $currentDefID = false;
321
322       // All lines except comments
323       elseif(!preg_match('/^\s*#/', $line)) {
324
325         // Drop white spaces and CR
326         $line = trim($line);
327
328         // This line is not part of the current definition
329         // (it's the begin of a new definition)
330         if(!$currentDefID) {
331           $description = NULL;
332           $refID = false;
333
334           // If there is an ID found for this global service, it's recovered
335           // and deleted
336           if(preg_match('/\s*ID\s*=\s*([a-z0-9_-]+)\s*$/i', $line, $id)) {
337             $refID = strtolower($id[1]);
338             $line = str_replace($id[0], NULL, $line);
339           } else
340             die("ERROR (conf 1.$l) : No ID found");
341
342           // If the user have gave a service ID in the URL, and this global service
343           // not corresponding to this ID, it's not considered
344           if(!$serviceID || $serviceID == $refID) {
345
346             // This global service was not in the cache
347             !isset($globalStates[$refID])
348               && $globalStates[$refID] = array(
349                 'description' => NULL,
350                 'state' => -1,
351                 'newState' => OK,
352                 'time' => -1,
353                 'newTime' => -1
354               );
355
356             // If there is a ! delimiter in the label, this means that there is
357             // a description attached (label ! description)
358             if(strpos($line, '!')) {
359               list($label, $description) = preg_split('/\s*!\s*/', $line);
360
361               $globalStates[$refID]['label'] = $label;
362               $globalStates[$refID]['description'] = $description;
363             } else
364               $globalStates[$refID]['label'] = $line;
365
366             $currentDefID = $refID;
367           }
368
369         // This line is part of the current definition.
370         // It's considered only if the global service state is not already critical.
371         // If a serviceID is defined (the user want only one picture for one service),
               it's considered if
372         // it's in the global service corresponding to the service ID.
373         } elseif($globalStates[$currentDefID]['newState'] != CRITICAL && (!$serviceID
               || $currentDefID)) {
374
375           // Change hostgroup definitions by its hosts
376           $line = extendHG($line, $l);
377
378           // Drop the optional linking words
379           $line = preg_replace('/\s*(OR|AND)\s+/', ' ', $line);
380
381           // Split each host[!service] definition
382           $orConditions = preg_split('/[\s]+/', trim($line));
383
384           // By default, for have a critical state for the line,
385           // it should that at least one host!service is critical
386           $nbCrit4Crit = $nbCrit4Warn = count($orConditions);
387
388           // Last condition is either a thresholds parameter, or a classical condition
389           $lastCondition = array_pop($orConditions);
```

```
390
391         // Last condition is a thresholds parameter (!nbCritForWarn!nbCritForCrit)
392         if(preg_match('/^!(\d+%?)!?(\d+%?)?$/', $lastCondition, $limits)) {
393
394           // There is two thresholds ? (warning and critical)
395           (count($limits) > 2) ?
396             list(, $nbCrit4Warn, $nbCrit4Crit) = $limits : // Warn and crit are
                  extracted
397             $nbCrit4Crit = $nbCrit4Warn = $limits[1]; // Crit is extracted, and warn
                  is overloaded with its value
398
399           // Replace the thresholds percentages by absolute values
400           $nbServices = count($orConditions);
401           $nbCrit4Warn = replacePercent($nbCrit4Warn, $nbServices);
402           $nbCrit4Crit = replacePercent($nbCrit4Crit, $nbServices);
403
404         // Last condition is really a condition, it's re-pushed in the array
405         } else
406           array_push($orConditions, $lastCondition);
407
408         // The critical threshold is necessarily higher than the warn threshold
409         ($nbCrit4Crit >= $nbCrit4Warn)
410           || die("ERROR (conf l.$l) : Warn threshold is lower than crit threshold (W:
                $nbCrit4Warn, C:$nbCrit4Crit)");
411
412         // Nb critical services (or down hosts) in the line
413         $nbCrit = 0;
414         $req = false;
415
416         // For each condition : what's its state ?
417         foreach($orConditions AS $orCondition) {
418
419           // If the ! delimiter is found, there is a service associated with a host
420           // (else, there is an host alive test)
421           $isCheckAlive = !strpos($orCondition, '!');
422           $host = $service = false;
423
424           // It's an host alive test
425           if($isCheckAlive) {
426             $host = $orCondition;
427
428             // Recover the state of the host
429             // (0 or 1 => in this case 1 is therefore critical/down, and not warning)
430             $req = mysql_query(
431               sprintf("SELECT nhs.last_hard_state, UNIX_TIMESTAMP(nhs.
                      last_hard_state_change)
432                   FROM nagios.nagios_hoststatus nhs, nagios.nagios_objects no
433                   WHERE nhs.host_object_id = no.object_id
434                   AND no.name1 = '%s'
435                   LIMIT 1",
436                 mysql_real_escape_string($host)
437               ));
438
439             mysql_num_rows($req)
440               || die("ERROR (conf l.$l) : No state found for <em>$host</em>");
441
442           // It's a service associated with a host
443           } else {
444
445             // Separate host and service
446             list($host, $service) = explode('!', $orCondition);
447
448             // Recover the of the service for this host (0, 1 or 2)
449             $req = mysql_query(
450               sprintf("SELECT nss.last_hard_state, UNIX_TIMESTAMP(nss.
                      last_hard_state_change)
451                   FROM nagios.nagios_servicestatus nss, nagios.nagios_objects no,
                        nagios.nagios_services ns
452                   WHERE no.object_id = nss.service_object_id
453                   AND no.object_id = ns.service_object_id
454                   AND no.name1='%s'
455                   AND no.name2='%s'
456                   LIMIT 1",
457                 mysql_real_escape_string($host),
```

```
458                 mysql_real_escape_string($service)
459               ));
460
461             mysql_num_rows($req)
462               || die("ERROR (conf 1.$l) : No relation found between <em>$host</em>
                      and <em>$service</em>");
463           }
464
465           list($state, $lastChange) = mysql_fetch_row($req);
466
467           // For a host alive test, anything that is higher than ok (0) is critical
                (2)
468           if($isCheckAlive)
469             $state > OK && $state = CRITICAL;
470
471           // Else, anything that is not OK, warning or critical is ok (i.e. unknown
                state)
472           else
473             ($state < OK || $state == WARNING || $state > CRITICAL)
474               && $state = OK;
475
476           // Display state for each service, in the debug mode
477           $allowDebugMode && isset($_GET['debug'])
478             && printf("DEBUG : <strong>%s</strong>!%s => <em>%s</em><br />\n",
479               $host,
480               $service ? $service : 'alive',
481               strtoupper($states[$state]));
482
483           // If this service/host is critical, it's a new critical for this line
484           ($state == CRITICAL)
485             && $nbCrit++;
486
487           // Save the newest service state, for this global service
488           ($globalStates[$currentDefID]['newTime'] < $lastChange)
489             && $globalStates[$currentDefID]['newTime'] = $lastChange;
490         }
491
492         // If the nb of critical is higher than the critical threshold, and the
              global service state is not already
493         // critical, this one becomes critical
494         if($nbCrit >= $nbCrit4Crit && $globalStates[$currentDefID]['newState'] <
              CRITICAL)
495           $globalStates[$currentDefID]['newState'] = CRITICAL;
496
497         // If the nb of critical is higher than the warning threshold, and the global
                service state is not higher
498         // than warning (therefore ok...), this one becomes warning
499         elseif($nbCrit >= $nbCrit4Warn && $globalStates[$currentDefID]['newState'] <
              WARNING)
500           $globalStates[$currentDefID]['newState'] = WARNING;
501       }
502     }
503   }
504
505   // The End
506   fclose($fconf);
507   mysql_close();
508
509   // Update the cache file and $globalStates values
510   updateCacheAndValues("./caches/cache_$confID", $lastRefresh, $page_title,
          $globalStates);
511
512 // All values are recovered from the cache file
513 } elseif($allowDebugMode && isset($_GET['debug']))
514   echo "DEBUG : The cache file is used";
515
516
517 $favicon = OK;
518
519 // If a least one global service is critical, the favicon (traffic lights) displays
      critical.
520 // Else, if a least one global service is warning, the favicon is warning, else it's
      ok.
521 foreach($globalStates AS $infos) {
```

```
522     if($infos['state'] == CRITICAL) {
523       $favicon = CRITICAL;
524       break;
525
526     } elseif($infos['state'] == WARNING)
527       $favicon = WARNING;
528   }
529
530
531   /*******************************
532    ************ HTML *************
533    *******************************/
534
535   // The user want the full page
536   if(!$serviceID) : ?>
537
538     <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
          xhtml1/DTD/xhtml1-strict.dtd">
539     <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
540     <head>
541     <title><?php echo $page_title ?></title>
542     <meta name="author" content="Julien VAUBOURG / CRI-SRT" />
543     <meta name="generator" content="vim -p index.{php,css}"/>
544     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
545     <meta http-equiv="refresh" content="<?php echo $refresh ?>" />
546     <link href="img/<?php echo $states[$favicon] ?>.ico" rel="icon" type="image/x-icon"
          />
547     <link rel="stylesheet" type="text/css" href="index.css" title="Default" />
548     </head>
549
550     <body>
551     <h1><?php echo $page_title ?></h1>
552     <h2>Etat des services</h2>
553
554     <div id="body">
555     <?php
556       $explanations = Array(
557         "Le service est parfaitement fonctionnel.",
558
559         "Le service est perturbe.
560         Nous mettons tout en oeuvre pour resoudre ces desagrements.",
561
562         "Le service n'est pas fonctionnel.
563         Nous mettons tout en oeuvre pour le retablir au plus vite."
564       );
565
566       foreach($globalStates AS $id => $infos) {
567         $relativeTime = getRelativeTime($infos['time']);
568
569         echo <<<HTML
570           <div class="state state_{$states[$infos['state']]}">
571             <div class="label"><a href="http://contrats.cri.inpl-nancy.fr/$id.html"
                    title="Voir le contrat de service">{$infos['label']}</a></div>
572             <div class="description"><span>{$infos['description']}</span></div>
573
574             <div class="explanation">
575               {$explanations[$infos['state']]}
576             </div>
577
578             <div class="since">Depuis $relativeTime.</div>
579           </div>
580
581   HTML;
582       }
583     ?>
584     </div>
585
586     <div id="copy">
587       <a href="http://gpii-reverse-proxy.inpl-nancy.fr/glpi/front/helpdesk.php" title="
            GLPI">Ouvrir un ticket</a> -
588       <a href="mailto:svp-proximite@inpl-nancy.fr" title="Service proximite">Contacter
            le service informatique</a><br />
589
590       <em>
```

```
591         Dernier rafraichissement le <?php echo date('d/m/y a H:i:s') ?>
592         (toutes les <?php echo ($refresh % 60) ? "$refresh secondes" : ($refresh / 60).
                ' minutes' ?>)
593       </em>
594     </div>
595     </body>
596     </html>
597
598
599   <?php // The user want only one global service id state (traffic lights)
600   else :
601
602     // No global service with this ID found
603     isset($globalStates[$serviceID])
604       || die("ERROR : Global service ID not found");
605
606     $img = sprintf('img/%s%s.png',
607       $states[$globalStates[$serviceID]['state']],
608       $rotate ? '_h' : NULL);
609
610     header('Content-Length: '.filesize($img));
611     header('Content-Type: image/png');
612
613     // Display image with type/mime PNG
614     exit(file_get_contents($img));
615
616   endif ?>
```

U. Sonde NRPE intermédiaire pour la génération de graphiques

```
1   #!/usr/bin/perl -w
2
3   ## Author <julien@vaubourg.com>
4   ## Sonde intermediaire pour le reformatage des donnees de metrologie de differents
        checks
5   ## pour Centreon
6
7   use strict;
8   use warnings;
9   use POSIX qw(strftime);
10
11  # Recuperation des arguments
12  my $check = shift;
13  my $host = shift;
14  my ($pluginsPath) = $0 =~ /(.*)\/.*/;
15
16  # Lancement du vrai check par NRPE
17  my $values = `$pluginsPath/check_nrpe -H $host -c $check -a @ARGV 2>&1`;
18  my $return = $?;
19
20  # Checks de mesure de valeur relative a une autre
21  # => donc trois valeurs : total, free, used
22  if($check =~ /_(disk|swap|mem)_/) {
23    my ($status, $disk, $used, $w, $c, $total);
24
25    # Check de disque dur
26    if($check =~ /_disk_/) {
27      if($values =~ /^(.*)\|\s*([^=]+)=(\d+)MB;(\d+);(\d+);(\d+);(\d+)/) {
28        $status = $1;
29        $disk = $2;
30        $used = $3 * 1024*1024;
31        $w = ($4 - $4/100*5) * 1024*1024; # 5% root et passage en octets
32        $c = ($5 - $5/100*5) * 1024*1024;
33        $total = ($7 - $7/100*5) * 1024*1024;
34
35        $status =~ s/- free/(PARTITION MERE) - free/ if $ARGV[0] ne $disk;
```

```
36      } else {
37        print $values;
38      }
39
40    # Check de swap
41    } elsif($check =~ /_swap_/) {
42      if($values =~ /^(.*)\|\s*swap=(\d+)MB;(\d+);(\d+);(\d+);(\d+)/) {
43        $status = $1;
44        $total = $6 * 1024*1024;
45
46        $used = $total - ($2 * 1024*1024);
47        $w = $total - ($3 * 1024*1024);
48        $c = $total - ($4 * 1024*1024);
49      } else {
50        print $values;
51      }
52
53    # Check de RAM
54    } elsif($check =~ /_mem_/) {
55      if($values =~ /^(.*)\|\s*(\d+);(\d+);(\d+);(\d+)/) {
56        $status = $1;
57        $total = $2;
58        $used = $total - $3;
59        $w = $total - $total/100*$4;
60        $c = $total - $total/100*$5;
61      } else {
62        print $values;
63      }
64    }
65
66    print "$status| 0-Free=$total; 1-Used=$used; 2-Warning=$w; 3-Critical=$c\n";
67
68  # Check salles de Benoit
69  } elsif($check =~ /_salles/) {
70
71    # Format initial : <status | nbUSERWin;nbUSERLin;nbOK;nb;tauxWarning;tauxCritical\n
        >
72    if($values =~ /^(.*)\|\s*(\d+);(\d+);(\d+);(\d+);(\d+)/) {
73      my $nbUsers = $2 + $3;
74      print "$1| 2-Salle_Utilisateurs=$nbUsers; 4-Salle_UtilisateursWindows=$2; 3-
          Salle_UtilisateursLinux=$3; 1-Salle_MachinesAllumees=$4; 0-Salle_Machines=$5;
           seuilWarning=$6; seuilCritical=$7";
75    } else {
76      print $values;
77    }
78  }
79
80
81  exit 0 if $return == 0;
82  exit 1 if $return == 256;
83  exit 2;
```