

École Privée des Sciences informatiques

73, rue de Marseille

33000 Bordeaux

SOGETI FRANCE

26, avenue des Lilas

64000 Pau

Comment passer d'une supervision basique à une supervision Hautement Disponible et Haute Performance ?

-

Cluster de Supervision

Kévin CAMPAGNE

Sylvain LABASSE

Directeur de recherche EPSI

Promotion EPSI 2013

Soutenance: 27 Septembre 2013





Remerciements

Tout d'abord, je tiens à remercier vivement Mr Sigrid HARO pour m'avoir permis de réaliser cette alternance au sein de SOGETI France sur le site de Pau.

Je tiens à remercier également les personnes qui m'ont accompagné au cours des mes différentes missions au sein de SOGETI :

- Laurent SIMARD, Sylvain PALOUS et Jérémy VIGIE, Administrateurs de l'administration nationale de la prestation EDF.
- Christophe CARRAUD et Caroline DESERT qui m'ont permis d'intégrer ces projets de supervision au sein de différents clients.
- Vincent MAUGEIN, Jérôme RIEUCROS, David DALLAGO, et Jean-Noël FREYD, pour l'aide précieuse qu'ils m'ont apportée durant les projets de supervision.
- L'ensemble du personnel que j'ai pu côtoyer durant cette année d'alternance, pour leur disponibilité et leur sympathie à mon égard.

Enfin je tiens à remercier et à exprimer toute ma reconnaissance à l'équipe pédagogique de l'EPSI Bordeaux ainsi qu'à ses intervenants et tout particulièrement à la directrice de l'EPSI Madame Marie-José CORNILLE et au directeur de recherche Monsieur Sylvain LABASSE qui m'ont permis d'évoluer au sein de leur école.



Sommaire

Introduction	6
I. La Supervision	12
a. Les solutions de supervision.....	13
1. Solutions Open-Sources de Supervision.....	13
2. Solutions Propriétaires de Supervision	22
3. Comparaison des solutions de supervision	28
4. Comparaison de Nagios et de Shinken.....	29
b. Présentation de l'outil Shinken	31
II. Haute Disponibilité et Haute Performance	40
a. La haute disponibilité	40
b. Cluster Informatique	41
c. Fonctionnalités des Clusters	43
1. Cluster Actif/passif (Fail Over).....	43
2. Cluster Actif/Actif (Load balancing)	44
d. Présentation d'un Cluster de Supervision.....	45
e. Shinken et la Haute Disponibilité et la Haute Performance	47
f. Mise en place du Clustering	53
g. Mise en place d'une infrastructure de cluster sur site distant	62
III. Résolez la problématique	64
a. Supervision Classique	64
1. Contraintes.....	64
2. Descriptif des serveurs de Supervision	64
3. Solution de supervision.....	65
b. Cluster de Supervision.....	67
1. Contraintes.....	67
2. Descriptif des serveurs de Supervision	68
3. Solution de supervision	68
c. Cluster de Supervision redondé	71
1. Contraintes.....	72
2. Descriptif des serveurs de Supervision	73
3. Solution de supervision	73
d. Aspect contractuel	77



Qualité de service SLA	77
Conclusion.....	82
English Part	91
Introduction	91
Case Study	92
High Availability and High Performance (Cluster)	93
Monitoring Cluster	93
Monitoring Cluster with Shinken	94
Study of highly available architectures	95
Conclusion	95
Glossaire	96
Bibliographie	101
Illustrations	102
Annexes	Erreur ! Signet non défini.
Annexe 1 - La première architecture qui a été pensée pour le client CACF	Erreur ! Signet non défini.
Annexe 2 - La première architecture qui a été pensée pour le client TOTAL EP ...	Erreur ! Signet non défini.
Annexe 3 - Configuration de l'architecture de TOTAL EP	Erreur ! Signet non défini.



Introduction

La supervision est devenue incontournable depuis de nombreuses années dans les systèmes d'information. La supervision a pour vocation d'apporter une vue d'ensemble sur le fonctionnement des systèmes d'Information, de les auditer, d'en surveiller la disponibilité et la performance, et d'alerter les services en charge de son exploitation.

La supervision est un ensemble de fonctions qui consiste à mettre en place des indicateurs d'état d'un système et d'un réseau.

Elle va donc nous permettre de :

- Vérifier l'état d'un hôte ou d'un service
- Remonter des alertes
- Détecter des comportements anormaux

Les systèmes d'information étant par nature complexes, leur supervision est devenue indispensable.

Les systèmes d'information sont tous différents de par leur taille, leur nature, leur criticité. Ils ont cependant pour point commun d'être le théâtre d'incidents, à un moment ou à un autre. Les administrateurs doivent concevoir l'architecture du système d'information de telle manière qu'une panne est un impact minimal sur le reste du système.

Les Administrateurs informatiques doivent aussi gérer les éventuels problèmes – (ce qui est une part importante de leur travail). Même avec des architectures robustes, on estime généralement à 80% l'activité d'un administrateur sur la résolution de problèmes. Cette valeur-ci est assez conséquente et c'est pour cela qu'on cherche à la diminuer.



Les systèmes d'information ont pour but final de servir les utilisateurs finaux. Ceux-ci ne manquent pas de signaler les problèmes qu'ils rencontrent aux administrateurs (problèmes qui surviennent soit par manque de formation des utilisateurs, soit par réelle carence du système).

Dans ce dernier cas, il n'est pas bon, pour l'administrateur, de l'apprendre par un autre utilisateur, car celui-ci n'est généralement ni conciliant ni très précis. Cette imprécision peut faire perdre un temps précieux lors de la résolution du problème. L'administrateur, par manque d'informations pertinentes, cherche le problème au mauvais endroit. Une solution de supervision permet justement d'éviter ce genre de soucis. L'administrateur est prévenu rapidement d'une situation anormale, bien souvent, en moins de temps qu'il n'en faut à un utilisateur pour venir déplorer l'incident. Il obtient des informations pertinentes et peut immédiatement s'atteler à la résolution du problème.

Si ce dernier est mineur, sa résolution est rapide. L'utilisateur qui tente de nouveau d'accéder à la ressource y parvient et n'appelle pas le support. Il y a gain de temps, de part et d'autre, sans compter que l'utilisateur finit par se faire une meilleure opinion du service informatique qu'il appelle moins souvent. De plus, certaines ressources ne sont utilisées qu'occasionnellement. En cas de problème, en période de non-utilisation, sans outil de supervision, l'erreur n'est pas remontée ; ce n'est que lors de l'utilisation de l'application que les utilisateurs s'en aperçoivent et sont bloqués. Avec un outil de supervision, les administrateurs ont tout le temps nécessaire pour résoudre le problème en période creuse ou d'avertir ses utilisateurs du problème rencontré...

Il faut bien évidemment éviter l'effet domino. Bien souvent, une ressource peut engendrer plusieurs erreurs à la chaîne. C'est là que la vitesse de détection et de réaction face au problème reste primordiale pour éviter un effet domino dévastateur. Si l'administrateur arrive à résoudre le problème rapidement, les autres éléments peuvent être épargnés.



Il gagne donc doublement du temps en prévenant l'apparition d'autres problèmes. Un outil de supervision permet de conserver un historique des alertes. L'administrateur peut alors regarder les incidents qui se sont produits juste avant le problème constaté et ainsi facilement remonter à la source.

Certains problèmes sont précédés de signes avant-coureurs avant de devenir bloquants. Repérer ces signes est une plus-value non négligeable des outils de supervision, afin de régler le problème avant même qu'il ne se produise.

Surveiller le système d'information y compris pendant les périodes non ouvrées présente bien des avantages. Durant ces périodes, aucun utilisateur n'est présent pour relever les éventuels problèmes. Cela ne signifie pas que le système en est exempt. Ils peuvent être annonciateurs de problèmes futurs, voire en être la cause.

La proactivité passe bien entendu par la supervision. Être proactif face aux problèmes est une demande croissante de la part des directions des systèmes d'information. On ne peut être proactif, c'est-à-dire résoudre les problèmes avant même qu'ils ne se présentent, qu'à condition d'être bien informé.

Comme on vient de le voir, les outils de supervision sont cruciaux pour atteindre cet objectif. Les administrateurs ont donc tout intérêt à en utiliser un, ce qui facilitera leur travail et leur permettra de passer plus de temps sur des actions ayant une véritable valeur ajoutée, comme la conception d'architectures plus robustes.

Les ressources nécessaires pour mettre en place une telle solution se justifient parce qu'elle permet, d'une part, de répondre à la demande de pro activité et, d'autre part, de réaliser des gains de temps importants sur l'ensemble du système d'information.



Les administrateurs ne doivent pas oublier leur rôle premier qui est de fournir aux utilisateurs des applications exploitables dans les meilleures conditions.

Le ressenti utilisateur est très différent de celui des administrateurs quand on parle de performances. S'il est bien connu que les systèmes d'information ne sont jamais assez rapides pour ceux qui les utilisent, il en va tout autrement pour ceux qui les administrent.

Certains utilisateurs se plaignent de la lenteur des applications. Si, dans bien des situations, cela est dû à une mauvaise utilisation du système, dans d'autres cas, ces problèmes sont réels.

Sans outil de supervision, l'administrateur a beaucoup de mal à faire la différence entre les plaintes abusives des utilisateurs et les véritables soucis de performance.

Un outil de supervision est un mécanisme fiable. Il peut remonter des valeurs objectives. Il est possible de repérer très simplement s'il y a une dégradation de la situation par rapport à une période où tout fonctionnait ou bien si le ressenti des utilisateurs est en cause.

Sans un tel outil, les administrateurs n'auraient aucune vue sur les performances et maintiendraient que tout va bien du côté des performances. Mais les utilisateurs ne feront que remonter les différentes lenteurs observées. Ce genre de situation peut s'envenimer très rapidement. Les utilisateurs peuvent facilement généraliser à toute une application une lenteur ressentie sur seulement quelques écrans particuliers.

Les administrateurs ont un objectif clair : **le maintien en production du système d'information.**

Cependant, tous les éléments ne sont pas logés à la même enseigne en ce qui concerne la criticité.

Certaines parties sont vitales pour l'entreprise, d'autres beaucoup moins. Les utilisateurs auront tendance à considérer que tout est critique ; ce qui empêche d'établir des hiérarchies par priorité : si tout est prioritaire, plus rien ne l'est. Or les administrateurs doivent traiter en priorité les problèmes qui surviennent sur les systèmes critiques.



Sans outil de supervision, il est presque impossible pour un administrateur de garder en tête ces différents niveaux de criticité. Lorsqu'un problème survient, l'administrateur doit pouvoir accéder à cette information très rapidement et facilement. On ne peut pas s'en remettre aux remontées des utilisateurs, car ces derniers n'ont pas de vision globale du système d'information et ils diront que leur souci est le plus critique de tous.

L'outil de supervision peut ainsi aider à mettre des priorités sur les interventions des administrateurs et leur permettre de se concentrer sur l'essentiel. Avec un taux de disponibilité des applications ainsi pondéré par leur criticité, la disponibilité globale du système d'information est améliorée.

Un chemin important a été parcouru depuis la naissance des outils de supervisions, cependant il est loin d'être achevé. Ces derniers progressent jour après jour, diminuent au maximum les incidents et garantissent une qualité de service optimal.

La trame de mon document se constituera :

Dans un premier temps, nous allons faire un récapitulatif sur la supervision puis nous allons voir les différentes solutions qui s'offrent à nous entre les solutions open-sources et propriétaires. Après avoir étudié les avantages et inconvénients des différents logiciels, nous allons voir un comparatif des solutions. En suivant, nous allons expliquer le choix de l'outil de supervision et la présenter afin de continuer sur le sujet suivant.



Dans un second temps, nous allons aborder l'implémentation de la haute disponibilité et de la haute performance à travers une solution de supervision. Il ne faut pas oublier que la supervision mûrit rapidement et a pour but de gérer des parcs de plus en plus importants avec le moins de configuration possible. Les architectures doivent être malléables, mais doivent aussi progresser rapidement. Nous aborderons donc une notion de « Cluster » afin d'avoir une solution de supervision hautement disponible et hautement performant. Avec la mise en place d'un cluster de Supervision, nous sommes à un carrefour de la mutation de la supervision. Les environnements informatiques grandissent constamment, et sont demandeurs de plus en plus de performance, d'hautes disponibilités et veulent également une simplicité d'administration.

Nous présenterons en suivant l'étude de la mise en place d'un cluster de supervision puis nous enchaînerons sur la mise en place d'architecture hautement disponible et performante.

Dans un troisième temps, nous apercevrons les différentes architectures de cluster de supervision que nous avons étudiées et mises en place pour plusieurs clients. Chaque client a des besoins et des contraintes différents et nous avons donc mis en place des solutions de supervision différentes qui correspondent aux besoins. Les architectures correspondent donc à des besoins classiques et à des besoins beaucoup plus complexes.

En suivant, nous verrons que ces architectures hautement disponible et performante qui optimisent considérablement les qualités de service et donc indirectement la productivité des entreprises.

Nous concluons par un petit résumé de ce document puis nous commenterons la prochaine « grande évolution » de la supervision.



I. La Supervision

La supervision informatique permet d'avoir une vue d'ensemble sur le fonctionnement des systèmes d'information. Elle remonte des statuts, des informations techniques et fonctionnelles d'un système d'information.

La supervision englobe plusieurs activités comme :

- Surveiller
- Visualiser
- Analyser
- Piloter
- Agir
- Alerter

Dès qu'un incident est identifié, la supervision reconnaît l'incident et transmet des informations pertinentes à l'administrateur et peut immédiatement s'engager dans la résolution du problème. La supervision peut être également un outil proactif, c'est-à-dire résoudre les problèmes avant même qu'ils ne se présentent.

L'informatique est devenue un des piliers principaux de l'entreprise ces dernières années. Le système d'information doit maintenir les différentes applications métier de son entreprise et les administrateurs doivent tout mettre en œuvre pour garantir en permanence une qualité de service envers ses applications métiers et ses utilisateurs.

Les outils de supervision sont indispensables à la bonne administration d'un système d'information. Sans eux, l'administrateur est privé de moyens fiables et rapides de vérifier que les éléments de l'infrastructure et les applications sont opérationnels. Ces incidents informatiques peuvent impacter des pertes de productivité dans une entreprise, c'est la raison pour laquelle, les outils de supervision sont présents pour diminuer ce risque.



Tout simplement, la supervision est devenue incontournable depuis quelques années dans tous les systèmes d'information pour garantir une qualité de services optimums. Nous allons vous présenter l'étude que nous avons effectuée sur les solutions de supervision.

a. Les solutions de supervision

1. Solutions Open-Sources de Supervision

Nagios

Nagios est une application permettant de surveiller l'état des hôtes d'un réseau et des différents services présents sur ces derniers. Il est codé en C sous licence GPL et est disponible sur les systèmes Linux et Unix.

L'architecture logicielle est modulaire et divisée en trois parties :

- Moteur : Ordonnance des tâches de supervision (test du fonctionnement du serveur web, etc.).
- Interface web : Affiche l'état du réseau et les anomalies.
- Les plug-ins : Chaque plug-in est un logiciel indépendant chargé de faire une vérification puis de transmettre le résultat à Nagios.

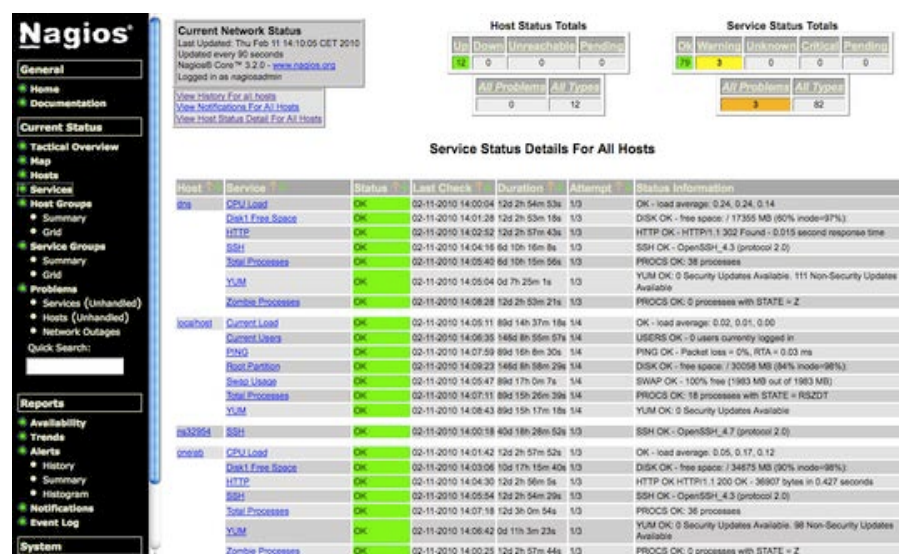


Image 1 – Interface Nagios



Il existe notamment des plug-ins Nagios nommée NRPE (et NSClient) et NCSA. NRPE est un esclave qui attend les ordres du moteur Nagios et NCSA envoie de lui-même des trappes de données (trapping).

Nagios est certainement le logiciel libre le plus connu dans le milieu de la supervision. Malgré sa grande popularité, beaucoup d'utilisateurs n'en sont pas pleinement satisfaits. Certains développeurs ont tenté de proposer des modifications du code source à Ethan Galstad.

Ce dernier étant peu réactif et acceptant rarement les propositions d'améliorations, des développeurs ont commencé à créer des forks de Nagios. C'est de là que sont nés la plupart des nombreux fork de Nagios.

Avantages

- Reconnu auprès des entreprises, grande communauté
- Nombre de plug-ins qui permettent d'étendre les possibilités (agents, reporting amélioré, etc...)
- Une solution complète permettant le reporting, la gestion de panne et d'alarmes, gestion utilisateur, ainsi que la cartographie des réseaux
- Beaucoup de documentations sur le web
- Performances du moteur

Inconvénients

- Interface non ergonomique et peu intuitive
- Configuration fastidieuse via beaucoup de fichiers
- Pour avoir toutes les fonctionnalités il faut installer des plug-ins, de base c'est assez limité.
- **N'arrive pas à gérer totalement la Haute Disponibilité**

Site officiel : <http://www.nagios.org>

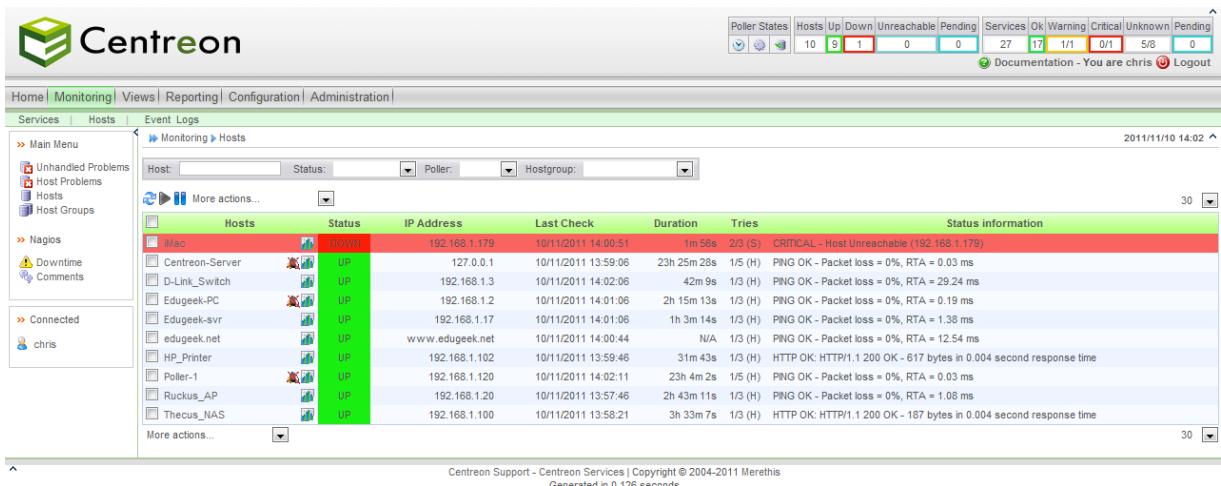


Centreon

Centreon est un fork de Nagios, il se présente comme une évolution de celui-ci pour son interface, mais aussi ses fonctionnalités. Créé en 2003 par des Français souhaitant améliorer Nagios et son interface très austère, Centreon (anciennement Oréon) a été repris par une nouvelle entreprise nommée Merethis. Centreon reprend donc les avantages du moteur de Nagios et permet ainsi d'être entièrement compatible avec des solutions existantes. Il est sous licence GPLv2.

Son interface reprend un découpage classique :

- Home : Page d'accueil avec Le "Tactical Overview" de Nagios permettant un coup d'œil rapide aux problèmes survenus et accès aux statistiques des performances du moteur et de ses composants.
- Monitoring : Possède plusieurs vues, mais reprend la grande idée de l'arbre des groupes d'équipements. Reprends également la vue Nagios.
- Views : Permet d'accéder à tous les graphiques avec un menu arborescent et donne accès à une cartographie du réseau en applet Java.
- Reporting : Un tableau de bord ressemblant à celui de Zabbix en ajoutant une frise chronologique de la disponibilité de l'équipement.
- Configuration : Pour tout configurer de A à Z.
- Administration : Configuration des accès utilisateurs.



The screenshot shows the Centreon web interface. At the top, there's a navigation bar with links like Home, Monitoring, Views, Reporting, Configuration, and Administration. Below this, there's a sidebar with a menu for Services, Hosts, and Event Logs. The main content area is titled 'Monitoring Hosts' and displays a table of hosts. The table has columns for Hosts, Status, IP Address, Last Check, Duration, Tries, and Status information. The hosts listed include 'Mac', 'Centreon-Server', 'D-Link_Switch', 'Edugeek-PC', 'Edugeek-svr', 'edugeek.net', 'HP_Printer', 'Poller-1', 'Ruckus_AP', and 'Thecus_NAS'. The status of each host is indicated by a color-coded icon (green for OK, red for DOWN, yellow for WARNING, etc.).

Hosts	Status	IP Address	Last Check	Duration	Tries	Status information
Mac	DOWN	192.168.1.179	10/11/2011 14:00:51	1m 56s	2/3 (S)	CRITICAL - Host Unreachable (192.168.1.179)
Centreon-Server	OK	127.0.0.1	10/11/2011 13:59:06	23h 25m 28s	1/5 (H)	PING OK - Packet loss = 0%, RTA = 0.03 ms
D-Link_Switch	OK	192.168.1.3	10/11/2011 14:02:06	42m 9s	1/3 (H)	PING OK - Packet loss = 0%, RTA = 29.24 ms
Edugeek-PC	OK	192.168.1.2	10/11/2011 14:01:06	2h 15m 13s	1/3 (H)	PING OK - Packet loss = 0%, RTA = 0.19 ms
Edugeek-svr	OK	192.168.1.17	10/11/2011 14:01:06	1h 3m 14s	1/3 (H)	PING OK - Packet loss = 0%, RTA = 1.38 ms
edugeek.net	OK	www.edugeek.net	10/11/2011 14:00:44	N/A	1/3 (H)	PING OK - Packet loss = 0%, RTA = 12.54 ms
HP_Printer	OK	192.168.1.102	10/11/2011 13:59:46	31m 43s	1/3 (H)	HTTP OK: HTTP/1.1 200 OK - 617 bytes in 0.004 second response time
Poller-1	OK	192.168.1.120	10/11/2011 14:02:11	23h 4m 2s	1/5 (H)	PING OK - Packet loss = 0%, RTA = 0.03 ms
Ruckus_AP	OK	192.168.1.20	10/11/2011 13:57:46	2h 43m 11s	1/3 (H)	PING OK - Packet loss = 0%, RTA = 1.08 ms
Thecus_NAS	OK	192.168.1.100	10/11/2011 13:58:21	3h 33m 7s	1/3 (H)	HTTP OK: HTTP/1.1 200 OK - 187 bytes in 0.004 second response time

Image 2 – Interface Centreon



Il permet de générer des rapports sur les incidents. Les données sont stockées dans une base de données MySQL. L'interface web est multiutilisateur et gère des listes de contrôle d'accès pouvant être paramétrées de manière très précise.

Bien qu'il fonctionne par défaut avec Nagios, il est possible d'utiliser Centreon avec Shinken.

Avantages

- La robustesse et la renommée de Nagios
- Une interface beaucoup plus agréable, permettant de tout configurer, de garder une vision globale sur tout le réseau en permanence
- Les utilisateurs de Nagios ne seront pas perdus pour autant, l'interface reprenant avantageusement certaines vues Nagios
- Une solution complète permettant le reporting, la gestion de pannes et d'alarmes, gestion utilisateur, ainsi que la cartographie des réseaux
- Une entreprise qui fait le développement de cet outil et qui propose un support
- Peut-être décoller du serveur Nagios et tourner tout seul sur un autre serveur

Inconvénients

- L'interface peut paraître complexe, car il existe beaucoup d'options, de vues
- **Un développement qui n'est pas encore en phase avec celui de Nagios : parfois des problèmes de compatibilité**
- Outil plus lourd que Nagios

Site officiel : <http://www.centreon.com/>



Shinken

Shinken est une application récente, gratuite et dont le développement est très actif. Elle a vu le jour en 2009. Son code est sous licence AGPL. Shinken peut fonctionner sur toutes les plateformes supportant le logiciel Python.

C'est une réimplémentation de Nagios en langage Python qui ajoute de nouvelles fonctionnalités orientées performance. Parmi ces fonctionnalités, on peut noter la haute disponibilité et la possibilité d'être distribuée. Pour réaliser ces améliorations, le système a été découpé en plusieurs parties (appelées démons dans le reste de ce document).

Ces différents démons peuvent être répartis sur plusieurs machines. Le passage de Nagios à Shinken se fait de manière aisée, car Shinken utilise le même format de fichiers de configuration que Nagios. Il est uniquement nécessaire d'adapter les fichiers de configuration spécifiques à Shinken si l'on ne souhaite pas garder les paramètres par défaut.

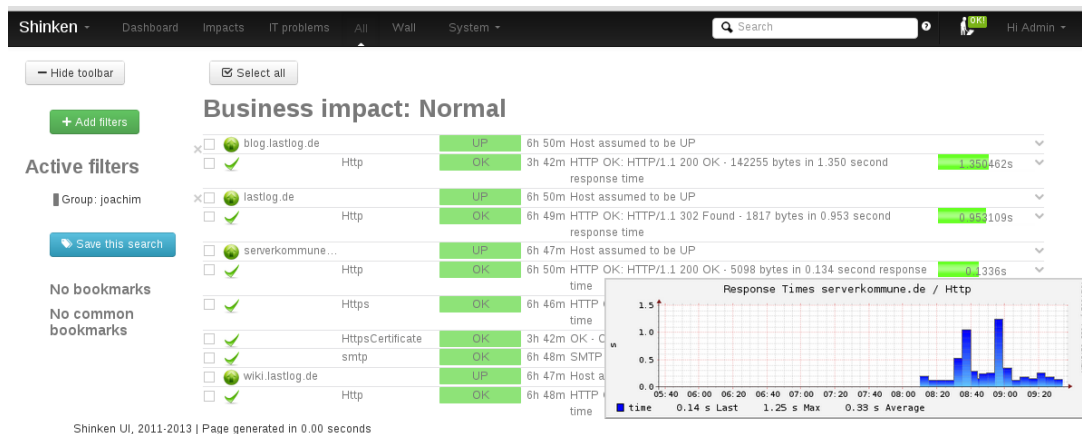


Image 3 – Interface Shinken

Shinken intègre le facteur business (business impact sur la figure ci-dessus). Le déclenchement des alertes et les notifications sont paramétrables de manière précise. Il dispose d'un système de détection permettant d'éviter des tempêtes d'alertes.



Avantages

- Tous les avantages de Nagios
- Une supervision distribuée hautement disponible
- Un système avancé de corrélation
- Une vue et des règles de Business Impact et notion d'importance
- Performance 150000 Check / 5 minutes
- Évolution constante

Inconvénients

- Shinken est un jeune projet
- Pas assez de tutoriaux

Site officiel : <http://www.shinken-monitoring.org/>

Cacti

Cacti est un logiciel open source (sous licence GPL) de supervision de réseau. Il est basé sur le système de base de données RRDTool. Son interface est principalement peuplée de graphiques indiquant l'état du réseau. Il gère nativement le SNMP et peut facilement être intégré à d'autres sources de données telles que Nagios ou Shinken



Image 4 – Interface Cacti



Cacti dispose d'un poller qui se charge de tester le bon fonctionnement de services à des intervalles prédéterminés. Ce poller existe en deux versions, l'une codée en PHP pour les petites architectures et une autre codée en C pour les architectures plus grandes.

Avantages

- Interface : Beaucoup plus claire que celle de NetMRG elle permet également beaucoup plus de choses (Plus de modes d'affichages et plus de possibilités de configuration)
- Configuration : Avec l'utilisation des templates pour les machines, les graphiques, et la récupération des données tout se configure aisément et entièrement via l'interface web. Import/ Export très simple des templates au format XML. On peut aussi très facilement utiliser des options poussées de RRDTOOL
- Performance : avec le choix du moteur de récolte des données, on peut opter pour la performance ou la simplicité
- Gestion des utilisateurs
- Communauté sur le web, présence d'une dizaine de plug-ins permettant d'étendre les fonctionnalités

Inconvénients

- Pas de gestion d'alarmes
- Pas de gestion de panne et absence d'une cartographie de réseau
- Un développement lent
- **Un outil avant tout de métrologie et non de supervision**

Site officiel : <http://www.cacti.net/>



Zabbix

Zabbix est une solution de supervision gratuite facile à mettre en place. Il est sous licence GPLv2. Une licence commerciale est également disponible pour les entreprises désirant intégrer Zabbix à leur produit propriétaire.

Actuellement cette solution ne contient pas de mécanisme performant pour éviter des tempêtes d'alertes. Zabbix contient un système d'affichage en temps réel de l'état des machines surveillées. Il contient également de nombreux graphiques permettant de visualiser l'évolution au cours du temps de l'infrastructure surveillée. Il dispose d'un système de gestion de SLA. Il fournit une gestion fine des droits d'accès des utilisateurs.

L'architecture de Zabbix peut être décomposée en trois parties:

- Le serveur de stockage de données (MySQL, PostgreSQL ou Oracle)
- L'interface web de gestion
- Le serveur de traitement

Zabbix dispose d'un système de répartition de charge et de haute disponibilité. Les trois parties de Zabbix peuvent être sur des machines différentes. Le serveur de traitement peut également avoir des "proxy Zabbix". Un proxy Zabbix est une machine supplémentaire qui peut être utilisée pour traiter une partie des données lorsque la charge de la machine principale devient trop élevée.

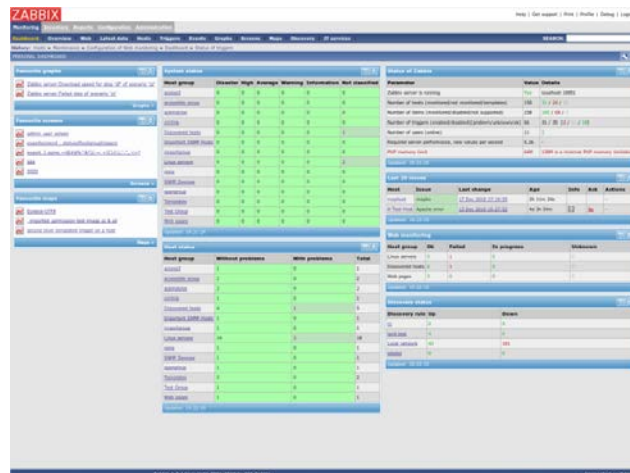


Image 5 – Interface Zabbix

Avantages

- Une solution complète
- Une entreprise qui pousse le développement, et une communauté croissante
- Une interface vaste
- Une gestion des templates poussée, avec import/export xml, modifications via l'interface
- Performant
- Une supervision distribuée hautement disponible

Inconvénients

- Interface est un peu vaste
- **L'agent zabbix communique par défaut en clair les informations**
- Commence à être connu, mais pas encore auprès des entreprises

Site officiel : <http://www.zabbix.com>



2. Solutions Propriétaires de Supervision

NetCrunch

NetCrunch est un logiciel propriétaire édité par la société AdRem Software. Son prix est de 1'500 \$ pour 50 périphériques surveillés.

Il fonctionne selon l'architecture client-serveur. Il y a le serveur NetCrunch et le client (console d'administration) qui est un logiciel avec une interface graphique se connectant au serveur pour afficher les données. Un client web est également disponible.

Il fonctionne sans agent et peut utiliser des sources d'informations telles que SNMP, les journaux d'événements de Windows et les serveurs Linux. Il contient un système de module permettant d'étendre le système. NetCrunch 6 contient plus de 2'000 modules préinstallés.

NetCrunch intègre des fonctionnalités de gestion de la disponibilité, de gestion de la performance des serveurs et de génération de rapports. Il dispose également d'un système de découverte automatique de la topologie du réseau.

Il peut aussi être utilisé pour faire un inventaire du matériel et des logiciels.

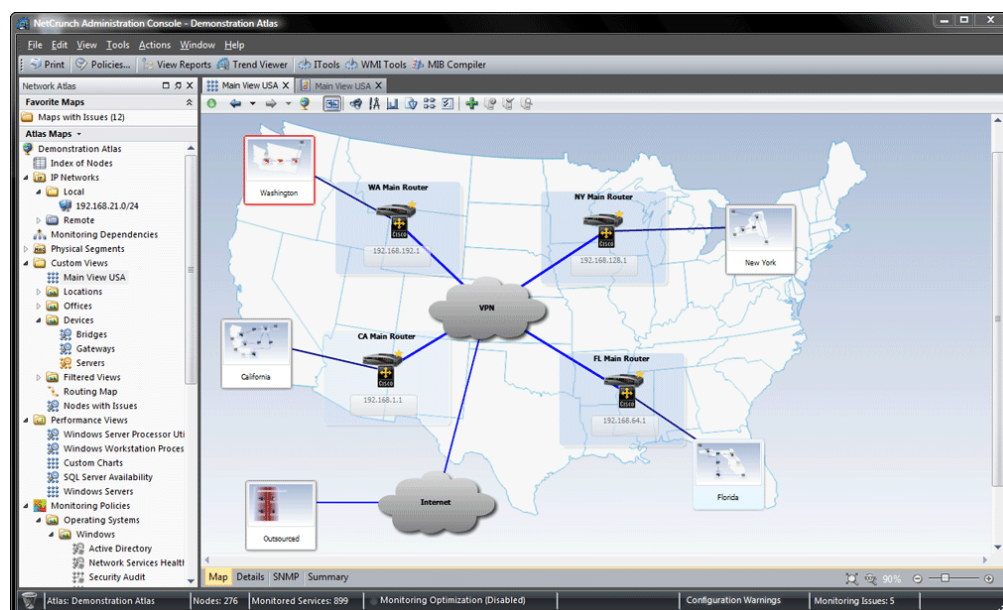


Image 6 – Interface Netcrunch



Avantages

- Fonctionnement sans agent
- Vue globale du système d'informations
- Scan du réseau

Inconvénients

- Solution lourde
- Données volumineuses
- Prix exorbitant

Site officiel : <http://www.adremsoft.fr/netcrunch/>

Traverse

Traverse est une solution propriétaire éditée par Zyrion. Le prix de l'édition standard varie avec la taille du réseau surveillé et commence à 50'000 \$. Une édition gratuite de démonstration est disponible. Traverse intègre des fonctionnalités de gestion de SLA, de performance du réseau et de gestion de parcs de machines virtuelles et de Clouds. Il gère les environnements virtualisés VMware, XenServer, KVM et Hyper-V. Il propose des vues représentant la topologie physique du réseau et des vues business permettant de voir en temps réel la performance du service informatique.

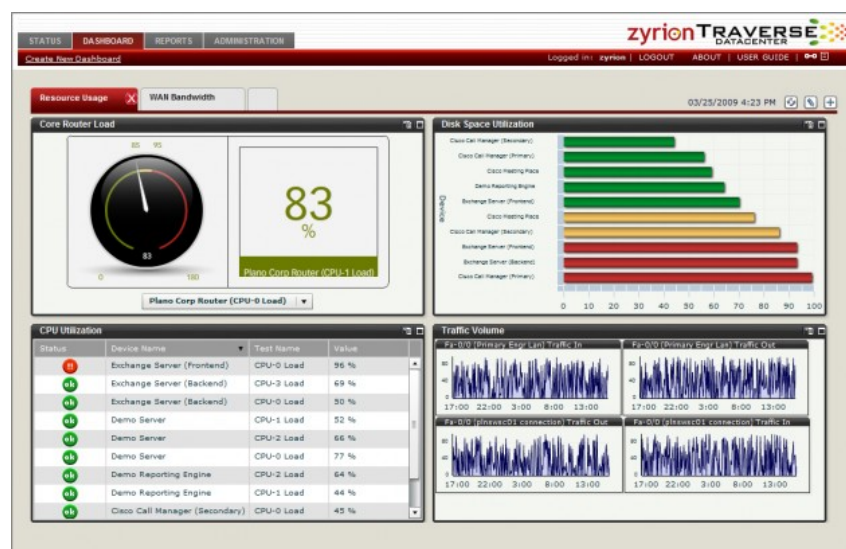


Image 7 – Interface Traverse



Avantages

- Interface simple d'administration
- Règle des Business Impact
- Performant

Inconvénients

- **Prix exorbitant**
-

Site officiel : <http://www.zyrion.com/products/>

Tivoli Monitoring

Tivoli Monitoring est une solution propriétaire développée par la société IBM. Il fait partie de la suite de logiciel Tivoli. Depuis la version 6.1 il ne repose plus sur le framework Tivoli, il utilise le moteur d'une solution concurrente (rachetée par IBM) nommée Omegamon (éditée à l'origine par Candle Corp).

Il est composé des éléments suivants :

- Tivoli Data Warehouse (TDW) : Sauvegarde les données de supervision dans une base de données SQL.
- Tivoli Enterprise Monitoring Server (TEMS) : Il lance les tests sans agent et communique avec les agents puis il envoie les données à sauvegarder au TDW.
- Tivoli Enterprise Portal Server (TEPS) : Permet d'accéder à distance au TDW grâce à une interface web.
- Tivoli Common Reporting (TCR) :

Tivoli Monitoring fournit des agents permettant de récupérer des informations sur l'état des machines. Ces agents sont prévus pour continuer à fonctionner lorsque le réseau est interrompu ou dans le cas où le serveur de supervision ne fonctionne plus. Si un tel cas se produit, les agents distribués sur les machines vont sauvegarder les informations et les



transmettre au serveur de supervision dès que ce dernier est à nouveau joignable. Tivoli Monitoring est aussi utilisable sans agent (agentless).

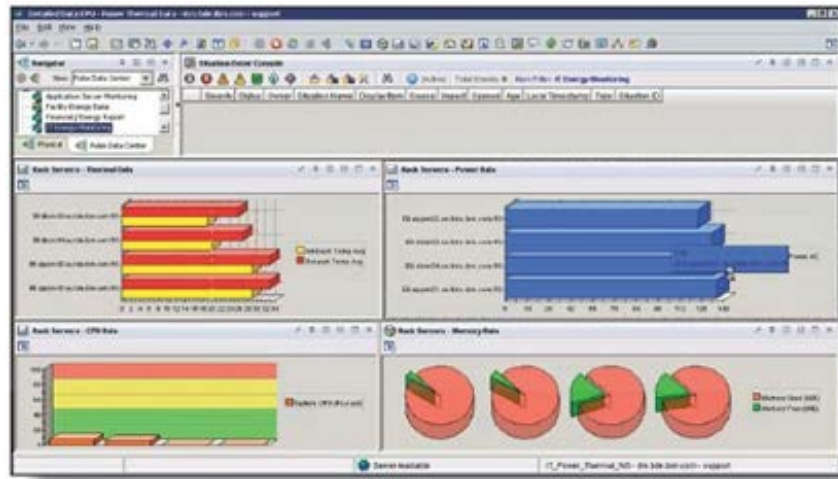


Image 8 – Interface Tivoli Monitoring

Avantages

- Solution la plus répandue sur le marché
- Facile d'adaptabilité aux besoins

Inconvénients

- Il faut acquérir le pack entier de la solution afin de pouvoir superviser au mieux le réseau.

Site officiel: <http://www-142.ibm.com/software/products/fr/fr/tivomoni>

OpenView

OpenView est un logiciel propriétaire développé par la société HP. En 2007, il a été renommé HP Network Management Center.

La suite HP OpenView contient une multitude de logiciels. Parmi ces logiciels, on trouve notamment:

- HP OpenView Network Node Manager (OV NNM) : chargé de surveiller le réseau et les services réseau à l'aide du protocole SNMP.



- HP Operations Manager (OM) : Surveille les systèmes qui utilisent des agents. Il existe un agent pour les systèmes Windows (OMW) et un agent pour les systèmes Unix (OMU).
- HP OpenView Performance (OVP*) : Logiciels permettant de surveiller les performances de différents équipements.
- HP OpenView Storage : Permettant de gérer le stockage des informations de supervision.
- HP OpenView Smart Plug-ins (SPI) : Plugins permettant de superviser une base de données, un système VMware, un système SAP, etc.
- HP Software Business Availability Center (BAC) : Fournis des services de gestion de SLA, de gestion d'utilisateurs, etc.

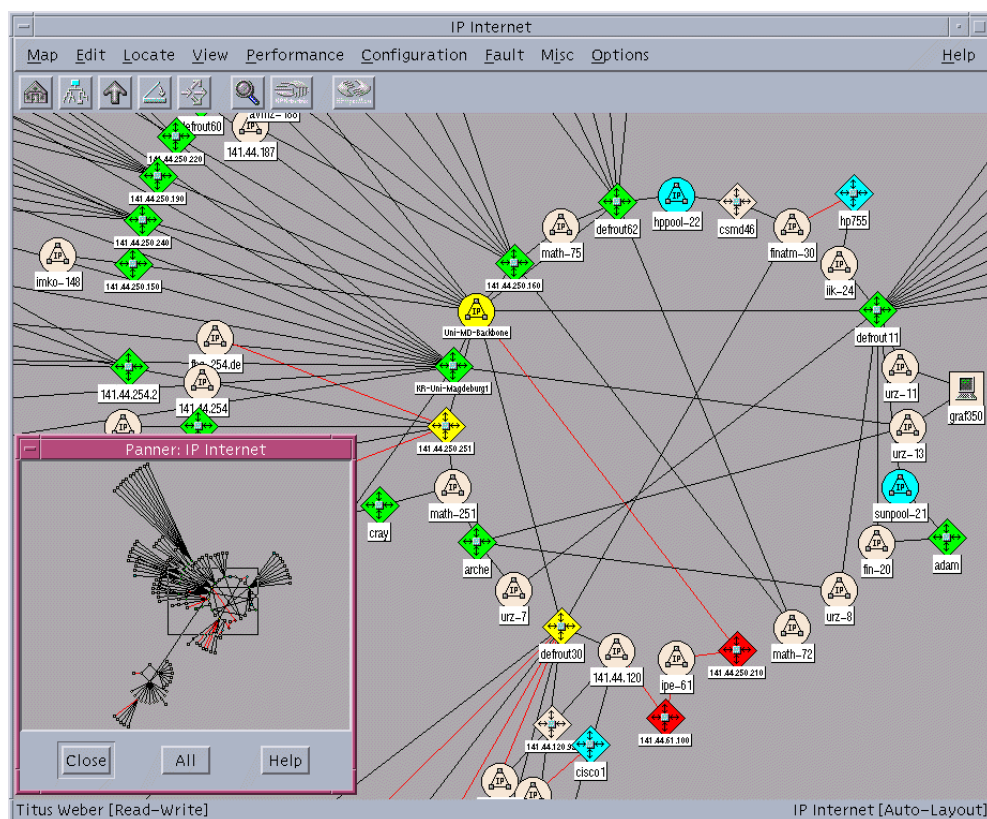


Image 9 – Interface HP OpenView



Avantages

- Vue globale du système d'informations
- Visions des différents incidents
- Contrôle homogène des différents matériels

Inconvénients

- Interface complexe
- Données volumineuses
- **Capacités d'optimisation moindre**

Site officiel: <http://www8.hp.com/us/en/software/enterprise-software.html>

Après une étude des différentes solutions possibles, nous allons faire une comparaison des solutions open-source et propriétaire.



3. Comparaison des solutions de supervision

Nom	Licence	Langage	OS Cible	Commentaires	Coût
Nagios	GPL	C	Linux, Unix		Gratuit
Centreon	GPLv2	PHP/Perl/C	Linux, Unix	Utilise Nagios,RRDTool	Gratuit
Shinken	AGPL	Python	Linux, Windows, ...	Compatible Nagios	Gratuit
Cacti	GPL	PHP/Shell/C	Linux, Unix	Basé sur RRDTool	Gratuit
Zabbix	GPLv2	PHP/C	Linux, Mac, Unix		Gratuit
NetCrunch	Propriétaire	N/A	Windows		> 1500\$
Traverse	Propriétaire	N/A	Linux, Windows		>50000\$
Tivoli	Propriétaire	N/A	Linux, Windows		> 750\$
Open View	Propriétaire	N/A	Windows, HP-UX		Payant

Pourquoi une solution Open Source plutôt qu'une solution Propriétaire ?

Les solutions propriétaires ont porté la supervision pendant des longues années. Elles offrent des solutions satisfaisantes. Mais ces solutions propriétaires sont assez coûteuses, ne sont pas compatibles et elles sont également assez fermées.

Ces derniers aspects entraînent la montée en puissance des solutions Open-Source. De nos jours les solutions offrent une alternative sérieuse avec quelques atouts dans leurs sacs :

- Faibles coûts
- Respect des standards
- Bonne qualité technique
- Compatibilité
- Évolution constante des solutions

Maintenant, nous allons comparer les deux solutions Open-Source les plus complètes et utilisés.



4. Comparaison de Nagios et de Shinken

Shinken étant une réimplémentation de Nagios, il est intéressant de comparer leur état de développement. Est-ce que Shinken a implémenté toutes les fonctionnalités de Nagios ? Quelles sont les fonctionnalités phares de Shinken ?

Parmi les fonctionnalités phares de Shinken on peut relever:

- Stabilité et haute disponibilité.
- Utilisable sur toutes les plateformes supportées par Python.
- Notion d'importance pour le business.
- Possibilité de mettre un Poller dans la DMZ et un Poller dans le LAN pour éviter une configuration complexe et risquée du pare-feu.

Nagios possède néanmoins certaines fonctionnalités qui ne sont pas disponibles dans Shinken:

- Pearl embarqué.
- Modules binaires.
- Planification automatique des tests.

Il est intéressant de comparer l'évolution du nombre de lignes de code au cours du temps.

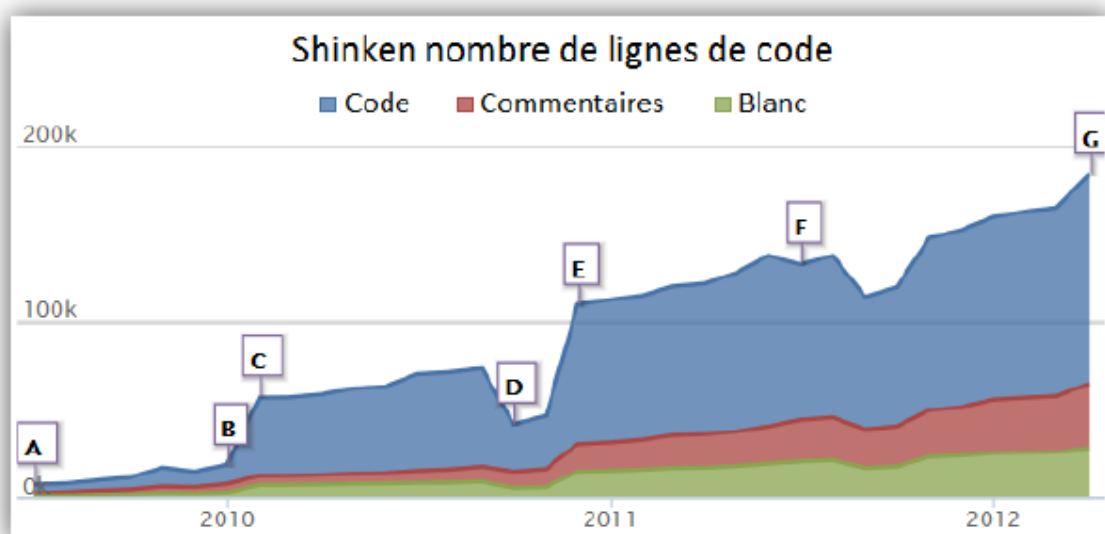


Image 10 - Shinken et ses Nombres des lignes de code

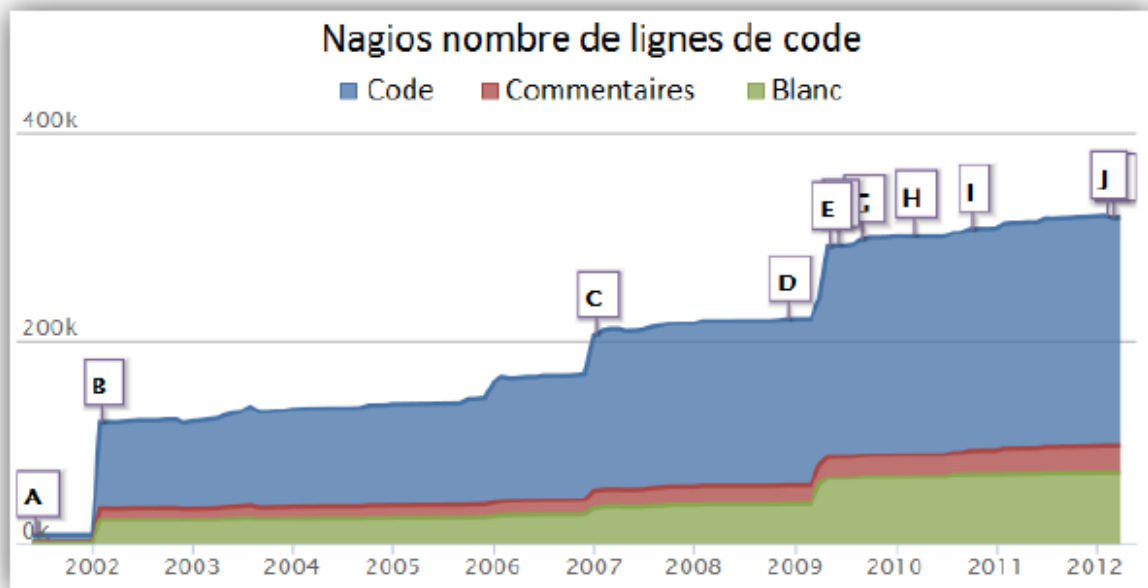


Image 11 - Nagios et ses Nombres des lignes de code

On remarque sur les schémas ci-dessus (fournis par www.ohloh.net) que Shinken est nettement plus jeune que Nagios. On remarque aussi les méthodes de développement des deux logiciels. En effet, Nagios se développe peu, mais progressivement. Shinken, quant à lui, se développe rapidement (200'000 lignes de codes en 3 ans).

La baisse brutale du nombre de lignes de code en septembre 2010 correspond à la suppression des fichiers XML contenant l'aide au profit du Wiki (<http://www.shinken-monitoring.org/wiki>).

Voici un lien vers un site internet pour montrer le comparatif entre Nagios et Shinken : <http://www.shinken-monitoring.org/what-is-in-shinken-not-in-nagios-and-vice-versa/>

Nous allons maintenant présenter Shinken dans le prochain chapitre.



b. Présentation de l'outil Shinken



Comme nous l'avons vu précédemment, Shinken est un outil de supervision créé par Jean Gabés.

Devant le manque d'ouverture des développeurs de Nagios et le risque de voir ce dernier devenir propriétaire, le projet Shinken devient complètement autonome.

Shinken qui est une refonte complète du cœur de Nagios en langage Python, ce qu'il lui apporte une nouvelle architecture plus souple et plus facile à maintenir que le démon monolithique de Nagios. Shinken utilise cinq processus différents pour plus de flexibilité et beaucoup plus rapides que Nagios en termes de ressources consommées.

Ce logiciel multiplateforme permet donc de faire une supervision classique :

- Vérifications des états SOFT/HARD
- Gestion des dépendances réseau et logiques (applicatives)
- Gestion des actions correctrices
- Supervision active et passive
- Vérification de la fraîcheur des informations dans le cadre de la supervision passive

Mais également d'obtenir une supervision bien plus complète :

- Supervision hautement disponible (*Possibilité d'Haute disponibilité*)
 - *Minimiser les temps d'indisponibilités en ne reliant aucune configuration à un serveur physique*
- Supervision Hautement performante (*5 fois plus de performances que le Nagios classique*)
 - *Performance = 150000 Check/5 minutes sur un serveur adéquate suivant l'infrastructure*



- *Gérer la répartition de charge*
- *Possibilité de pondérer le découpage de la configuration*
- *Multiplateforme (compatible sur toutes les plateformes où Python est installable)*
 - *Windows*
 - *Linux*
 - *Solaris*
 - *FreeBSD*
 - *Android*

Shinken a pour objectif de simplifier la vie des Administrateurs de grands parcs et de permettre une évolution maximum à un service informatique.

Voici un schéma de description sur le fonctionnement détaillé sur les rôles de Shinken :

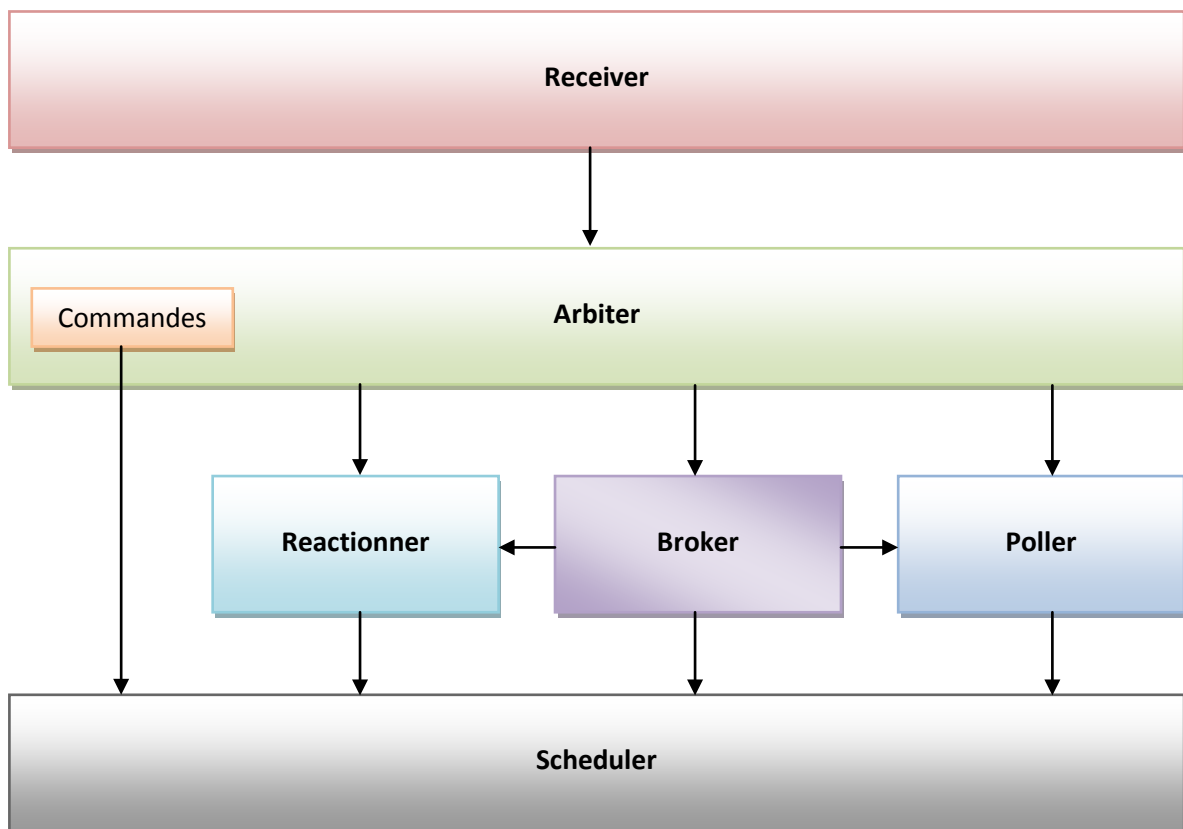


Schéma 1 – Fonctionnement des rôles de Shinken



Comme on peut le voir sur ce schéma, nous avons 6 rôles différents :

- Le Receiver
- L'Arbiter
- Le Reactionner
- Le Broker
- Le Poller
- Le Scheduler

Chacun de ces rôles est d'une importance capitale pour le bon fonctionnement de Shinken.

Les modules sont les suivantes :

L'arbiter :

Il est chargé de lire la configuration, de la découper en autant de parties qu'il y a d'ordonnanceurs dans l'architecture, et leur envoyer ainsi qu'aux autres éléments de l'architecture. Il est également garant de la haute disponibilité : si un élément tombe, il envoie la configuration qu'il avait en charge à un élément spare défini par l'administrateur. Enfin, son dernier rôle est de lire les ordres des administrateurs fournis dans le pipe nommé nagios.cmd et de les transmettre à tous les ordonnanceurs ou un seul suivant la demande.

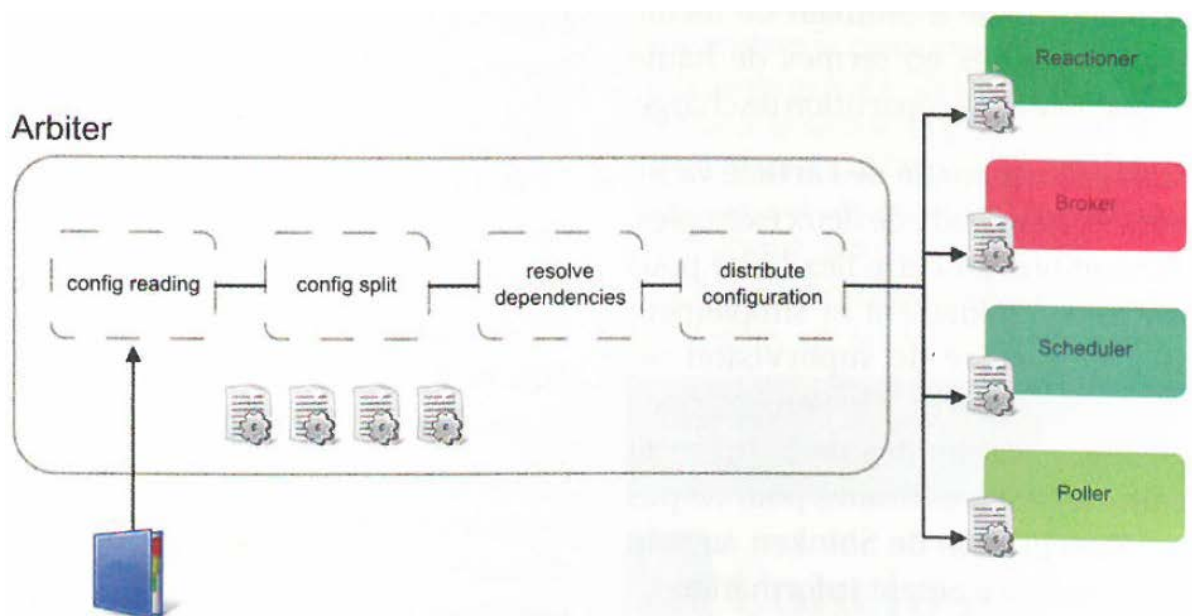


Schéma 2 – Fonctionnement de l'Arbiter



Le Scheduler :

Ils sont chargés d'ordonner les vérifications et de lever des actions en cas de soucis avec ces dernières. Ils ne lancent pas directement les vérifications ni les notifications. Ils ne font que les proposer dans des files d'attentes où vont venir piocher d'autres éléments de l'architecture. L'administrateur peut en avoir autant qu'il veut, l'Arbiter va découper la configuration en fonction du nombre de *schedulers* définis.

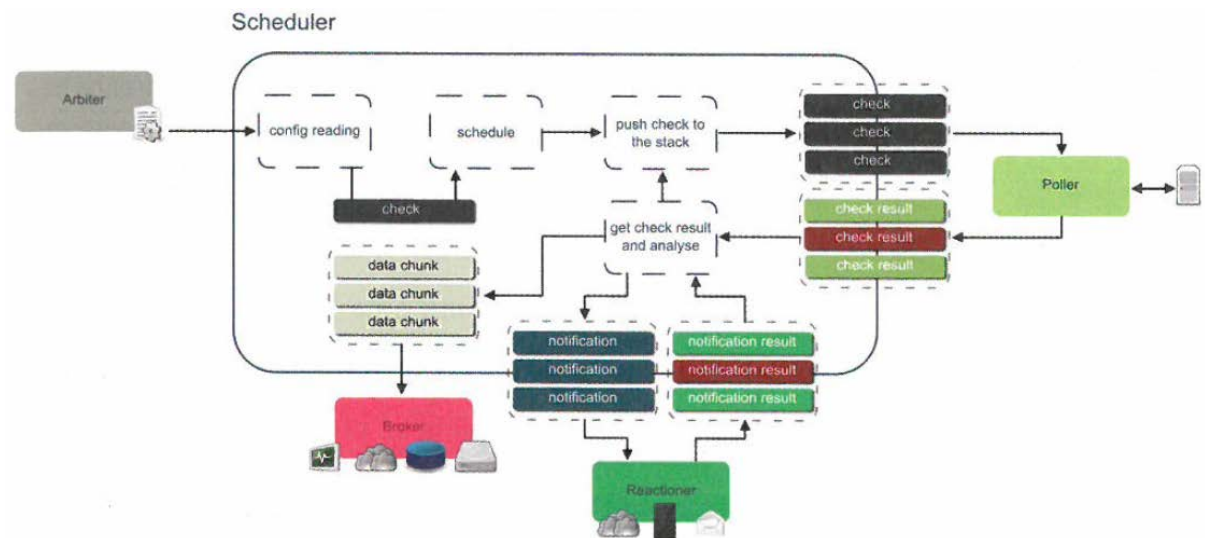


Schéma 3 – Fonctionnement du Scheduler

Le Poller :

Ils ont pour rôle de lancer les sondes demandées par les ordonnanceurs dans lesquelles ils vont piocher les demandes. Ils utilisent un *royaume* de *processus* qui effectueront les vérifications. Une fois par seconde, ils vont retourner les résultats aux ordonnanceurs pour que ces derniers puissent réaliser des actions supplémentaires si besoin (comme une notification) et réordonner les tests. Ils sont en première ligne pour les performances, et l'administrateur peut en avoir autant qu'il en a envie/besoin.

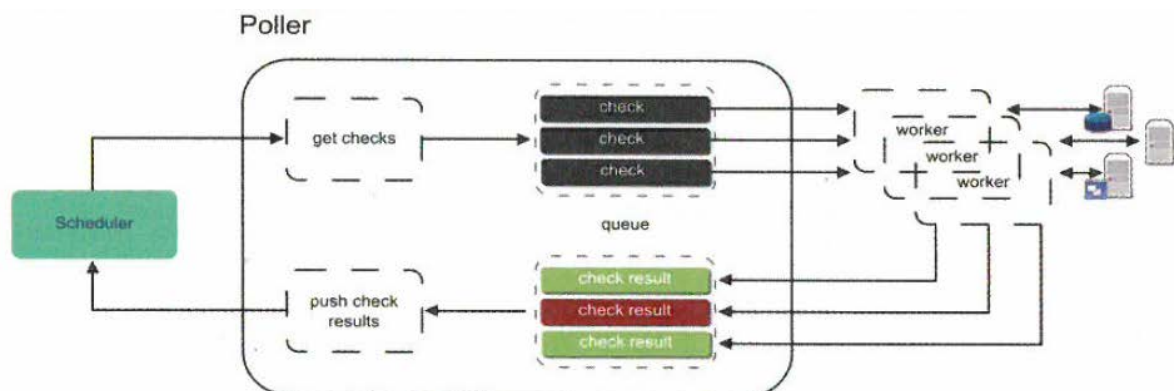


Schéma 4 – Fonctionnement du Poller



Le Reactionner :

Ils sont en charge de lancer les vérifications et les actions correcteurs. Ils sont découplés des *pollers* car il est plus pratique d'avoir un unique *reactionner* (avec un spare) pour n'avoir qu'un seul lieu où se remplissent les flux RSS d'alertes ou les autorisations SMTP par exemple.

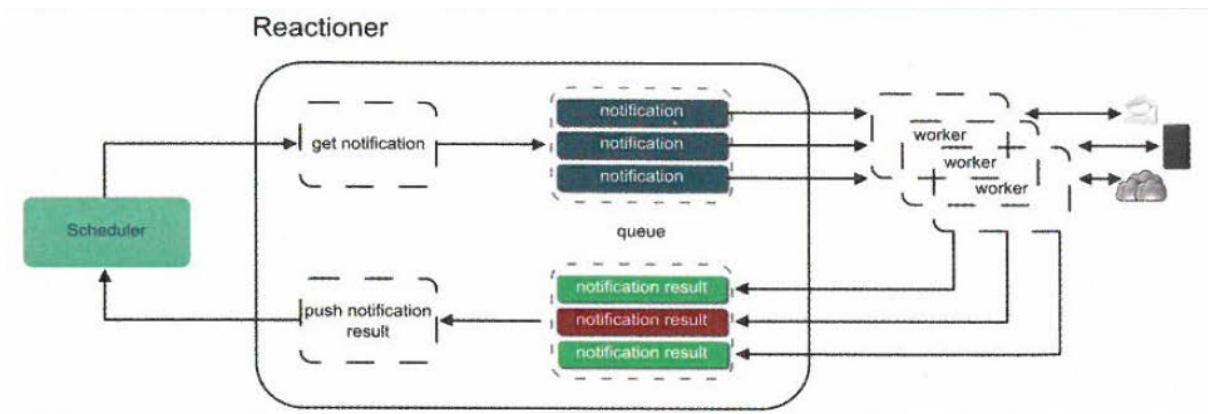


Schéma 5 – Fonctionnement du Reactionner

Le Broker :

Il est en charge de récupérer les informations d'états des ordonnanceurs et de les traiter. Ces traitements sont faits par des modules.

Ces traitements sont faits par des modules dont de nombreux types existent :

- un pour l'export en base merlin (MySQL),
- un autre pour l'export des données de métrologie dans le fichier service perfdata
- une présentation des données de style Livestatus,
- un export en base Oracle,

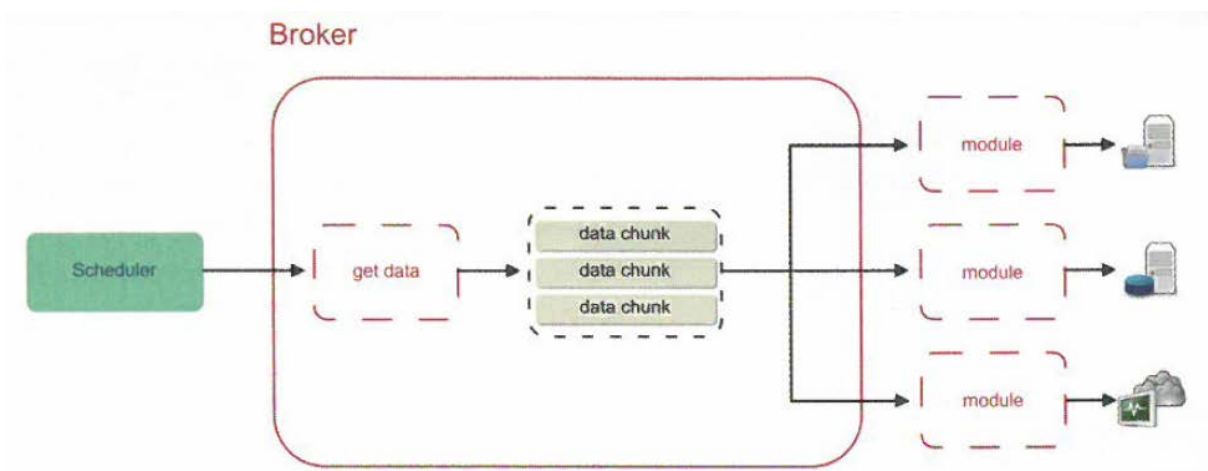


Schéma 6 – Fonctionnement du Broker



Le Receiver :

Son rôle est de recevoir les données d'acquisition passive et de les acheminer vers le bon Scheduler responsable de faire la corrélation et le traitement des statuts.

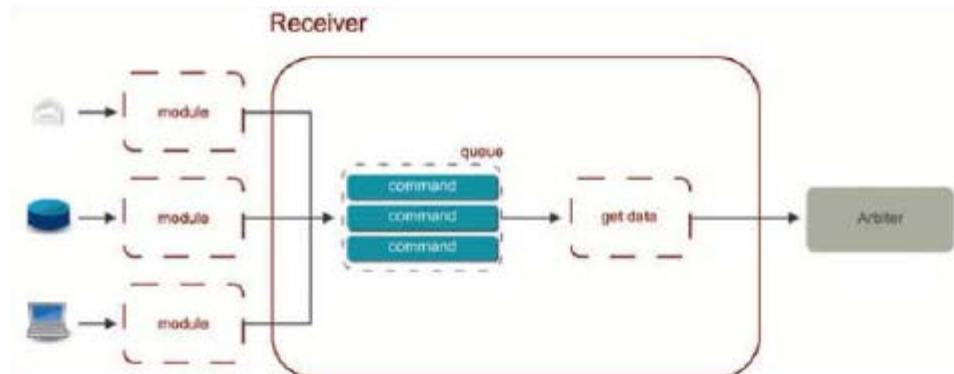


Schéma 7 – Fonctionnement du Receiver

Modes d'acquisition des données issues des contrôles

Shinken utilise deux types de modules d'acquisition afin de récupérer les données issues des contrôles :

1- Module d'acquisition actif des données

Un module d'acquisition intégré est une pièce optionnelle d'un logiciel qui est lancé par un démon Shinken. Le module est responsable de l'acquisition de données en utilisant un protocole spécifique dans une méthode très haute performance.

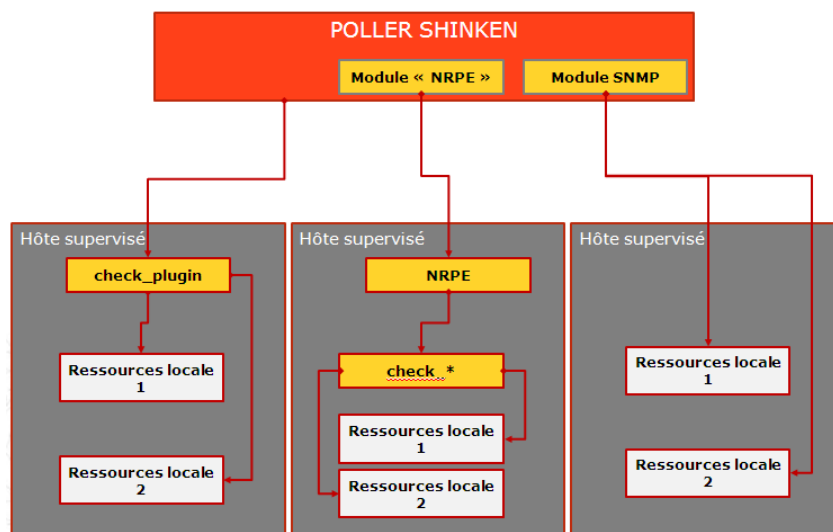


Schéma 8 – Fonctionnement du Poller avec les agents



Shinken fournit en standard :

- un module intégré d'acquisition de données SNMP: SnmpBooster,
- un module intégré NRPE d'acquisition de données : NRPEBooster,

A cela, se rajoute la possibilité d'utiliser des packs de supervision «prêts à utiliser» fournis par la communauté. Ces packs sont directement accessibles par le module d'acquisition du Poller.

2- Module d'acquisition passif des données

Ce module permet de récupérer les résultats de contrôles passifs, c'est-à-dire reçus depuis une source externe d'une manière spontanée.

Les résultats de contrôles passifs sont reçus par le démon du processus « Arbitrer » ou le démon du « Receiver ». Les modules chargés sur ces processus permettent la réception de divers types de formats de données, de type :

- Port d'écoute, ou LISTENER,
- TSCA (Apache Thrift),
- Web services.

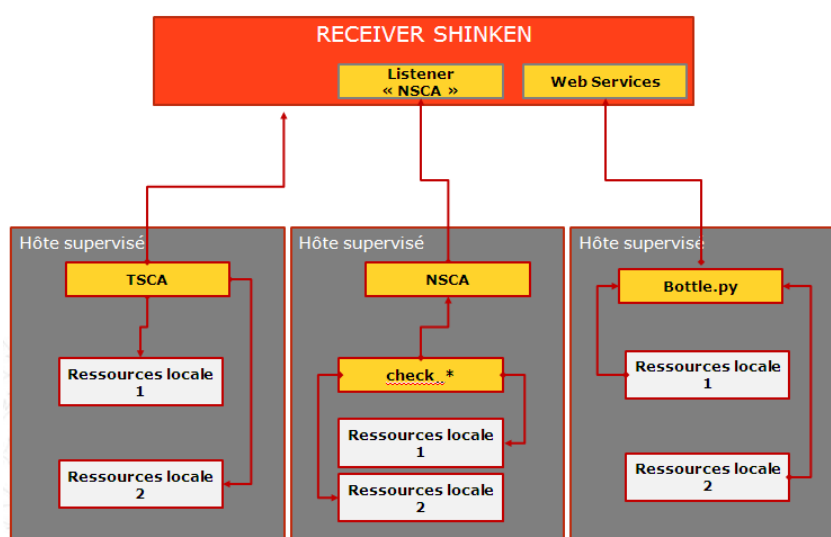
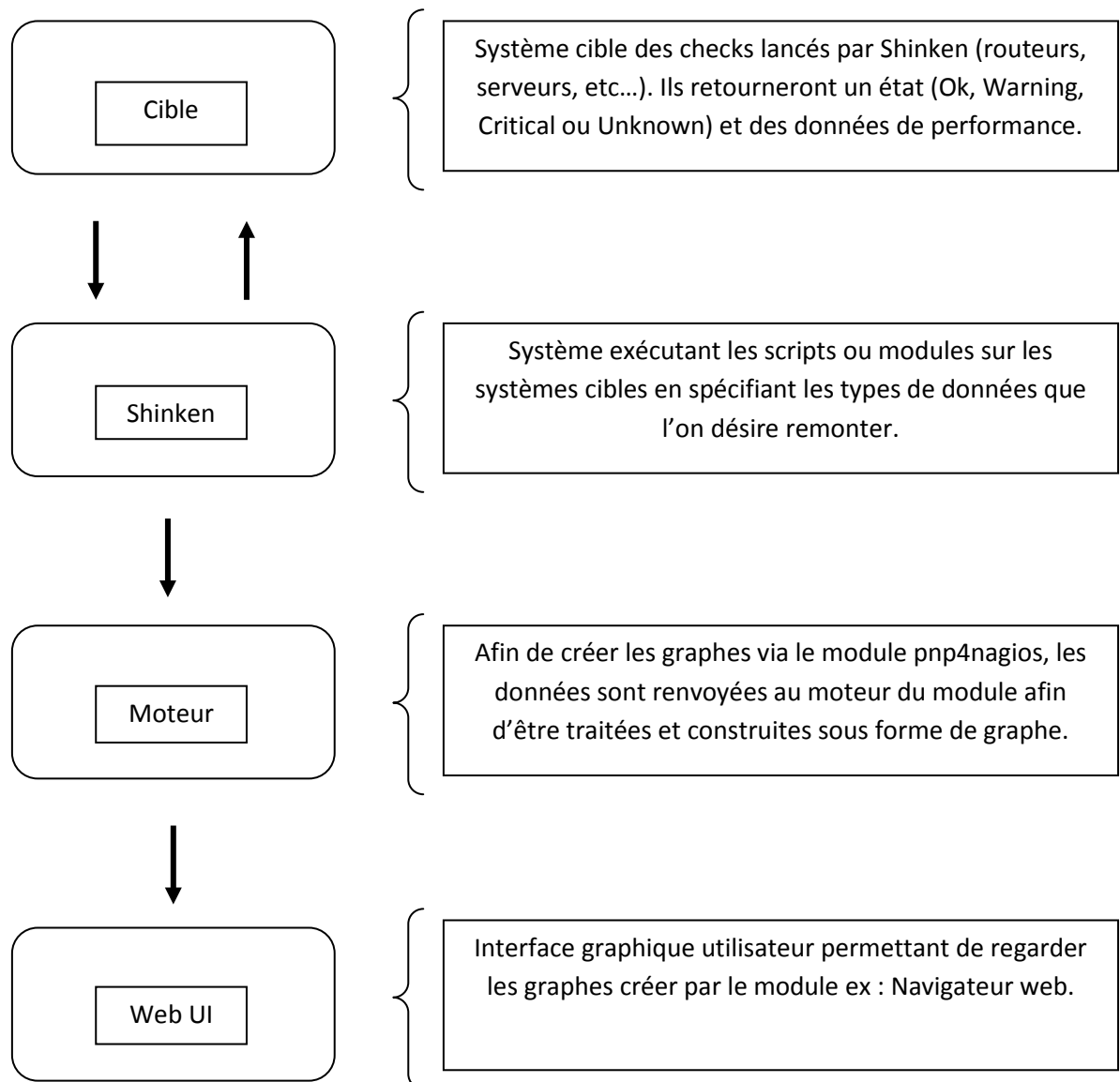


Schéma 9 – Fonctionnement du Poller avec les traps



On peut résumer le fonctionnement de Shinken par :



La web UI utilisée conjointement avec Shinken, il existe plusieurs Web UI.

Cependant nous travaillons avec Thruk, qui est une web UI qui se rapproche de l'UI de Nagios, ce choix était volontaire afin de ne pas dérouter les administrateurs systèmes étant habitués à utiliser Nagios comme système de supervision.

Elle possède certains avantages par rapport à celle de Nagios :

- La possibilité 'd'agréger' plusieurs Nagios dans une même interface
- L'utilisation de Livestatus, qui peut supporter des environnements très importants



- L'ajout de nombreuses fonctionnalités utiles, telles que :
 - L'utilisation de filtres, permettant (par exemple) de ne pas afficher les hôtes dont l'interruption, est prévue
 - La possibilité de sélectionner plusieurs hôtes/services d'un coup pour toutes les "commandes externes", comme l'ajout de commentaires sur un ensemble d'hôtes ou d'interruptions programmées
 - La possibilité d'avoir des infos sur le serveur en passant la souris au-dessus des services ou des hôtes (pas besoin d'ouvrir la page spécifique à l'hôte), et de voir les commentaires dans un pop-up
 - L'auto-complétion dans la barre de recherche d'hôtes
- La compatibilité avec Shinken

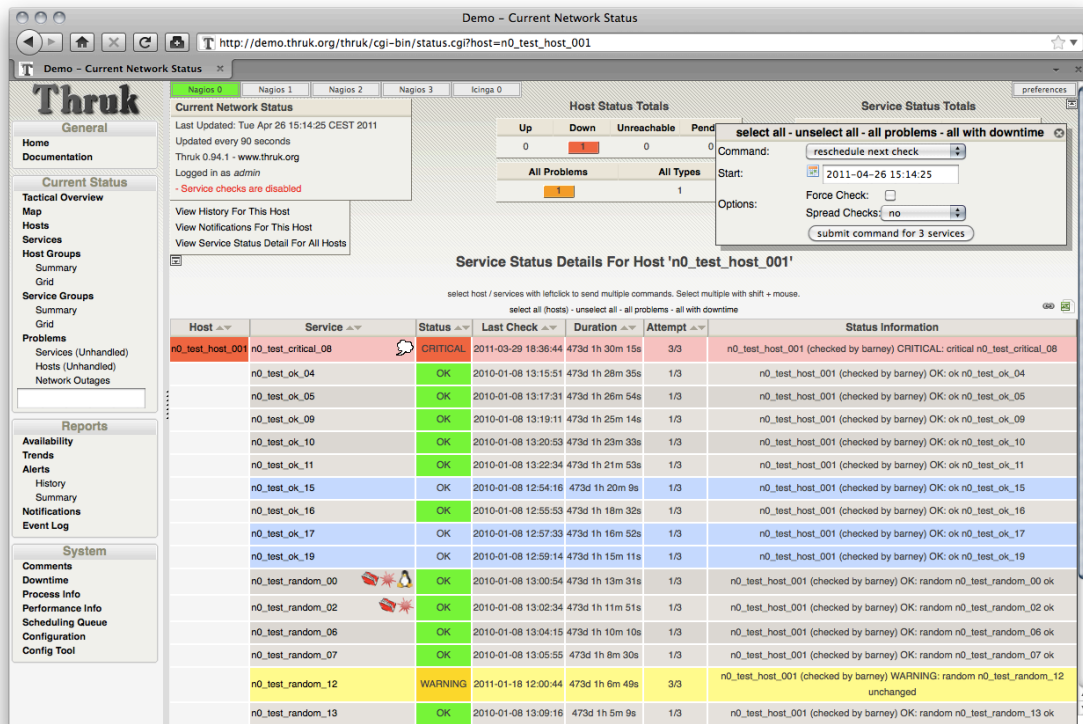


Image 12 – Interface Thruk

Nous venons de voir que Shinken gère des architectures hautement disponibles et performantes, nous allons voir en suivant ce qu'apportent la haute disponibilité et la haute performance.



II. Haute Disponibilité et Haute Performance

a. La haute disponibilité

La haute disponibilité est un service qui distribue un taux de disponibilité maximum. La disponibilité est aujourd'hui primordiale, pour une entreprise qui a des applications et services métiers vitaux. En cas d'indisponibilité, les répercussions peuvent être énormes si ce sont des applications qui impactent la production de l'entreprise.

En regardant le tableau ci-dessous, nous pouvons s'apercevoir qu'on atteint le seuil d'haute disponibilité à partir de 99,9999%. Soit en moyenne, 32 secondes d'indisponibilité.

Taux de disponibilité	Durée d'indisponibilité
97%	11 jours
98%	7 jours
99%	3 jours et 15 heures
99,9%	8 heures et 48 minutes
99,99%	53 minutes
99,999%	5 minutes
99,9999%	32 secondes

Image 13 – Taux de disponibilité

Mais il ne faut pas confondre la haute disponibilité et le PRA (plan de reprise d'activité). Ce sont bien deux tâches distinctes. Cependant, elles peuvent être complémentaires pour atteindre une disponibilité optimum.

La haute disponibilité a besoin d'une architecture bien adaptée et structurée, mais également des locaux isolés abritant cette architecture. Pour avoir la meilleure haute disponibilité possible, nous devons tout mettre en place au niveau des infrastructures des équipements (Alimentation, réseaux, climatisation...), mais également, une architecture matérielle complémentaire par la redondance des équipements. Par ailleurs, pour parfaire une structure hautement disponible, l'implémentation des clusters et la sécurisation des



données sont indispensables pour mettre en place une solution de haute disponibilité sur le long terme.

b. Cluster Informatique

Le Cluster est une architecture composée de plusieurs hôtes (ordinateurs formant des nœuds, où chacun des nœuds sont capables de fonctionner indépendamment des autres.



Deux principaux usages se distinguent :

- Un Cluster de Haute Disponibilité/ Haute Performance permet de répartir une charge de travail parmi un grand nombre de serveurs et d'assurer l'accomplissement de la tâche même en cas de défaillance d'un des nœuds.
- Le cluster de calcul permet de répartir une charge de travail parmi un grand nombre de serveurs afin d'utiliser la performance réunie de chacun des nœuds.

Historique du Cluster

Un Cluster est un regroupement de deux serveurs ou plus, en vue de créer un "super serveur virtuel". La théorie des clusters date du début des années 70.

Un cluster fournit des fonctions de haute disponibilité et de répartition de charge. Il facilite aussi l'évolutivité de la montée en charge. Originellement créé au milieu des années 80 par Digital Equipment Corporation, sous le nom de VAXCluster.

En 1995, un accord de partenariat signé entre Microsoft et DEC, donne naissance à Windows NT 4 Enterprise Server ; version qui intègre MSCS (Microsoft Cluster Serveur). Il existe aujourd'hui des solutions cluster sous Windows 2000/2003, Linux, Unix, AS/400.



Plusieurs constructeurs proposent des solutions propriétaires (Tandem, Siemens, Veritas, Novell, IBM, Sun...). Lotus fournit également une solution logicielle pour créer un cluster Notes entre des machines d'horizons différents (AS/400, Netware, Windows, Unix, Linux...).

Fonctionnement d'un Cluster

Un cluster exprime l'idée de grappe, ce terme regroupe un groupe de serveurs indépendants fonctionnant en une seule grappe. Le client ne voit pas la différence entre un cluster et une machine unique.

Les clusters sont couramment constitués de nœuds de calculs, de nœuds de stockages, et d'un ou plusieurs nœuds frontaux. On peut les relier entre eux sur différents réseaux.

L'utilisation d'un Cluster est de plus en plus importante dans les services Informatiques, où les besoins d'Haute Disponibilité et Haute Performance sont toujours aussi croissants et devient primordiaux pour le bon fonctionnement du SI.

Les Clusters sont utiles pour minimiser l'impact d'une panne serveur sur la disponibilité d'une application. Afin que l'application soit toujours disponible pour les clients du SI.

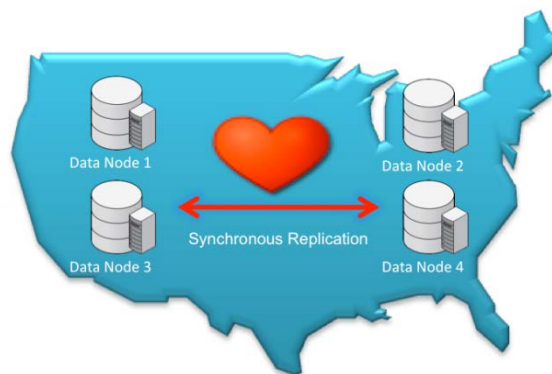


Image 14 – Exemple de cluster

Lors d'une défaillance d'un serveur dans la grappe (Cluster), le clustering réagit en isolant le système/hôte défaillant sans perte de données et garantit un service minimum de l'application envers ses clients internes/externes.

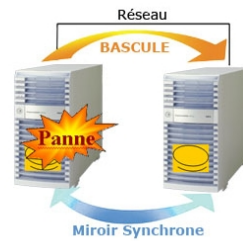


Image 15 – Exemple de panne d'un cluster

De même pour le partage des tâches d'un serveur surchargé avec un autre serveur au sein du Cluster (si les ressources sont partagées entre plusieurs tâches).

c. Fonctionnalités des Clusters

1. Cluster Actif/passif (Fail Over)

Un cluster actif/passif a pour but d'augmenter la disponibilité des applications métiers et de services d'une entreprise.

Dans une architecture haute disponibilité, le Fail-over permet une reprise automatique pratiquement immédiate si un équipement d'une grappe est défaillant. Le cluster actif/passif a donc la capacité de basculer automatiquement un serveur vers un autre serveur.

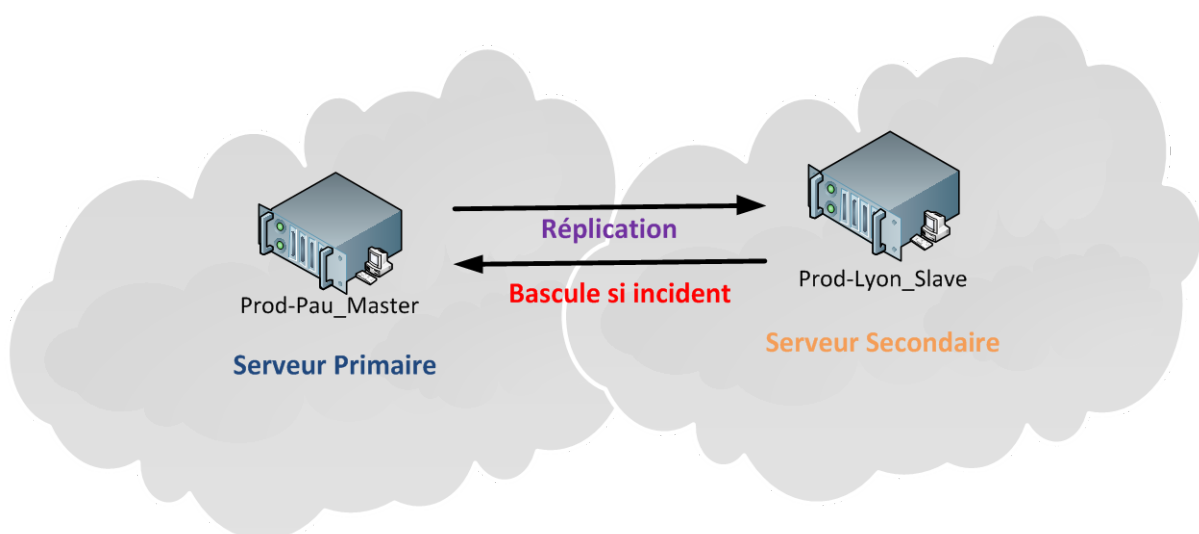


Schéma 10 – Cluster Actif/passif (Fail-Over)



Il existe deux modes de configurations du Fail-over :

- Automatique : Basculement automatique, de façon à être transparent pour les utilisateurs.
- Manuel : Intervention technique sur l'équipe afin de le faire basculer vers l'équipement de secours

2. Cluster Actif/Actif (Load balancing)

Le cluster actif/actif est un mécanisme du cluster «répartiteur de charge». Ce mécanisme sait diriger les processus/ calculs vers les ressources les moins occupées du cluster.

Le load balancing permet d'assurer la disponibilité des équipements au maximum, et offre le meilleur temps de réponse.

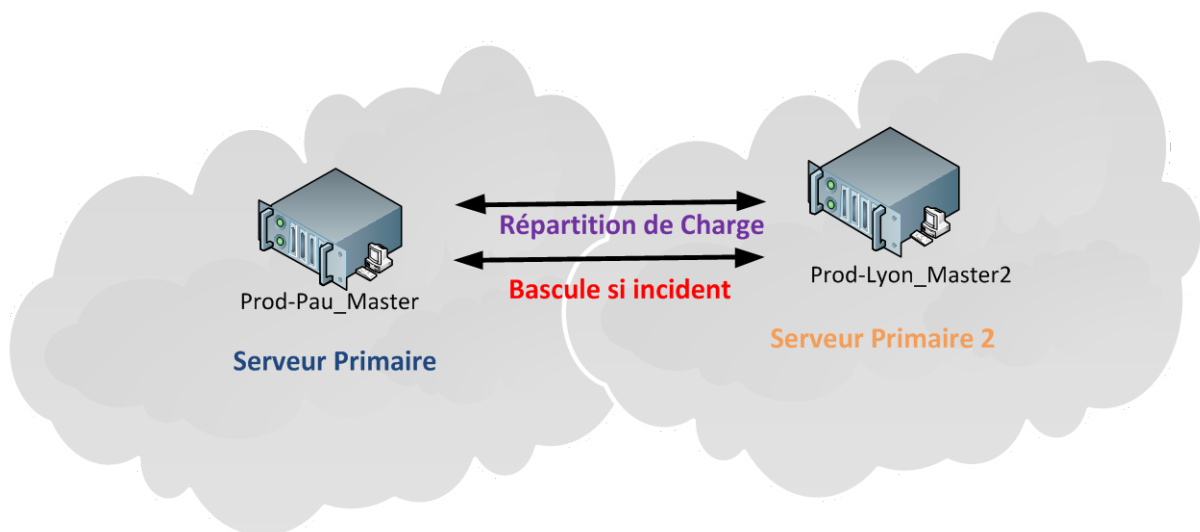


Schéma 11 – Cluster Actif/Actif (Load-Balancing)

Le cluster actif/actif a la particularité :

- D'augmenter la disponibilité
- De répartir les charges (Performance, car solution qui offre un meilleur temps de réponse)

Après avoir vu le fonctionnement des clusters, nous allons voir en suivant les clusters de supervision.



d. Présentation d'un Cluster de Supervision

Un cluster de supervision est caractérisé par deux serveurs ayant la même solution (système d'exploitation et logiciel de supervision).

Étant donné que les solutions de supervision surveillent la disponibilité de nombreux hôtes et services. La supervision se doit d'être la plus disponible possible.

La supervision dite « classique » contient quelques défauts comme :

- La charge
- L'Haute Disponibilité
- La gestion des configurations
- La Possible perte de site distant

La mise en place d'un cluster de supervision supprime ces défauts.

On va parler de supervision hautement disponible et hautement performante.

Caractéristiques d'un Cluster de supervision

Le cluster de supervision peut utiliser deux types de cluster :

- Le cluster actif/passif (Fail Over) est une réplication de données en temps réel et gère la reprise automatique si un dysfonctionnement survient.
- Le cluster actif/actif (Load Balancing) est une répartition de charge et gère également la reprise automatique si un dysfonctionnement survient.



Cluster Actif/Passif :

Le cluster Actif/Passif comporte deux serveurs hautement disponibles :

- Serveur primaire qui réalise les travaux de supervision
- Serveur secondaire qui prend le relai lorsque le serveur primaire ne fonctionne plus

Le contexte est simple, le serveur primaire va superviser les hôtes et services. Le second serveur contrôlera le serveur primaire, et il aura toute la configuration si le serveur maître est défaillant.

En cas de défaillance, il sera amené à prendre le relai et supervisera les hôtes et services en attendant que le serveur primaire soit rétabli.

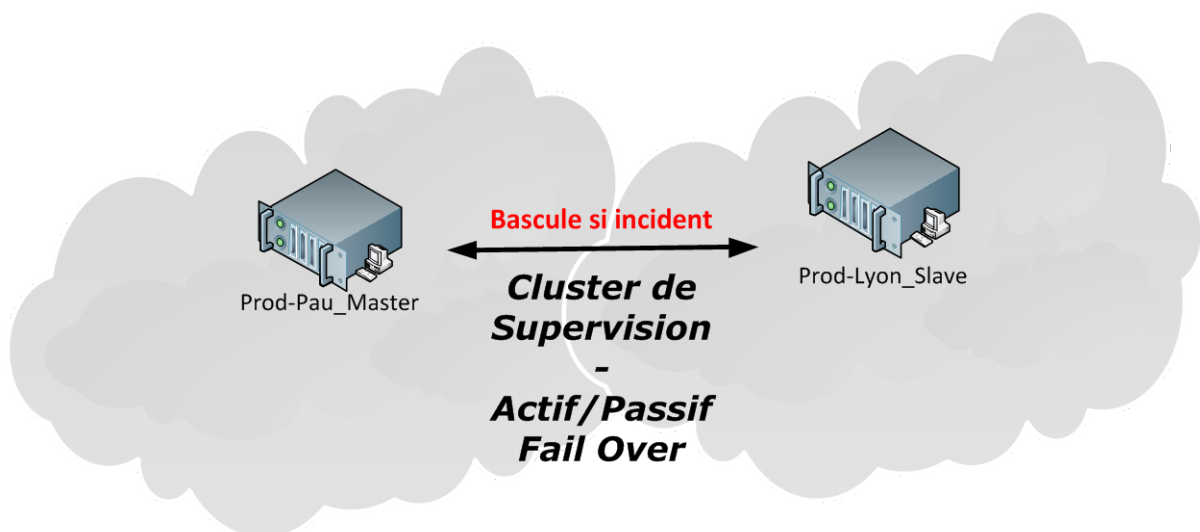


Schéma 12 – Cluster de supervision Actif/passif (Fail-Over)

Cluster Actif/Actif :

Le cluster Actif/Actif comporte deux serveurs hautement disponibles :

- Serveur primaire qui réalise les travaux de supervision
- Serveur secondaire qui se partage les travaux de supervision avec le serveur primaire, mais prend complètement le relai si le serveur primaire est défaillant.



Ici, le contexte est un peu plus complexe, le cluster actif/actif est une solution d'haute disponibilité et répartit la charge équitablement entre deux serveurs de la même grappe.

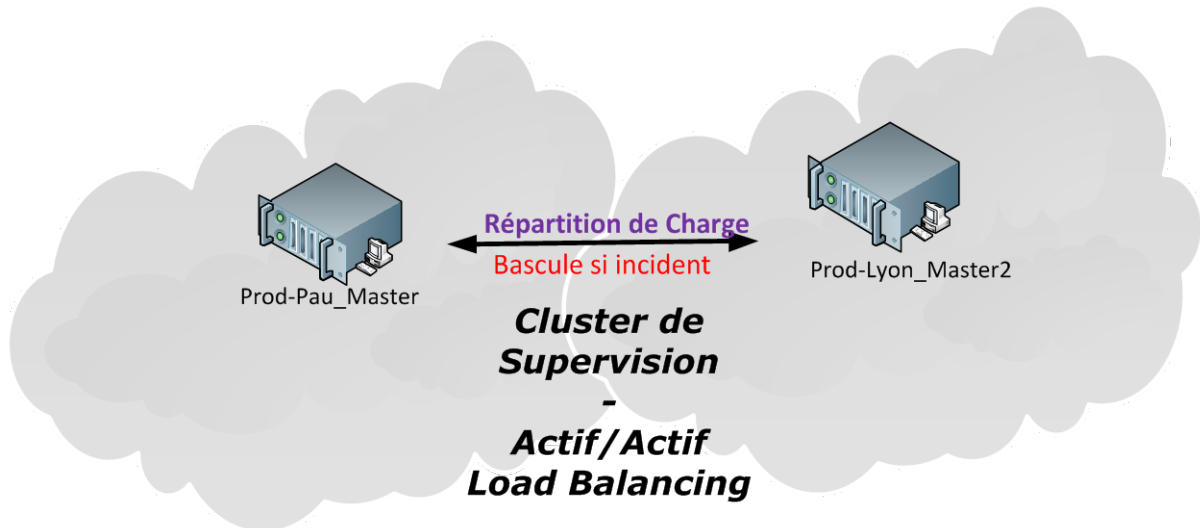


Schéma 13 – Cluster de supervision Actif/Actif (Load-Balancing)

e. Shinken et la Haute Disponibilité et la Haute Performance

L'outil Shinken est une solution distribuée, chaque composant peut être défini, configuré, dupliqué et réparti de manière autonome. Cette flexibilité donne à Shinken de très bonnes capacités en termes de haute disponibilité et haute performance.

Les fondements d'architecture distribuée de Shinken ont été créés rapidement dans son développement. Les principes de répartition des rôles et des tâches ont été découpés en plusieurs éléments, dans Shinken ce sont des nouveautés avec une simplicité d'administration.

Si on regarde Nagios, le mode distribué n'a pas été pensé lors de la conception de l'outil en 1999. Le mode distribué a été ajouté quelques années plus tard par les développeurs Nagios. Ces services Nagios ne se connaissent pas entre eux, et font uniquement des échanges de leurs données au sein d'une même base de données, permettant à des outils de présenter les informations de plusieurs serveurs sur une même interface.



Ce fonctionnement s'explique par la nature même du fonctionnement de Nagios, qui fait tout au sein du même processus. Ceci a atteint ses limites lorsqu'il est question de grands environnements et d'architectures réparties.

Afin de résoudre ce problème d'un seul même processus qui ne peut pas monter en charge, Shinken découpe le processus en plusieurs rôles. Chaque rôle/service se focalise sur sa tâche et ces services se connaissent parfaitement entre eux et peuvent se répartir de manière intelligente les tâches et la charge.

Les besoins recensés de Shinken sont :

- Consulter la configuration (*Arbiter*)
- Gérer et ordonner les contrôles (*Scheduler*)
- Lancement des checks par les sondes (*Poller*)
- Envoie des notifications (*Reactionner*)
- Exportation/Sauvegarde/Présentation des données sur les interfaces (*Broker*)
- Écoute et gère les informations passives (*Receiver*)

Ces besoins exprimés par Shinken sont les rôles/services que nous avons définis à la présentation de la solution Shinken.

Je vais reprendre un exemple fourni par Shinken, pour vous expliquer correctement ces services, puis nous regarderons comment mettre en place des architectures hautement disponibles.

The diagram illustrates the architecture of a distributed system, organized into three main layers: Receiver, Arbitrer, and Scheduler.

- Receiver Layer (Red):** Contains component 12, which receives input from the top and sends data to the Arbitrer.
- Arbitrer Layer (Green):** Contains component 13, which receives input from the Receiver and sends data to the Scheduler. It also contains a sub-component labeled "Commandes" (orange) which receives input from component 1 and sends data to the Scheduler.
- Scheduler Layer (Grey):** Contains component 8, which receives input from the Arbitrer and sends data to the Reactionner. It also contains component 10, which receives input from the Broker and sends data to the Poller. Component 4 is also present in this layer.
- Intermediate Components:**
 - Reactionner (Blue):** Receives input from component 3 and sends data to the Scheduler (component 8).
 - Broker (Purple):** Receives input from component 3 and sends data to the Scheduler (component 10) and the Poller.
 - Poller (Blue):** Receives input from component 3 and sends data to the Scheduler (component 5) and the Broker.

Numbered components (1-14) are distributed across the layers, indicating specific data flows and interactions within the system.

La configuration de Shinken est lue par l'Arbiter en phase 1, cette configuration est envoyée au Scheduler en phase 2. L'Arbiter va donc communiquer à tous les autres services qu'une configuration est disponible sur le Scheduler en communiquant l'adresse IP de ce dernier.

Le Scheduler commence à ordonner les sondes de supervision et gère la file d'attente des demandes (phase 4). Le Poller connaît désormais l'adresse du Scheduler, s'y connecte et sollicite les demandes de supervision en attente (phase 5). Puis il traite ces demandes en lançant les sondes sur les hôtes à superviser. Il renvoie les résultats qu'il reçoit au Scheduler (Phase 7).

Les résultats des contrôles effectués vont permettre au Scheduler de faire son travail de corrélation. Si un contrôle retourne un résultat négatif, il va créer une nouvelle demande de vérification, qui sera faite par un nouveau Poller afin trouver l'origine du problème du souci.



Une fois ce travail effectué, il peut alors créer une demande de notification dans une file d'attente (phase 8). Le Reactionner scrute cette liste régulièrement et lancera les commandes nécessaires.

Les informations sont transmises aux interfaces de supervision. À chaque changement d'état ou d'arrivée d'une nouvelle information, le Scheduler génère un morceau de données sur une file d'attente. Ces données sont vidées par le Broker (phase 10), qui va alors donner ces données à ses propres modules, qui vont mettre à jour les bases RRD – graphiques de performance. (Phase 11)

Si l'utilisateur souhaite donner un ordre à Shinken, il peut passer par le Receiver pour lui envoyer cet ordre (Phase 12). L'Arbiter va régulièrement récupérer ces éléments (Phase 13) et les redistribuer vers le Scheduler pour les traiter (Phase 14).

Multiplier les services ?

Nous venons de voir les instances de chaque service et nous allons voir les solutions Haute Disponibilité et Haute Performance de Shinken.

Shinken peut effectivement garder des services en « Réserve » au cas où qu'un de ceux qui sont actifs tombe en panne. (*Cluster Actif/Passif –Fail Over*)

C'est-à-dire que tous les services peuvent être multipliés de manière indépendante. Cela permet d'assurer la disponibilité de votre environnement de supervision avec tous les intérêts que cela apporte. Mais Shinken peut également répartir la charge des services, si le besoin se fait ressentir. (*Cluster Actif/Actif –Load Balancing*)

Dans un mode Load Balancing, nous pouvons effectivement multiplier des rôles dans un même Royaume afin d'éviter une surcharge trop importante sur un même service.

Comment pouvons-nous définir les royaumes ? Nous allons aborder ce sujet au prochain chapitre ci-dessous, afin d'y voir plus clair.



Des royaumes dans Shinken ?

Une multiplication de services dans de grands environnements montre rapidement des limites. Pour retirer ces fameuses limites, nous avons besoin d'une notion qui permet de multiplier des services en les découpant sur différents sites. Cette notion existe sous le nom technique de « Realm » ou encore de « Royaume ».

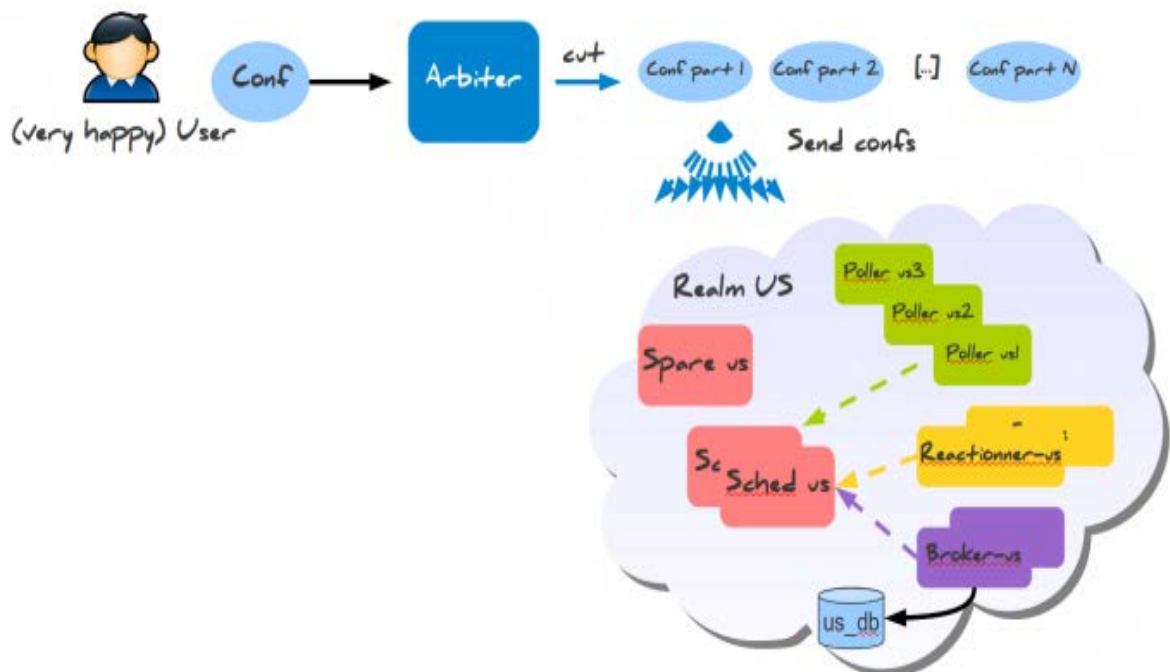


Schéma 15 – Fonctionnement des royaumes

Un Realm est un ensemble de services sur un même site, tous les services peuvent être affectés et multiplier dans un Royaume, excepté l'Arbiter car il ne peut avoir qu'un seul Arbiter Maître par Realm.

Les services d'un Realm vont se connecter uniquement entre eux. Aucune connexion ne sera établie vers les services d'autres Realms.

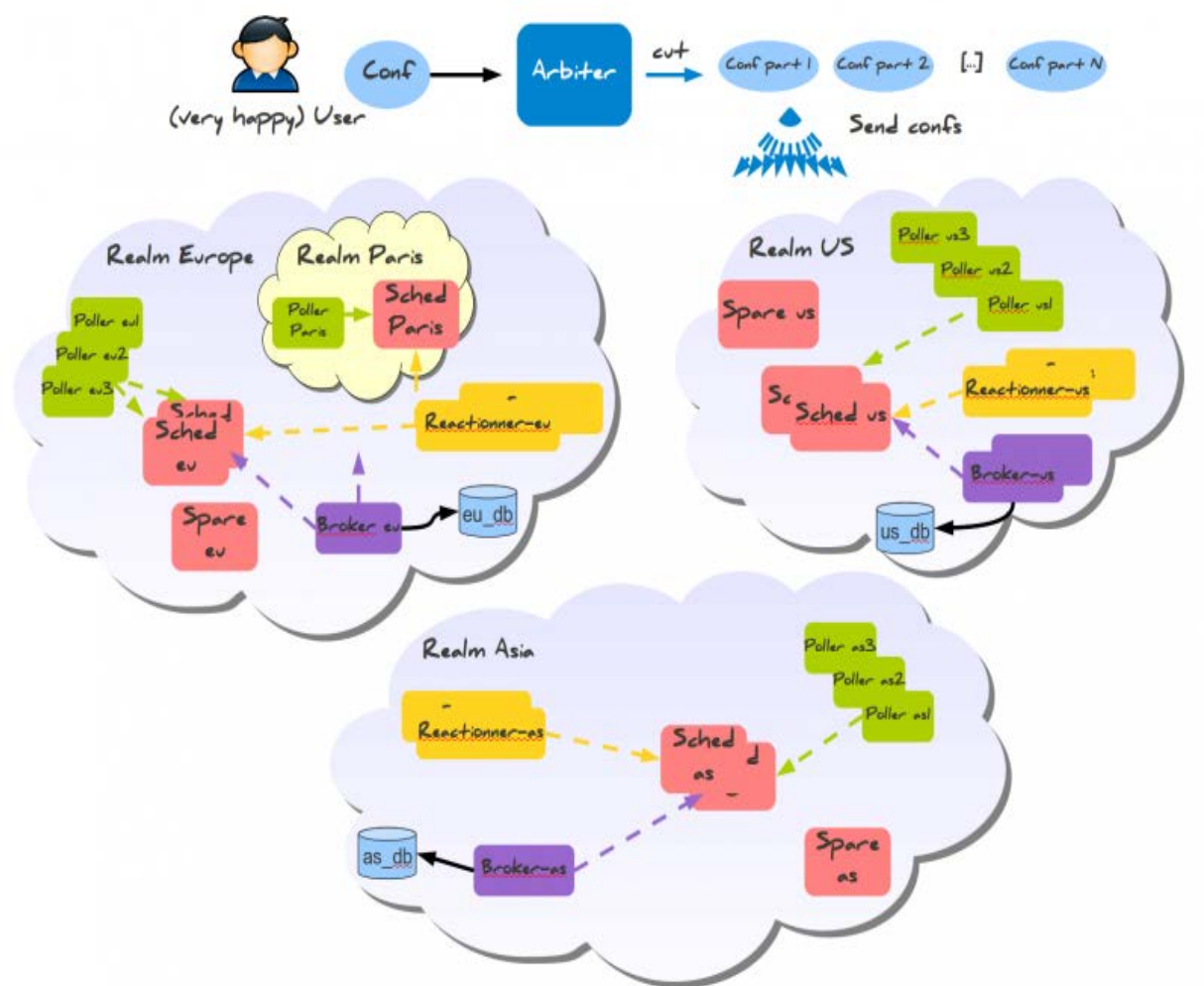


Schéma 16 – Fonctionnement avec plusieurs royaumes

Les Royaumes permettent d'avoir des possibilités d'architectures très larges.

Nous avons vu le fonctionnement global de Shinken avec des environnements hautement disponibles et performants, maintenant nous allons voir un peu plus en détail la mise en place d'un cluster de supervision.



f. Mise en place du Clustering

Comment la distribution sur plusieurs environnements fonctionne sous Shinken ?

Nous allons partir sur une architecture totalement doublée.

À chaque instance de service, il y aura un autre service de secours qui sera sur un second serveur et serait le remplaçant si le premier venait à être défaillant.

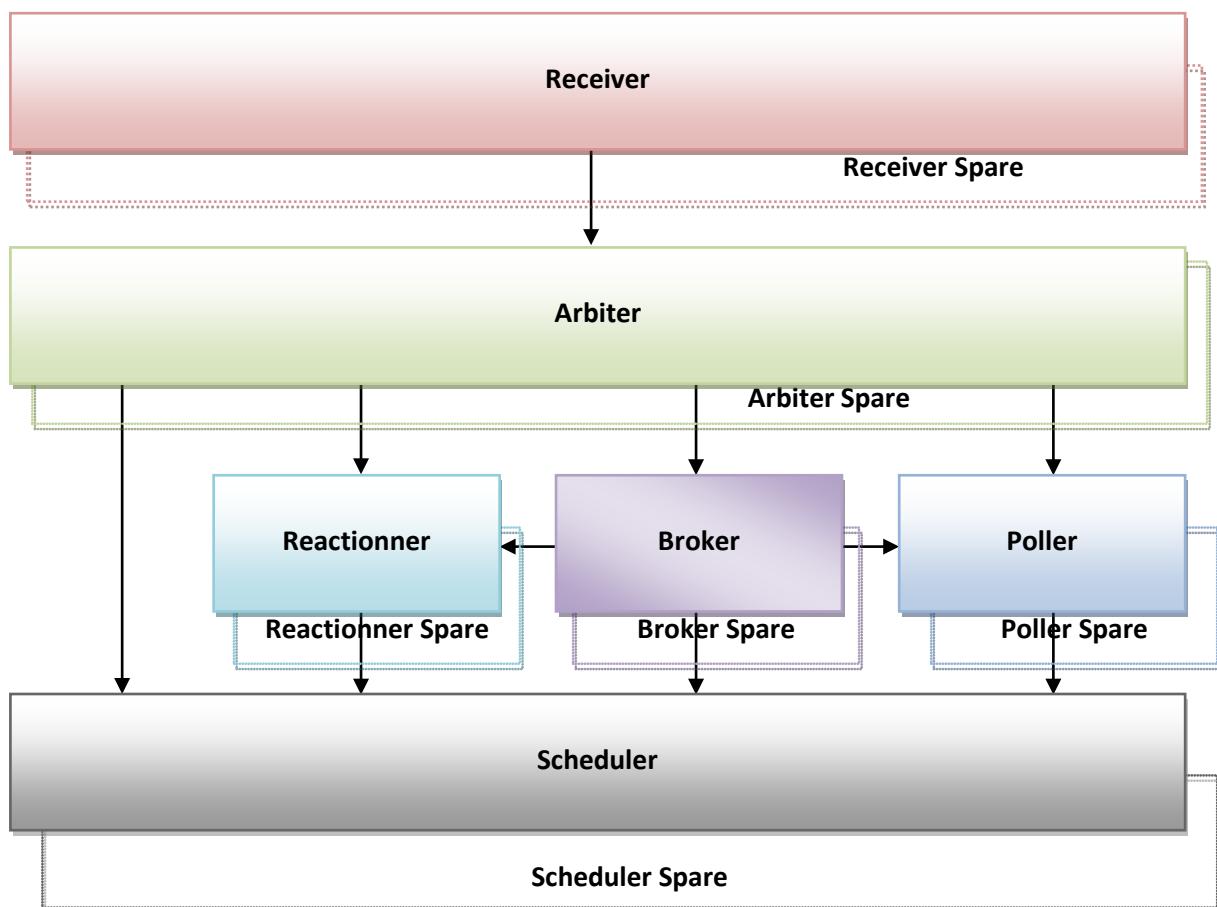


Schéma 17 – Mise en place d’une architecture hautement disponible avec multiplication des modules (spare)

Les services remplaçants sont nommés « spare » dans cet exemple. Le surveillant de l’architecture est l’Arbiter maitre, car il connaît parfaitement tous les services. Il demande régulièrement des informations aux services, et vérifie que leurs configurations sont correctes et fonctionnent correctement.



Si un service est défaillant, il lui renvoie la bonne configuration, s'il n'y a pas de changement d'état du service en question, il envoie la configuration à son secondaire (spare). Le service secondaire reprend le relai du service primaire qui a connu le défaut.

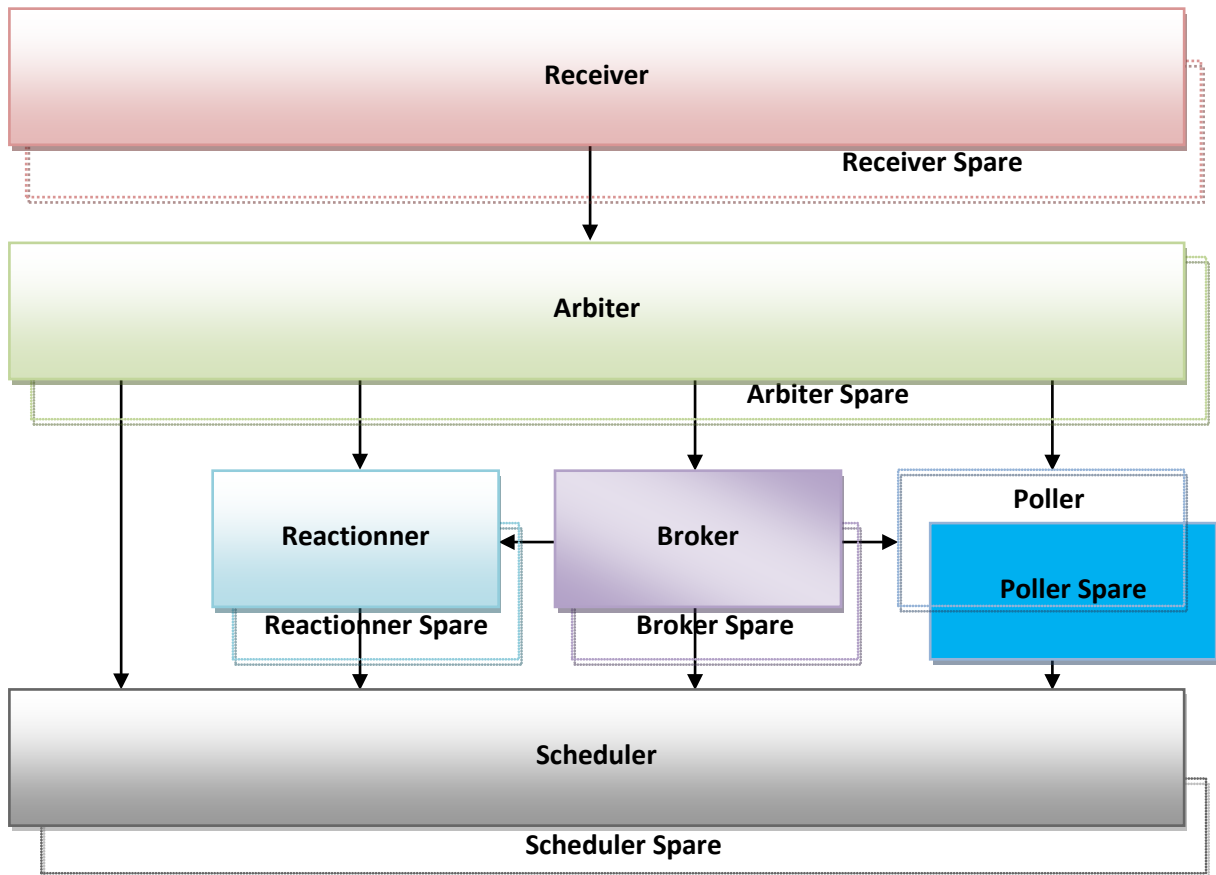


Schéma 18 – L'arbitre détecte une défaillance et envoie la nouvelle configuration au Poller « spare »

Suite à notre exemple, les composants objets de Shinken sur le second serveur seront identiques au premier. Il faut donner à l'Arbiter maître les connaissances des différents services de l'environnement en question.

Il faut éditer le fichier « `/usr/local/shinken/shinken-specific.cfg` », ce fichier contient la liste des services qui sont connus par l'Arbiter.

Il faut dupliquer chaque configuration de service, en remplaçant son nom et son adresse par ceux du second serveur.



Afin de définir le service primaire ou secondaire, le petit paramètre dans le fichier de configuration qui servira à distinguer les versions « actives » ou « passives » doit être défini au niveau du [*spare*]:

- Si le Spare est positionné à « 0 », le service sera actif

```
define arbiter{
    arbiter_name    Poller-master
    address         10.11.120.209
    host_name       server-master
    port            7770
    spare           0
}
```

L'Arbiter envoie sa configuration au service primaire régulièrement du moment qu'il est opérationnel.

- Si le Spare est positionné à « 1 », le service sera donc passif

```
define arbiter{
    arbiter_name    Poller-slave
    address         10.11.120.210
    host_name       server-slave
    port            7770
    spare           1
}
```

L'Arbiter enverra sa configuration à ce service secondaire si l'actif est amené à disparaître.

Configuration de l'Arbiter

La configuration de l'Arbiter est un peu plus complexe que la configuration des autres objets.

L'Arbiter a besoin d'identifier le rôle (maitre ou esclave) qu'il aura au sein de l'architecture. Il effectue cette vérification en regardant dans la configuration s'il y a d'autres Arbiter. S'il n'y a aucun autre Arbiter dans la configuration, un objet est créé à la première lecture de la configuration. Cet objet définit l'hôte en question comme Arbiter et l'ajoute à la



configuration automatiquement. Cette configuration sera envoyée aux différents services en suivant.

S'il y a d'autres Arbiter avec la même configuration, il y aura forcément des complications. D'une part, les deux Arbiters sont considérés comme maitres et d'autre part le hostname configuré sera faux dans l'un des deux.

Il faut donc bien configurer ce paramètre sur les deux serveurs.

```
define arbiter {  
    spare 0  
    address 10.11.120.209  
    port 7770  
    arbiter_name Arbiter-prime  
    host_name srv-master  
}
```

```
define arbiter {  
    spare 1  
    address 10.11.120.210  
    port 7770  
    arbiter_name Arbiter-slave  
    host_name srv-spare  
}
```

Nous pouvons avoir un Arbiter Master et un Arbiter Slave mais nous ne pouvons pas avoir deux Arbiter Master sur la même infrastructure.

Afin de garantir une solution hautement disponible et utilisable, il faut conserver toutes les configurations et les données mises en place. Pour cela il faudra mettre en place des modules de rétention.

Grâce aux modules de rétention, les services comme le Scheduler et l'Arbiter qui ont des rôles différents, doivent garder impérativement en mémoire toutes les informations de l'infrastructure.



Avec les modules de rétention, nous assurons la persistance des informations dans les services, même si une défaillance est signalée sur un de ces services.

Plusieurs modules de rétention sont proposés :

- MongoDB
- MemCached
- Redis
- Pickle

Il faut rééditer le fichier «*/usr/local/shinken/shinken-specific.cfg*», on configure les modules de rétention depuis ce fichier.

Ici, nous devons renseigner l'adresse du serveur dans le module MongoDB par exemple :

```
define module {  
    module_name Mongodb  
    module_type mongodb  
    uri mongodb://srv-master/?safe=true  
    database shinken  
}
```

Il faut également activer le module de rétention sur les services en question :

```
define arbiter {  
    modules ,PickleRetentionArbiter  
    spare 0  
    address 10.11.120.209  
    port 7770  
    arbiter_name Arbiter-prime  
    host_name srv-master
```

Maintenant nous avons une solution de supervision hautement disponible et utilisable, mais pas encore optimum.

Si nous insérons la notion de royaume dans la mise en place d'une architecture totalement doublée sur différents data-center/sites.



Royaume non commun

Si nous prenons deux data-center séparés, nous pouvons mettre chaque data-center dans un Realm afin d'avoir un cloisonnement complet. Sur chaque data-center, nous avons une instance de chaque service. Excepté l'Arbiter qui reste unique avec au maximum un Arbiter de secours si l'arbiter primaire est défaillant.

On définit les realms dans le fichier «*/usr/local/shinken/shinken-specific.cfg*», nous configurons par exemple un data-center en France.

```
define realm {  
    realm_name France  
}
```

Nous configurons le deuxième royaume pour l'autre data-center

```
define realm {  
    realm_name Espagne  
}
```

Nous affectons les realms aux différents services. Pour chaque instance de service, il faut rajouter le paramètre « realm » pour lui indiquer dans quel royaume il se trouve.

```
define scheduler {  
    modules ,PickleRetention  
    spare 0  
    address 10.11.120.209  
    scheduler_name Scheduler-prime  
    realm Espagne  
    port 7768  
}
```

Dans ce cas-là, le scheduler primaire est dans le realm Espagne.



Pour avoir une solution fonctionnelle, nous devons avoir tous nos services affectés dans un royaume.

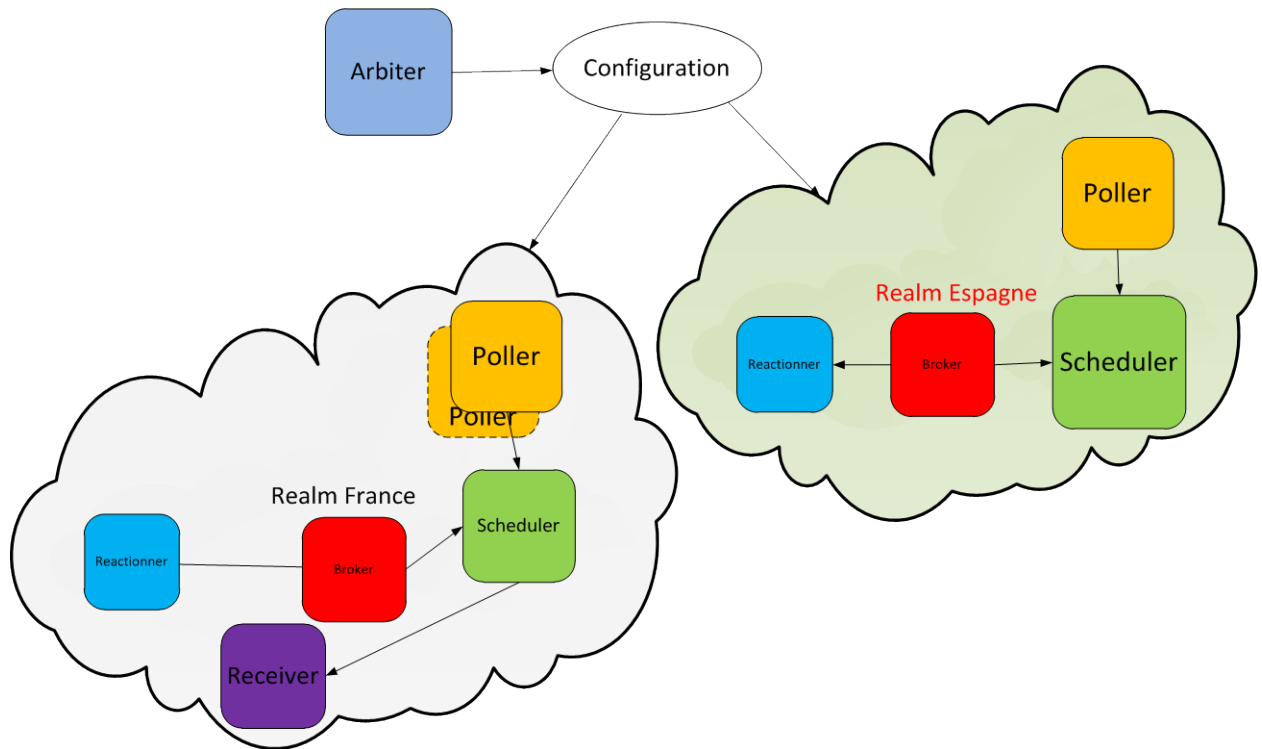


Schéma 19 – Création de deux royaumes non communs « France et Espagne »

Nous avons donc deux data-centers et deux Realms, ces deux royaumes opèreront comme deux sites totalement séparés. Ils sont juste reliés par la même configuration, mais ne peuvent pas se partager les ressources.



Royaume commun et hiérarchique

Pour avoir une architecture centralisée et avoir une unique interface pour visualiser la solution de supervision, nous pouvons mettre en place une fonctionnalité des Realms qui est la hiérarchisation.

On peut définir des sous-royaumes à un royaume qui est au niveau au-dessus. Le realm qui contient les sous-realm peut atteindre les services de ces sous-Realms sans soucis.

Par exemple, nous définissons un Realm Europe et il contiendra deux sous-realms :

```
define realm {  
    default 1  
    realm_name Europe  
    realm_members France, Espagne  
}
```

Pour que la configuration soit complète, il faut affecter le sous-royaume qui convient au service.

Pour cet exemple, nous plaçons le « receiver » dans le sous-royaume « France » et donc automatiquement dans le royaume parent qui est « Europe ».

```
define receiver {  
    port 7773  
    spare 0  
    address 10.11.120.209  
    realm France  
    receiver_name Receiver-prime
```

Pour cette architecture, on ne garde qu'une instance pour les services Brokers et Reactionner, donc on les affecte au royaume parent. Dans cet exemple, nous affectons donc le broker dans le royaume de haut niveau qui est « Europe ».



```

define broker {
    broker_name Broker-Prime
    modules Livestatus, Simple-log, WebUI, NPCDMOD
    manage_arbiters 1
    manage_sub_realms 1
    spare 0
    address 10.11.120.209
    realm Europe
    port 7772
}
  
```

Suite à nos différents exemples, nous avons l'infrastructure suivante :

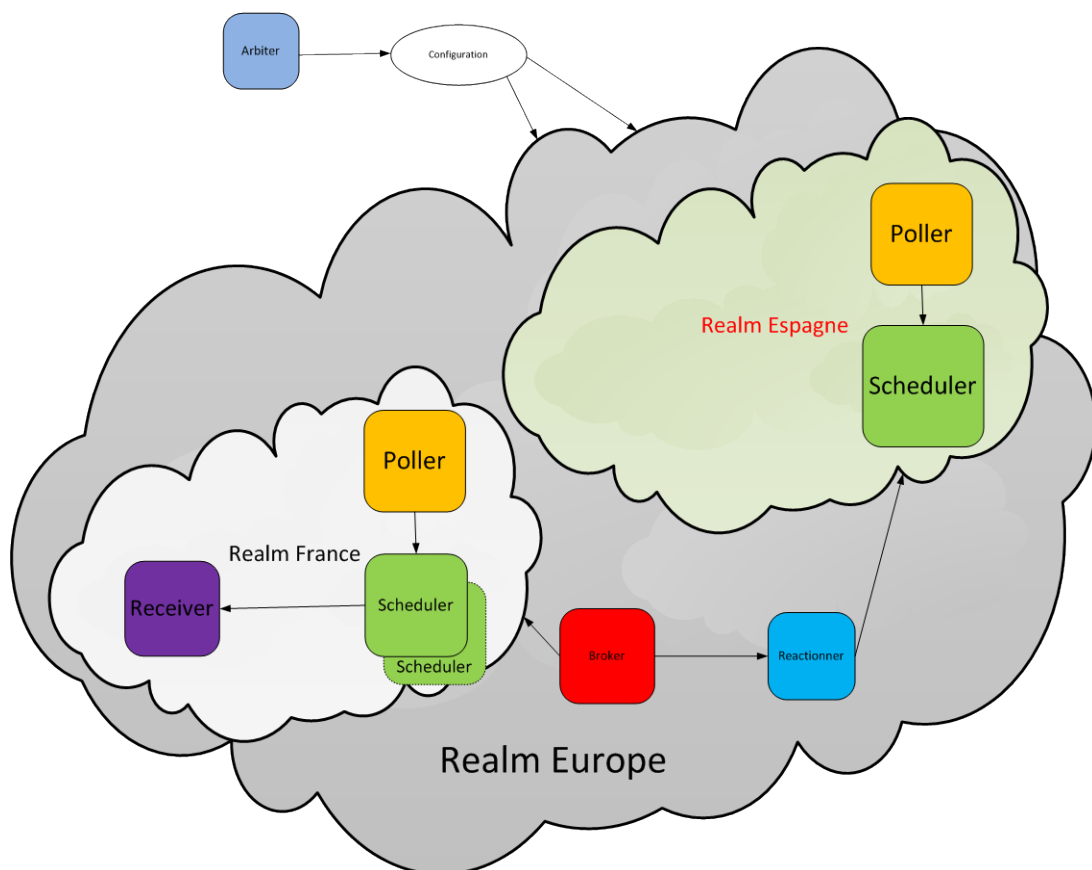


Schéma 20 – Création d'un royaume commun « Europe » avec deux sous royaume « France et Espagne »



L'Arbiter a pour rôle de découper les configurations en différentes forêts, il envoie également les configurations aux différents services. Dès que les services reçoivent toutes les informations nécessaires, ils effectuent leurs tâches.

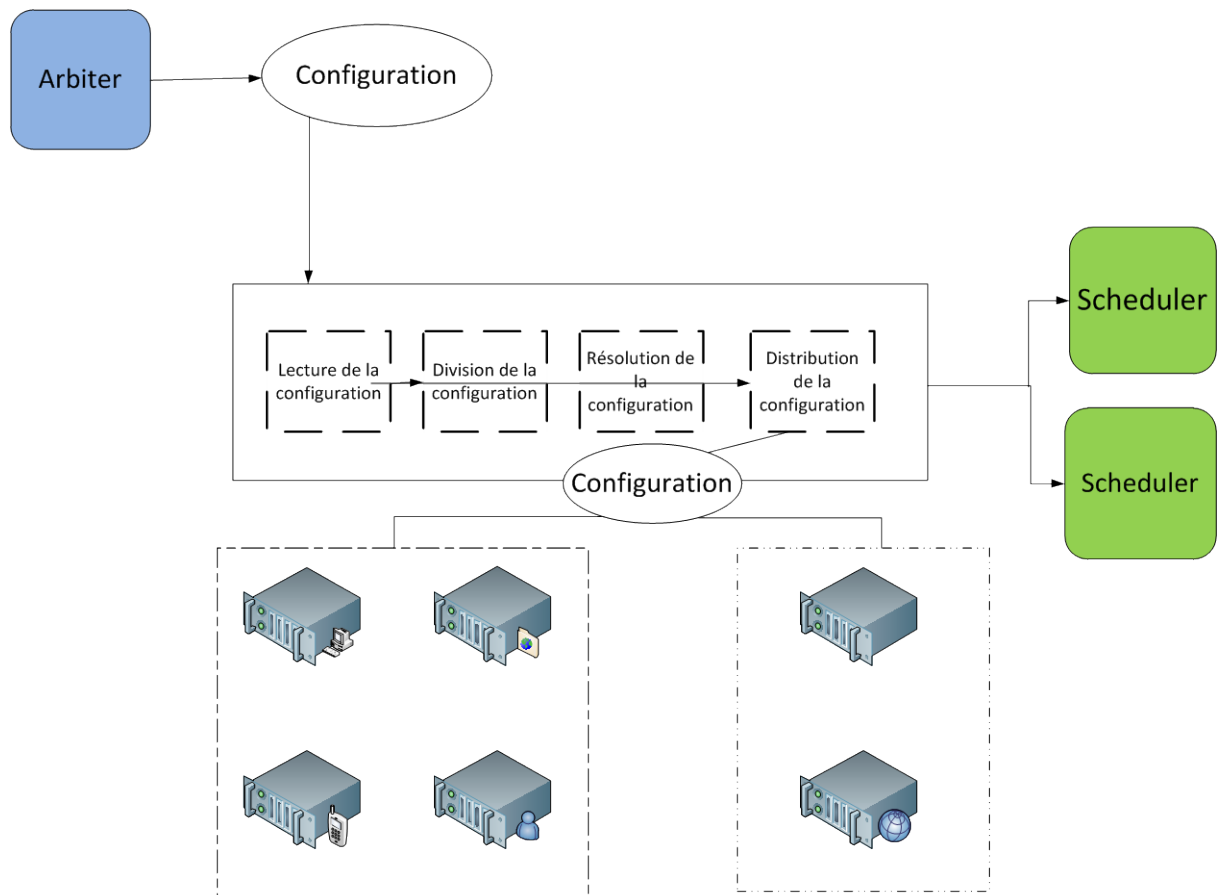


Schéma 21 – Fonctionnement du découpage de la configuration par l'Arbiter

g. Mise en place d'une infrastructure de cluster sur site distant

De nos jours les entreprises ont leurs sites globalement répartis sur l'ensemble des différents continents.

Pour répondre au mieux à cette problématique d'infrastructure de cluster sur site distant, nous avons identifié deux grands risques :



- Perte de lien entre sites distants

Si nous avons une perte de lien avec un site distant, nous pouvons ni accéder ni superviser.

- Saturation des liens intersites

Le trafic peut être minime entre les différents rôles dans un LAN, mais il peut d'autant plus augmenter sur une architecture de WAN et cela peut causer des grandes répercussions sur les bandes passantes entre les sites.

La solution la plus cohérente est d'avoir une instance sur chaque site des différents services. Pour assurer la continuité de la supervision sur un site en cas de perte de lien, il faut mettre en place les services principaux à savoir un Scheduler, un Poller, un Broker et un Reactionner.

Il faut rappeler que l'Arbiter est toujours unique et peut gérer des sites distants avec un point de contrôle central. Cependant les autres services devront être présents sur chaque site, cette solution permettra de réduire la bande passante entre les sites.

Néanmoins, il ne faut pas oublier les royaumes (Realms) vus au chapitre précédent pour faciliter la mise en place de cluster sur sites distants. Les royaumes permettent de gérer des grands environnements, mais aussi des infrastructures avec des clusters sur sites distants.

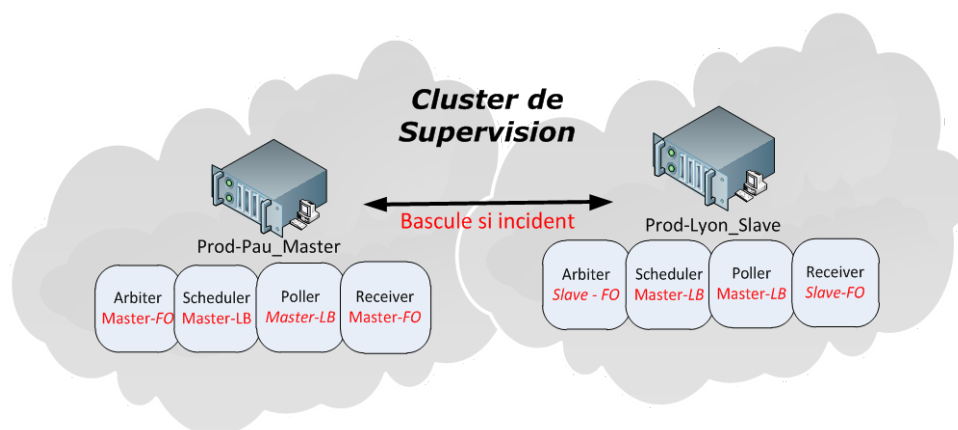


Schéma 22 – Fonctionnement d'un cluster de supervision sur site distant

Après avoir vu la mise en place d'un cluster de supervision, nous allons vous présenter les architectures mises en place sur différents clients, d'une supervision basique à un cluster hautement disponible et performant.



III. Résolvez la problématique

a. Supervision Classique

Nous avons mis en place une solution classique de supervision pour une société anonyme.



Les besoins exprimés par cette société sont :

- Gain de productivité
- Amélioration des engagements de services
- Une vision capable de mesurer les impacts sur les métiers

1. Contraintes

L'infrastructure informatique de cette société anonyme est sur un seul site.

La société souhaite une solution de supervision malléable et évolutive dans un souci d'agrandissement de l'entreprise dans les prochaines années.

2. Descriptif des serveurs de Supervision

Chaque entreprise met à disposition des serveurs pour la mise en place de la solution de supervision. Cependant toutes les entreprises ne dépendent pas des mêmes ressources.

La société nous attribue des serveurs de supervision basique, cependant ces machines correspondent suffisamment au besoin de l'entreprise. Mais si une évolution de la solution de supervision est envisagée, il faudra effectivement augmenter les caractéristiques du serveur.



Les caractéristiques du serveur attribué sont :

Caractéristiques	Physique
Volume Utile (Go)	120
CPUs	2*3 core
Mémoire (GB)	4

3. Solution de supervision

Les besoins primordiaux de cette entreprise sont de gagner en productivité et améliorer les engagements de services. Ces besoins entraînent la mise en place d'une solution de supervision classique, car il n'y a pas d'importance particulière d'un service continu sur la solution de supervision actuellement. Une supervision classique répond largement aux besoins de cette entreprise. Cependant ils ont émis la contrainte d'évolutivité à moyen terme. Nous allons mettre une architecture de supervision classique pour commencer, si l'infrastructure informatique de cette entreprise évolue rapidement et massivement, on aura la possibilité de faire évoluer la solution de supervision, en une solution hautement disponible et hautement performante.

Pour cette solution, nous avons un seul royaume « France », étant donné qu'il n'y a qu'un site pour le moment, si d'autres sites apparaissent à moyen terme, ils seront déclarés sous-royaumes.

L'infrastructure est construite sur un seul serveur, nous avons les six rôles sont unique. Pour une évolution, il sera possible de redonner ces six rôles sur un autre serveur.

Schéma explicatif de la solution ci-dessous :

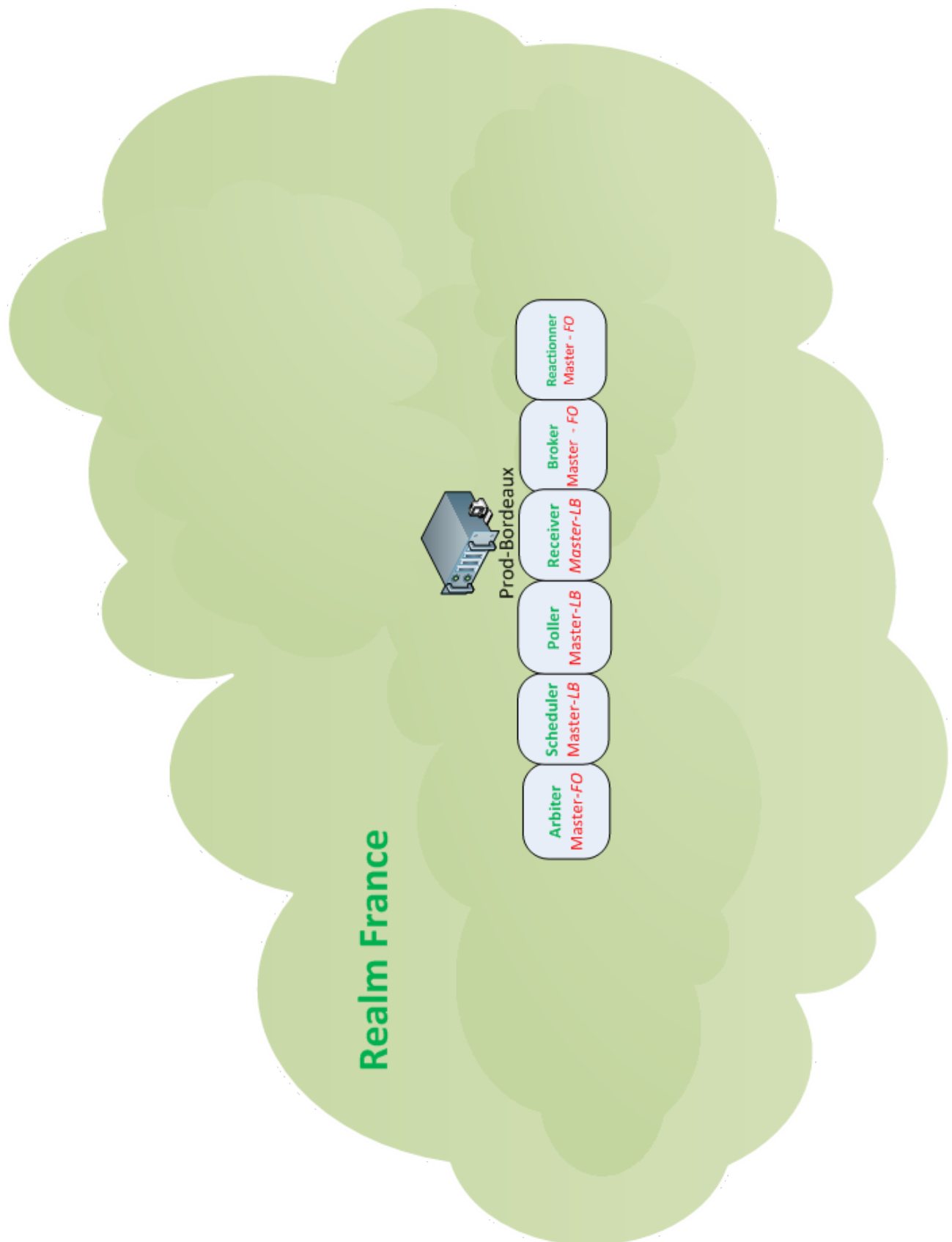


Schéma 23 – Solution retenue pour une architecture classique



b. Cluster de Supervision

Nous avons étudié une solution de supervision pour l'infrastructure de CACF.
L'ancienne solution de supervision de CACF qui était Nagios ne répondait plus aux besoins de CACF.



Les besoins exprimés par le Crédit Agricole sont :

- Gain de productivité
- Amélioration des engagements de services
- Supervision continue
- Possibilité d'évolution
- Une vision capable de mesurer les impacts sur les métiers de CACF

Une vision globale en temps réel des équipements des deux sites de CACF

1. Contraintes

L'infrastructure informatique de CACF se divise en deux sites distants Evry et Roubaix.
La majeure partie du parc informatique à superviser se concentre sur le site d'Evry.

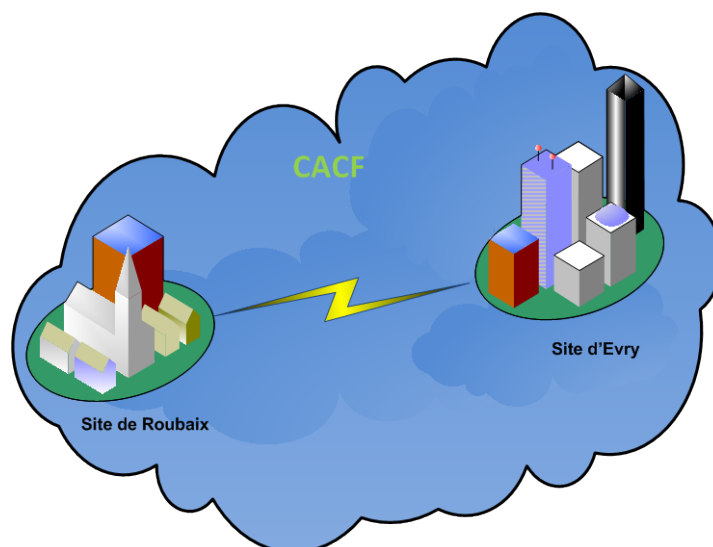


Image 16 – Sites distants de CACF



2. Descriptif des serveurs de Supervision

CACF nous attribue deux serveurs avec des caractéristiques correctes, ces serveurs correspondent au besoin de l'architecture de supervision.

Les caractéristiques des serveurs attribués sont :

	Physique	
	Bundle1	Bundle2
Volume utile (Go)	60 Go	
CPUs		2 * 2 core
Mémoire (GB)		6

3. Solution de supervision

Le besoin primordial est d'avoir une supervision accessible à n'importe quel moment pour arriver à gagner en productivité et améliorer les engagements de services. Ce besoin entraîne la mise en place d'une architecture hautement disponible et hautement performante. Pour un souci d'évolution de l'entreprise, cela entraînera également la mise en place d'une architecture distribuée.

Une première architecture a été pensée et a été mise en place avec un seul royaume et de rattacher tous les hôtes à superviser à ce royaume, qu'ils soient de Roubaix ou encore d'Evry. (*Annexe 1*)

Cependant cette solution n'était pas optimum pour une évolution possible de l'infrastructure de CACF. Il était donc plus judicieux d'avoir un royaume parent « France » et de créer deux sous-royaumes « Roubaix » et « Evry » avec chaque serveur sur chaque site distant pour avoir une certaine modularité. Si l'infrastructure de CACF évolue avec l'intégration de filières ou d'autres sites, la solution de supervision pourra se rendre malléable à cette évolution.

Nous avons donc un Royaume France qui englobe deux sous royaumes « Roubaix » et « Evry », un serveur de supervision sera sur chaque site, afin de superviser l'ensemble de l'infrastructure.

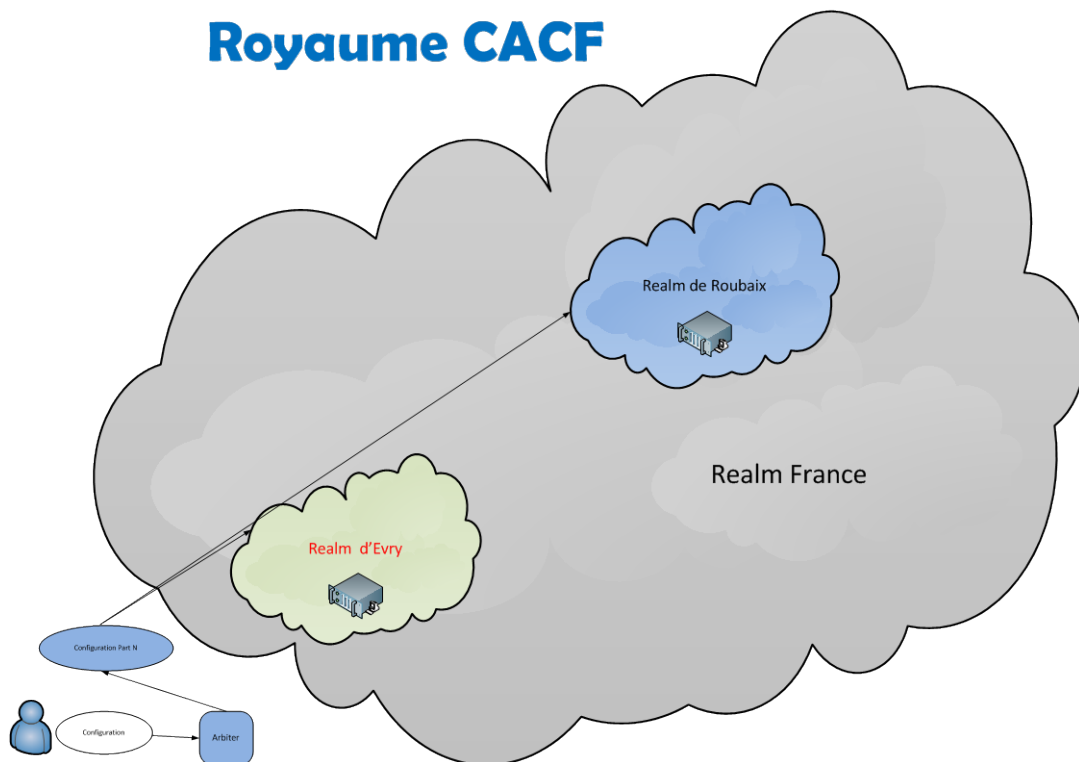


Schéma 24 - Mise en place des royaumes sur l'infrastructure CACF

L'infrastructure retenue est construite sur deux serveurs distants, les rôles/services ne sont donc redondés qu'une seule fois.

L'Arbiter, le Broker et le Reactionner sont en mode Actif/Passif (Fail Over) afin d'avoir un rôle en réserve qui peut prendre la main à n'importe quel moment si un rôle du serveur primaire est défaillant. Il ne peut avoir qu'un seul Arbiter primaire et un seul Arbiter secondaire. Ces trois rôles se situent dans le realm de la France. Les rôles du serveur de supervision d'Evry1 (realm Evry) sont actifs et les rôles du serveur de supervision d'Evry2 (realm Roubaix) sont passifs.

Afin de partager la charge de travail entre les deux sous royaumes « Roubaix » et « Evry » et assurer un service continu en cas de problème, les rôles scheduler, poller, receiver sont en mode Actif/Actif (Load Balancing). Ces rôles-ci se situent sur les royaumes d'Evry et Roubaix.

Schéma explicatif de la solution ci-dessous :

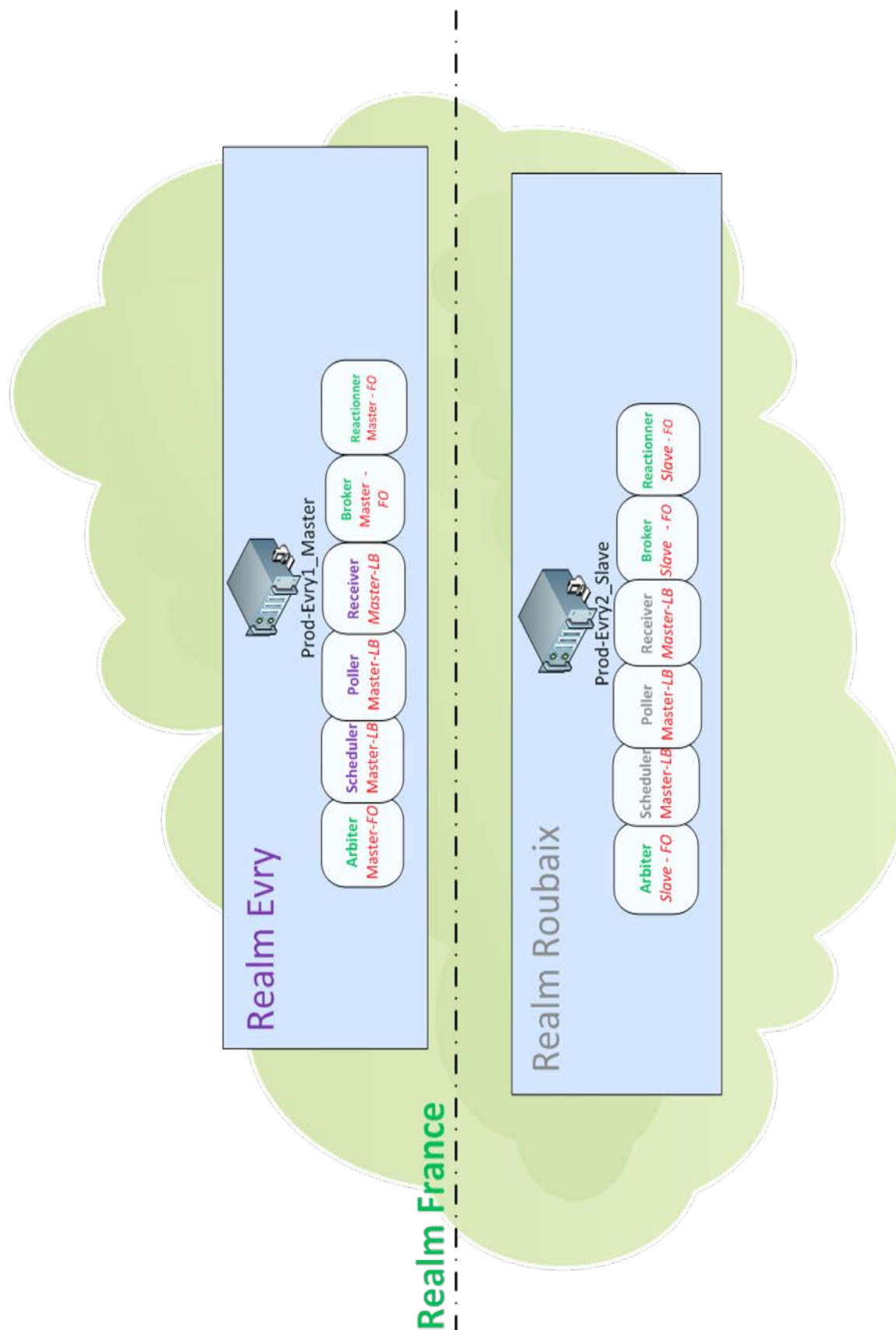


Schéma 25 – Solution retenue pour l'infrastructure de CACF



c. Cluster de Supervision redondé

Nous avons étudié une solution de supervision pour l'infrastructure de TOTAL. Il existait plusieurs solutions de supervision au sein de la branche de TOTAL EP mais ceci ne répondait plus aux nouveaux besoins de TOTAL.



Les besoins exprimés par TOTAL sont :

- Gain de productivité
- Amélioration des engagements de services
- Possibilité d'évolution
- Aucune interruption possible du service de supervision
- Une vision capable de mesurer les impacts sur les métiers de TOTAL EP
- Une vision globale en temps réel des équipements sur tous les sites et filiales de Total EP

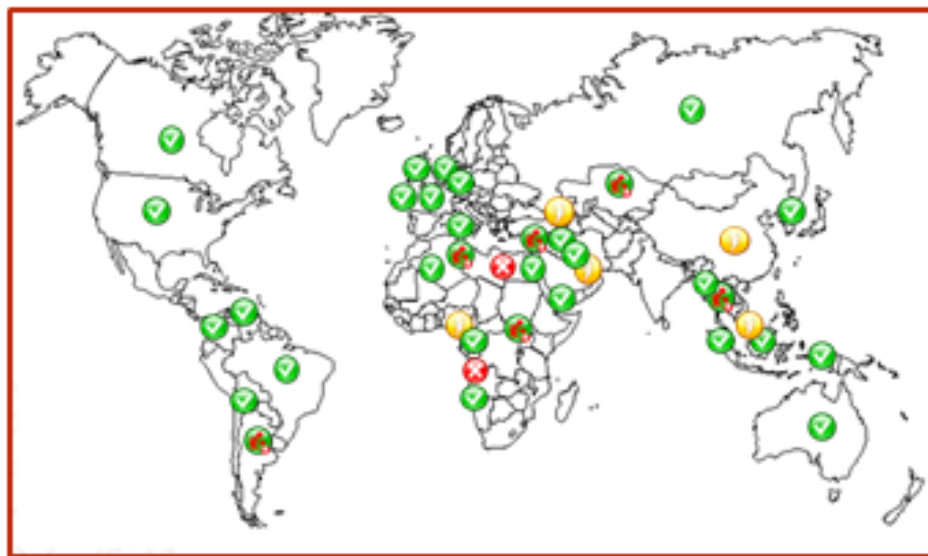


Image 17 - Vision globale en temps réel sur tous les sites de Total EP



1. Contraintes

L'infrastructure informatique principale de TOTAL se divise en deux sites distants :

- Paris
- Pau

Le site de Paris est un site administratif et le site de Pau est un site scientifique et de recherche de TOTAL EP. Les deux sites se partagent la majeure partie du parc informatique à superviser.

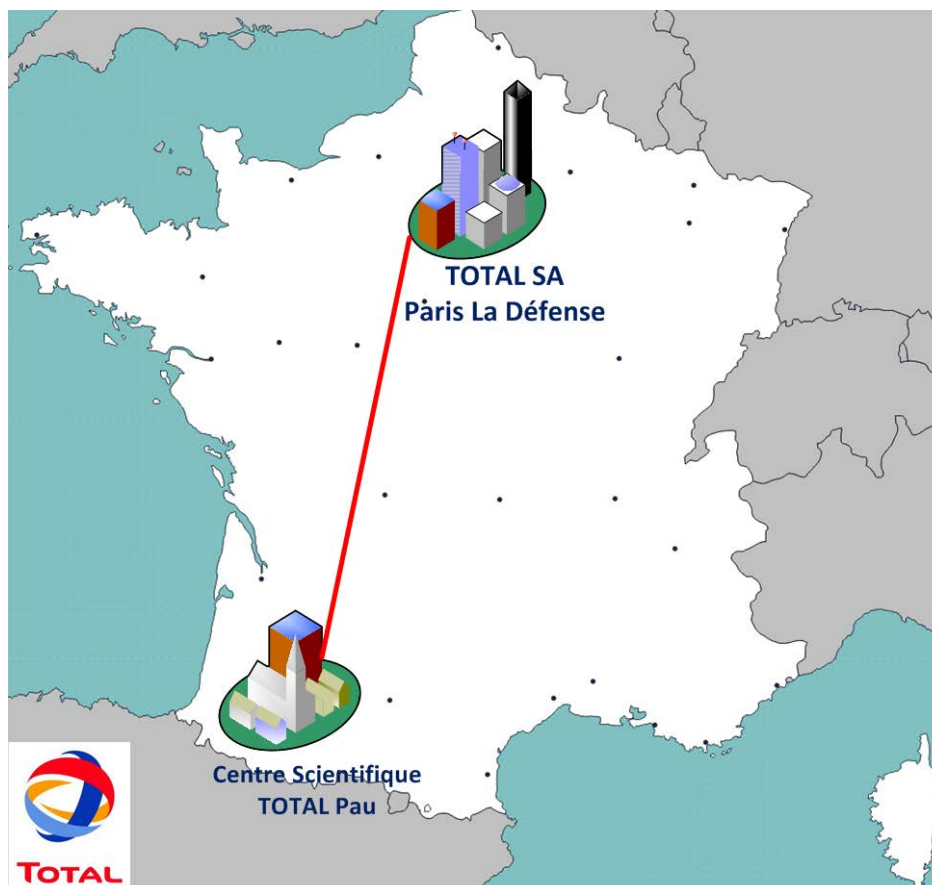


Image 18 – Sites distants de TOTAL EP

Cependant les filiales de TOTAL EP sont également à superviser à court terme. Elles sont une cinquantaine et sont réparties partout dans le monde (image 17).



2. Descriptif des serveurs de Supervision

TOTAL nous attribue quatre serveurs avec des caractéristiques optimums, ces serveurs correspondent complètement au besoin de l'architecture de supervision.

Les caractéristiques des serveurs attribués sont :

	Physique	
	Bundle1	Bundle2
Disques Système – Raid1	2*146Go	
Volume Utile (Go)	115 Go	
Disques Données – Raid5		4*146Go
Volume Utile (Go)		350
CPUs		2 * 6 core
Mémoire (GB)		16

3. Solution de supervision

Le besoin primordial est d'avoir une supervision accessible en continu pour arriver à gagner en productivité et améliorer les engagements de services. Ce besoin entraîne la mise en place d'une architecture hautement disponible et hautement performante. Pour un souci d'évolution de l'entreprise et de ses filiales, cela entraînera également la mise en place d'une architecture distribuée.

Plusieurs architectures ont été étudiées, avec différentes solutions possibles, avec un des deux royaumes distincts pour chaque site. Pour que chaque site ne supervise que leur royaume. (*Annexe 2*)

Mais cependant cette solution ne répondait pas suffisamment aux divers besoins exprimés par le client. Effectivement il est plus judicieux d'avoir un royaume parent « France » et d'avoir deux sous-royaumes « Paris » et « Pau » avec deux serveurs de supervision sur chacun de ses royaumes pour avoir cette fameuse modularité qui sera demandée pour l'intégration de la supervision des filiales de TOTAL EP mais également l'évolution de l'infrastructure de TOTAL. Cette solution pourra donc se rendre malléable à cette évolution.



Nous avons donc un Royaume France qui englobe deux sous royaumes « Paris » et « Pau », deux serveurs de supervision seront répartis sur chaque site afin de superviser l'ensemble de l'infrastructure. Quand nous rajouterons les filiales à superviser, chaque filiale aura son propre sous-royaume.

Royaume TOTAL WORLD

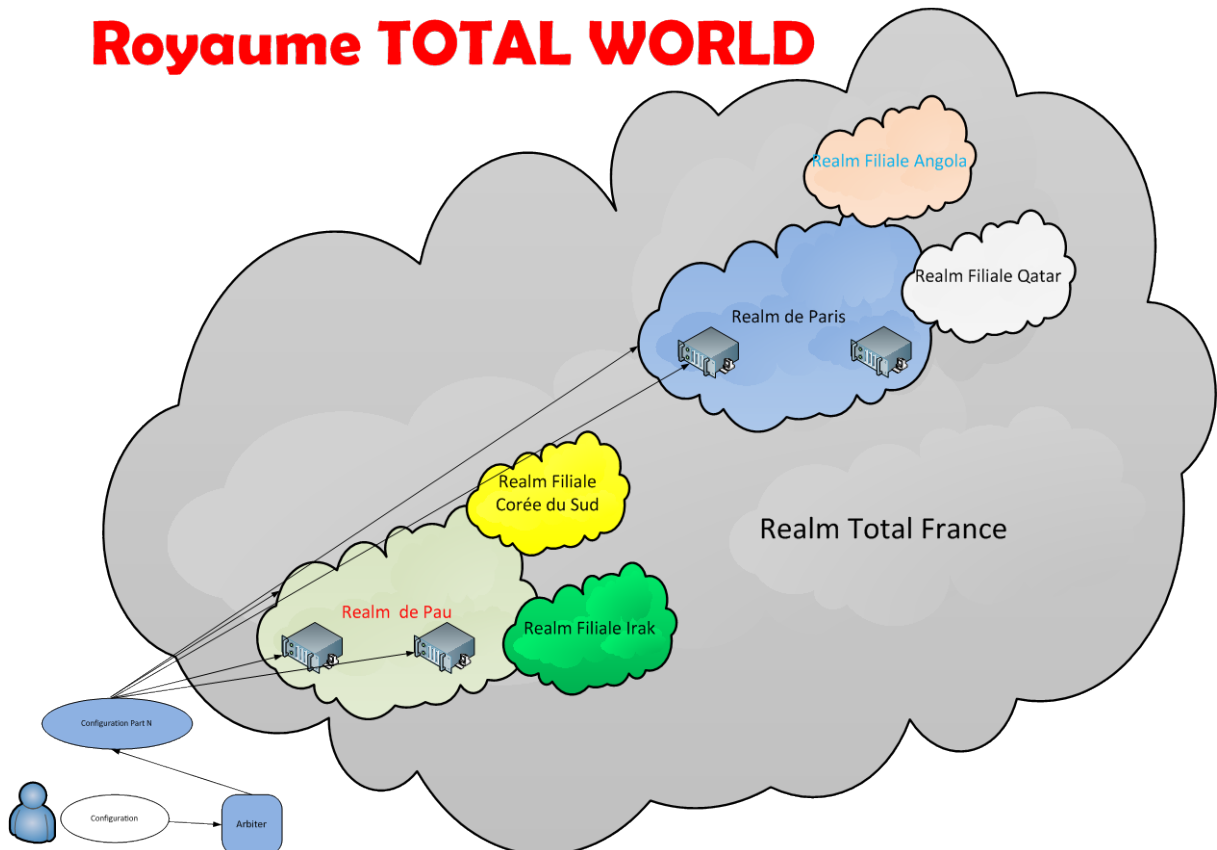


Schéma 26 – Mise en place des royaumes sur l'infrastructure TOTAL EP

L'infrastructure retenue est construite sur quatre serveurs, avec deux serveurs par royaume, sachant que les deux royaumes sont distants. Les rôles/services sont redondés deux fois. La solution Shinken repose sur plusieurs modules, c'est cette répartition de rôles/services sur plusieurs serveurs qui permet le mode Actif/Actif (Load Balancing) et Actif/Passif (Fail Over).

Disposant de deux serveurs de supervision sur chaque site, le cluster mode Actif/Actif (Load Balancing) se fait sur les rôles Scheduler, Poller et Receiver des serveurs primaires de Paris et de Pau afin d'alléger la charge de travail. Le mode cluster Actif/Passif (Fail Over) permet



d'assurer un service continu si les serveurs primaires rencontrent des problèmes.

En premier lieu, le serveur primaire de Paris est redondé par le serveur secondaire de Paris.

L'Arbiter, le Broker et le Reactionner sont en mode Actif/Passif (Fail Over) afin d'avoir un rôle en réserve qui peut prendre la main à n'importe quel moment si un rôle du serveur primaire est défaillant. Les trois rôles (Arbiter, Broker, Reactionner) du serveur primaire de Paris sont actifs et les rôles du serveur secondaire de Paris sont passifs.

Il ne peut avoir qu'un seul Arbiter primaire et un seul Arbiter secondaire, étant donné que l'Arbiter secondaire est sur le serveur primaire de Pau. On peut enlever de la configuration les autres Arbiter qui sont sur les serveurs secondaires. On commente donc l'Arbiter sur le serveur secondaire de Paris, il ne sera donc pas pris en charge dans la configuration de Shinken. Si l'Arbiter slave qui est sur le serveur primaire de Pau connaît une défaillance, on configurera cet Arbiter comme Slave. Ces six rôles se situent dans le realm « France ».

En second lieu, le serveur primaire de Pau est redondé par le serveur secondaire de Pau. L'Arbiter, le Broker et le Reactionner sont en mode passif (Fail Over) afin d'avoir des rôles en réserve qui peut prendre la main à n'importe quel moment si un rôle du serveur primaire de Paris ou de Pau est défaillant. Il faut souligner que l'Arbiter sur le serveur secondaire de Pau est également commenté et donc n'est pas pris en charge dans la configuration de Shinken. Si les Arbiter slave qui sont sur le serveur primaire de Pau et le serveur secondaire de Paris connaissent une défaillance, on configurera cet Arbiter comme Slave.

Ces six rôles se situent dans le realm « France ». (*Configuration de l'architecture Annexe 3*)

Après la présentation de l'architecture hautement disponible et performante mis en place sur l'infrastructure de TOTAL EP, nous vous expliquerons ce qu'apporte la Qualité de Service (SLA).

Schéma explicatif de la solution ci-dessous :

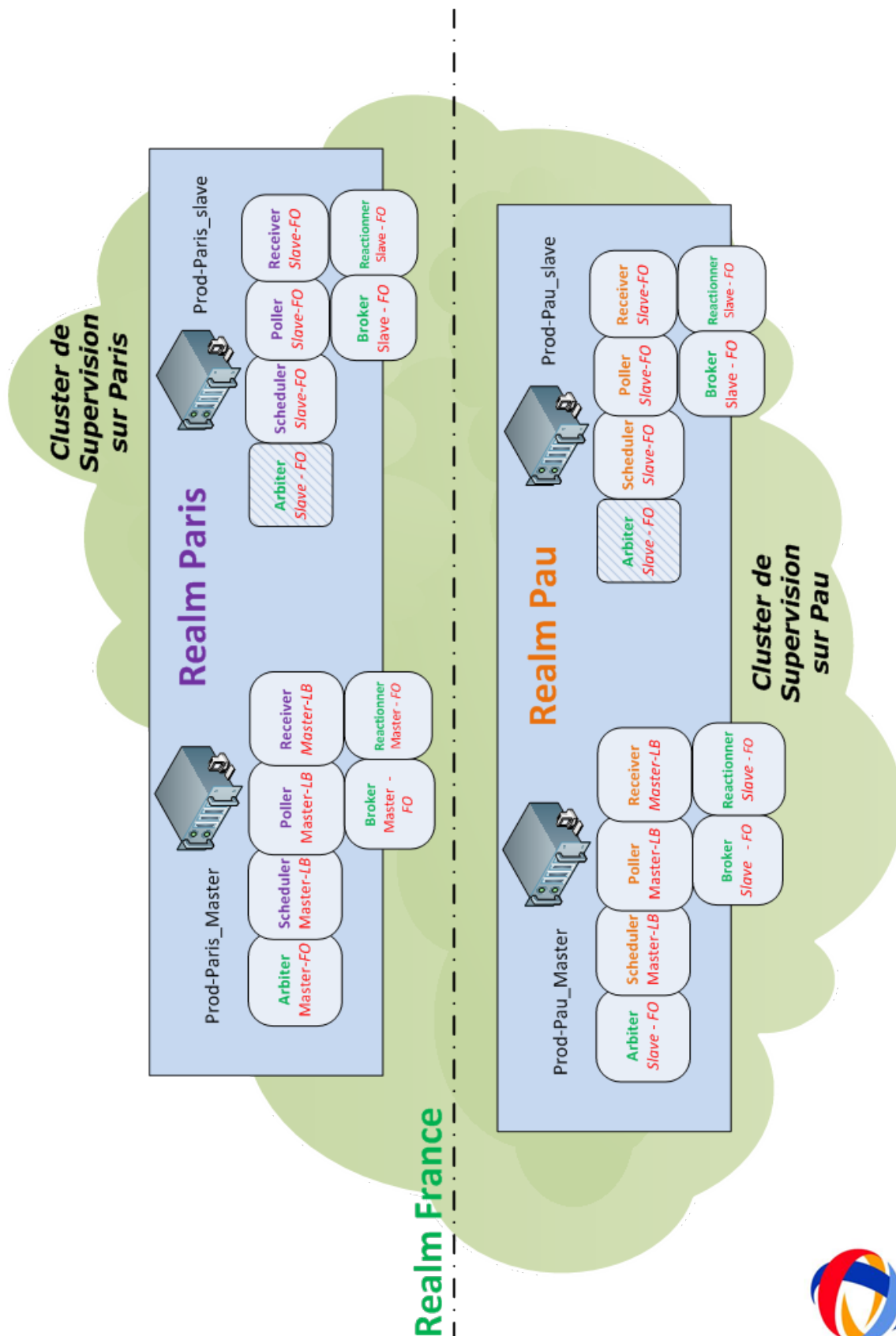


Schéma 27 – Solution retenue pour l'infrastructure de TOTAL EP





d. Aspect contractuel

Qualité de service SLA

Une solution complète de supervision (systèmes, réseaux, stockage, et applications) contrôle le suivi, le pilotage de la performance, de la qualité de leurs infrastructures et de leurs services informatiques. La supervision, le reporting sur la performance, la gestion d'alertes et le support proactif doivent apporter aux utilisateurs une compréhension complète des applications et de l'infrastructure associée, et aux clients une garantie sur les niveaux de services fournis.

Les applications sont l'essence même du service fourni par le système d'information aux utilisateurs finaux et aux métiers. La qualité de service, fournie aux utilisateurs par un outil informatique est évaluée via de multiples indicateurs. Généralement déterminés à partir d'un accord avec le client sur la qualité de service (Service Level Agreement - SLA), ceux-ci varient fortement d'une application à l'autre en fonction de sa spécificité et de sa criticité au regard de l'activité de l'entreprise.

Ces indicateurs sont généralement regroupés en trois grandes familles :

- La disponibilité du service,
- La performance de l'application,
- L'intégrité des données.

Outre l'intégrité des données, la supervision en tant que service de contrôle et d'assurance de service régulier, mesure la qualité de service fourni aux utilisateurs en termes de performance et de disponibilité des différents services.



Pour réduire la qualité de service (SLA), il faut obtenir les renseignements ci-dessous :

- Confirmer l'incident : dégradation des temps de réponse, indisponibilité totale ou partielle de l'application, impossibilité d'accès aux serveurs, renseigner la date et plage horaire du problème.
- Mesurer l'impact : nombre et pourcentage d'utilisateurs impactés, pages ou transactions longues ou inaccessibles.
- Isoler l'incident en cause : réseau, serveur applicatif, base de données, application, poste client ... (voir Business Impact de Shinken en Conclusion)
- Alerter les exploitants, les administrateurs de l'application, par messagerie électronique, mais également par le système de ticketing mis en place dans les entreprises pour suivi l'évolution de l'incident.

La supervision a pour but d'aider à identifier et résoudre des problèmes, pour pouvoir agir le plus rapidement possible au regard de l'urgence ou la criticité de celui-ci, en fonction de l'impact sur l'activité de l'entreprise et des risques encourus.

La supervision permettra donc de répondre au mieux aux renseignements ci-dessus. Dans la conclusion nous parlerons d'une fonctionnalité qui est le Business Impact, qui aide grandement à cibler la source de l'incident, mais également l'importance de l'application impactée.



Une SLA signifie « Service level agreement » en anglais et « convention de service » en français, mais plus communément appelée « Qualité de service ». Les SLA sont des contrats qui lient le client et le prestataire. Ce contrat définit la qualité de service minimum qu'un prestataire doit assurer à son client. Les accords en termes de qualité doivent être définis entre le client et le prestataire. Durant un contrat d'infogérance ou d'externalisation entre le client et le prestataire, le contrat SLA est le document référence qui juge la bonne qualité des services vendus.

Dans des contrats de SLA un taux de disponibilité inférieur à un taux souvent très élevé (par exemple 99.999%) se traduit par une pénalité financière au bénéfice du client. L'historique est également nécessaire pour prouver et mesurer le taux de disponibilité d'un serveur. Il est toujours nécessaire de garder les taux de disponibilités de ces serveurs par les systèmes de monitoring, de métrologie ou de logs.

L'historique permet, dans ce cas, d'éviter qu'un client tente d'abuser le prestataire de service en lui faisant croire que son serveur était indisponible alors que ce n'était pas le cas. L'historique permet également de calculer correctement la part financière qui doit être redistribuée au client en cas d'indisponibilité. Les SLA sont assimilés à des processus business.

Voici les chiffres que les clients ont souvent sous la main pour baser les SLA :

Rappel sur le pourcentage d'indisponibilité	Si disponible...	Indisponibilité annuelle
	90%	876 heures
	95%	438 heures
	99%	87 heures, 36 minutes
	99.9%	8 heures, 45 minutes, 36 secondes
	99.99%	52 minutes, 33.6 secondes
	99.999%	5 minutes, 15.36 secondes
	99.9999%	31.68 secondes



Les SLA englobent les cinq critères suivants :

- La Disponibilité dans le temps
- Performance d'accès
- Surveillance proactive
- GTI : Garantie de temps d'intervention
- GTR : Garantie de temps de rétablissement



Image 19 – Qualité de service SLA

Pour garantir une qualité de service (SLA : Service Level Agreement) maximal couplé avec une solution de supervision, il faut mettre en place des :

- ✚ Garantie de taux de disponibilité
On définit en moyenne une indisponibilité.
- ✚ Garantie de temps d'intervention (GTI) et de temps de rétablissement (GTR)
Un contrat permet de garantir une GTI et une GTR pour des criticités.



Dans les contrats de Qualités de services (SLA) entre les prestataires et les clients, il existe des contrats qui identifient des groupes de classe de service avec des caractéristiques et des priorités spécifiques. Ces classes de service constituent des catégories fondamentales où se baseront les qualités de service entre le prestataire et le client.

Les classes de services sont généralement :

- **Platinum**
- **Gold**
- **Silver**
- **Bronze**

Par exemple, le service Platinum est la priorité la plus haute et doit être résolu dans le plus bref délai. Le service Bronze est la priorité la moins importante, mais doit être résolu dans un délai convenable. Les applications métiers, autres applications ou services particuliers sont classés dans ces classes de service, pour différencier les applications importantes et moins importantes. Le temps de résolution des problèmes dépendra des classes de service.

Je vous présente les indicateurs de qualité au sein de la prestation SPIRIT envers TOTAL EP.

KPI – Indicateur clé de performance de TOTAL EP

KPI			Cible
Prise en compte des tickets incidents	90%	Taux de respect du délai de prise en compte des incidents pour les services Platinum	10 minutes
	90%	Taux de respect du délai de prise en compte des incidents pour les services Gold	15 minutes
	90%	Taux de respect du délai de prise en compte des incidents pour les services Silver	20 minutes
	90%	Taux de respect du délai de prise en compte des incidents pour les services Bronze	30 minutes
Délais de traitement des incidents	90%	Taux de respect du délai de traitement des incidents pour les services Platinum	2 heures
	90%	Taux de respect du délai de traitement des incidents pour les services Gold	4 heures
	90%	Taux de respect du délai de traitement des incidents pour les services Silver	8 heures
	90%	Taux de respect du délai de traitement des incidents pour les services Bronze	24 heures



Conclusion

Un outil de supervision a pour but premier de surveiller tous les autres outils et autres serveurs. Il se doit d'être le plus résistant possible ; afin d'optimiser une qualité de service ainsi que la productivité de l'entreprise pour qu'elle ne soit pas impactée dans un incident informatique.

Les solutions de supervisions évoluent de jour en jour. Mais les solutions de supervision qui évoluent le plus rapidement sont les solutions open source, elles ont rattrapé le retard qu'elles avaient face aux solutions propriétaires ces dernières années. Maintenant elles évoluent plus rapidement que celles-ci grâce à des communautés qui les font évoluer chaque jour.

Shinken montre par exemple une certaine stabilité, performance et son atout primaire est l'évolutivité.

Les clusters de supervision sont la suite logique de l'évolution de la supervision. Les solutions propriétaires et open sources l'ont bien compris et intègrent dorénavant cette fonctionnalité dans leurs outils de supervision.

Il ne faut pas oublier que la mise en place d'une architecture de cluster de supervision dépend des besoins et des contraintes du client.

Après avoir vu différentes architectures de cluster de supervision sur plusieurs infrastructures, dont une architecture de cluster optimum et plus complexe au sein de l'infrastructure de TOTAL EP.

L'étude de cette architecture fût complexe et compliquée suite aux problèmes et contraintes rencontrés.

Cependant nous avons pu concevoir des architectures qui répondent aux besoins de nos différents clients et ainsi garantir au maximum une qualité de service.

Avant de conclure, je vais vous présenter deux évolutions de la supervision qui sont en pleine expansion actuellement ou qui vont devenir des fonctionnalités très demandées à court terme.



La supervision continue son évolution

BUSINESS IMPACT

La plupart des administrateurs accordent de l'importance à leur infrastructure. Cependant il faut dépasser la vision purement technique, cette vision devient petit à petit dépassée.

Maintenant on attend qu'un service informatique gère son système avec une vision « Business ». Il ne faut pas oublier que c'est la production de l'entreprise, qui fait vivre l'entreprise et ses salariés.

Il faut donc tout mettre en œuvre pour maintenir cette production, et cette fameuse vision « Business » qui intègre petit à petit les services, car l'informatique est devenue incontournable dans les entreprises depuis plus d'une dizaine d'années.

L'infrastructure informatique sert à maintenir la production d'une entreprise. Un serveur a de l'importance que si l'application qui y tourne dessus a de l'importance. Car le plus important c'est que l'utilisateur puisse faire son travail correctement. Si une instance application est défaillante, mais qu'une autre prend le relais, l'utilisateur de cette entreprise ne remarquera rien et la production continuera à fonctionner.

Si un problème informatique survient et que les utilisateurs sont impactés dans cet incident, les administrateurs feront tout pour résoudre au plus vite cet incident, mais le responsable des administrateurs ne comprendra pas et ne cherchera pas à comprendre la source du problème.

Il demandera qu'une seule chose, si « le service rendu à l'utilisateur est à nouveau disponible ? »

Certains administrateurs prennent quelques minutes pour résoudre cet incident, car ils connaissent parfaitement leur architecture. Ce n'est pas le cas de tous les administrateurs qui pendant ce temps effectuent une vérification de chaque élément pour trouver la source



du problème, il n'est donc pas en train de le résoudre. Et cela est vraiment problématique, car il y a une perte de temps qui peut impacter la productivité de l'entreprise.

Afin d'éviter de perdre trop de temps pour cibler le problème exact de l'incident, Shinken permet une belle évolution de la supervision et fournit automatiquement un indicateur de disponibilité d'applications importantes pour le métier. Il est possible d'avoir une vue simple et de cibler directement le problème. Cette solution est le « Business Impact » qui est géré nativement par Shinken.

Avec le Business Impact, on peut identifier directement la source du problème quelque qu'il soit.

Par exemple, pour un problème réseau, il trouve directement la défaillance au niveau du firewall.



Schéma 28 – Exemple Business impact – Défaillance Firewall

Une seule alerte remonte à l'interface de Shinken, c'est l'alerte la plus pertinente en ciblant l'équipement défaillant et en détaillant l'erreur rencontrée.



La fonctionnalité Business Impact peut trouver l'incident en remontant à la source même en ayant plusieurs dépendances logiques.

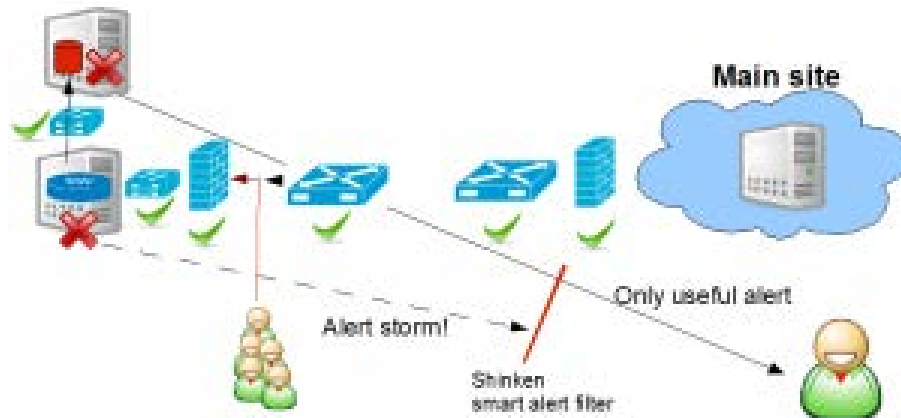


Schéma 29 – Exemple Business impact – Défaillance Base de données

Une seule alerte remonte à l'interface de Shinken, l'alerte remonte que la base de données du site internet est défectueuse, et cette erreur met le site internet hors service.

Avec tous les services réels de chaque application et serveur, les administrateurs peuvent être vite perdus, cependant Shinken a mis en place un paramétrage de business impact pour classer les applications.

Dans la configuration, le paramètre business impact indique un ordre d'importance. Les ordres d'importance vont de 0 (pas d'importance) à 5 (importance maximum). La valeur par défaut est de 2 (importance moyenne).

Si nous prenons une application web d'une entreprise, elle est certainement très importante, mais elle ne le sera pas autant qu'un ERP ou qu'une application métier propre à l'entreprise.

Il sera donc crucial de se focaliser sur les applications les plus importantes pour le business de l'entreprise.



Un exemple d'interface où apparaît la vue « Impacts » de Shinken.

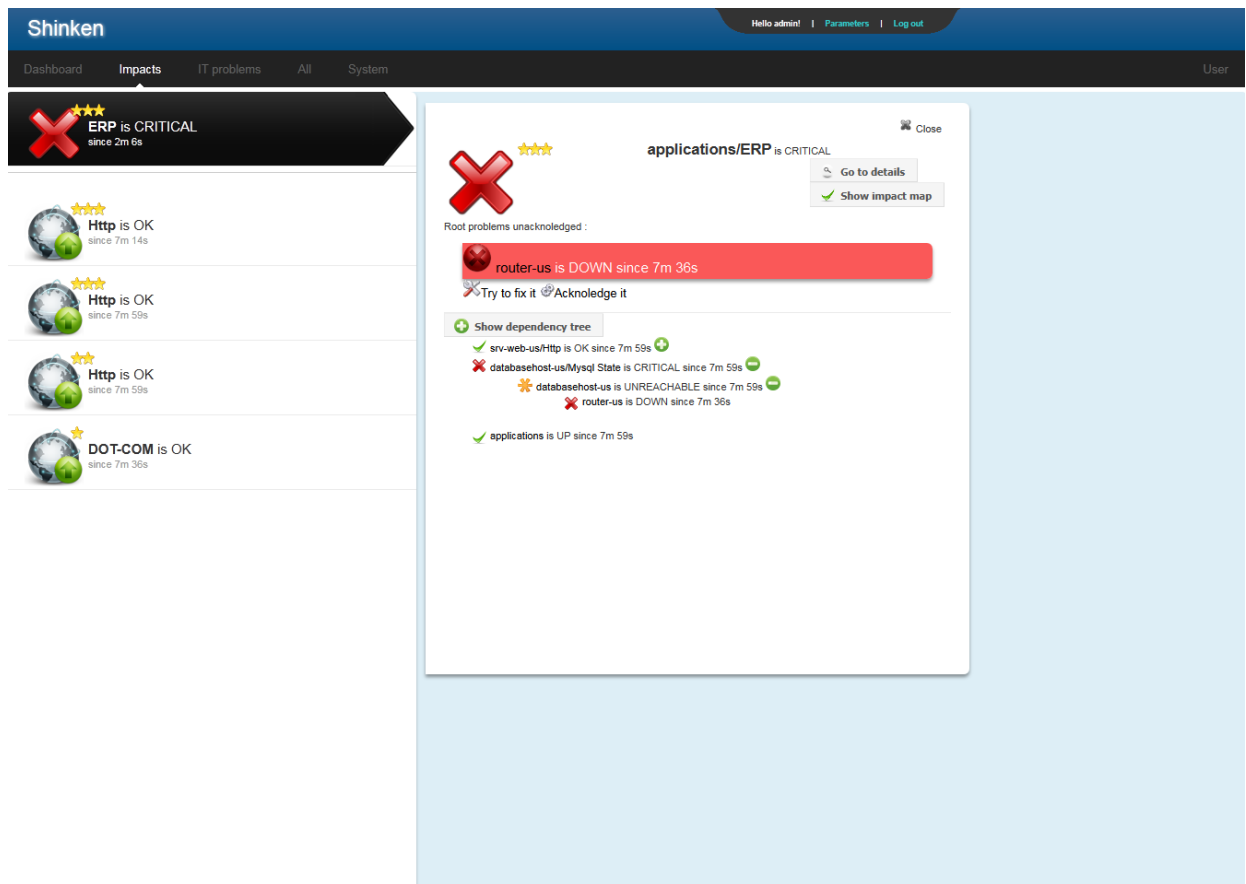


Image 20 – Interface Business Impact sur Shinken

Nous remarquons que l'application ERP est critique, et de plus l'application ERP est de la plus haute importance, il est donc impératif de la rendre disponible aux utilisateurs le plus rapidement possible.

Nous voyons qu'un routeur est défaillant et cela entraîne une perte de connectivité sur une base MySQL.

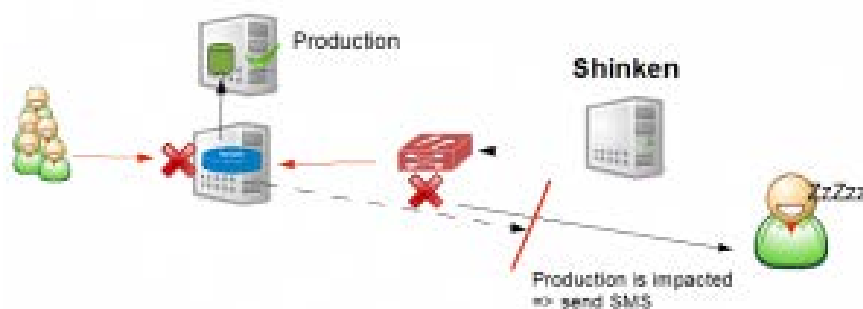
Shinken définit l'importance des applications comme nous l'avons vu précédemment, mais Shinken peut filtrer les notifications entre la Production et la Qualification et avertit les administrateurs de l'erreur en suivant par SMS.



Schémas 31 – Défaillance rencontrée sur l'infrastructure Qualification

Par exemple un switch est la source d'un problème et son impact peut-être critique, cependant cela reste de la qualification, il n'est donc pas nécessaire de réveiller l'astreinte des administrateurs.

Shinken n'envoie pas d'alerte par SMS aux administrateurs, mais affichera cette erreur sur l'interface des impacts.



Schémas 32 – Défaillance rencontrée sur l'infrastructure Production

La source du problème est toujours un switch, son impact est critique, car cela touche la production, il est donc nécessaire d'alerter l'astreinte par SMS pour que l'administrateur règle le plus rapidement ce problème technique.



APPLICATION DE SUPERVISION SUR SMARTPHONE

Shinken et les autres solutions de supervisions commencent à se tourner vers la technologie du mobile pour alerter les administrateurs si la production est mise en péril, et par des Smartphones pour superviser leur infrastructure informatique depuis une application.

Avec l'utilisation croissante des smartphones, les développeurs ressentent l'évolution de la supervision depuis une application de smartphone. Les administrateurs ont accès instantanément à la supervision de l'infrastructure de leur entreprise à distance et à n'importe quel endroit.

Du côté de Shinken, Jean Gabès (Fondateur de Shinken) et ses développeurs sont en train d'implémenter cette fonctionnalité dans la solution Shinken.

Suite à un de ces articles sur « le blog de nicolargo », il dit au sujet d'une application de supervision :

[L'interface sera ainsi bien plus proche d'une interface "Smartphone" que d'une UI classique d'un outil d'administration, ce qui explique pourquoi elle ne va pas plaire à tout le monde également. Donc pas de soucis, les tablettes et Smartphones sont prévus 😊 D'un point de vue technique, je ne vois pas l'intérêt de l'application native quand on a toutes les possibilités d'HTML5 sous la main et que l'on n'a pas besoin de prendre en photo son utilisateur. Il est donc à prévoir d'ailleurs une non-compatibilité avec IE6 ou 7, et là clairement ils ne sont pas dans le cahier des charges. Je ne serai pas contre un patch (non intrusif...) qui permettrait de les supporter, mais ces navigateurs étant tellement mal faits et limitatifs dans l'interface qu'il serait non efficace de tenter de les gérer.]

Les applications de supervision sur les Smartphones, sont une petite évolution de la supervision qui peut être très utile pour les administrateurs où qu'ils soient.



Je viens de vous énumérer les prochaines évolutions de la supervision avec le Business Impact et les applications supervision sur les Smartphones. Cependant ce n'est pas le sujet principal de mon mémoire. La supervision est en constante évolution, et c'est cela qui rend cette technologie tellement intéressante et primordiale pour les entreprises de nos jours.



English Part

Introduction

The information technology is the backbone of the enterprise.

The monitoring is very important for business since years. The monitoring brings an overview of infrastructure of the enterprise but also the high availability of all business applications.

The monitoring consists to implement the indicators to have an overview of the status of an infrastructure.

The monitoring solutions permit generally:

- Check the states an host and service
- Alert incident
- Detect failures

The monitoring permits to have live information about infrastructure.

The monitoring activities are:

- Monitor
- View
- Analyze
- Driving
- Acting
- Alert

The monitoring tools are good to administrate an Information System. For improving the quality of service and enterprise productivity, the monitoring solution perfectly meets the needs.



Case Study

A Study on open-source solutions and commercial solutions was done to find the best monitoring solution. We have selected a new monitoring tool to improve the quality of service of the business which costs nothing.

The selected tool is the open-source solution and an evolution of the monitoring reference which is Nagios.

The tool takes all advantages for Nagios and implements new features.

Shinken is the selected monitoring tool because it's a new monitoring tool and a new project of monitoring very interesting, which brings many new features and possibility of evolution.

The Shinken project is a complete redesign of the Nagios kernel, the monitoring reference since 10 years. The redesign of the Nagios kernel was made in a computer language called 'Python', giving it a new and more flexible architecture and easier to maintain than the monolithic daemon of his older brother.

Shinken offers flexible monitoring and many capabilities.

Shinken give new features:

- High availability and High performance (Monitoring cluster)
- Business impact

These new features are very benefic for the small and large infrastructures. Shinken was designed for all types of architecture but also for the large architectures with simplicity of configuration.

The High Availability, high performances and Business Impact are the monitoring evolution.



High Availability and High Performance (Cluster)

The high availability is a service which delivers a maximum availability rate. The availability is very important today for enterprise with vital business applications. In case of unavailability, the impact can be massive if there are applications that affect the production of the enterprise.

The high availability needs architecture adapted and structured to implement. To get the highest possible availability, it needs a redundancy of the infrastructures equipments but also to implement a cluster.

The cluster is a structure composed of multiple hosts, where all of them are able to operate independently of other.

The independent servers operate grouped in a single cluster. The client doesn't see the difference between a single machine and a cluster.

The clusters are useful to minimize the impact of a server failure and the availability of a business application.

When a failure on a cluster server occurs, the cluster reacts by activating another server in the cluster to limit downtime.

Monitoring Cluster

To have a good monitoring solution of the infrastructure you need the monitoring tool to be available at all time (highest availability of all the systems in your infrastructure)

The new monitoring tool needs the same requirements as the classical monitoring:

- The Load
- The High Availability
- Configuration Management
- A loss of remote site

The monitoring cluster can use two types of clustering:



- The Cluster “Fail Over” it’s a live data replication and manages failover if a malfunction occurs.

Functionality:

- Primary Server which performs the work of monitoring
 - Secondary Server which one takes it over when the primary server doesn’t work anymore.
- The Cluster “Load Balancing” does load balancing between the configured servers and manages failover if a malfunction occurs.

Functionality:

- Primary Server which performs the work of monitoring
- Secondary Server which shares the work of monitoring with the primary server, but completely takes the over if the primary server fails.

The monitoring cluster permits to have a permanent vision on infrastructure at all time.

This allows a very high reactivity on computing incidents.

Monitoring Cluster with Shinken

Shinken’s architecture offers the high availability and high performance with two modes of clusters “Load Balancing” and “Fail Over”. The administrator manage a single configuration for any infrastructure, the system automatically cuts the configuration in several parts (as much parts as installed modules) and sends them to different modules.

Shinken has cut the monitoring process in several roles:

- Check the configuration ([Arbiter](#))
- Manage and Schedule control ([Scheduler](#))
- Launch of check by the probes ([Poller](#))
- Sends notifications if necessary ([Reactionner](#))
- Export / Backup / Presentation of data interfaces ([Broker](#))
- Listening and manages passive information ([Receiver](#))



To have high availability and high performance, the system needs the different roles to be multiplied.

Shinken keeps roles in spare in case of one of the active roles will fail but it also distribute the load between these roles in case of overload.

To improve the high availability on the architecture, after the multiplication of roles, we add a notion of realm which can be multiplied by splitting those services at different sites.

Study of highly available architectures

To start implement the architecture of high availability, you first need to identify the needs and constraints of the company. With this information, we can define the perfect architecture for the enterprise.

Simple and high available architectures can be implemented but also much more complex architectures with a percentage of maximum availability. Watch the memory to see the different cases of architectures.

Conclusion

A monitoring tool is primarily intended to monitor all others tools and other servers. It must be the most resistant tool within the architecture, to optimize the service of quality and productivity of the company. The monitoring solutions evolve from day to day. The monitoring Cluster is the evolution of monitoring.



Glossaire

Solution open-source :

L'Open Source signifie « Libres de Droits » ce qui indique qu'un logiciel Open Source est à priori accessible dans l'intégralité de son code source. L'Open Source est un *monde de partages où chacun peut apporter une pierre à l'édifice pour améliorer encore et toujours les solutions retenues*.

Les solutions Open Source révolutionnent donc le marché du logiciel. L'intérêt majeur est leur évolutivité au sein de la communauté qui les entretient.

Solution propriétaire :

Un logiciel propriétaire est écrit, la plupart du temps, par une entreprise. Il est distribué uniquement sous forme binaire (sans source). On ne connaît pas sa recette, car elle reste la propriété de l'entreprise. Souvent, il est disponible sous une licence d'utilisation assez restrictive. Par exemple, si vous achetez un ordinateur fourni avec Microsoft Windows, vous n'aurez le droit de l'utiliser uniquement sur cet ordinateur. Si vous décidez de mettre à jour un ancien ordinateur avec votre nouveau Windows, vous serez donc contraint d'acheter une nouvelle licence. Ces licences restrictives placent les utilisateurs dans un état de division. Ceux qui les violent en faisant des copies pour ses amis sont appelés "pirates".

Du fait que ces entreprises gardent leurs secrets de fabrication, les développeurs et les utilisateurs (nous) sont également divisés. Les développeurs ont le pouvoir de faire leurs logiciels comme ils le veulent, et les utilisateurs n'ont comme seul choix d'accepter ou de refuser ces logiciels. Non seulement vous n'aurez jamais le droit de les modifier, mais vous ne pourrez même pas étudier leur fonctionnement

Plug-In :

Un plugin (ou greffon) est un logiciel qui sert à étendre les fonctionnalités du programme de base. De par leur nature même, les plugins ne peuvent (en grande majorité) fonctionner seuls. Les plug-ins sont écrits spécifiquement pour leur programme hôte, et doivent de ce fait respecter des règles précises afin de communiquer avec cette dernier

Licence GPL :

Soit Licence Publique Générale GNU Version 2, Juin 1991

La copie et la distribution de copies exactes de ce document sont autorisées, mais aucune modification n'est permise.

NRPE, NSClient :

NRPE (Nagios Remote Plugin Executor) permet d'exécuter à distance sur un pc sous Linux les commandes Nagios. La partie « serveur » de NRPE doit être installée sur le pc à superviser, et elle se chargera de lancer, en local sur ce pc, les plugins concernés. Ce serveur NRPE obéit lui-même aux instructions qu'il reçoit du serveur hébergeant Nagios, où s'exécute



check_nrpe, qui transmet les commandes de Nagios. NSClient est un agent identique à NRPE sauf qu'il est dédié pour les hôtes windows.

Fork :

Programme développé à partir des sources d'un autre logiciel

Licence AGPL :

C'est une licence libre dérivée de la Licence publique générale GNU avec une partie supplémentaire couvrant les logiciels utilisés sur le réseau.

Elle a été écrite par Affero pour autoriser les droits garantis par la GPL à couvrir les interactions avec des produits propriétaires à travers un réseau comme Internet, ce que la GPL ne fait pas.

Langage Python :

Python est un langage de script de haut niveau, structuré et open source. Il est multi-paradigme et multi-usage.

Conçu pour être orienté objet, il n'en dispose pas moins d'outils permettant de se livrer à la programmation fonctionnelle ou impérative; c'est d'ailleurs une des raisons qui lui vaut son appellation de « langage agile ».

Parmi les autres raisons, citons la rapidité de développement (qualité propre aux langages interprétés), la grande quantité de modules fournis dans la distribution de base ainsi que le nombre d'interfaces disponibles avec des bibliothèques écrites en C, C++ ou Fortran. Il est également apprécié pour la clarté de sa syntaxe, ce qui l'oppose au langage Perl.

RRDTool :

RRDtool est un outil de gestion de base de données RRD (round-Robin database) créé par Tobi Oetiker. Il est utilisé par de nombreux outils open source, tels que Cacti, collectd, Lighttpd, et Nagios, pour la sauvegarde de données cycliques et le tracé de graphiques, de données chronologiques. Cet outil a été créé pour superviser des données serveur, telles la bande passante et la température d'un processeur. Le principal avantage d'une base RRD est sa taille fixe.

RRDTool inclut également un outil permettant de représenter graphiquement les données contenues dans la base.

RRDTool est un logiciel libre distribué selon les termes de la GNU GPL.

**SNMP :**

SNMP signifie Simple Network Management Protocol (traduisez *protocole simple de gestion de réseau*). Il s'agit d'un protocole qui permet aux administrateurs réseau de gérer les équipements du réseau et de diagnostiquer les problèmes de réseau.

PHP :

Le langage PHP est un langage de programmation web côté serveur, ce qui veut dire que c'est le serveur qui va interpréter le code PHP (langage de scripts) et générer du code HTML qui pourra être interprété par votre navigateur.

Le php permet d'ajouter des fonctionnalités de plus en plus complexes, d'avoir des sites dynamiques, de pouvoir gérer une administration de boutique en ligne, de modifier un blog, de créer des réseaux sociaux...

Le php fut créé en 1994 par Rasmus Lerdorf, c'est un langage libre et gratuit, avec une grande communauté mondiale

Langage C :

Le C est un langage de programmation de bas niveau très populaire, crée dans les années 1970 par D.Ritchie et B.W.Kernighan. Il est portable, libre, faiblement typé (peu de types de variables différents : son fonctionnement est donc proche de l'ordinateur (gain en rapidité), mais un brin plus difficile à manipuler pour le programmeur). Le C n'est sans doute pas le langage le plus facile à apprendre (notamment à cause de l'adoption du concept parfois un peu obscure des pointeurs), ni le plus récent, mais ses qualités font de lui un langage incontournable en matière de programmation.

Templates :

Template est un anglicisme utilisé en informatique pour désigner un modèle de conception de logiciel ou de présentation des données. On parle aussi de « patron » comme en couture, de gabarit ou de kit graphique.

Format XML :

Le XML, acronyme de eXtensible Markup Language (qui signifie: langage de balisage extensible), est un langage informatique qui sert à enregistrer des données textuelles. Ce langage a été standardisé par le W3C en février 1998 et est maintenant très populaire. Ce langage, grosso-modo similaire à l'HTML de par son système de balisage, permet de faciliter l'échange d'information sur l'internet.

Contrairement à l'HTML qui présente un nombre fini de balises, le XML donne la possibilité de créer de nouvelles balises à volonté.

**SLA :**

Accords sur la qualité de service. Ils sont définis contractuellement sur des points précis avec des pénalités en cas d'incident. Les SLA ne peuvent être définis et respectés qu'avec des outils permettant de surveiller et de mesurer efficacement les engagements.

Proxy :

Le proxy est un ordinateur faisant office de passerelle entre le réseau d'un particulier ou d'une entreprise et Internet.

Machines virtuelles :

Logiciel installé dans un ordinateur et permettant de simuler le fonctionnement d'un dispositif matériel qui, normalement, ne pourrait pas fonctionner avec cet ordinateur. Grâce à une machine virtuelle, on peut, par exemple, simuler l'utilisation d'un processeur, ou d'une console de jeux sur un PC.

Cloud :

Le Cloud Computing est l'accès via un réseau de télécommunications, à la demande et en libre-service, à des ressources informatiques partagées configurables.

SAP :

C'est un logiciel d' ERP (Engineered Ressource Planner ou Product), en français PGI (Progiciel de Gestion intégrée). Il s'agit d'une suite logicielle pour entreprise où tous les modules sont reliés entre eux : gestion des stocks, comptabilité, facturation, paye...

DMZ :

Zone tampon d'un réseau d'entreprise, située entre le réseau local et Internet, derrière le coupe-feu, qui correspond à un réseau intermédiaire regroupant des serveurs publics (HTTP, SMTP, FTP, DNS, etc.), et dont le but est d'éviter toute connexion directe avec le réseau interne et de prévenir celui-ci de toute attaque extérieure depuis le Web.

LAN :

Réseau local de petite ou grande taille qui connecte plusieurs ordinateurs d'une même organisation.

Pare-feu :

Un pare-feu est un logiciel ou un matériel qui consiste à protéger un ordinateur d'intrusion venant d'autres machines.

Langage PERL :

Perl est un langage de programmation interprété très adapté à la manipulation de chaînes de caractères.

**Spare :**

Pièce d'équipement ou logiciel en réserve.

SMTP :

SMTP (pour Simple Mail Transfer Protocol ou Protocole Simple de transfert de Courrier) est un protocole de communication introduit dans les années 80 et utilisé lors de l'adressage des courriers électroniques sortants

Web UI :

Interface Web d'une application

AS 400 :

AS/400 est une gamme de mini-ordinateurs IBM apparue début février 1987

GTI :

La GTI est une garantie certifiant que l'intervention aura commencé dans le délai défini par cette dernière

GTR :

La GTR est un délai contractuel dans lequel on s'engage à rétablir un service interrompu.



Bibliographie

Sites Internet :

<http://www.monitoring-fr.org/supervision/>

<http://www.nagios.org/>

<http://www.shinken-monitoring.org/>

<http://wiki.monitoring-fr.org/>

<http://Nagios-exchange.fr>

<http://wiki.monitoring-fr.org/shinken>

<http://doc.ubuntu-fr.org/shinken>

<http://www.gabes.fr/jean/2009/10/06/shinken-les-grandes-lignes/>

<http://www.gabes.fr/wiki/index.php?title=Nagios>

http://www.telecom-valley.fr/sites/default/files/files/Shinken-J_%20gabes.pdf

[http://www.gabes.fr/wiki/index.php?title=Nagios#La supervision .C3.A0 haute disponibilit .C3.A9](http://www.gabes.fr/wiki/index.php?title=Nagios#La_supervision_.C3.A0_haute_disponibilit_.C3.A9)

<http://blog.nicolargo.com/2011/08/interview-de-jean-gabes-le-createur-de-shinken.html>

<http://www.shinken-monitoring.org/features/focus-on-business-critical-root-problems/>

<http://www.shinken-monitoring.org/wiki/official/part-problemsandimpacts>

http://www.journaldunet.com/expert/cgi/expert/impression_article.php?f_id_article=49340

Livre :

Le linux hors série sur Shinken a également été une grande source d'information.

Nagios 3 ; pour la supervision et la métrologie



Illustrations

Images :

- Image 1 – Interface Nagios – Page 13
- Image 2 – Interface Centreon – Page 15
- Image 3 – Interface Shinken – Page 17
- Image 4 – Interface Cacti – Page 18
- Image 5 – Interface Zabbix – Page 21
- Image 6 – Interface Netcrunch – Page 22
- Image 7 – Interface Traverse – Page 23
- Image 8 – Interface Tivoli Monitoring – Page 25
- Image 9 – Interface HP OpenView – Page 26
- Image 10 - Shinken et ses Nombres des lignes de code – Page 29
- Image 11 - Nagios et ses Nombres des lignes de code – Page 30
- Image 12 – Interface Thruk – Page 39
- Image 13 – Taux de disponibilité – Page 40
- Image 14 – Exemple de cluster – Page 42
- Image 15 – Exemple de panne d'un cluster – Page 43
- Image 16 – Sites distants de CACF – Page 67
- Image 17 - Vision globale en temps réel sur tous les sites de Total EP – Page 71
- Image 18 – Sites distants de TOTAL EP – Page 72
- Image 19 – Qualité de service SLA – Page 81
- Image 20 – Interface Business Impact sur Shinken – Page 87

Schémas :

- Schéma 1 – Fonctionnement des rôles de Shinken – Page 32
- Schéma 2 – Fonctionnement de l'Arbiter – Page 33
- Schéma 3 – Fonctionnement du Scheduler – Page 34
- Schéma 4 – Fonctionnement du Poller – Page 34
- Schéma 5 – Fonctionnement du Reactioner – Page 35
- Schéma 6 – Fonctionnement du Broker – Page 35



- Schéma 7 – Fonctionnement du Receiver – Page 36
- Schéma 8 – Fonctionnement du Poller avec les agents – Page 36
- Schéma 9 – Fonctionnement du Poller avec les traps – Page 37
- Schéma 10 – Cluster Actif/passif (Fail-Over) – Page 43
- Schéma 11 – Cluster Actif/Actif (Load-Balancing) – Page 44
- Schéma 12 – Cluster de supervision Actif/passif (Fail-Over) – Page 46
- Schéma 13 – Cluster de supervision Actif/Actif (Load-Balancing) – Page 47
- Schéma 14 – Fonctionnement concrètement des rôles de Shinken – Page 49
- Schéma 15 – Fonctionnement des royaumes – Page 51
- Schéma 16 – Fonctionnement avec plusieurs royaumes – Page 52
- Schéma 17 – Mise en place d'une architecture hautement disponible avec multiplication des modules (spare) – Page 53
- Schéma 18 – L'arbitre détecte une défaillance et envoie la nouvelle configuration au Poller « spare » – Page 54
- Schéma 19 – Création de deux royaumes non communs « France et Espagne » – Page 59
- Schéma 20 – Création d'un royaume commun « Europe » avec deux sous royaume « France et Espagne » – Page 61
- Schéma 21 – Fonctionnement du découpage de la configuration par l'Arbitre – Page 62
- Schéma 22 – Fonctionnement d'un cluster de supervision sur site distant – Page 63
- Schéma 23 – Solution retenue pour une architecture classique – Page 66
- Schéma 24 - Mise en place des royaumes sur l'infrastructure CACF – Page 69
- Schéma 25 – Solution retenue pour l'infrastructure de CACF – Page 70
- Schéma 26 – Mise en place des royaumes sur l'infrastructure TOTAL EP – Page 74
- Schéma 27 – Solution retenue pour l'infrastructure de TOTAL EP – Page 76
- Schéma 28 – Exemple Business impact – Défaillance Firewall – Page 85
- Schéma 29 – Exemple Business impact – Défaillance Base de données – Page 86
- Schéma 30 – Les différentes vues « Technique et métiers » – Page 88
- Schémas 31 – Défaillance rencontrée sur l'infrastructure Qualification – Page 89
- Schémas 32 – Défaillance rencontrée sur l'infrastructure Production – Page 89