

Rapport de stage

Monitoring et supervision

Mathieu Dreyer

Année 2018-2019

Stage de 1^e année réalisé dans l'entreprise *LMI Solutions*



Maître de stage : *Rémi Santato*

Encadrant universitaire : *Olivier Airaud*

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Dreyer Mathieu

Elève-ingénieur(e) régulièrement inscrit(e) en 1^{ère} année à TELECOM Nancy

N° de carte d'étudiant(e) : 1215022015D

Année universitaire : 2018 - 2019

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Monitoring et supervision

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Vandœuvre-lès-Nancy le 13/05/ 2019

Signature : Mathieu Dreyer

Table des matières :

Table des matières

1.Introduction	5
2.LMI Solutions et mon rôle au sein de l'entreprise	6
2.1La supervision à LMI Solutions.....	6
2.2Description de Zabbix	7
2.3Fonctionnement de Zabbix	8
2.4Installation de Zabbix	9
2.5Les alertes Zabbix.....	10
3.Mon travail sur Zabbix	13
3.1Création des différents Zabbix locaux.....	13
3.2Résolution du problème sur les triggers	17
3.3Proposition de solution face au problème.....	17
Evaluation de la solution apportée	19
Résumé	20
Abstract (250 words max)	20

1. Introduction

Pour satisfaire au mieux les clients, LMI Solutions a pour but d'agir vite en cas de panne, ou de problème sur une machine chez un client.

Pour réussir cela, il faut donc un système de supervision, pour pouvoir être alerté en temps réel, des dysfonctionnements, ou des problèmes rencontrés pour ensuite pouvoir éviter la panne de la machine.

La question qui se pose est comment pouvoir récupérer leurs informations, de manière instantanée, chez les différents clients ?

Pour ce faire, LMI Solutions a mis en place un système de monitoring, qui permet de récupérer les données d'un hôte. Cependant, il faut installer le client sur chaque hôte que l'on souhaite monitorer. La principale contrainte est que lorsque, par exemple, il y a une coupure de courant chez un client, toutes les machines, ne répondent plus, donc on reçoit un nombre élevé d'alertes, alors qu'une seule signalant une coupure de courant suffit.

Dans ce rapport, je vais vous décrire la mission qui m'a été confiée, les problèmes que j'ai rencontrés, et les solutions apportées sur cette mission. Ensuite ce rapport traitera des alternatives explorées, et d'un regard critique de la solution apportée.

.

2. LMI Solutions et mon rôle au sein de l'entreprise

LMI Solutions est une entreprise, qui fournit différents services informatiques. Elle se positionne comme le partenaire de référence pour déployer tout le potentiel technologique au service des entreprises et des collectivités ou administrations.

C'est une Société à responsabilité limitée, au capital de 50 000 euros, dirigée par Frédéric Declé. C'est une petite et moyenne entreprise, qui emploie entre 10 et 19 personnes. L'entreprise dispose de deux agences, une située à Millery et la seconde à Reichstett.

Son domaine géographique d'action est principalement basé dans le Grand Est, dans le but de pouvoir agir vite sur place en cas de problème.

Mon rôle au sein de l'entreprise est de pouvoir effectuer plusieurs tâches, pour décharger mes collègues. J'ai principalement travaillé sur Zabbix, qui sera traité dans la suite de ce rapport.

2.1 La supervision à LMI Solutions

La supervision à LMI Solutions se fait via le biais de Zabbix. C'est un outil permettant de récupérer en temps réel des informations sur les différents hôtes, pour ensuite avertir l'utilisateur, des différents dysfonctionnements, quels qu'ils soient (légers, important, grave...).

Lors de mon arrivée, j'ai d'abord dû comprendre comment fonctionnait Zabbix, pour ensuite pouvoir répondre à la problématique.

2.2 Description de Zabbix

Le fonctionnement de Zabbix, se divise en 4 parties, qui sont le Server Zabbix, la partie Zabbix frontend, le Zabbix proxy et enfin l'agent Zabbix (Zabbix agent).

En premier, nous avons la partie Server Zabbix qui permet une surveillance à distance (et en local) du bon fonctionnement de différents services systèmes et réseaux. Pour récupérer ses informations, le Server Zabbix a besoin d'agents. Il peut très bien fonctionner sans, mais le nombre d'informations sera beaucoup plus limité.

Ensuite, nous avons la partie Zabbix frontend, qui permet de visualiser toutes ces informations. Il s'agit d'une interface web, qui est accessible depuis n'importe quel poste à condition d'avoir un accès à internet.

Voici une capture d'écran de l'interface web mise en place au sein de LMI Solutions.

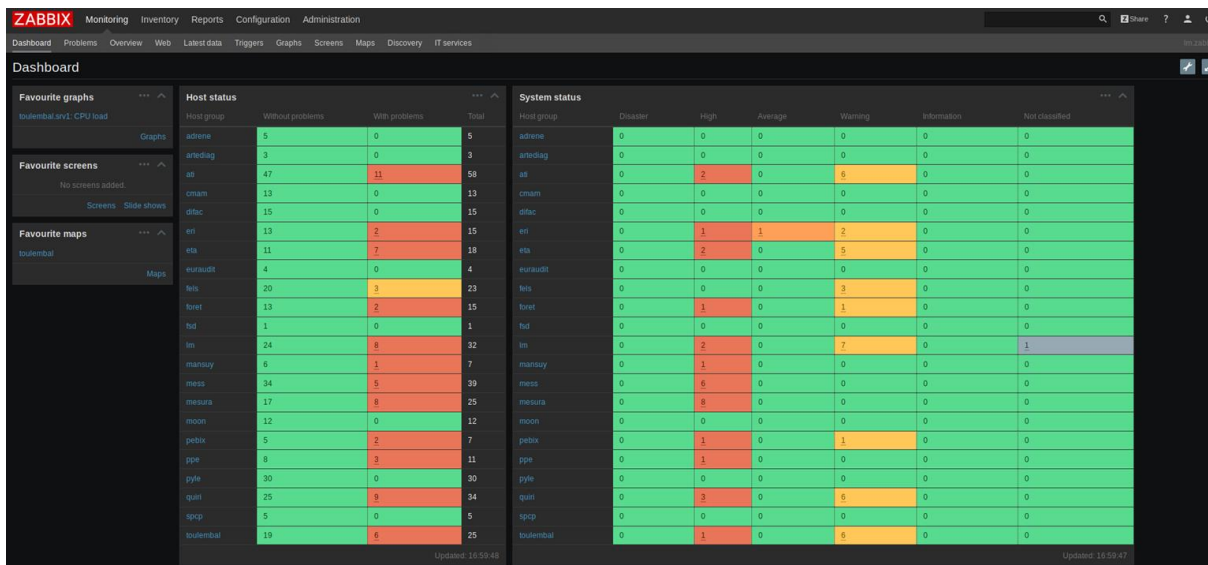


Figure 2.2.1 – interface web

Après cela, nous avons la partie Zabbix proxy, qui est très importante. C'est la partie qui permet de collecter des informations des hôtes locaux, et de transmettre ces informations au Server Zabbix. Cela réduit la charge du Server Zabbix. Cette méthode est celle qui est appliquée à LMI Solutions, car elle est appropriée pour superviser les sites distants, il va collecter les données, agissant comme une sonde.

Enfin, nous avons la partie Zabbix agent. Cette partie n'est pas nécessaire pour que le server Zabbix fonctionne correctement. Cependant, si nous voulons récupérer des informations comme l'espace restant sur un disque, ou voir le niveau d'utilisation du CPU, il faut avoir des Zabbix Agents

2.3 Fonctionnement de Zabbix

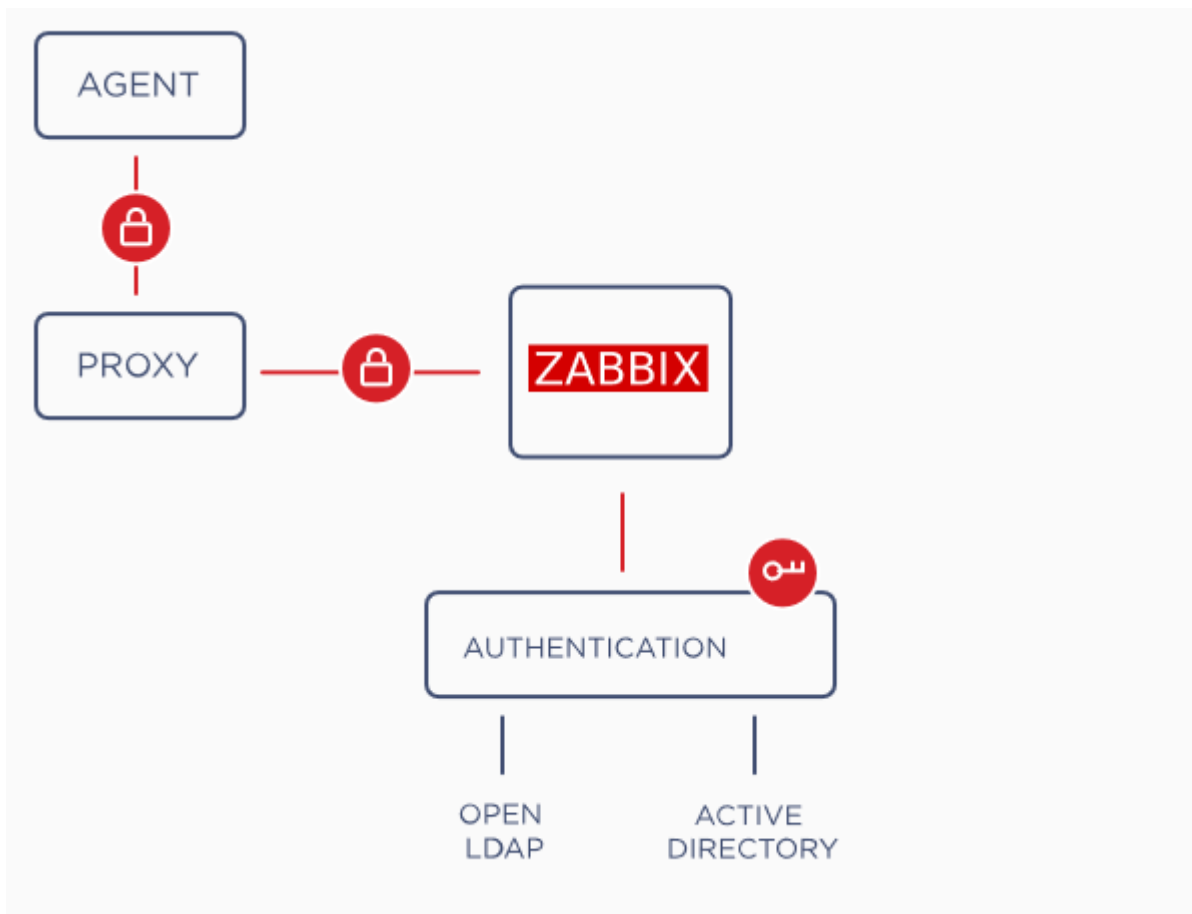


Figure 2.3.1 Fonctionnement de Zabbix

(Source : <https://www.zabbix.com/features>)

Dans ce schéma, Zabbix est divisé en 3 parties, comme dans le fonctionnement à LMI Solutions. La partie centrale, appelée Zabbix regroupe le Server Zabbix, et le frontend. Les agents renvoient au proxy les informations relatives aux différents hôtes, puis celui-ci les transmet au Server Zabbix.

Pour chaque client, nous définissons un site. Le Server Zabbix va ainsi récupérer dans chaque site un fichier init, qui correspond à ce qui sera installé sur les différentes machines en fonctions du système d'exploitation, ou du nom. (Par exemple si une machine contient le mot Zabbix dans son nom, elle sera définie comme proxy Zabbix du site).

Les différentes machines à superviser sont stockées dans le fichier client.sls. Ce fichier est stocké dans le dossier du client, et il est essentiel au fonctionnement de Zabbix. Dans ce fichier, sont contenues différentes informations sur les hôtes, telle que l'adresse IP des hôtes, la version du système d'exploitation (exemple : Windows server 2012), si on souhaite monitorer la machine ou non.

2.4 Installation de Zabbix

Pour installer Zabbix, il faut créer en premier temps la machine Zabbix proxy. Cette machine est, souvent, une machine virtuelle sous Ubuntu. Pour ce faire, il suffit d'exécuter un script qui va installer l'agent Salt et de configurer l'identifiant de la machine Server Zabbix, et définir le nom de la machine actuelle. Dans notre cas, le Server Zabbix est défini sous ce nom :

```
master: salt.lmgroupe.fr
```

Et le nom du proxy est sous la forme client.zabbix.

Une fois cela effectué, il faut accepter la clé du côté du server Zabbix, ce qui autorisera la communication du proxy vers le server Zabbix. Ensuite, il faut configurer le proxy dans l'interface web du server Zabbix.

Voici une capture d'écran de la fenêtre de création d'hôte. Il faut remplir les différents champs. En effet, en se basant sur le schéma précédent, on constate que tous les hôtes dépendent tous du serveur Zabbix.

The screenshot shows the Zabbix web interface with the 'Proxies' tab selected. The 'Proxy name' field contains 'foret.zabbix' and 'Proxy mode' is set to 'Active'. Below these fields, there are two lists: 'Proxy hosts' (empty) and 'Other hosts' (containing various hostnames like 'lm.web', 'lm.zabbix', 'mess.ckel', 'mess.dcl', 'mess.firewall.master', 'mess.firewall.slave', 'mess.hyper-v', 'mess.salto', 'mess.switch.baie-camera', 'mess.switch.baie-etage'). A 'Description' text area is also present. At the bottom are 'Add' and 'Cancel' buttons.

Figure 2.4.1 Création d'un hôte

Après cela, il faut installer le client Salt sur chaque machine Windows que nous souhaitons superviser.

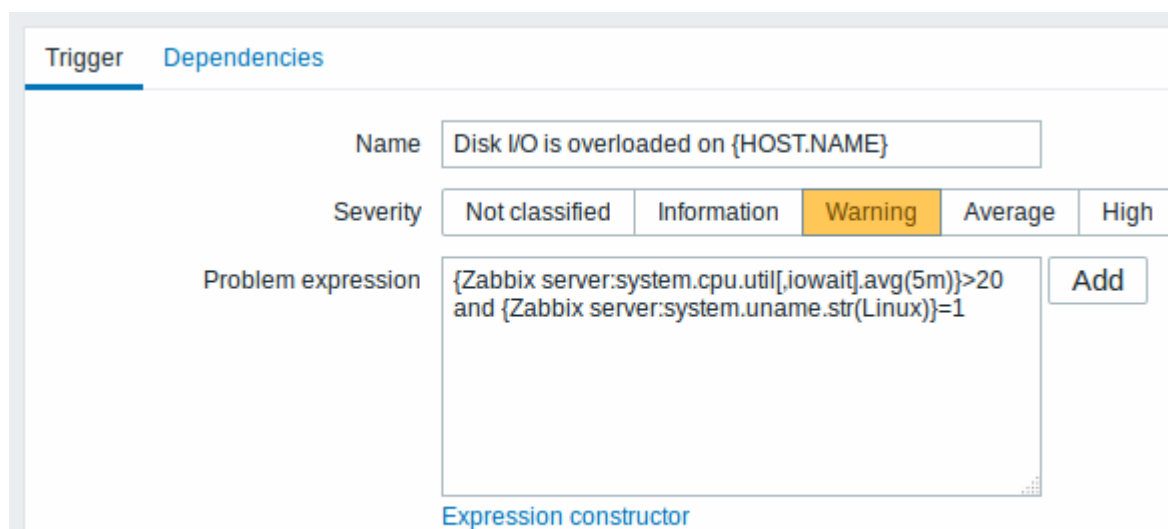
Une fois toutes ces manipulations effectuées, il ne reste plus qu'à lancer un script sur la machine server Zabbix, qui va automatiquement créer les hôtes, ajouter le bon Template. Un Template est attribué aux hôtes en fonction de leurs systèmes d'exploitation et des besoins des différents clients.

2.5 Les alertes Zabbix

Le système d'alerte de Zabbix fonctionne de la manière suivante :

Il existe 5 niveaux d'alerte sur Zabbix qui sont : non classifié, information, alerte, moyenne, et élevée.

Une fois qu'un Template est affecté à un hôte, il obtiendra des triggers, qui sont des déclencheurs définis dans les différents templates.



The screenshot shows the Zabbix web interface for configuring a trigger. It has two tabs: 'Trigger' (active) and 'Dependencies'. The 'Name' field contains 'Disk I/O is overloaded on {HOST.NAME}'. The 'Severity' section has five buttons: 'Not classified', 'Information', 'Warning' (highlighted in orange), 'Average', and 'High'. The 'Problem expression' field contains the Zabbix expression: '{Zabbix server:system.cpu.util[,iowait].avg(5m)}>20 and {Zabbix server:system.uname.str(Linux)}=1'. To the right of this field is an 'Add' button. Below the expression field is a link labeled 'Expression constructor'.

Figure 2.5.1 Trigger

(Source : <https://www.zabbix.com/documentation/3.4/manual/config/triggers/trigger>)

Un trigger est défini sous cette forme, avec un nom, une expression qui caractérise ce que le trigger va surveiller, et la condition pour que le trigger soit déclenché.

Lorsqu'une condition est atteinte, le trigger va envoyer un mail aux utilisateurs concernés, comprenant la machine hôte. Il contient également quel trigger a été activé, et la gravité du problème.

Voici une capture d'écran d'un mail Zabbix.

```
2019.05.17 12:47:30
Trigger: Zabbix agent on client.zabbix is unreachable for 3 minutes
Trigger status: PROBLEM
Trigger severity: High
Trigger URL:

Item values:

1. Agent ping (client.zabbix:agent.ping): Up (1)
2. *UNKNOWN* (*UNKNOWN*:*UNKNOWN*): *UNKNOWN*
3. *UNKNOWN* (*UNKNOWN*:*UNKNOWN*): *UNKNOWN*

Original event ID: 8558471
```

Figure 2.5.2 Mail de prévention Zabbix

Le principal problème de ce type de fonctionnement, est que lorsque le proxy Zabbix est déconnecté d'internet, aucune information n'est remontée jusqu'au server Zabbix, donc les hôtes sont considérés comme ne répondant plus au Zabbix proxy et par conséquent au lieu de simplement recevoir une alerte pour le proxy Zabbix, nous recevons aussi les alertes pour tous les hôtes dépendant du proxy.

3. Mon travail sur Zabbix

Ma tâche principale a été de m'occuper de Zabbix, d'importer les nouveaux clients, ou ceux qui n'étaient pas encore configurés. Ensuite, j'ai également dû m'occuper du problème d'alerte quand un proxy Zabbix n'est pas connecté à internet.

Je diviserai donc mon travail en deux parties, la première traitera de la première partie de ma mission et la seconde de la résolution du problème des triggers, avec un regard critique sur la solution apportée.

3.1 Création des différents Zabbix locaux

Pour créer les Zabbix correspondant aux différents clients, j'ai dû suivre une procédure, qui consiste à installer un proxy Zabbix, puis à configurer chaque hôte qui doit être supervisé, de manière qu'il communique avec le proxy Zabbix. Pour commencer, la procédure se divise en deux temps.

Dans le premier, il faut au préalable installer la machine virtuelle qui sera ensuite utilisée pour le proxy Zabbix. Cette machine permettra donc de relayer les informations des différents hôtes vers le serveur Zabbix. La machine virtuelle doit être créée en respectant certaines conditions sur la mémoire ram par exemple.

Pour la machine virtuelle, nous choisissons comme système d'exploitation Ubuntu. Cela est plus simple, car nous disposons d'un noyau linux, pour tous les types de dépannage.

Une fois que la machine virtuelle est finalisée, il faut ensuite créer les différents hôtes. Selon les clients, nous devons monitorer différents types de machines. Comme Saltstack nécessite des pré requis, il faut s'intéresser aux différents systèmes d'exploitation des différents hôtes.

Si le système d'exploitation est compatible avec Saltstack, alors j'ai effectué l'installation de l'agent Salt. Pour ce faire, il me suffit de me renseigner sur l'architecture de la machine, et sur la version de python de cette machine. En fonction de cela, je peux choisir la bonne version à télécharger sur le site de Saltstack.

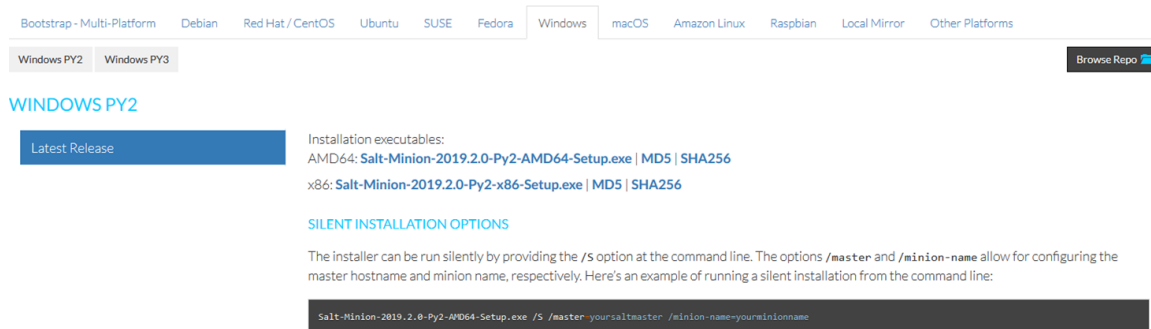


Figure 3.1.1 – Téléchargement Saltstack

(Source : <https://repo.saltstack.com/#windows>)

Une fois que toutes ces opérations sont effectuées, il faut ensuite se connecter au serveur Zabbix. Ainsi, je me connecte en SSH (un protocole qui permet de faire des connexions sécurisées) au serveur Zabbix. Ensuite, il faut donc que j'accepte les clés des différents hôtes. Les clés sont acceptées avec une simple commande.

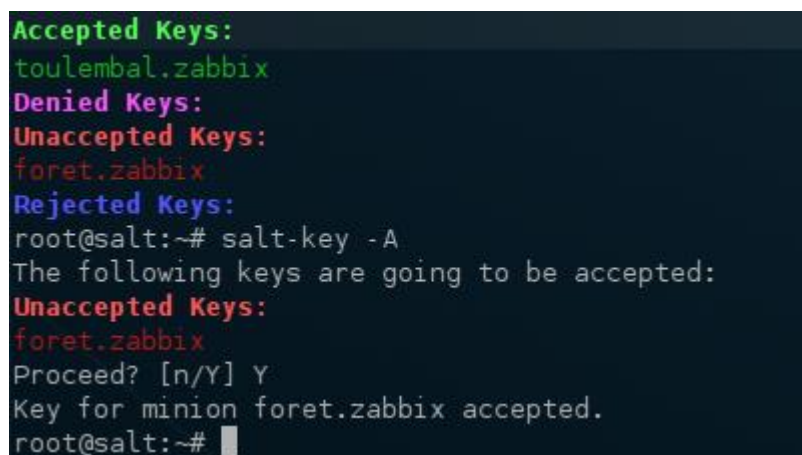


Figure 3.1.2 – Acceptation des clés

Puis, il faut faire une commande qui permet d'effectuer une suite de tests sur les machines pour vérifier l'état de l'agent Salt.

Cependant, les hôtes peuvent être aussi supervisés sans agent. Si la machine n'est pas compatible avec l'agent Salt, ou s'il ne faut pas installer d'agent Salt sur la machine (par exemple sur une baie de stockage) il est tout de même possible de surveiller ces hôtes.

Pour ce faire, on utilise le protocole ICMP (permet le contrôle des erreurs de transmission), qui permet d'avoir un minimum d'informations sur une machine.

Une fois que toutes ces manipulations sont achevées, les hôtes sont donc accessibles depuis l'interface web.

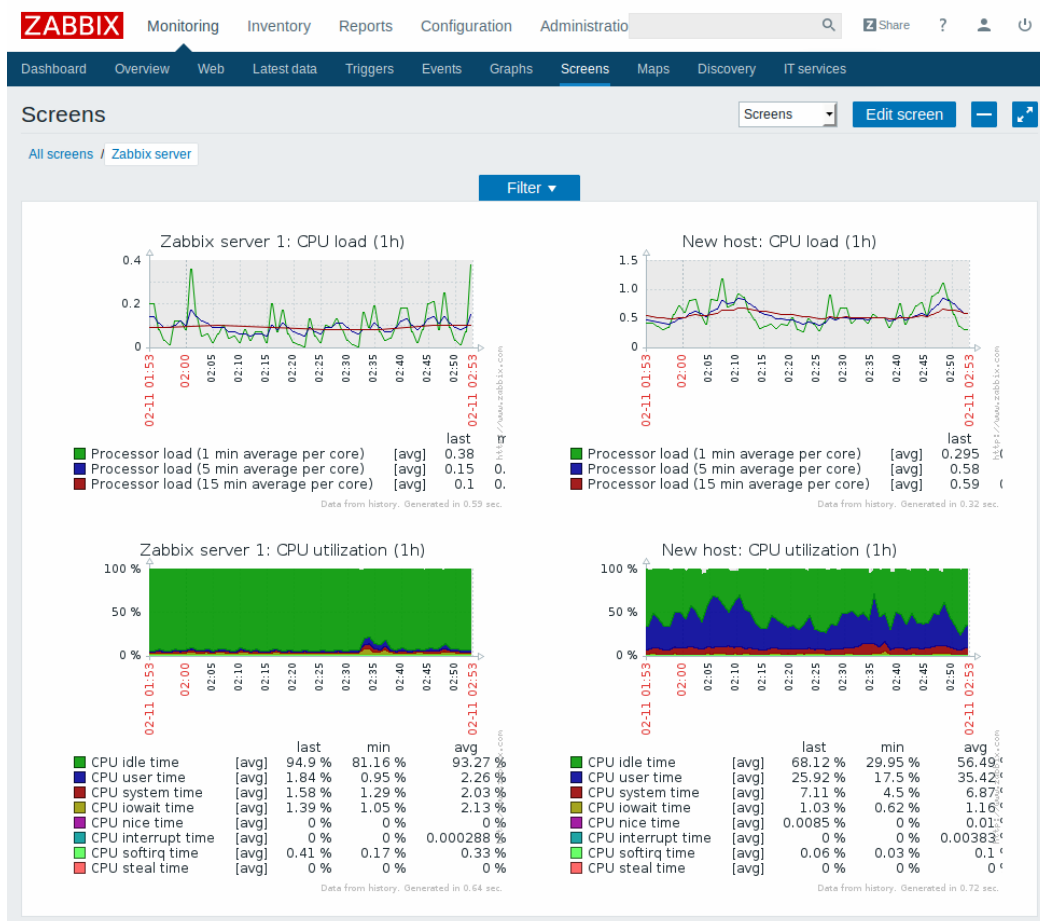


Figure 3.1.3 – supervision d'un hôte

(Source : https://www.zabbix.com/zabbix_web_frontend/)

Nous pouvons donc maintenant superviser tous les hôtes. Zabbix offre un suivi complet de chaque hôte avec un compte rendu en temps réel, des graphiques permettant de suivre en continu une donnée, par exemple le pourcentage de CPU utilisé. Et en cas de problème avec un hôte, nous pouvons savoir depuis combien de temps ce problème est survenu.

Cependant une fois ces opérations effectuées, un problème survient. En effet, en se basant sur la figure 2.3.1, comme un hôte dépend de son proxy, si celui-ci se retrouve hors ligne et si on venait à tomber en panne, chaque hôte étant relié à celui-ci seront donc considéré comme hors ligne. Donc, avec les différents triggers et la dépendance des hôtes au proxy, nous avons un problème d'alerte, qui peuvent être des faux positifs, ou trop de messages pour un même problème.

3.2 Résolution du problème sur les triggers

Lorsque la création de tous les hôtes est finie, et que le fichier client.sls est à jour, il faut exécuter une commande sur le server Zabbix, qui va importer tous les hôtes et monitorer ceux qui doivent l'être. La commande est issue d'un script, qui permet de configurer chaque hôte en fonction des informations renseignées dans le fichier client.sls. Il faut donc modifier le script, afin de pouvoir limiter ou supprimer ces faux positifs.

3.3 Proposition de solution face au problème

Avant de proposer une solution, j'ai fait une recherche dans le but de trouver si ce problème avait déjà été soulevé, et si une solution face à ce problème existe.

Je n'ai pas trouvé de solution qui s'adaptait bien avec la mise en place actuelle de Zabbix au sein de l'entreprise. Cependant, cela a permis de nous mettre sur une piste, qui consiste à définir des dépendances dans un trigger.

Mon choix de solution s'est donc fait grâce à ce qui existait déjà sur Zabbix. En effet, chaque hôte dispose d'un Template qui lui définit ses triggers.

Mon idée a donc été de mettre en place un Template sur chaque hôte, qui rajoute une dépendance. Si le proxy dont dépend l'hôte ne répond plus au ping, alors aucun trigger ne sera déclenché sur les hôtes qui sont relié à ce proxy.

J'ai donc mis en place, cette solution avec un seul hôte pour faire des tests. Pour réaliser ces tests, j'ai défini une dépendance à un hôte, puis j'ai déconnecté le proxy, pour vérifier les dépendances et pour pouvoir adapter le script XML.

Je ne recevais pas d'alerte venant de l'hôte. Ces tests ont été concluants. Pour généraliser cette solution, j'ai donc fait un import de l'hôte pour voir comment était définies les dépendances

Figure 3.3.1 – Dépendance d'un hôte

```
<dependencies>
  <dependency>
    <name>Zabbix agent on {HOST.NAME} is unreachable for 3 minutes</name>
    <expression></expression>
  </dependency>
</dependencies>
```

Une fois que nous avons compris la façon dont une dépendance est définie, nous avons pu définir une dépendance automatique à tous les hôtes. C'est-à-dire qu'à chaque import, il faut créer la dépendance et l'affecter à l'ensemble des hôtes du même proxy.

Pour réaliser cela, il a fallu que je modifie le script mis en place pour qu'il ajoute automatiquement la bonne dépendance aux différents hôtes.

Le script est en format XML. Chaque hôte est défini grâce à des Templates Jinja. Pour chaque client un fichier `device.sls` qui contient toutes les catégories d'hôte présent et qu'il faut monitorer chez le client. Grâce à cela, un fichier `zabbixs-import` a été créé permettant l'import de chaque hôte en fonction de leur type. Il m'a donc fallu adapter ce qui existait pour que chaque hôte, sauf le proxy, ait une dépendance sur le proxy. En mettant cette solution en place, j'ai eu un problème.

En effet, au départ ma première idée a été d'ajouter un Template à chaque hôte, et de désactiver le trigger `icmp` sur l'hôte. Ensuite d'ajouter un trigger `icmp` sur le nouveau Template, avec une dépendance sur le proxy Zabbix. Le problème que j'ai rencontré est que sur Zabbix, il n'est pas possible de prendre un trigger existant pour l'affecter à un autre Template.

Il m'a donc fallu supprimer ce trigger dans le template existant, pour faire un second template, avec uniquement un trigger `icmp`, qui dépend du proxy. Cependant pour ne pas le faire à la main pour chaque hôte, je me suis servi des templates Jinja.

Grâce à cela, j'ai pu définir un template propre à toutes les machines, en vérifiant si ce n'est pas la machine proxy.

J'ai donc défini en en-tête le template, qui sera propre à chaque proxy, et l'affectation de celui-ci à tous les hôtes. Le template est donc défini de cette manière.

```
<templates>
  <template>
    {%- if server_name != zabbix_hostname -%}
    <name>{{ template }}</name>
    {%- else -%}
    <name>template.proxy</name>
    {%- endif -%}
  </template>
```

Figure 3.3.2 – Définition des templates

Ensuite, il m'a fallu également définir à la main, pour chaque hôte, les triggers, et les dépendances. Le trigger est un trigger ICMP, qui est défini sur une vérification toutes les minutes. Pour un hôte s'il y a 5 vérifications d'affilées qui sont en échec, et que le proxy répond, il y a un envoi d'alerte. Pour le proxy on regarde s'il y a 3 vérifications d'affilées qui sont en échec.

Evaluation de la solution apportée

Cette solution mise en place correspond à l'attente. Pour tester cette solution, nous avons essayé d'importer plusieurs hôtes et les résultats étaient concluants. Cette solution est durable dans le temps, du moment qu'il ne faut pas modifier le template, ou que le trigger ICMP n'est utilisé que par ce template. De plus, il est possible d'adapter cette solution dans le futur, de la modifier facilement. Elle est adaptable aux différents besoins, et il n'existe pas de solutions similaires sur internet. Les solutions proposées, sont avec des plugins, ou en modifiant directement à la main les hôtes. Comme le fichier avec le code source est propre à l'entreprise, il n'y avait pas de solution avec ce type de fonctionnement.

Résumé

Durant ma période d'apprentissage, j'ai dû effectuer la supervision d'hôte, avec Zabbix. Ma mission a été de monitorer les hôtes qui devaient être monitorés. Pour ce faire, je me suis servi des outils qui étaient à ma disposition. Ensuite j'ai dû améliorer la solution en place, pour permettre de limiter les mails d'alertes. En effet, pour transférer toutes les informations des différentes machines, une machine sert de relais, pour les transmettre au serveur. Si l'une des machines relais venait à tomber en panne, ou à être déconnectée du réseau, toutes les machines qui sont reliées à elle seront considérées comme hors-ligne aussi. Pour ce faire, j'ai donc dû adapter le script en place, pour qu'il prenne en compte cette éventualité, et qu'il puisse donc s'adapter et n'envoyer qu'un seul mail au lieu d'un mail par hôte dépendant du même relai.

Mots-clés : (3 mots-clés)

Zabbix, saltstack, triggers

Abstract

During my apprenticeship period, I had to do host supervision with Zabbix. What I had to do was to monitor the one hosts that needed to be monitored. In order to do it, I used different tools which were at my disposal. Then I had to improve the existing solution to limit the number of alert emails. Indeed, to transfer all the information from the different machines, considering the fact that each machine serves as relay to transmit information to the server. If one of the relay machines would have failed, or be disconnected from the network, all the machines connected to it, will also be considered as offline. To do this, I had to adapt the current script, so that it would take into account this possibility, and so that it could adapt and send only one mail instead of one mail per host which depend on the same relay.

Keywords : (3 keywords)

Zabbix, saltstack, triggers