UNIVERSITÉ DE TOULON

KALANDA.NET
SOLUTIONS PROFESSIONNELLES

RAPPORT DE STAGE

Technicien Informatique et Réseaux

DE SOUZA Hugo

Avril 2021 à Juin 2021

Tuteur de stage : PIERRET Maxime (Technicien support)
Responsable pédagogique : SPANO Eric

Établissement / Formation : Université de Toulon – Licence Pro. Métiers Des Réseaux Informatique
Entreprise d'accueil : Kalanda – 94 Avenue de la Sarriette, 13600 La Ciotat

Remerciements

Avant de commencer le développement du rapport de cette expérience professionnelle, il me paraît tout naturel de commencer par remercier les personnes qui m'ont permis d'effectuer ce stage, ainsi que ceux qui m'ont permis d'en faire un moment agréable et enrichissant.
Je tiens tout d'abord à remercier Mr Christian Jaubert qui m'a accordé sa confiance et accueilli au sien de son entreprise Kalanda.
Je tiens à remercier vivement mon maitre de stage, Mr Maxime Pierret, pour le temps passé ensemble et le partage de son expertise au quotidien.
Je remercie également l'ensemble de l'équipe de Kalanda, pour leur accueil, ils ont su se rendre disponibles quand cela été nécessaire et ont toujours pris soin de m'expliquer les choses de façon pédagogique.

Table des matières

Introduction

Dans le cadre de ma Licence professionnel à l'Université de Toulon, j'ai eu l'opportunité d'effectuer mon stage au sein de l'entreprise Kalanda située à La Ciotat. Au cours de ce stage dans le département technique, j'ai pu m'intéresser au support technique ainsi qu'au déploiement de service et d'infrastructure.

Dans un premier temps, nous décrirons l'entreprise et son activité. Ensuite nous aborderons mes missions, et notamment mes différentes réalisations et le savoir-faire que j'ai pu acquérir au cours de ce stage. Enfin, nous dresserons un bilan global de ce stage

Présentation de l'entreprise

La société Kalanda, forte de l'expérience qu'elle conforte depuis 1996, d'abord dans le domaine de l'expertise et du développement spécifique sur de fortes technologies, ainsi que de sa faculté à choisir et s'adapter. Kalanda s'impose sur le marché de la conception de logiciels pour le compte de services publics et grands comptes, ainsi que de sociétés privées. Kalanda est plus particulièrement spécialisée dans la conception de logiciels spécifiques ainsi que de la mise en place d'Intranet, Extranet, sites Internet dynamiques, et de SIG (Systèmes d'Information Géographiques).

Dès 2001, Kalanda s'engage naturellement dans l'hébergement des technologies WEBDEV, ainsi que PHP/MySQL, .NET, HTML et bases de données.

Aujourd'hui, Kalanda employant 5 techniciens(nes) et certifié ISO 27001 ainsi que HDS (Hébergement de données de santé) propose un catalogue d'hébergement adapté aux besoins de sa clientèle, sans sacrifier la spécificité sans laquelle la notion du meilleur service ne saurait exister.

Description du travail demandé

Réalisation du scénario de migration de l'ensemble de l'infrastructure Zabbix de la version 3.4.15 vers la dernière version en LTS (tests préalables, tuning sur le nombre d'éléments analysés).
Général :
Zabbix 3.4.15 à Zabbix 5.0

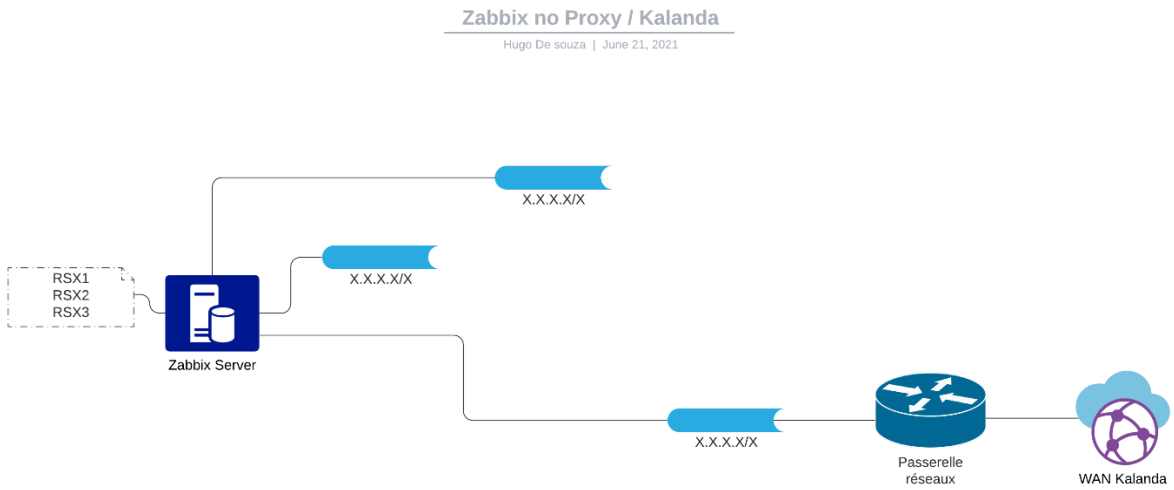
- Tests de répartition de charge via des proxys
- Utiliser la même @ip que l'hosts.zabbix.kalanda.net
- Éviter une coupure du service

- Utilisation snmpv2c

Points plus précis :

- Tuning MySQL
- Tuning Zabbix
- Répartition des hôtes sur les proxys
- Mise à jour des templates
- Mise à jour des scripts
- Utilisation d'un chiffrement entre le serveur Zabbix et les proxys
- Vérification de la rétrocompatibilité des agents Zabbix 3.0
- (Option : voir pour snmpv3)
- Compatibilité avec Grafana v6.7.3, si impossible (option préparer une migration vers Grafana v7)

Le travail qui m'a était demander est donc la migration de l'infrastructure de supervision actuellement en place vers une version mise à jour, une optimisation des bases de données et une répartition de charges. Le tout en évitant une coupure de service.
Voici le schéma de l'infrastructure actuelle :



Travail fourni

Actuellement la société utilise donc le logiciel libre Zabbix en version 3 qui vient de sortir du support et ne disposera donc plus de mises à jour. Zabbix est un logiciel qui supervise de nombreux paramètres réseaux ainsi que la santé et l'intégrité des serveurs. Zabbix utilise un mécanisme de notification flexible qui permet aux utilisateurs de configurer une base d'alerte e-mail pour pratiquement tous les événements. Cela permet une réponse rapide aux problèmes serveurs. Zabbix offre un excellent reporting et des fonctionnalités de visualisation de données basées sur les données stockées.

Le service est actuellement hébergé sur une machine virtuelle sous Debian 9 et la migration se fera sur une machine virtuelle similaire sous Ubuntu Serveur 20.04.
En premier, j'ai étudié les principaux changements au niveau de la compatibilité de Zabbix 5.0.12 TLS (Long terme support), les changements importants sont :

- PHP >7.2.0
- MySQL >5.5.62 ou Maria DB >10.3.3
    - + Activation des “Extended float”
    - Si bdd crée avec Maria DB <10.2.1 voir https://support.zabbix.com/browse/ZBX-17690
- Zabbix Agent >1.4
- Fin du support PolarSSL

Seuls deux changements nous concernent et serons à effectuer la base de données lors de la migration, l’activation des “Extended float” et la modification du type de ligne.
Afin d’établir de plan de migration, j'ai déployé sur un environnement de test (Windows Serveur 2019 + Hyper-V) une machine avec une installation représentant l’actuelle à l’aide de la documentation fourni sur l’existant ainsi qu’une seconde qui sera la cible de la migration.
Je me suis donc penché sur la question, “Que faut-il transférer sur le nouveau serveur ?”.
Le service se repose principalement sur un fichier “zabbix_server.conf” ou les paramètres du service sont définis et la base de données ou tout le reste est stocké (templates, hôtes, dashboard, etc). Le fichier de configuration doit être reporter sur le nouveau et non-copier, car certaines options ont été rajouter avec les mises à jour. La base de données elle doit être exporté puis réimporter
Ma première idée fut d’utiliser l’utilitaire “mysqldump” inclus avec MariaDB, il permet de réaliser un fichier contenant toutes les directives SQL permettant de recréer une base de données à l’identique de l’état dans lequel elle se trouvait au moment de la sauvegarde, mais après de nombreux test, il s’est révélé très lents dans l’importation de la base dans notre cas environ 350 Go.

Je me suis donc tourné vers un autre utilitaire sous le nom “mariabackup”, il est explicitement conçu pour les sauvegardes à chaud. De manière générale, XtraBackup assure la cohérence des sauvegardes à chaud comme suit : il copie les fichiers de données en direct de votre serveur MySQL, ouvre un petit serveur MySQL, puis applique à la copie les journaux de toutes les transactions effectuées entre-temps. Ce processus génère une sauvegarde cohérente qui ne nécessite pas de temps d'arrêt. Il s’est révélé parfait pour cette opération, de plus il supporte la compression ce qui nous réduit le temps de transfert vers le nouveau serveur.

Je commence donc par l’exportation et le transfert de l’ancienne bdd(base de données).

```
# mariabackup --backup --compress --target-dir=/home/kalanda/backup/
# scp -r -p /home/kalanda/backup/ kalanda@Ip-newsrv:/home/kalanda/
```

Une fois le transfert terminé la bdd peut être décompressé et réimporter.

```
# mariabackup --decompress --parallel=4 --target-dir=/home/kalanda/backup/ --remove-
original
# mariabackup --prepare --target-dir=/home/kalanda/backup/
# service mysql stop
# rm -rf /var/lib/mysql/*
# mariabackup --copy-back --target-dir=/home/kalanda/backup/ --
datadir=/var/lib/mysql/
```

Réattribution des droits sur le répertoire MySQL et changement du type de ligne.

```
# chown -R mysql:mysql /var/lib/mysql/
# service mysql start
```

Création de l'utilisateur et attribution des droits.

```
$ mysql -uroot -p
password
mysql> create user zabbix@localhost identified by 'password';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
mysql> quit;
```

Activation des "Extended float" a l'aide du script fournis par Zabbix. Cela est nécessaire, car Le type de données numérique (flottant) prend désormais en charge une précision d'environ 15 chiffres. C'est par défaut pour les nouvelles installations. Cependant, lors de la mise à niveau d'installations existantes, un correctif de mise à niveau manuelle de la base de données doit être appliqué.

```
$ wget
"https://git.zabbix.com/projects/ZBX/repos/zabbix/raw/database/mysql/double.sql"
$ mysql -uzabbix -p zabbix < double.sql
# sed -i '/DOUBLE/c\$DB['DOUBLE_IEEE754'] = true;/' /etc/zabbix/web/zabbix.conf.php
```

Une fois la base de données migrée le service Zabbix peut être installé normalement en omettant l'importation du schéma SQL de base, car nous venons de l'importer avec la base. Un processus de mise à niveau devrait se lancer automatiquement au lancement de Zabbix, ce processus peut se montrer assez long en fonction de la taille de la base, l'avancement peut être suivi dans les fichiers de logs.
Lors de la mise à niveau j'ai rencontré l'erreur suivante :

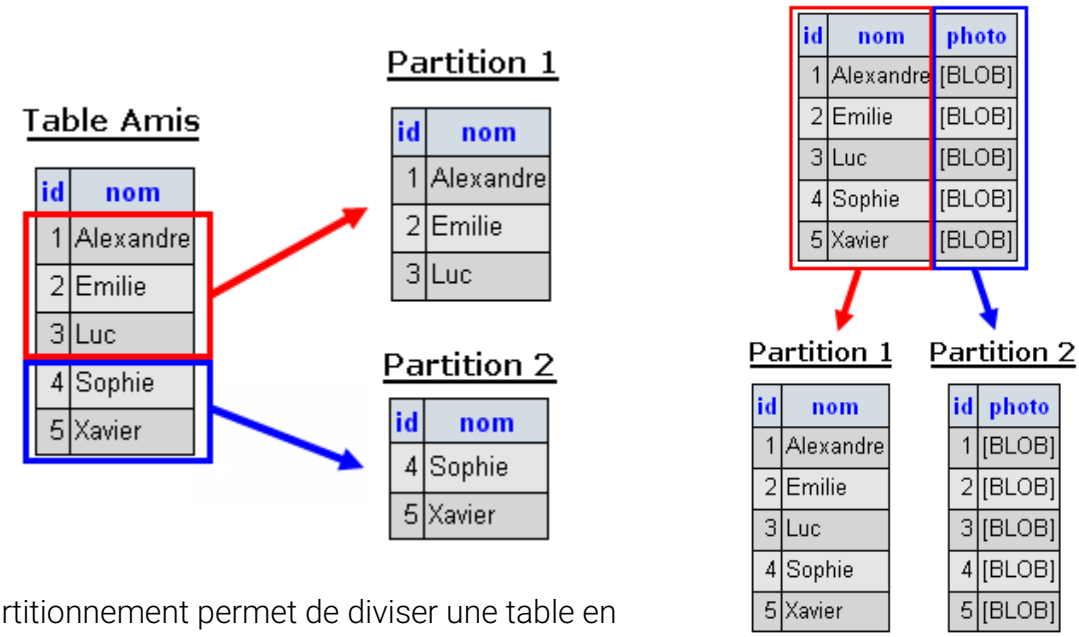
```
[Z3005] query failed: [1118] Row size too large.(...)
```

Après quelque recherche, il s'est révéler que le problème vient du changement de format de ligne par défaut de MySQL qui est passé de compact a dynamique. Il nous faut donc répercuter ce même changement sur la base fraichement migrée.

```
$ mysql -uroot -p zabbix -e "alter table hosts row_format = dynamic;"
```

Après une vérification rapide du bon fonctionnement du nouveau service et que toutes les données migrées était bien présente, je me suis attelé à une tâche primordiale au fonctionnement optimal de Zabbix, l'optimisation de MySQL et le partitionnement de certaine table de la base de données.

Par défaut MySQL stock chaque table d'une base de données dans un fichier ce qui dans notre cas peut engendrer des fichiers de plusieurs centaines de giga octets. Afin de ne pas stocker une quantité de donnée infinie, Zabbix dispose d'un processus appelé "housekeeper", il permet à intervalle défini de supprimer les données qui dépasse la politique de rétention configurer. Ce même processus est grandement affecté pas la taille de la base de données, en plus de prendre un temps considérable à la suppression des données, la base est pendant ce temps énormément ralentis ce qui engendre un ralentissement, un blocage global du service.
Afin de ne plus subir de ralentissement le partitionnement de certaine table de la base de données permet de palier à ce problème.
Il existe deux types de partitionnements, horizontal et vertical :



Le partitionnement permet de diviser une table en plusieurs partitions et donc en plusieurs fichiers de petite taille comme le montre les deux schémas ci-dessus. Dans notre cas, c'est le partitionnement horizontal basé sur l'horodatage des données que nous allons utiliser. Ce partitionnement va nous permettre de ne plus se baser le "housekeeper" et de gérer la rétention de donnée directement via MySQL. Contrairement au "housekeeper" qui effectue une opération « select » sur la base entière puis supprime des enregistrements de la table en fonction de leur horodatage, le partitionnement permet de supprimer directement les partitions antérieures à une date ce qui est presque instantané.
Les tables principales de Zabbix contant les données brutes remonter, sont les tables :

- history
- history_log
- history_str
- history_text
- history_uint
- trends

- trends_uint

La politique de rétention sera laissée comme l'actuellement de 30 jours pour l'historique et passera de 30 jours à 1 an soit 365 jours pour les trends. Les tendances(*trends*) représentent trois valeurs de l'historique par heure ce qui permet un stockage à long terme des évolutions survenues durant la supervision tout en limitant la quantité de donnée.

Pour mettre en place ce partitionnement, j'ai utilisé un script .sql fournis par le site "bestmonitoringtools.com". Le script va créer les procédures de maintenance nécessaires pour créer, supprimé et vérifié les partitions.
Il faut juste l'adapter à la politique de rétention.

```
...
CALL partition_maintenance(SCHEMA_NAME, 'history', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_log', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_str', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_text', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'history_uint', 30, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'trends', 365, 24, 3);
CALL partition_maintenance(SCHEMA_NAME, 'trends_uint', 365, 24, 3);
...
```

Une fois les procédures créer, il faut définir un éventement (tâche planifier) exécutant la maintenance des tables toute les 12 h.

```
$ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CREATE EVENT zbx_partitioning
ON SCHEDULE EVERY 12 HOUR DO CALL partition_maintenance_all('zabbix');"
```

Ensuite j'ai forcé l'exécution de la maintenance, car dans notre cas la base de données existe déjà il faut donc crée des partitions avec les données précédentes contenues dans la base de données.

```
$ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "CALL
partition_maintenance_all('zabbix');"
```

La première exécution peut prendre un certain temps, car elle va s'effectuer sur une base déjà remplie et crée une partition par jour d'historique et de tendance. Une fois terminer, j'ai vérifié la bonne création des partitions.

```
$ mysql -u 'zabbix' -p'zabbixDBpass' zabbix -e "show create table history\G"
```

Il est nécessaire de désactiver le "housekeeper" interne de Zabbix pour l'historique et les tendances afin de ne pas créer de conflit.
Une fois le partitionnement terminer je me suis penché sur l'optimisation de la configuration de MySQL.

Je me suis appuyé sur 2 utilitaires :

- mysqltuner.pl
- tuning-primer.sh

Ainsi qu'une configuration de MySQL (GitHub) et pré-optimiser pour répondre à 40 000 NVPS (Nouvelle Valeur Par Seconde), les nouvelles valeurs par secondes est un indicateur de charge et de performance de Zabbix.
Afin de pouvoir tester l'efficacité des changements apporter à la configuration de MySQL j'ai utilisé un outil me permettant de générer un grand nombre de valeurs par second reçu par Zabbix.

```
[mysqld]  #Fichier /etc/mysql/my.cnf
large-pages       #Activation des huges-pages
disable_log_bin   #Désactivation des "binary log"

event_scheduler=ON     #Activation du planificateur d'événement
skip-name-resolve=ON   #Désactivation de la résolution DNS

innodb_log_file_size=256M            #Taille max des fichiers de log
innodb_log_buffer_size=32M           #Cache RAM pour les transactions
innodb_open_files=1024               #Nombre maximum de fichier ouvert
innodb_flush_log_at_trx_commit=0     #Equilibre entre ACID et performance
innodb_flush_method=O_DIRECT         #Méthode de modification de la bdd
innodb_buffer_pool_instances=2       #Nombre d'instances de cache
innodb_buffer_pool_size=2G           #Taille des instances de cache

join_buffer_size=2M   #Taille du cache d'opération de jointure
thread_cache_size=15  #Nombre de "thread" en cache
sync-binlog=0         #Désactivation de la synchronisation des "binary log"

optimizer_switch='index_condition_pushdown=off'

table_definition_cache=500 #Uniquement pour MariaDB

query-cache-size=0 #Uniquement pour MariaDB
query_cache_limit=0 #Uniquement pour MariaDB
```

Après de nombreux test et exécution des utilitaires, je suis arrivé à la configuration ci-dessus.
Cette configuration permet sur la configuration de la machine (4vCPU, 8 Go, SSD) de traiter plus de 10000 VPS, ce qui laisse une bonne marge d'évolution étant donnée les 1300 VPS actuelle.

Au cours de mes recherches pour optimiser le service MySQL un changement supplémentaire a apporté mon semblé intéressant, l'activation des "Hugepages".
Les "Hugepages" permettent de définir une zone RAM dans laquelle les pages ne sont pas gérées avec une taille classique, généralement 4 Ko, mais d'une taille bien supérieure, 4 ou 2 Mo. Il y a plusieurs intérêts à l'utilisation des "Hugepages", cette zone mémoire est bloquée en RAM et ne peut pas aller en SWAP, et surtout elle permet d'optimiser la gestion du TLB (Cache CPU) et a ainsi un impact significatif sur l'utilisation du CPU lors des accès mémoire.
Pour les activer il faut simplement éditer le fichier "/etc/sysctl.conf" et y ajouter les lignes suivantes :

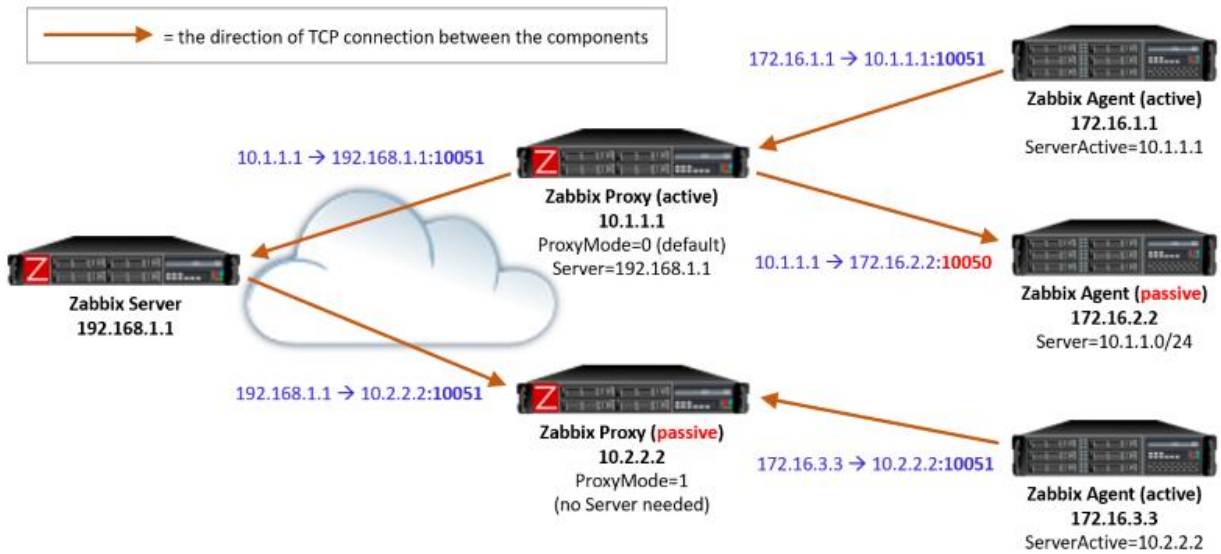
- vm.hugetlb_shm_group= 120
- kernel.shmmax = 1073741824
- kernel.shmall = 2883584
- vm.nr_hugepages = 512

La mise en place du nouveau serveur terminer l'IP peut maintenant être transféré, la migration est terminée.
Une fois la mise en place du service Zabbix Server terminer je me suis penché sur la répartition de charge à l'aide d'un Proxy Zabbix.

Le proxy Zabbix est un processus qui peut collecter des données de surveillance en travaillant essentiellement pour le compte du serveur. Toutes les données collectées sont stockées localement puis transférées au serveur Zabbix auquel appartient le proxy.
Le déploiement d'un proxy est facultatif, mais peut s'avérer très utile pour répartir la charge d'un seul serveur Zabbix. Si seuls les proxys collectent des données, le traitement sur le serveur devient moins gourmand en CPU et en E/S disque.

Le service de proxy Zabbix dispose de sa propre base de données dans laquelle les données son stocké et traité si nécessaire avant d'être renvoyer vers le serveur, tout comme le serveur que nous avons vu ce dessus il est nécessaire d'optimiser le service MySQL afin d'en tirer les meilleures performances.

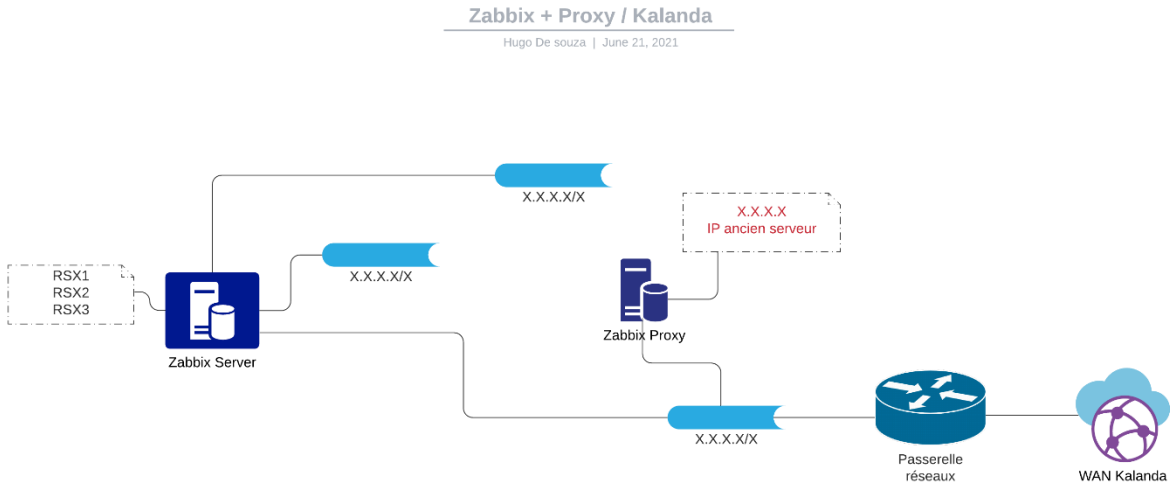
Contrairement au serveur le proxy n'utilise qu'une seule table pour stocker la majorité des données, la table "proxy_history" j'ai donc partitionné que cette table avec une durée de rétention d'un jour.



Le proxy peut être configuré en tant que proxy passif ou actif, dans notre cas de proxy actif permet de réduire la charge sur les "pollers"(processus de récupération de donnée) du serveur.

Le déploiement d'un proxy Zabbix est très similaire au déploiement d'un serveur. Il suffira de renseigner l'adresse du serveur dans la configuration du proxy dans le cadre d'un proxy actif et renseigner son "hostname" dans la section proxy de l'interface web.

Une fois ce déploiement terminer on se retrouve avec la nouvelle infrastructure suivante :



Pour une question de facilité de migration le proxy utilisera l'IP de l'ancien serveur Zabbix.

Dans une optique de cybersécurité la mise en place d'un chiffrement entre proxy, serveur et agent est nécessaire.

Zabbix utilise OpenSSL 1.1.1 pour le support du chiffrement.
Deux choix son offert pour le chiffrement :

- À l'aide de certificats (Plus sécurisé)
- À l'aide d'une clé PSK(Pre-shared key) (Plus simple à mettre en place)

Il m'a semblé plus judicieux de mettre en place des certificats pour les échanges serveur → proxy / proxy → serveur, et d'utiliser des clés PSK pour les échanges agent → proxy / proxy → agent. Il serait fastidieux de générer des certificats pour chaque agent déployer.

Pour la génération des certificats, il m'a fallu commencer par la création d'une autorité de certification Zabbix ne supportant pas les certificats auto signés. Cette autorité de certification nous servira à signer l'ensemble des certificats utiliser pour les échanges Zabbix.
Voici la procédure que j'ai utilisée :

```
# mkdir zabbix_pki
# chmod 700 zabbix_pki
# cd zabbix_pki
### Création de l'autorité de certification ###
# openssl genrsa -aes256 -out zabbix_ca.key 4096
# openssl req -x509 -new -key zabbix_ca.key -sha256 -days 3560 -out
zabbix_ca.crt
### Création du certificat et signature pour un proxy ###
# openssl genrsa -out zabbix_proxy.key 2048
# openssl req -new -key zabbix_proxy.key -out zabbix_proxy.csr
# openssl x509 -req -in zabbix_proxy.csr -CA zabbix_ca.crt -CAkey zabbix_ca.key
-CAcreateserial -out zabbix_proxy.crt -days 1460 -sha256
### Déplacement et changement des permissions ###
# mkdir /etc/zabbix/cert
# mkdir /etc/zabbix/cert/private
# mv ./zabbix_proxy.crt /etc/zabbix/cert/zabbix_proxy.crt
# mv ./zabbix_proxy.key /etc/zabbix/cert/private/zabbix_proxy.key
# chmod 644 /etc/zabbix/cert/*.crt
# chmod 640 /etc/zabbix/cert/private/*.key
```

Une fois les certificats générer il suffit de les renseigner dans le fichier de configuration du serveur et du proxy Zabbix. En plus de l'emplacement des fichiers, il est préférable de renseigner aussi un "Issuser" et un "Subject". On les obtient dans le bon format avec la commande suivante :

```
$ openssl x509 -noout -issuer -subject -nameopt
esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,s
name -in /etc/zabbix/*.crt ### Adapter au proxy ou serveur ###
```

Il est aussi possible en cas de fuite de certificat de renseigner dans le fichier de configuration une liste de révocation.

La génération de clés PSK (Pre-sharde Key) est on ne peut plus simple :

```
# openssl rand -hex 256 > /etc/zabbix/key.psk #Clé 256-byte / 2048-bit
# chown root:root /etc/zabbix/key.psk
# chmod 600 /etc/zabbix/key.psk
```

Il suffira ensuite comme pour les certificats de définir l'emplacement de clé dans le fichier de configuration ainsi qu'une "PSKIdentity". Le tout sera à reporter sur l'interface web Zabbix.

Afin de confirmer le bon fonctionnement du chiffrement j'ai mis en place un analyseur réseau (Wireshark) ce qui m'a permis de récupérer les paquets émis et reçu par le serveur Zabbix.

On peut voir ci-dessous que sans chiffrement les données sont en claire et lisible.

No.	Time	Source	Destination	Protocol	Length	Info
10	0.276464	10.0.0.148	10.0.0.158	TCP	66	10051 → 37564 [ACK] Seq=47 Ack=6
11	0.516733	10.0.0.158	10.0.0.148	TCP	74	44168 → 10051 [SYN] Seq=0 Win=64
12	0.516767	10.0.0.148	10.0.0.158	TCP	74	10051 → 44168 [SYN, ACK] Seq=0 A
13	0.516929	10.0.0.158	10.0.0.148	TCP	66	44168 → 10051 [ACK] Seq=1 Ack=1
14	0.568477	10.0.0.158	10.0.0.148	TCP	7306	44168 → 10051 [PSH, ACK] Seq=1 A
15	0.568477	10.0.0.158	10.0.0.148	TCP	7306	44168 → 10051 [PSH, ACK] Seq=724
16	0.568505	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=72
17	0.568514	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=14
18	0.568873	10.0.0.158	10.0.0.148	TCP	102…	44168 → 10051 [PSH, ACK] Seq=144
19	0.568874	10.0.0.158	10.0.0.148	TCP	130…	44168 → 10051 [PSH, ACK] Seq=246
20	0.568874	10.0.0.158	10.0.0.148	TCP	4410	44168 → 10051 [PSH, ACK] Seq=376
21	0.568903	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=24
22	0.568918	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=37
23	0.568923	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=41
24	0.569140	10.0.0.158	10.0.0.148	TCP	4410	44168 → 10051 [PSH, ACK] Seq=419
25	0.569141	10.0.0.158	10.0.0.148	Zabbix/JSON	9547	Zabbix Proxy data from "PROXY1",
26	0.569152	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=46
27	0.569194	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [ACK] Seq=1 Ack=55
28	0.661292	10.0.0.148	10.0.0.158	Zabbix/JSON	127	Zabbix Protocol, Version=3, Len=
29	0.661312	10.0.0.148	10.0.0.158	TCP	66	10051 → 44168 [FIN, ACK] Seq=62

Alors qu'a contrario une fois le chiffrement mis en place le protocole TLS 1.3 est utilisé et chiffre les données.

1	0.000000	10.0.0.158	10.0.0.148	TCP	74	48972 → 10051 [SYN] Seq=0 Win=64240 Len=0
2	0.000029	10.0.0.148	10.0.0.158	TCP	74	10051 → 48972 [SYN, ACK] Seq=0 Ack=1 Win=6
3	0.000117	10.0.0.158	10.0.0.148	TCP	66	48972 → 10051 [ACK] Seq=1 Ack=1 Win=64256
4	0.000647	10.0.0.158	10.0.0.148	TLSv1.3	379	Client Hello
5	0.000716	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=1 Ack=314 Win=6489
6	0.001781	10.0.0.148	10.0.0.158	TLSv1.3	324	Server Hello, Change Cipher Spec, Applicat
7	0.001937	10.0.0.158	10.0.0.148	TCP	66	48972 → 10051 [ACK] Seq=314 Ack=259 Win=64
8	0.002687	10.0.0.158	10.0.0.148	TLSv1.3	130	Change Cipher Spec, Application Data
9	0.002697	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=259 Ack=378 Win=64
10	0.002892	10.0.0.148	10.0.0.158	TLSv1.3	145	Application Data
11	0.002975	10.0.0.158	10.0.0.148	TCP	66	48972 → 10051 [ACK] Seq=378 Ack=338 Win=64
12	0.028462	10.0.0.158	10.0.0.148	TCP	7306	48972 → 10051 [PSH, ACK] Seq=378 Ack=338 W
13	0.028462	10.0.0.158	10.0.0.148	TCP	7306	48972 → 10051 [PSH, ACK] Seq=7618 Ack=338
14	0.028499	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=7618 Win=6
15	0.028523	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=14858 Win=
16	0.028779	10.0.0.158	10.0.0.148	TLSv1.3	102…	Application Data [TCP segment of a reassem
17	0.028780	10.0.0.158	10.0.0.148	TLSv1.3	130…	Application Data [TCP segment of a reassem
18	0.028780	10.0.0.158	10.0.0.148	TCP	4410	48972 → 10051 [PSH, ACK] Seq=38026 Ack=338
19	0.028812	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=24994 Win=
20	0.028827	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=38026 Win=
21	0.028830	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=42370 Win=
22	0.029566	10.0.0.158	10.0.0.148	TCP	4410	48972 → 10051 [PSH, ACK] Seq=42370 Ack=338
23	0.029566	10.0.0.158	10.0.0.148	TLSv1.3	130…	Application Data [TCP segment of a reassem
24	0.029566	10.0.0.158	10.0.0.148	TLSv1.3	8644	Application Data, Application Data
25	0.029587	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=46714 Win=
26	0.029604	10.0.0.148	10.0.0.158	TCP	66	10051 → 48972 [ACK] Seq=338 Ack=59746 Win=

Cependant, sur une grande partie des serveurs et équipements surveiller par Zabbix n'utilise pas l'agent, mais le snmp plus précisément le snmpV2c qui lui aussi transmet toutes les informations en claire sur le réseau. La principale différence entre snmpV2c et snmpv3 réside dans les améliorations apportées au modèle de sécurité et de configuration à distance. Snmpv3 ajoute une sécurité cryptographique à snmpV2c. snmpv3 remplace le partage de mot de passe simple (en texte clair) dans snmpV2c par des paramètres de sécurité codés beaucoup plus sécurisés.

Sans snmpV3, les réponses son lisible :

No.	Time	Source	Destination	Protocol	Length	Info
702	16.434323	10.0.0.148	10.0.0.128	SNMP	86	get-request 1.3.6.1.2.1.1.3.0
705	16.435361	10.0.0.128	10.0.0.148	SNMP	88	get-response 1.3.6.1.2.1.1.3.0
1556	46.451976	10.0.0.148	10.0.0.128	SNMP	102	get-request 1.3.6.1.2.1.1.3.0 1
1579	46.461520	10.0.0.128	10.0.0.148	SNMP	105	get-response 1.3.6.1.2.1.1.3.0
2803	76.489561	10.0.0.148	10.0.0.128	SNMP	86	get-request 1.3.6.1.2.1.1.3.0
2818	76.504168	10.0.0.128	10.0.0.148	SNMP	88	get-response 1.3.6.1.2.1.1.3.0

Avec snmpv3, les réponses sont chiffrées :

811	21.865970	10.0.0.148	10.0.0.128	SNMP	185	encryptedPDU: privKey Unknown
812	21.866019	10.0.0.148	10.0.0.128	SNMP	186	encryptedPDU: privKey Unknown
813	21.866126	10.0.0.148	10.0.0.128	SNMP	186	encryptedPDU: privKey Unknown
814	21.868909	10.0.0.128	10.0.0.148	SNMP	454	encryptedPDU: privKey Unknown
815	21.868924	10.0.0.128	10.0.0.148	SNMP	515	encryptedPDU: privKey Unknown

La configuration du snmpv3 est très rapide et les modèles par défaut de Zabbix qui permette la remonter des éléments son directement compatible.
Mise en place sur une machine Linux (Ubuntu) :

```
# /etc/snmp/snmpd.conf
rouser username authpriv # Ici l'utilisateur est user

# net-snmp-create-v3-user -ro -a authpass -X private -a SHA -x AES username
```

Il suffira ensuite de configurer le snmpv3 sur zabbix en renseignant :

- authpass
- private
- username
- protocoles

Une fois la partie migration de Zabbix terminer, je me suis orienté vers la migration de Grafana, Grafana est une application web libre sous licence Apache qui permet d'agréger une ou plusieurs sources de données, afin d'en réaliser des visualisations à travers de tableaux de bord qui se composent de graphiques dynamiques.
Grafana permet également de créer des alertes personnalisées, par SMS ou par Email notamment. L'application est extensible à souhait, grâce à un grand nombre de plug-ins disponibles notamment un plugin Zabbix.
Kalanda utilise Grafana car elle dispose de plusieurs services de monitoring (Zabbix, LibreNMS) et cela permet d'agréger les informations en un seul service.

Il m'a donc était demandé de vérifier la compatibilité de Grafana v6.7.3 avec Zabbix 5.0LTS.
J'ai donc déployé Grafana dans sa version 6 sur une VM et à l'aide du plugin Zabbix qui permet de mettre en relation Grafana et l'api de Zabbix.
Après quelque vérification que les données était bien disponible sur les tableaux de bord, j'en ai conclu qu'ils étaient parfaitement compatibles cependant il reste préférable d'effectuer la mise à niveau afin disposé des derniers correctifs de sécurité et dernière fonctionnalité.

Dans le cadre de la migration du serveur Zabbix, le serveur Grafana sera migré aussi.
Une fois la nouvelle version de Grafana installer il nous suffira de transférer le dossier '/var/lib/grafana/' ainsi que reporter la configuration si nécessaire.
La réinstallation des plugins est nécessaire.
Il ne faut pas oublier de réattribuer le droit sur le dossier après le transfert.

Dans le cas particulier de l'utilisation de Zabbix avec Grafana, il est préférable d'activer la connexion directe à la base de données plutôt que l'utilisation de l'api, cela permet de réduire de temps de chargement des "dashboard" ainsi que la quantité de donnée transférée. Lors de mes tests (à l'aide de l'analyseur réseaux de Google Chrome) j'ai obtenu ces résultats :

- ~/15 temps de requêtes
    - 150 ms → 10 ms

- ~/4 quantité de donnée
    - 20 ko → 5 ko

Ces résultats sont donnés par requête, il peut en avoir plusieurs centaines lors du chargement de la page. Pour cela il nous faudra donc crée un utilisateur MySQL a cet effet.

```
$ mysql -uroot -p
password
mysql> create user grafana@IP identified by 'password';
mysql> grant select on zabbix.* to grafana@IP;
mysql> quit;
```

Puis renseigner ses paramètres dans la configuration des sources de Grafana.
Une fois la configuration terminée la migration de Zabbix et de Grafana est maintenant terminé.

Proposition d'une solution différente

Lors de toutes les recherches et teste au cours de l'élaboration de ce scénario de migration, l'utilisation d'un autre moteur de base de données m'a paru être une idée pertinente à développer.

Je me suis donc penché sur l'idée d'une migration vers une base de données sous PostgreSQL, plus précisément PostgreSQL avec l'utilisation de l'extension TimescaleDB. TimescaleDB est une base de données open source optimisée pour le stockage de données de séries chronologiques. Elle est implémentée comme extension de PostgreSQL et combine la facilité d'utilisation des bases de données relationnelles et la rapidité des bases de données NoSQL.

Avantage de TimescaleDB :

- Meilleure performance globale
- Support de la compression (jusqu'à 96% de gain réel)
- Support des sauvegardes à chaud
- Performant « out of the box »
- Simple

Pour effectuer cette migration, j'ai utilisé "pgloader" pour le transfert de la base de données.
Il faut prendre soin de créer l'utilisateur ainsi que la base de données initiale.

```
sudo -u postgres createuser --pwprompt zabbix
sudo -u postgres createdb -O zabbix zabbix
```

Pour l'utilisation du SSL lors du transfert de la base, il nous faut aussi récupérer les certificats.

```
# scp -p /var/lib/mysql/ca.pem user@newsrv:/home/user/ca.pem.crt
# scp -p /var/lib/mysql/client-cert.pem user@newsrv:/home/user/client-cert.pem.crt

### Sur le nouveau serveur ###
# mv *.pem /usr/local/share/ca-certificates/
# update-ca-certificates
```

Ensuite on peut transférer la base de données à partir du nouveau serveur, le temps nécessaire peut être long :

```
./pgloader mysql://user:mdp@ipmysqlsrv/zabbix?sslmode=require
postgresql://user:mdp@ippsqlsrv/zabbix
```

Une fois l'importation terminée, il est nécessaire d'attribuer les droits et de rétablir le bon schéma pour le bon fonctionnement.

J'ai effectué cette opération à l'aide d'un script SQL que j'ai adapté pour cette utilisation.

```
DO
$$
DECLARE
row record;
BEGIN
FOR row IN SELECT table_name FROM information_schema.tables WHERE table_schema =
'zabbix' ORDER BY table_schema,table_name
LOOP
EXECUTE 'ALTER TABLE ' || quote_ident(row.table_name) || ' owner to zabbix;';
EXECUTE 'ALTER TABLE ' || quote_ident(row.table_name) || ' set schema public;';
END LOOP;
END;
$$;

\c zabbix
drop schema if exist zabbix;
alter table alerts alter column parameters drop not null;
```

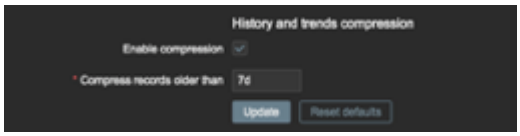
La base de données est maintenant utilisable par Zabbix. Il nous faut maintenant activer l'extension TimescaleDB, en plus de l'activation il faut exécuter le script fourni avec Zabbix.

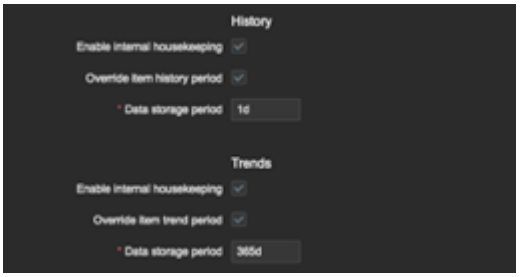
```
echo "CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;" | sudo -u
postgres psql zabbix cat timescaledb.sql | sudo -u zabbix psql zabbix
```

Une fois TimescaleDB activer, tout comme MySQL une optimisation des paramètres de PostgreSQL ainsi que de Timescale son nécessaire afin d'en tirer les meilleures performances.
Pour ce faire j'ai utilisé 2 utilitaires et un site :

- timescale-tune
- postgresqltuner.pl
- pgtune.leopard.in.ua

Le paramétrage de la politique et rétentions et de compression s'effectuera directement sur l'interface web de Zabbix.





La migration vers PostgreSQL est terminée.
Une fois les deux serveurs (MySQL et PostgreSQL) je me suis penché sur l'évaluation des performances, avantage et inconvénient des chacun.

Pour effectuer ces tests, j'ai utilisé un outil me permettant de générer plusieurs milliers de nouvelles valeurs envoyer à Zabbix ce qui me permet d'évaluer la charge que le service et la base de données peuvent supporter.

2000 NVPS Zabbix	Système PostgreSQL	Système MySQL
CPU	0.2659	0.2504
RAM	1.98/8 Go libre	2.37/8 Go libre
I/O	~125 o/s	~ 75 o/s
Stockage (7j d'historique)	5,76 Go (Compresser)	96 Go

On peut voir à l'aide de ces principaux indicateurs de performance que le système PostgreSQL est très légèrement plus gourmand en ressource, mais est beaucoup plus efficace dans le stockage des données.
La différence de consommation de ressource est négligeable cependant le gain en stockage est de l'ordre de 96% place gagner ce qui peut permettre un stockage des données sur une plus longue durée.

En ce qui concerne les avantages et inconvénients des deux gestionnaires de base de données :

- PostgreSQL et MySQL propose quasiment les mêmes performances avec Zabbix
- MySQL légèrement moins consommateur de ressource
- MySQL dispose d'une plus grande part de marché et donc plus de documentation
- PostgreSQL est plus stable à forte charge
- PostgreSQL est pleinement ACID
- PostgreSQL offre les sauvegardes à chaud
    - TimscaleDB :
        - Compression avec jusqu'à 97% de gain de stockage
        - Partitionnement facile à mettre en place
        - Gain de performance pour les "times-séries"
            - (Chiffres fournis par TimescaleDB)
            - "insert" x20 plus rapide
            - Requête x1,2-14000
            - "delete" x2000 plus rapide
        - Reste une technologie récente (2017)
        - Communauté très active

Étant donnée toutes ces informations que j'ai pu recueillir au cours de mes tests je pense que la migration de MySQL vers PostegreSQL est une solution viable et même une meilleure solution que le système actuel. Cella permettra de facilement améliorer les performances à l'aide du partitionnement automatique de TimescaleDB et de réduire la base de données d'environ 410 Go a 100 Go (La compression se faisant uniquement après sept jours minimum). En ce qui concerne les proxys et dans l'optique d'homogénéisé l'infrastructure en cas de passage à TimescaleDB sur le serveur, les proxys passeraient sur PostegreSQL avec un partitionnement, TimescaleDB n'étant pas compatible à ce jour avec les proxys.

Ma première idée était de, comme le proxy avec MySQL de partitionné la table « proxy_history » afin de ne pas impacter les performances lors du nettoyage par le « housekeeper ».
J'ai commencé par création de la table suivie de l'activation du partitionnement avec 1 h de rétention, voici les commandes utilisées :

```
$ sudo -u postgres createuser --pwprompt zabbix
$ sudo -u postgres createdb -O zabbix zabbix
```

```
$ sudo -u zabbix psql zabbix
CREATE TABLE public.proxy_history
(
    id                      bigserial                                   NOT NULL,
       itemid                  bigint                                      NOT NULL,
       clock                   integer         DEFAULT '0'                 NOT NULL,
       timestamp               integer         DEFAULT '0'                 NOT NULL,
       source                  varchar(64)     DEFAULT ''                  NOT NULL,
       severity                integer         DEFAULT '0'                 NOT NULL,
       value                   text            DEFAULT ''                  NOT NULL,
       logeventid              integer         DEFAULT '0'                 NOT NULL,
       ns                      integer         DEFAULT '0'                 NOT NULL,
       state                   integer         DEFAULT '0'                 NOT NULL,
       lastlogsize             numeric(20)     DEFAULT '0'                 NOT NULL,
       mtime                   integer         DEFAULT '0'                 NOT NULL,
       flags                   integer         DEFAULT '0'                 NOT NULL,
       write_clock             integer         DEFAULT '0'                 NOT NULL
) PARTITION BY RANGE (clock);
CREATE INDEX proxy_history_1 ON public.proxy_history USING btree (itemid, clock);


$ sudo -u postgres psql zabbix
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman schema partman;
GRANT ALL ON SCHEMA partman TO zabbix;
GRANT ALL ON ALL TABLES IN SCHEMA partman TO zabbix;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA partman TO zabbix;
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA partman TO zabbix;


$ sudo -u zabbix psql zabbix
SELECT partman.create_parent('public.proxy_history', 'clock', 'native', 'hourly', null, 7,
'on', null, true, 'seconds');
UPDATE partman.part_config set retention = '1 hour', retention_keep_table = false,
retention_keep_index = false WHERE parent_table = 'public_proxy_history';
SELECT partman.run_maintenance('public.proxy_history');
SELECT partman.run_maintenance(p_analyze := false);
UPDATE partman.part_config SET retention_keep_table = false, retention = '1 hour'
WHERE parent_table = 'public.proxy_history';
SELECT partman.run_maintenance('public.proxy_history');
```

Une fois la table partitionné le reste du schéma fournis avec l'installation du proxy peux être importé.

```
$ zcat /usr/share/doc/zabbix-proxy-pgsql/schema.sql.gz | sudo -u zabbix psql zabbix
```

Cependant, je me suis heurté a un gros problème de performance lors de l'utilisation du partitionnement avec PorsgreSQL. L'un des processus clé du proxy Zabbix le « data sender » qui renvoie les métriques remonter par le proxy au serveur se retrouvé rapidement saturer ce qui engendré un blocage du proxy.
Je me suis donc tourné vers le houskeeper interne de Zabbix.
Avec PostgreSQL le housekeeper arrive à soutenir la charge et prend seulement quelques dizaines de secondes à nettoyer la BDD pour 2000 NVPS.

Conclusion

Pour conclure, j'ai effectué mon stage de fin d'études de licences professionnel en tant que technicien informatique et réseaux au sein de l'entreprise Kalanda. Pendant trois mois, j'ai pu mettre en pratique les connaissances théoriques acquises durant ma formation sur les métiers des réseaux informatique et me suis confronté aux difficultés du monde du travail dans le secteur de l'hébergement informatique. Cette expérience a été très enrichissante pour moi, car elle m'a permis de découvrir le domaine de l'hébergement informatique ainsi que ses acteurs et contraintes. Aujourd'hui, cette expérience vient de confirmer le fait que j'ai fait le bon choix d'orientation et me permet d'affiner mon futur projet professionnel.