POLYTECH LILLE

Université de Lille 1 SCIENCES ET TECHNOLOGIES

POLYTECH LILLE

Rapport de projet intermédiaire

P31 - Supervision des serveurs de la plateforme informatique

Semestre 9 - Septembre à Décembre 2018

Taky DJERABA
Promo 2019

Resp. École : M. Thomas VANTROYS
M. Xavier REDON

Sommaire

Chapitre 1

Présentation du contexte

Le projet a pour objectif de réaliser un tableau de bord affichant l'état des serveurs physiques et des machines virtuelles de la plateforme informatique ainsi que de mettre en place un système de sauvegarde automatique. L'idée est de faciliter la gestion de la plateforme informatique ainsi que de permettre aux personnes se chargeant de son administration d'être alerté en cas d'incident.

La plateforme informatique en question se compose de trois serveurs : *bisban*, *sandbox* et *baleine* (le serveur de sauvegarde), possédant une distribution Debian ainsi qu'un hyperviseur Xen en mode para-virtualisation. Ces derniers sont reliés à un stack de commutateurs donnant accès aux réseaux ADSL, SDSL et Renater.

L'architecture en salle serveur se présente comme suit :

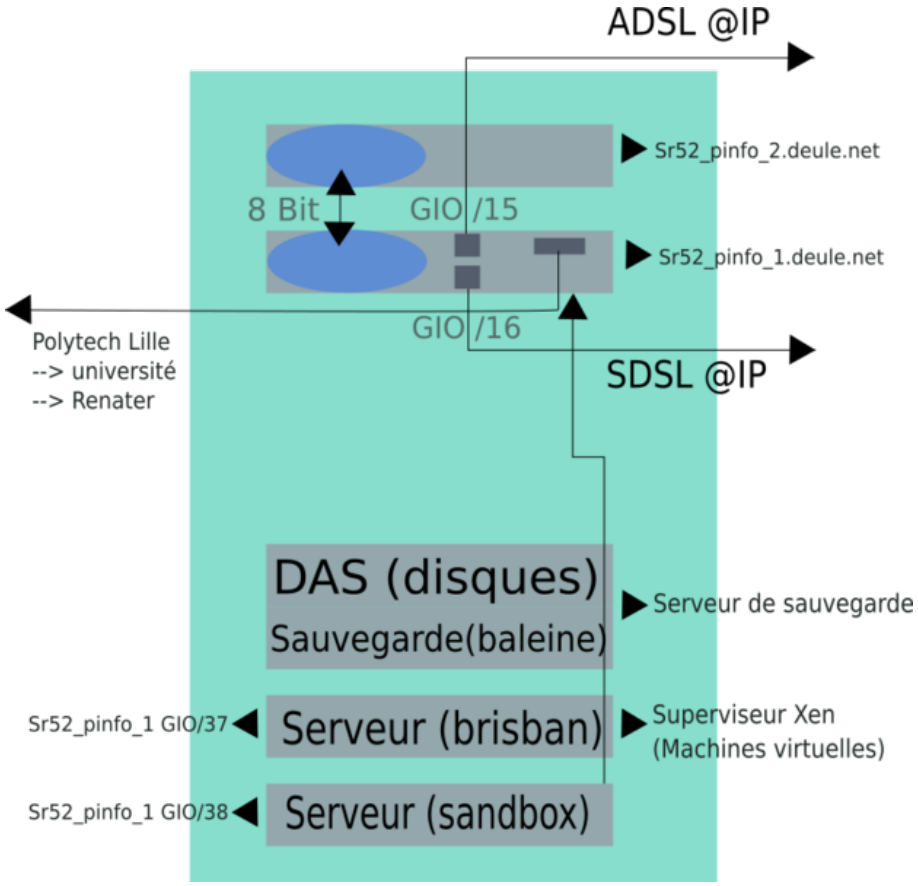

FIGURE 1.1 – Source : https ://wiki-ima.plil.fr/ - Onglet plateforme

Chapitre 2

Présentation du cahier des charges

Objectifs

Les éléments suivants doivent être monitorés :

— État de santé des machines physiques : température, état des disques, ...
— Occupation des machines physiques : utilisation CPU, utilisation espace disque, utilisation mémoire
— État de santé des connexions réseau : réseau Renater, réseau ADSL, réseau SDSL
— État de santé des machines virtuelles : temps d'exécution de chaque machine virtuelle
— Occupation des machines virtuelles : utilisation disque et mémoire
— État de certaines applications critiques :
    Date de validité des clefs DNSSEC
    Dates des dernières sauvegardes des machines virtuelles
— Vérification des certificats https de sites web.
— Éventuellement, l'affichage de la température en salle serveur

En ce concerne le système de sauvegarde automatique, les contraintes sont les suivantes :
— Le système de sauvegarde doit être simple et ne doit nécessiter aucune configuration.
— L'espace de stockage sur le serveur de sauvegarde est limité, ce qui exclue les backups complètes régulières.
— Le système doit idéalement permettre de garder en mémoire une backup datant d'un jour, d'une semaine et d'un mois.

Environnement de travail

Afin de tester la surveillance de serveurs et de machines virtuelles, j'ai eu à ma disposition l'accès au serveur de sauvegarde *baleine*. Sur cet hôte, deux machines virtuelles ont été installés :
— *supervise* : le serveur de supervision.
— *vmtest* : une machine virtuelle de test.

A cela, on peut ajouter la machine virtuelle *stargate*, installée sur le serveur *brisban*, et routé vers l'extérieur, afin de pouvoir tester les réseaux ADSL, SDSL et Renater ainsi que la date de validité des clés DNSSEC.

Chapitre 3

Travail effectué

L'essentiel du travail réalisé durant ce semestre s'est centré autour de la mise en place du serveur de supervision ainsi que sur la création des sondes nécessaire pour la récupération de certaines données critiques (DNSSEC, État des réseaux et disques).

3.1 Choix des technologies

En ce qui concerne la surveillance des machines virtuelles et physiques de la plateforme informatique, j'ai eu la possibilité de me reposer sur des solutions de monitoring déjà existante comme Nagios ou Prometheus. Ces solutions ont été accompagnées d'agents tel que NRPE ou encore Node_exporter, afin de prélever des métriques sur serveur à monitorer.

3.1.1 Nagios Core & NRPE

La première solution à avoir été employée est Nagios Core. Ce choix à surtout été motivé par la popularité de cet outil ainsi que par l'importance de sa communauté, mettant à disposition plugins et addons sur la plateforme communautaire Nagios Exchange.

Nagios est un outil de supervision permettant la surveillance d'équipements, de services ou de réseaux. Ce dernier effectue des contrôles intermittents sur les hôtes et services spécifiés en utilisant des plugins externes qui retournent un statuts d'état à Nagios.

Nagios utilise deux types de supervision :

— La supervision active : le serveur de supervision va chercher l'information à surveiller.
— La supervision passive : le serveur de supervision reçoit l'information à surveiller.

En plus de l'utilisation de Nagios Core, l'agent Nagios Remote Plugin Executor (NRPE) a été installé sur les serveurs à surveiller afin de permettre l'exécution de scripts à distance par Nagios via le plugin check_nrpe.

Configuration de Nagios

La configuration de Nagios se fait de manière assez simple par la modification des fichiers de configuration présent dans le répertoire d'installation de Nagios.

L'arborescence dans ce répertoire se présente comme suit :

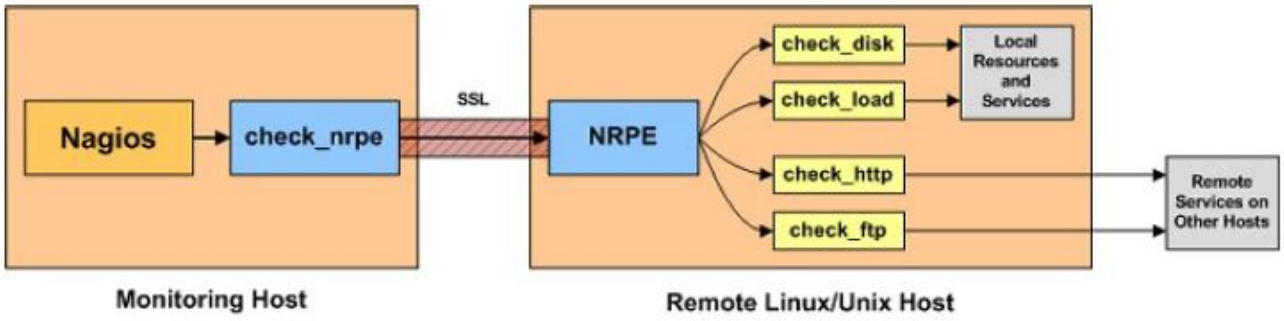

FIGURE 3.1 – Schéma de fonctionnement de Nagios+NRPE

```
|-- cgi.cfg                    Config de l'application web de nagios
|-- htpasswd.users             Contient les identifiant pour la connexion à l'application web
, ici: nagiosadmin:1234
|-- nagios.cfg                 Fichier de configuration de Nagios
|-- resource.cfg               Définition de chemin pour l'accès aux scripts lancé par Nagios
|-- objects                    Dossier contenant les fichier de configuration relatifs aux
services et hôtes
|    |-- commands.cfg          Fichier de configuration des commandes pouvant être lancé
par Nagios
|    |-- contacts.cfg          Configuration des contacts à avertir en cas de problème
|    |-- localhost.cfg         Configuration de la supervision du serveur hôte Nagios
|    |-- printer.cfg           Définition des imprimantes hôtes à superviser
|    |-- switch.cfg            Définition des switchs hôtes à superviser
|    |-- templates.cfg         Fichier contenant des templates pour la définition d'hôtes
ou de services
|    |-- timeperiods.cfg       Définition des configs de temps
|    |-- windows.cfg           Exemple d'un fichier de configuration pour une machine
Windows
|-- servers                    Dossier contenant les fichiers de configuration servant
à la définition des hôtes à monitorer et des services à exécuter
|    |-- baleine.cfg           Définitions pour l'hôte baleine
|    |-- vmBaleine.cfg         Définitions pour les VMs se trouvant sur baleine
```

Afin se surveiller une donnée, il est nécessaire de définir deux principaux élément :

— Un hôte à surveiller.
— Un service à mettre en place.

La définition de ces objets se fait au moyen de macros, dans des fichier de configuration situés dans un répertoire dédié : /usr/local/nagios/etc/servers.

Les définitions ci-dessous se situent dans le fichier /usr/local/nagios/etc/servers/baleine.cfg et permettent le lancement par Nagios du plugin check_load présent sur le serveur baleine.

On définie un hôte en lui attribuant une adresse :

```
define host
      use linux-box
      host_name baleine.insecserv.deule.net{
      alias Serveur de sauvegarde Baleine
      address 172.26.64.15
}
```

Et on met en place un service sur cet hôte :

```
define service{
     use generic-service
     host_name baleine.insecserv.deule.net
     service_description Current Load
     check_command check_nrpe!check_load
}
```

On peut voir dans la définition du service ci-dessus que Nagios exécute la commande "check_nrpe", laquelle prend en paramètre "check_load".

La définition de la commande check_nrpe est faite dans le fichier de configuration commands.cfg :

```
define command{
     command_name check_nrpe
     command_line /usr/lib/nagios/plugins/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
```

La commande check_nrpe va ensuite faire un appel au daemon NRPE exécuté sur la machine distante à monitorer pour lancer un plugin check_load.

Configuration de NRPE

L'installation de NRPE sur les machines à surveiller se fait de manière tout aussi simple en modifiant le fichier /usr/local/nagios/etc/nrpe.cfg :

On indique quels serveurs sont autorisés à communiquer avec NRPE :

```
allowed_hosts=127.0.0.1,172.26.64.13
```

Et on définit les commandes qui seront lancés :

```
command[check_load]=/usr/local/nagios/libexec/check_load -r -w .15,.10,.05 -c .30,.25,.20
```

L'exemple ci-dessus permet à NRPE de lancer la commande check_load lorsque le serveur de supervision Nagios en fait la demande.

3.1.2 Prometheus & Node_exporter

La deuxième solution à avoir été testé est Prometheus. Il s'agit, comme pour Nagios, d'un outil de supervision, à la différence que ce dernier est beaucoup plus moderne. Prometheus est inclus avec une base de donné en série temporelle, ce qui lui permet de stocker des métriques de manière optimisé, et donc, de surveiller un serveur pendant un temps beaucoup plus long. A cela, on peut ajouter un fonctionnement extensible, du fait d'une compatibilité avec d'autre modules comme Kubernetes (permettant d'automatiser le déploiement de conteneurs), Grafana (Affichage de métriques), ou PagerDuty (système d'alerte), etc ...

De la même manière que pour Nagios, Prometheus a besoin d'agent afin de collecter différentes métriques. Je me suis tourné pour cela vers Node_exporter, un job exporter proposé par Prometheus, permettant d'exporter un grande nombre de métriques en utilisant des modules inclus dans Linux (CPU, Température, Mémoire, Espace Disque, Interfaces réseaux, etc ...)

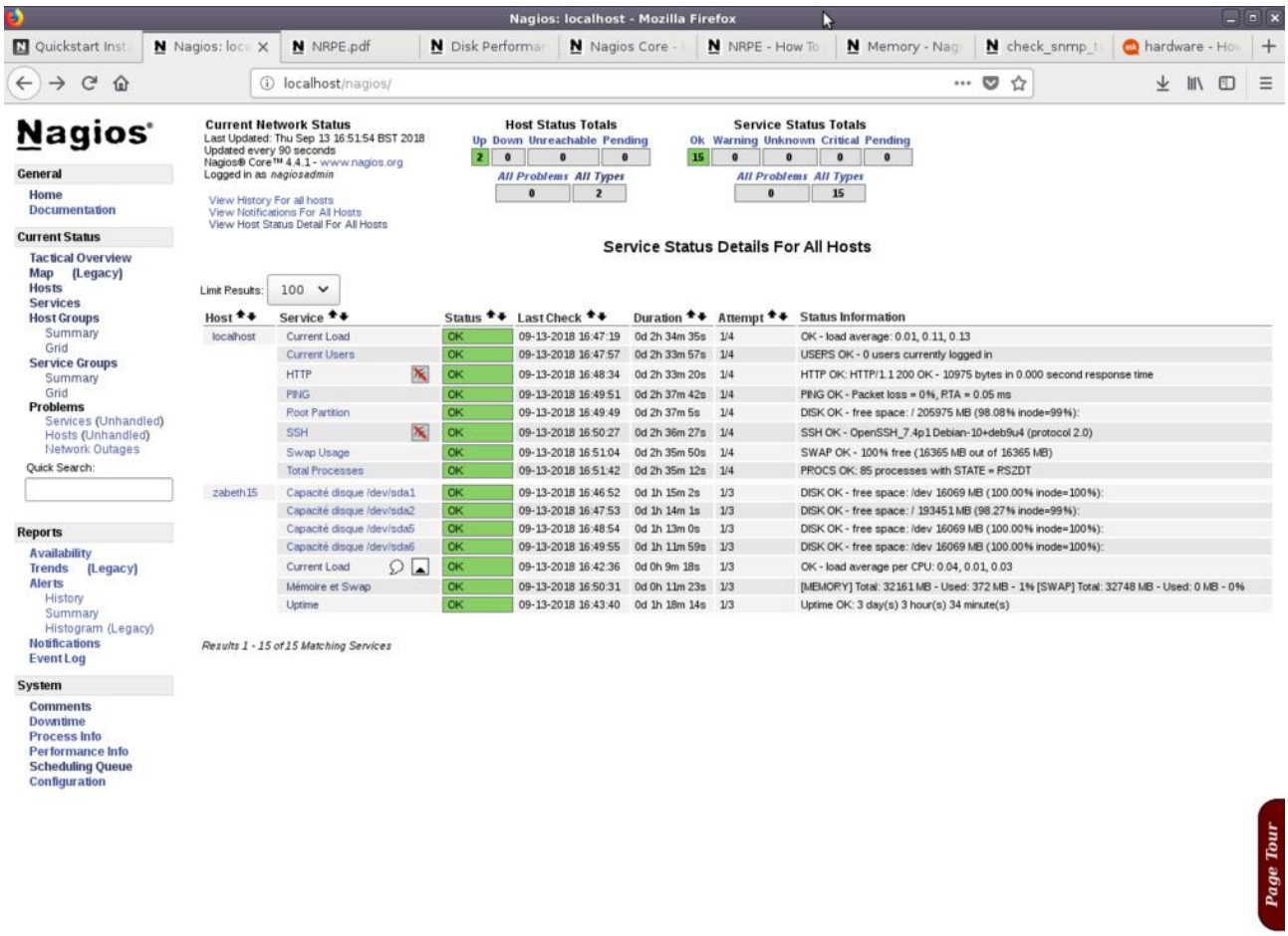

FIGURE 3.2 – Interface Nagios

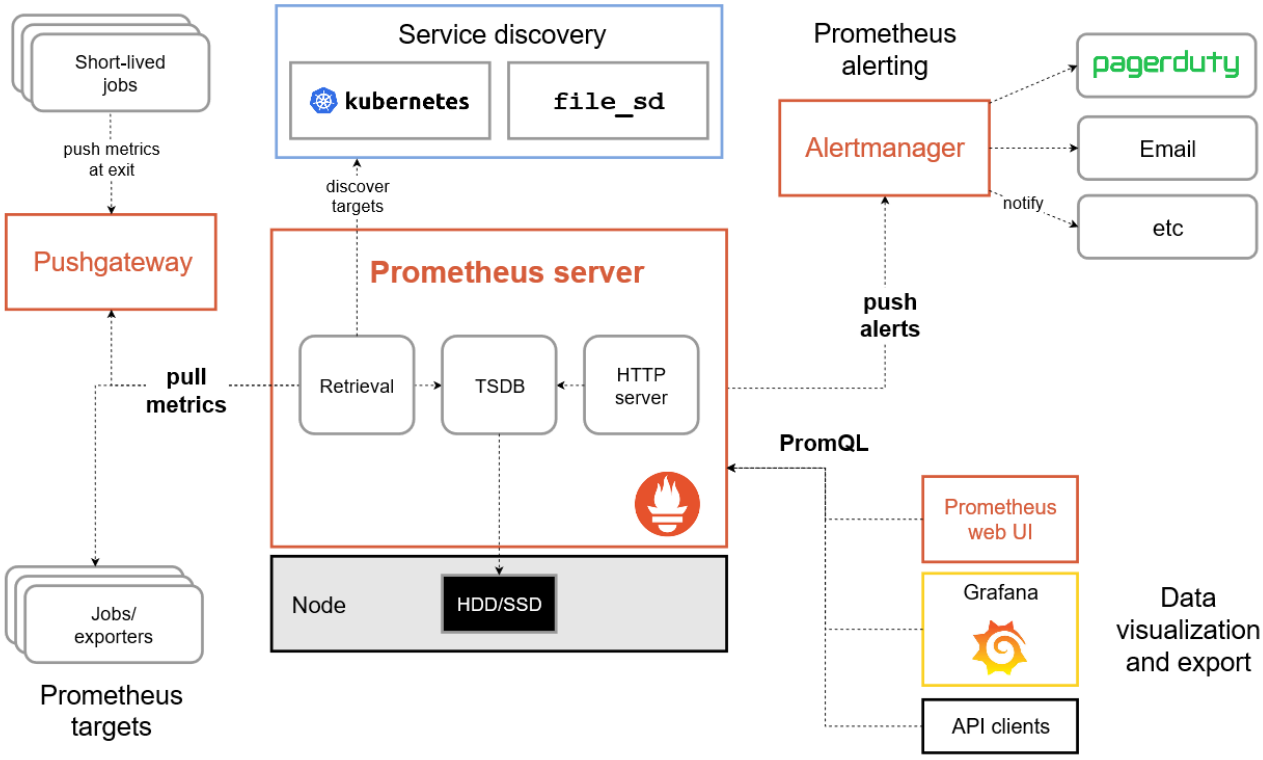

FIGURE 3.3 – Architecture de Prometheus

Configuration de Prometheus

La seule configuration à effectuer pour mettre en place Prometheus est de lui indiquer les adresses IP des machines à superviser dans le fichier prometheus.yml :

```
scrape_configs:
 # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
 - job_name: 'node'
   static_configs:
           - targets: ['172.26.64.15:9100','localhost:9100','172.26.64.16:9100']
```

Configuration de Node_exporter

Node_exporter possède l'avantage d'être prêt à l'emploi, aucune configuration n'est requise si ce n'est pour mettre en place le module Textfile Collector.

Le module Textfile Collector permet de récupérer des métriques à partir de fichier en .prom. De cette manière, on peut exporter des données issues de scripts lancé périodiquement (grâce à un outil comme cron, par exemple).

Voici la configuration d'un job cron proposé sur le git du Node_Exporter :

```
echo job_bash > /path/to/directory/my_batch_job.prom.$$
mv /path/to/directory/my_batch_job.prom.$$ /path/to/directory/my_batch_job.prom
```

L'activation du module Textfile Collector de fait de la manière suivante :

```
./node_exporter --collector.textfile --collector.textfile.directory=/path/to/promfile
```

3.1.3 Grafana

Afin de pouvoir afficher des métriques issues à la fois de Nagios et de Prometheus, un serveur Grafana a été installé sur le serveur de supervision.

FIGURE 3.4 – Métrique issue du serveur de supervision Nagios

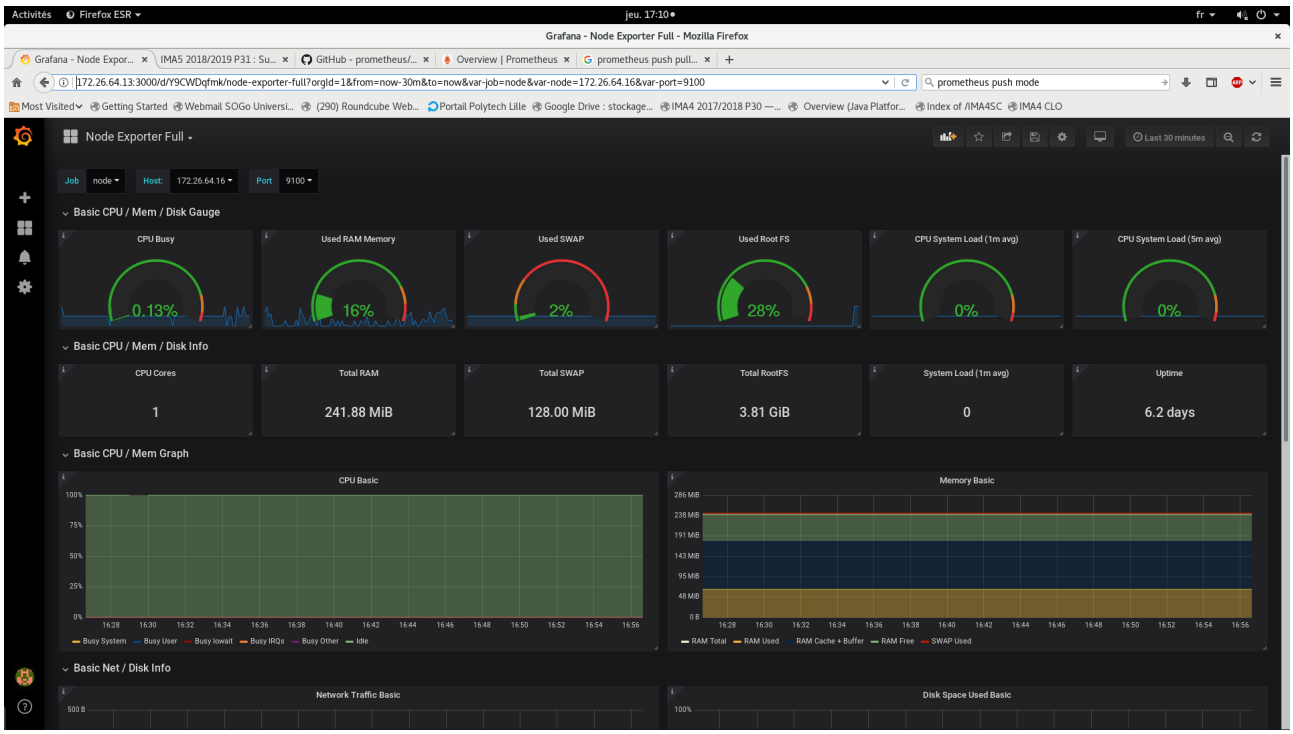

FIGURE 3.5 – Métriques issues de la VM "vmtest" affiché sur grafana

3.2 Création des scripts

Pour la récupération de certaines données spécifiques, comme la récupération de la date de validité des clés DNSSEC, la vérification de l'état de santé des disques ou encore l'état des réseaux ADSL, SDSL et Renater, la création de scripts a été nécessaire.

3.2.1 Output de retour

En fonction de la solution à utiliser (Nagios ou Prometheus), la valeur de retour des scripts est amené à être différente :

Nagios

```
Format:
Message de retour | PerfData_label=value[UOM];[warn];[crit];[min];[max]
```

```
Exemple:
Key OK - ZSK Expiration Date = 2018/12/21 11:12:57 | zsk_expiration_date_date=20181221111257
```

En plus du message de retour, le script doit retourner un code de retour correspondant à un statut :

```
0  OK
1  WARNING
2  CRITICAL
3  UNKNOWN
```

Prometeus

```
Format (minimaliste):
# HELP Label Documentation sur la métrique.
# TYPE Label <Type> (Type: counter, gauge, histogram, summary, ou untyped)
Label <Donnée>


Exemple:
# HELP Disque_0 État de santé du disque présent sur le slot 0 sur serveur baleine.
# TYPE Disque_0 untyped
Disque_0 0
```

3.2.2 Date de validité des clés DNSSEC

Préalablement à la rédaction du script, du temps a été consacré à étudier le fonctionnement du protocole DNSSEC, afin de déterminer quelle donnée critique devra être surveillé. Il a finalement été convenu avec le tuteur de projet que la donnée à récupérer est la date de validité des signatures de l'enregistrement DNSKEY. Un script utilisant l'outil *dig* a donc finalement été rédigé.

Le résultat de la commande dig plil.fr RRSIG +multi donne le résultat suivant :

```
plil.fr. 21599 IN RRSIG DNSKEY 5 2 259200 (
                        20181221111257 20181121110621 40768 plil.fr.
                        TWruW7vtrja1i37zJw/egTbLCPvlw5DTVDvEQf7qIPu0
                        P8eKTw0dD3N+JXGU3k1mmvidw+Ljy7YeqZn9A1ZbR1Db
                        qfQnXk6sg5MBRS0Xw3QFVB61cvnETz36bsZQvd7fO9ny
                        KMWpviQOEfjWbZaZk5vbDZXbgFrksj7Mt/55zMvyMFvJ
                        I5fAv+mlo5RGk8M/AUJ20TaVjUOfe/p9qTlDJoEFSjeW
                        A6qN7tFjn7qIvgHLhcghguVBsn7nogxzX9gIZ0uvZSFz
                        xC+xQrn8wEOISSZJWm9ZONwJuHcT8oIgjcSqnDx4rmcu
                        pF3YyB2OYTcxa9JT+Hx8wX0swoc0P+B/Nw== )
plil.fr. 21599 IN RRSIG DNSKEY 5 2 259200 (
                        20181221111257 20181121110621 51828 plil.fr.
                        Oc+8B84qbhy4lh2Uka1bkJnYJhQFFSZO3AfxW6YPbQha
                        vsVd9ozGc3Zbs3jumVm6TLK5sC8B+l24NnBeGD9QCiG4
                        29RbsRMyOi59YaemTA42BRuZCKJMWvJn6e/0FCyvTc2T
                        w/we6MWDXcRoIAqbCD+79Bniq+sbt4pBRU2zrWU= )
```

Les Hashs ci-dessus représentent les enregistrements RRSIG de type DNSKEY des clés ZSK et KSK. Le premier champ à l'intérieur des parenthèse représente la date de validité de la signature (voir figure 3.6 pour plus d'infos).

L'idée pour le script consiste donc à récupérer la date d'expiration et à la parser. La première étape consiste à récupérer l'ID des clés ZSK ou KSK. En raison du roulement des clés, il se peut que deux clés d'un même type soit enregistré sur un domaine.

```
dnssec-tools.org.       21 IN DNSKEY 256 3 5 (
                                AwEAAde0IFUPiuDwEQM8vGlR436nb+TzGLlMWxzMmWDP
                                mHfWMPv8OXgZafScErgijGRnwPfv+t9irTUsX4dkvJie
                                vkV549mtDwhXLb9Vyg9C5JsoqOBq//QJvZXEZHtcHPoQ
                                9tIGVt8W9uctLUKMsAdAVaozfMxl8CuLWjqkL2Gq57HP
                                ) ; ZSK; alg = RSASHA1; key id = 19221
dnssec-tools.org.       21 IN DNSKEY 256 3 5 (
                                AwEAAfDcFXK9ef/BppquoTm7LSk2rIE5x9LPu+WplVEA
                                To9i7OhqxxPy6K6uIomxLNtZAlcDbUybQEeopWhglovk
                                odkvRY72q6ZUxW1vwzBAvmDLZ7dfTjPa0LYKVQ7qsf7Q
                                jFBiyPkWARSAHH2xqBATry25ft8j9O9ULX4/DYdYPq9D
                                ) ; ZSK; alg = RSASHA1; key id = 3147
dnssec-tools.org.       21 IN DNSKEY 257 3 5 (
                                AwEAAazJbvxfEciFb3LmGsddun82dGsrZj4YpCvn8qHI
                                KRQ2c0Hra0LnfhOK09YKG7B9eey3aWzOKWUa50fv/0sl
                                8npI7/5tgWCQFBpIdqcDxoJxtvtFlrUlOpRQz45aFE3x
                                FJoquCfPGYgaNSZu5VGED4qAsZqxGXLvDU8SDX76Eo3G
                                oLBqYJGXG0inRVAQViFNk9ovtA2kZSP4oHTsWBCcnKDx
                                8CRZhF9jrbplzUdW7ahbTRaYv6tPYaKd7nzvH8GksO9b
                                9t5EJ1o1BhL6cDOHXOrkL+Twhc1Mwg5jBhYWH6r1LaTX
                                w2NQN1hAQzGrAZPt13kP0KQp/ybGhoKtunLxVsFiiDb6
                                4HQs+cRiw2wSf/vYaNVAiYesrMHTpq5BLeWTVrMLk2Kt
                                NtQMV972pOKipT9UN7At7Sugdpbm1g7jQ8G3eKP6iRg5
                                YAOPjbuXFNYjmKG9fMTjQGgs5IHbDVe/W3mPHYS+1970
                                AmNX/momejvYG8NlQykZdrcqPUw5dSAOmUnbehl3wzJJ
                                HIR59FaxeG9bO8vrFBlaRapz1eMtrZbqkYV7P8PnUP/3
                                i7WVGZO5AJxPpxpHaSNbd63d575pQKMQxEU7dPPfQsq8
                                wnNxcCHDixb6EetVcfv9Id9up4v9t0O33O4562kN2S9/
                                cwPOOlnobJ/g597cU2sF5Pmxn+r7
                                ) ; KSK; alg = RSASHA1; key id = 34816
```

On voit dans l'exemple ci-dessus qu'il y a deux clés ZSK. Le script enregistre donc les IDs des deux clés afin d'identifier laquelle est utilisé pour signer l'enregistrement RRSIG de type DNSKEY (une seule clé est utilisé pour signer les enregistrements d'un même type). Une fois l'ID de la clé ZSK ou KSK récupéré, le script cherche parmi les enregistrements RRSIG la date de validité correspondante et la retourne.

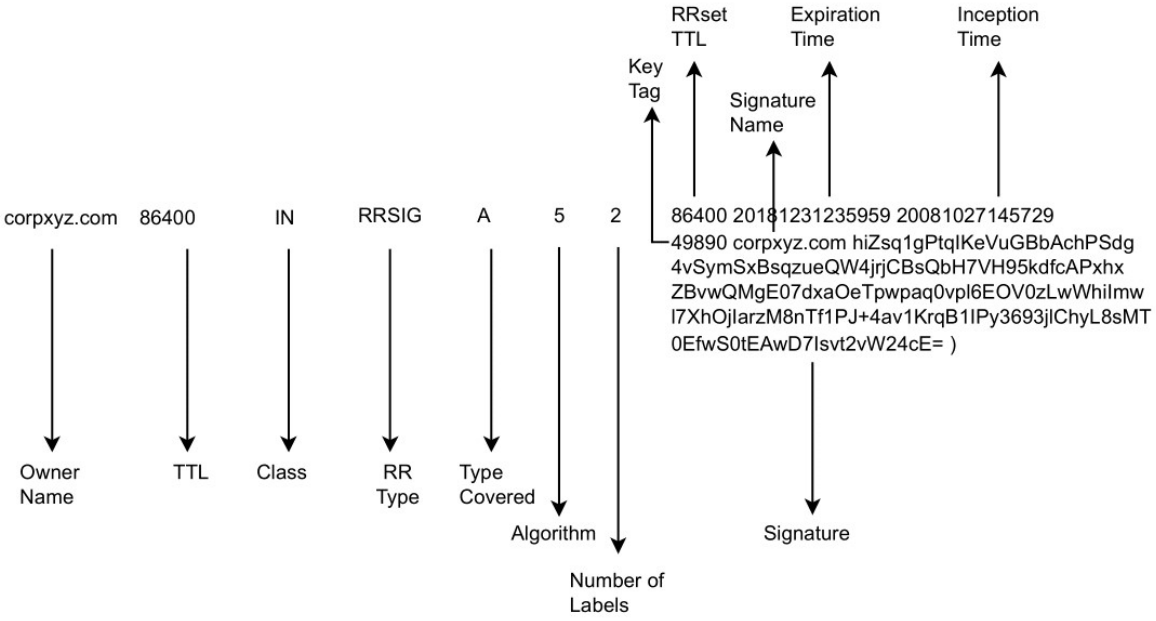

FIGURE 3.6 – Exemple d'un enregistrement RRSIG de type A

L'utilisation du script se fait de la manière suivante :

```
./check\_dnssec.sh <domain> <key\_type>
key_type: KSK ou ZSK
```

Le résultat suivant est obtenu (ici formaté pour une utilisation avec Nagios) :

```
root@stargate:~/PFE_IMA_5 ./check_dnssec.sh plil.fr ZSK
Key OK - ZSK Expiration Date = 2018/12/21 11:12:57 | 20181221111257
```

3.2.3 État des disques du serveur de sauvegarde baleine

Afin de vérifier l'état de santé des disques durs, un script fonctionnant grâce à l'outil smartctl à été mis en place.

Exemple de résultat pour la commande : smartctl -ad cciss,[0-7] /dev/cciss/c0d1 :

```
=== START OF INFORMATION SECTION ===
Vendor:               HP
Product:              DG146A4960
Revision:             HPDB
User Capacity:        146 815 737 856 bytes [146 GB]
...
=== START OF READ SMART DATA SECTION ===
SMART Health Status: OK
Current Drive Temperature:     35 C
Drive Trip Temperature:        70 C
```

```
Manufactured in week 34 of year 2007
Specified cycle count over device lifetime:  50000
Accumulated start-stop cycles:  118
Elements in grown defect list: 0
Vendor (Seagate) cache information
  Blocks sent to initiator = 6238686454743040
Error counter log:
           Errors Corrected by           Total   Correction     Gigabytes    Total
               ECC          rereads/    errors   algorithm      processed    uncorrected
           fast | delayed   rewrites  corrected  invocations   [10^9 bytes]  errors
read:          0        0         0         0          0          0,000           0
write:         0        0         0         0          0          0,000           0
Non-medium error count:      403
SMART Self-test log
Num  Test              Status                 segment  LifeTime  LBA_first_err [SK ASC ASQ]
     Description                              number   (hours)
# 1  Background long   Completed                  -    1819                 - [-   -    -]
# 2  Background short  Completed                  -    1819                 - [-   -    -]
```

On observe, en plus d'une indication sur l'état de santé du disque (SMART Health Status : OK), diverses informations telles que la température ou encore les dysfonctionnements (Elements in grown defect list).

Pour les disques de la baie DAS, on récupère des informations plus détaillées sur l'état de santé des disques, comme la valeur *Current_Pending_Sector*, qui indique le nombre de secteurs du disque qui ne peuvent plus être lus et qui sont en attente d'être remappé.

Commande : smartctl -ad cciss,[8-19] /dev/cciss/c0d1 :

```
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x002f   200   200   051    Pre-fail  Always       -       0
  3 Spin_Up_Time            0x0027   253   253   021    Pre-fail  Always       -       1150
  4 Start_Stop_Count        0x0032   100   100   000    Old_age   Always       -       6
  5 Reallocated_Sector_Ct   0x0033   200   200   140    Pre-fail  Always       -       0
  7 Seek_Error_Rate         0x002f   200   200   051    Pre-fail  Always       -       0
  9 Power_On_Hours          0x0032   034   034   000    Old_age   Always       -       48633
 10 Spin_Retry_Count        0x0033   100   253   051    Pre-fail  Always       -       0
 11 Calibration_Retry_Count 0x0033   100   253   051    Pre-fail  Always       -       0
 12 Power_Cycle_Count       0x0032   100   100   000    Old_age   Always       -       6
184 End-to-End_Error        0x0033   100   100   097    Pre-fail  Always       -       0
187 Reported_Uncorrect      0x0032   100   100   000    Old_age   Always       -       0
188 Command_Timeout         0x0032   100   100   000    Old_age   Always       -       0
190 Airflow_Temperature_Cel 0x0022   067   050   045    Old_age   Always       -       33
192 Power-Off_Retract_Count 0x0032   200   200   000    Old_age   Always       -       5
193 Load_Cycle_Count        0x0032   200   200   000    Old_age   Always       -       0
194 Temperature_Celsius     0x0022   117   100   000    Old_age   Always       -       33
196 Reallocated_Event_Count 0x0032   200   200   000    Old_age   Always       -       0
197 Current_Pending_Sector  0x0032   200   200   000    Old_age   Always       -       0
198 Offline_Uncorrectable   0x0030   200   200   000    Old_age   Offline      -       0
199 UDMA_CRC_Error_Count    0x0032   200   200   000    Old_age   Always       -       0
200 Multi_Zone_Error_Rate   0x0008   200   200   000    Old_age   Offline      -       0
```

L'idée pour la réalisation de ce script fut donc de reprendre cette commande afin d'avoir un retour sur l'état de santé du disque.

Le script vérifie dans un premier temps l'état de santé général du disque en vérifiant le résultat du test SMART (PASSED ou OK) et, dans un second temps, vérifie la présence ou non dysfonctionnement (Pre-Fail, Elements in grown defect list).

Selon l'état de santé du disque, le script retourne les outputs suivants :

NB : Le chiffre passé en argument représente le numéro de slot du disque (de 0 à 7, les disques sur le serveur baleine, et de 8 à 19, les disques durs de la baie DAS).

```
root@baleine:/usr/local/nagios/libexec# ./smart_check_disk.sh 0
Utilisation ./smart_check_disk.sh [0-19]
DISQUE OK - Aucune erreur
DISQUE OK - Attention, disque endommagé
ERREUR DISQUE
```

3.2.4 État des réseau ADSL, SDSL et Renater

Afin de pouvoir tester l'état de santé des réseaux ADSL, SDSL et Renater, une machine virtuelle nommé *stargate* à été mise en place sur le serveur *brisban*.

La machine virtuelle est routé aux trois serveurs via les interfaces suivantes :

— eth0 - 192.168.1.111 - Réseau ADSL
— eth1 - 2001 :660 :4401 :6011 :216 :3eff :fe47 :ba1f - Réseau Renater
— eth2 - 2001 :7a8 :116e :47 :216 :3eff :fe47 :ba20 - Réseau SDSL
— eth3 - Réseau Renater

Table de routage ipv4 :

```
default via 192.168.1.253 dev eth0
192.168.1.0/24 dev eth0  proto kernel  scope link  src 192.168.1.111
```

Table de routage ipv6 :

```
2001:660:4401:6011::/64 dev eth1  proto kernel  metric 256  expires 923sec
2001:660:4401:6048::/64 dev eth3  proto kernel  metric 256  expires 963sec
2001:7a8:116e:47::/64 dev eth2  proto kernel  metric 256  expires 982sec
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
fe80::/64 dev eth3  proto kernel  metric 256
fe80::/64 dev eth0  proto kernel  metric 256
default via fe80::211:5dff:fef2:5400 dev eth1  proto ra  metric 1024
expires 1723sec hoplimit 64
default via fe80::4e4e:35ff:fe5e:b943 dev eth2  proto ra  metric 1024
expires 1782sec hoplimit 64
default via fe80::211:5dff:fef2:5400 dev eth3  proto ra  metric 1024
expires 1763sec hoplimit 64
```

Pour la réalisation du script permettant de récupérer l'état de santé des réseaux ADSL, SDSL et Renater, la "difficulté" a surtout résidé dans le fait de forcer l'utilisation d'une interface pour pouvoir tester un réseau en particulier.

Étant donné que l'interface eth0 est la seule à posséder une adresse en ipv4, un simple ping ipv4 vers des machines distantes permet de tester l'état du réseau ADSL.

Pour tester les réseaux SDSL et Renater, il faut faire en sorte d'utiliser uniquement l'une de ces interfaces. On peut observer qu'en pingant une adresse en ipv6 depuis l'interface eth2, les paquets sont envoyés depuis l'adresses ipv6 de l'interface eth1 (Renater) :

```
root@stargate:~# ping6 -I eth2 2001:4860:4860::8888
ping6: Warning: source address might be selected on device other than eth2.
PING 2001:4860:4860::8888(2001:4860:4860::8888) from 2001:660:4401:6011:216:3eff:fe47:ba1f
eth2: 56 data  bytes
64 bytes from 2001:4860:4860::8888: icmp_seq=1 ttl=55 time=5.13 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=2 ttl=55 time=5.55 ms
64 bytes from 2001:4860:4860::8888: icmp_seq=3 ttl=55 time=5.19 ms
```

Afin de résoudre ce problème, on spécifie explicitement l'adresse IP source dans les paramètres du ping6.

Dans l'exemple ci-dessous, on tente de pinger un serveur distant depuis l'interface eth2 (réseau SDSL). On constate que les echo requests sont toujours envoyés via l'interface eth1 :

```
root@stargate:~# tcpdump -i eth1 icmp6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
16:36:49.282006 IP6 2001:7a8:116e:47:216:3eff:fe47:ba20 > google-public-dns-a.google.com:
ICMP6, echo request, seq 15, length 64
16:36:50.283237 IP6 2001:7a8:116e:47:216:3eff:fe47:ba20 > google-public-dns-a.google.com:
ICMP6, echo request, seq 16, length 64
16:36:51.284455 IP6 2001:7a8:116e:47:216:3eff:fe47:ba20 > google-public-dns-a.google.com:
ICMP6, echo request, seq 17, length 64
```

Les echo reply sont cependant reçus sur l'interface eth2, via le réseau SDSL :

```
root@stargate:~# tcpdump -i eth2 icmp6
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
16:32:17.341902 IP6 google-public-dns-a.google.com > 2001:7a8:116e:47:216:3eff:fe47:ba20:
ICMP6, echo  reply, seq 4, length 64
16:32:18.343265 IP6 google-public-dns-a.google.com > 2001:7a8:116e:47:216:3eff:fe47:ba20:
ICMP6, echo reply, seq 5, length 64
16:32:19.344553 IP6 google-public-dns-a.google.com > 2001:7a8:116e:47:216:3eff:fe47:ba20:
ICMP6, echo reply, seq 6, length 64
```

Des tests complémentaires ont ensuite été fait directement en salle serveur en débranchant les liaisons ADSL, SDSL et Renater pour simuler un dysfonctionnement.

Un script renvoyant l'état des réseaux ADSL, SDSL et Renater a finalement été rédigé :

```
usage: ./check_ping.sh <Network>
Network : choose: ADSL, SDSL or Renater
root@stargate:~/PFE_IMA_5# ./check_ping.sh ADSL
Connexion ADSL OK - 10/10 paquets reçus
```

3.2.5 Autres scripts

En plus de ce qui a déjà été évoqué, du temps a été passé sur des solutions qui n'ont finalement pas été retenues.

Script de vérification de température

Un script permettant de relever la température d'un CPU a été rédigé en reprenant un plugin déjà existant. Des modifications ont été faite afin de pouvoir avoir un retour dans le cas d'un serveur possédant plusieurs processeurs, comme c'est le cas sur baleine.

La solution a finalement été abandonné après l'installation de Prometheus puisqu'une sonde parvenant au même résultat est déjà présente dans le Node_eporter.

Script pour sauvegarde automatique

Un script permettant de sauvegarder automatiquement l'image d'une machine virtuelle a aussi été rédigé. Le script a par la suite été abandonné après la mise à disposition, par mon tuteur de projet, d'un meilleur script permettant de parvenir au même résultat.

Sauvegarde incrémentielle

Dans la même lignée, la solution de sauvegarde incrémentielle *rsnapshot* a aussi été testé sur la VM de test *vmtest*, puis abandonnée en raison de sa non conformité au cahier des charge (car nécessitant trop de configuration).

Chapitre 4

Présentation du travail restant et du planning

4.1 Travail restant

En ce qui concerne la supervision des serveurs, les tâches suivantes restent à faire :
— Liaison réseau entre la VM stargate et la VM de supervision (à voir avec le service informatique) et intégration de ces script à grafana.
— Configuration du système d'alerte.

Concernant le système de sauvegarde automatique, tout reste à faire, des premiers test réalisé avec rsync ont cependant été réalisé.

4.2 Planning

Dans l'idéal, toutes les tâches restantes devront être finies d'ici la fin du mois de Janvier voir en début février.

Une réunion avec les tuteurs de projet devra ensuite se faire afin de déterminer d'autres missions.