Antoine Barbare & Grégoire Martinache

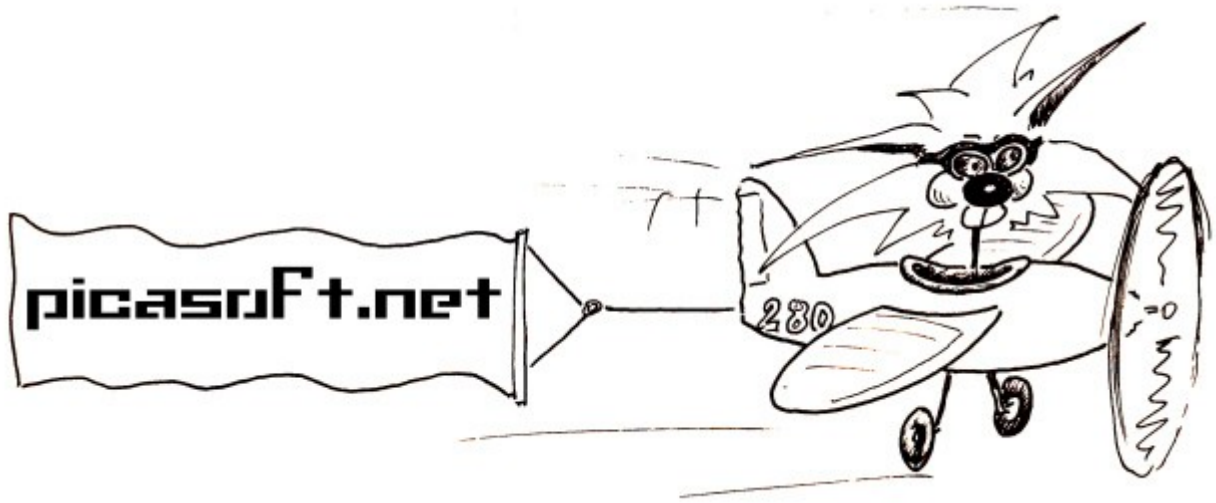
Rapport de TX « Chatons »

Le 8 janvier 2017

Mise en place d'une architecture expérimentale pour héberger des services dans le cadre du mouvement de redécentralisation d'Internet

Table des matières

1) Picasoft



L'objectif du projet Picasoft est de s'inscrire dans la mouvance de redécentralisation d'internet illustrée par la campagne Dégooglisons Internet initiée par Framasoft, et en particulier de participer à l'initiative CHATONS (Collectif des Hébergeurs Alternatifs, Transparents, Ouverts, Neutres et Solidaires).

Pourquoi ?

- Parce que la maîtrise des outils numériques est au cœur des enjeux de la formation de tous les ingénieurs
- Parce que l'étude et la maîtrise de l'hébergement de service est au cœur des enjeux des ingénieurs en informatique
- Parce que l'UTC travaille sur la littératie numérique, c'est à dire la capacité pour chacun de se repérer et s'adapter dans le monde numérique, et que au sein de cette problématique, des questions comme la maîtrise de ses données ou de son pouvoir de citoyen sont essentielles.

Comment ?

- Évangéliser : Organiser et participer à des conférences et ateliers
- Héberger : Proposer un service d'hébergement public géré par les étudiants de l'UTC
- Former : Faire connaître et former aux alternatives existantes

2) Introduction

Contexte :

Le but de la TX CHATONS est de valider ou non la viabilité de la mise en place de services web décentralisés par des étudiants.

Pour ce faire, nous avons décidé de choisir des services web décentralisés des géants du web (GAFAM) et d'héberger ces services chez un hébergeur qui respecte la vie privée de ses utilisateurs et qui a un côté éthique dans le service qu'il rend.

Cette TX a pour inspiration principale ce qui est actuellement effectué par l'association framasoft qui depuis maintenant 2 ans sélectionne et héberge des services web qui permettent de se substituer aux outils des GAFAM. Aujourd'hui Framasoft a décidé de ne plus être l'unique “fournisseur” concernant l'hébergement de ces services. C'est alors que l'initiative CHATONS est née (voir https://CHATONS.org/). Elle vise à mettre en avant des hébergeurs respectant les données des utilisateurs et ayant des principes éthiques et moraux concernant le service qu'ils rendent. L'objectif d'un CHATONS est également d'informer les utilisateurs sur tous les risques qui menacent nos libertés et leur permettre de s'en prémunir.

La TX à été initié par Stéphane Crozat. Elle a été conduite par Antoine Barbare et Grégoire Martinache.

Problématiques :

- Étudier les différentes options. Nous devons apporter une réflexion sur les décisions qui sont prises : choix de l'architecture, des services à proposer … Nous sommes partie prenante de la construction technologique, et, à ce titre, nous ne devons pas être esclave d'un choix technologique particulier.

- Se former par l'expérience. En nous dotant d'un projet et d'une infrastucture, nous pouvons nous former à la maitrîse du numérique et ainsi proposer aux autres étudiants en informatique d'apprendre à connaitre les systèmes et réseaux.

- L'enjeu sociétal : Il est aujourd'hui important pour les ingénieurs non informaticiens d'avoir la capacité de faire des choix entre les outils en comprenant les enjeux de ces outils dont le métier dépend totalement.

État de l'art :

Actuellement, différentes plateformes proposent des services se substituant aux services des GAFAM. D'autres services tels que Framasoft offrent en plus de la promotion et de la diffusion de logiciels libres un service d’hébergement permettant d'utiliser les outils sur leurs propres serveurs.

Le lancement du collectif CHATONS a également changé la donne et a lancé de nombreuses initiatives à travers la France. Par exemple, le collectif CHATONS à Orléans (installation technique) envisage de faire peu à peu la même chose que nous, en utilisant les mêmes technologies (Proxmox, Docker…)

Réalisations :

Dans ce rapport, toutes les informations concernant la réalisation du projet seront scindées en deux parties :

- Notre démarche lors de l'élaboration de Picasoft en expliquant les choix qui ont été fait : Démarche
- La documentation technique destinée aux personnes chargées de pérenniser le travail de Picasoft : Passation

L'aboutissement de ce travail à mené à la mise en place de différents services :

Etherpad

Site officiel de etherpad.
Disponible sur https://pad.picasoft.net/p/pads

Etherpad est un éditeur de texte libre en ligne fonctionnant en mode collaboratif et en temps réel.

Il permet à plusieurs personnes de partager l'élaboration simultanée d'un texte, et d'en discuter en parallèle, via une messagerie instantanée.

Pour l'association Picasoft, Etherpad a été le support de tous nos comptes rendus de réunions et permet régulièrement de réaliser des documents de facon collaborative.

Article de Framasoft : https://framasoft.org/article5066.html

Mattermost

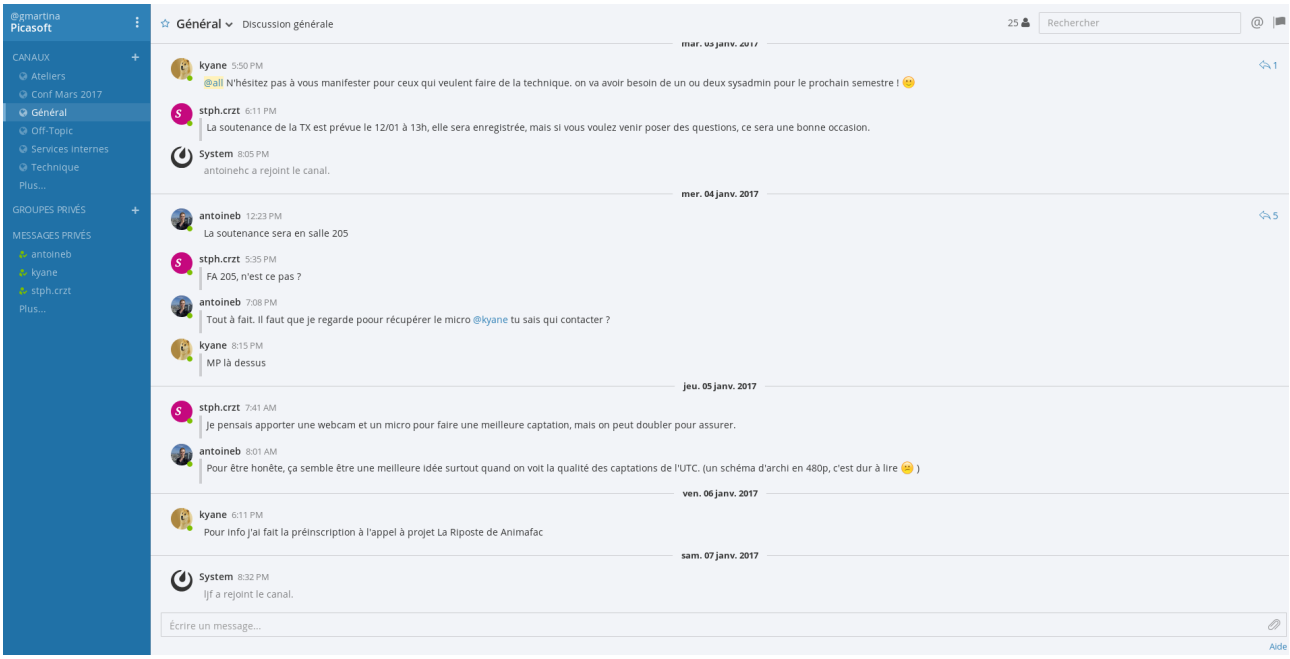
Site officiel de mattermost.
Disponible sur https://team.picasoft.net

Mattermost est un clone de Slack, open source, sous licence MIT.

Le Mattermost Picasoft est le mode de communication privilégié au sein de l'association.

Article de framasoft : https://framablog.org/2016/05/10/framateam-liberez-vos-equipes-des-groupes-facebook-et-de-slack/

Mattermost sur Picasoft :



Khanboard

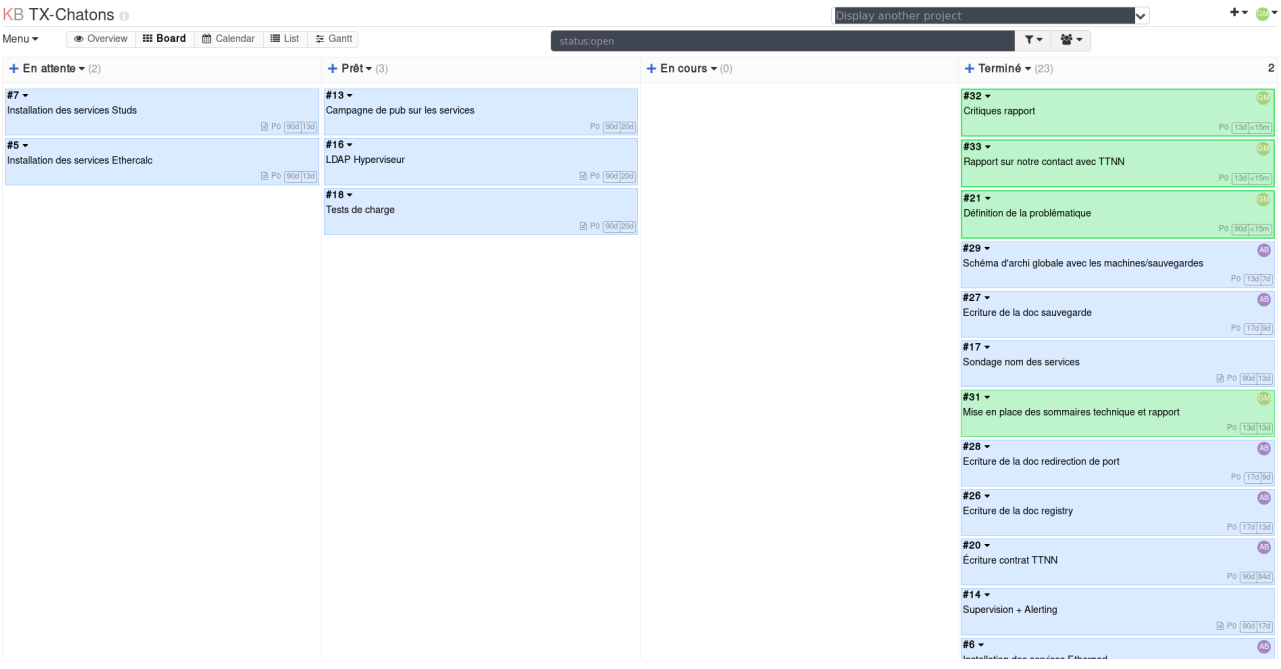
Site officiel de kanboard.
Disponible sur https://backlog.picasoft.net/ (Accès limité aux membres de Picasoft)

Kanboard est un gestionnaire de tâches visuel qui permet de gérer facilement des petits projets de manière collaborative

Au sein de Picasoft, Kanboard est utilisé suivre facilement l'ensemble des tâches effectuées et à venir.

Article de framasoft : https://framablog.org/2015/10/07/framaboard-les-libristes-ont-reinvente-le-tableau-blanc-mais-en-mieux/

A titre d'exemple, le Kanboard de Picasoft à la fin de la TX :



3) Démarche

Organisation au long du semestre

- Une réunion par semaine avec les membres de la TX et les suiveurs;
- Créneau du jeudi à partir de 12h30 et possibilité de la suivre à distance via des outils de téléconférence lors de l'indisponibilité d'un membre à se déplacer;
- Ordre du jour (ODJ) et compte-rendu (CR) pour chaque réunion, liste disponible ici;
- Organisation “agile” avec l'utilisation de Kanboard (voir ici). Disponible ici : https://backlog.picasoft.net/ (identifiants disponible ici).
- Communication via notre mattermost;
- Utilisation du GitLab UTC. Le repo tx-chatons contient les images docker mises en place dans le registry Docker ainsi que différents scripts de sauvegardes.

La communication par IRC a été abordée, mais l'impossibilité de voir l'historique de ce qui a été dit nous a amenés à abandonner cette solution au profit de mattermost.

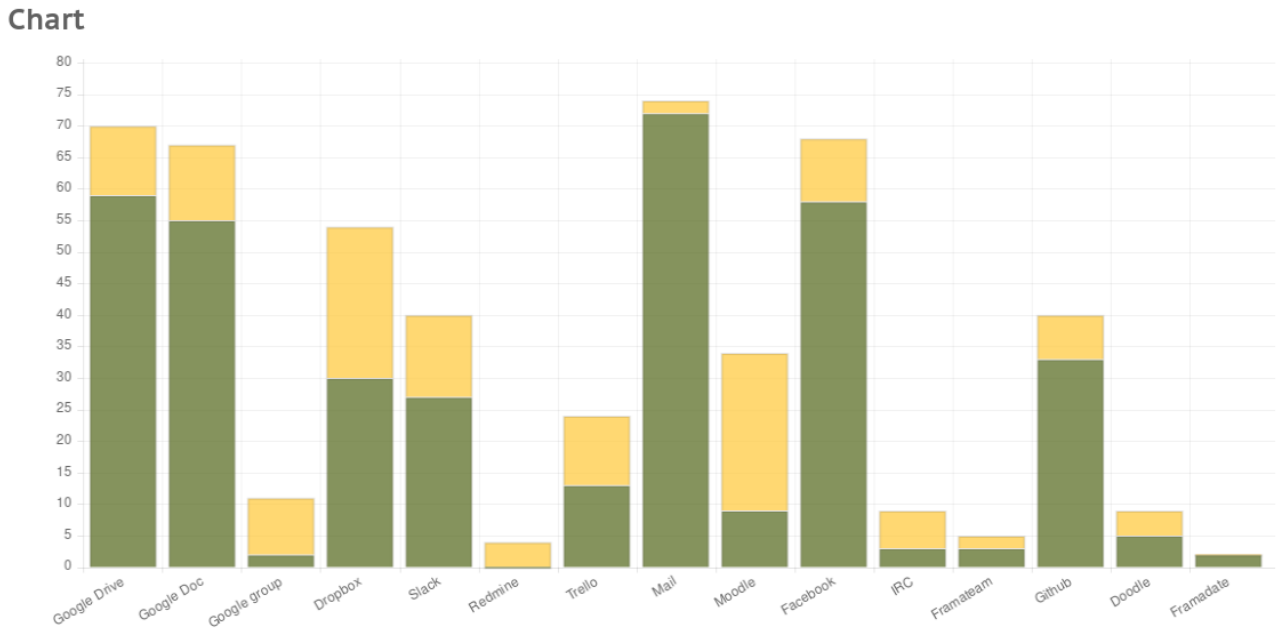
Problématiques

Choix des services

Afin de déterminer les services qui seront proposés, nous avons d'abord mené une étude fonctionnelle et technique des outils disponibles. (Voir le comparatif).

Les points forts et les points faibles de chacune des solutions ayant été abordés, nous avons noté un fort besoin dans un système de gestion de projet. Pour confirmer ces observations, nous avons mis en place un sondage lors de la journée des étudiants pour connaître les outils utilisés lors de la conduite d'un projet.

Le sondage à obtenu 75 réponses donnant les résultats suivants :



Résultat sondage

Nous avons alors remarqué que de nombreuses personnes utilisent Facebook ou Gmail comme outil de travail, qui sont des outils inadaptés mais qui provoquent également un mélange entre la vie privée et la vie scolaire.
Pour toutes ces raisons, notre choix s'est porté sur la proposition de différents services permettant la gestion de projet.

Choix de l'hébergement

Une autre problématique du projet a été de choisir la façon d'héberger les différents services en production :

Possibilités :

- Achat serveur physique : budget disponible;
- Location à moyen/long terme : difficile chez des grands hébergeurs;
- Location à une association;
- Hébergement chez Rhizome : écarté, car la machine utilisée actuellement n'a pas les ressources suffisantes.

Afin d'accomplir nos objectifs, nous avons d'abord décidé d'être hébergé chez Tetaneutral, qui s'inscrit dans la démarche de neutralité et de décentralisation du web que nous soutenons.

Détails des conditions de l'hébergement :

```
2 machines  de configuration
4 coeurs, 32G de Ram, RAID SSD 500G, RAID HDD 2T
5IP fixes en plus des 2 par machines  17.5€/mois
Cotisation 20€/an
Hébergement 25€/machines /mois
TOTAL:  4663€ sur 3 ans
```

Devenir CHATONS

On mentionnera également une autre problématique qui consistait à savoir si nous allions nous inscrire en temps que CHATONS officiel sur le site de Framasoft.

Deux possibilités :

- Soit devenir un vrai CHATONS au sens Frama (ouverture totale et une certaine redondance avec l'UTC possible)
- Soit devenir un CHATONS au sein de l'UTC, alors il faut prendre en compte ce qui existe à l'UTC.

Nous avons décidé de former un vrai CHATONS :

- La reconnaissance d'un travail abouti;
- Picasoft pourra profiter à des gens dans un cercle plus large;

Pour la TX, terminer en étant CHATONS est vraiment valorisant.

4) Comparatif des services abordés

Big Blue Button

Pour faire quoi ?

Pour proposer une alternative à Skype et permettre aux étudiants/professeurs de discuter en visioconférence.

Points positifs

- Facile à utiliser
- De nombreuses fonctions pour l'éducation (tableau blanc, pdf, chat ...)
- Fonctionne avec Moodle

Points négatifs

- Peu d'étudiants utilisent régulièrement la visioconférence
- Trop ciblé sur l'éducation et la formation, ne remplace pas Skype pour discuter avec ses proches

Messagerie instantanée (Zulip)

Pour faire quoi ?

Pour remplacer What's App & les SMS

Points positifs

- Touche un grand nombre de personnes

Points négatifs

- Necessite que tous les contacts migrent
- Difficulté d'entretien

Mail

Pour faire quoi ?

Pour proposer un serveur Mail sécurisé et respectueux de la vie privée.

Points positifs

- Touche un grand nombre de personnes
- Beaucoup de gens utilisent Gmail faute d'une autre solution

Points négatifs

- Une mise en place trop technique
- Une maintenance particulièrement compliquée

OwnCloud

Pour faire quoi ?

Pour proposer une alternative à Google Drive, Dropbox ...

Points positifs

- Service populaire

Points négatifs

- Un Owncloud est déjà en place à l'UTC
- Demande une grande capacité de stockage si le service est populaire

Gitlab

Points positifs

- Service populaire parmi les GI

Points négatifs

- Un Gitlab est déjà en place à l'UTC
- Service restreint aux informaticiens

Yunohost

Pour faire quoi ?

Proposer une façon pratique de s'autohébérger

Points positifs

- Permet aux novices de s'autohéberger

Points négatifs

- Pas vraiment un service à proposer, plutôt une idée d'atelier

Diaspora

Pour faire quoi ?

Points positifs

- Plus simple à gérer au niveau des cercles de projet
- Beaucoup de personnes travaillent en groupe via Facebook (Mélange vie privée et travail) : Permettrait de mieux séparer vie privée et scolaire.

Points négatifs

- Difficulté de faire migrer de nouveaux utilisateurs.

L'usage parallèle de Facebook pour la vie privée et de Diaspora pour la vie privée s'avérerait trop lourde pour la plupart des utilisateurs qui cherchent un service centralisé.

5) Services choisis

Ensemble de services pour la gestion de projet

Pour faire quoi ?

Proposer un ensemble de service pour permettre de gérer ses projets en respectant le partage vie privée / vie scolaire.

Services

- Mattermost : En plus de la communication de groupe, il y a la possibilité d'avoir des channels privés et de communiquer par messages privés ce qui permet de segmenter la communication et passer en privé pour faire le point sur quelque chose de précis si nécessaire.

- Framemo +: Simple à utiliser, mais pas de login ni de protection des données.

- Outils de gestion Kanboard : Permets de travailler de façon agile (avec login), mais plus lourd à l'usage de Framémo

- Outils de sondage : De nombreux sondages Google Forms sont présents sur Facebook : Alternative intéressante.

- Etherpad

- Ethercalc

- Zerobin : Idée abandonnée, car usage trop restreint aux GI

Points positifs

- Mode de gestion de projet assez pratique et utilisé par les étudiants. Par rapport à Facebook, il permet de regrouper des personnes autour d'un sujet commun et de se focaliser sur le sujet.

- Pas encore d'outils de ce genre, il serait sans doute rapidement utile pour les professeurs et les étudiants

Points négatifs

Demande la mise en place d'une quantité importante de services

6) Rapport technique

Afin de faciliter les passations futurs et assurer la pérénité du projet Picasoft, il est important que celui-ci soit documenté.

Installation :

Afin de mener à bien ce projet de TX, deux machines physiques ont été achetés. **Alice** et **Bob** c'est leur nom sont actuellement hébergées chez Tetaneutral à Toulouse (en savoir plus) Cette partie décrit les différentes opérations et installations effectuées sur ces machines pour la mise en place de nos premiers services au publique.

Contacts avec TetraNeutral

- Laurent Guerby : Président de l'asso et en charge de l'intendance
- Partie technique (Mailing-list de bénévoles)
- Partie financière (Mailing-list de bénévoles)

Ecriture du contrat :

L'acceptation du devis implique l'acceptation du règlement intérieur : https://chiliproject.tetaneutral.net/projects/tetaneutral/wiki/RI.

Facture :

https://doc.picasoft.net/lib/exe/fetch.php?media=passation:f20161114-0021-adt991.pdf

7) Les machines physiques de Picasoft

Pour la réalisation de cette TX, deux machines physiques ont été achetées (Alice et Bob).

Elles sont actuellement hébergées chez Tetaneutral (à Toulouse) qui est un fournisseur d'accès internet associatif qui donne accès à une connexion neutre et sans filtrage. Le fait d'être hébergé au sein d'une association a également eu un impact sur le choix des machines. En effet, les hébergeurs classiques dans leurs prestations d'housing (hébergement où vous ramenez votre machine) imposent d'avoir des serveurs rackables qui sont le plus souvent coûteux.

Chez Tetaneutral, les contraintes sont moins importante et il nous a été possible de faire héberger des serveurs au format “tour ATX” classique. L'avantage de cette solution est que nous avons pu monter des configurations sur mesure avec des pièces informatiques grand publique. Du fait de cette économie, nous avons eu la possibilité d'investir dans des configurations haut de gamme.

Configuration physique

Carte mère ASRock Q170M vPro
Intel Core i5-6500 (3.2 GHz)
2x Crucial DDR4 16 Go (2 x 8 Go) 2133 MHz CL15 DR X8
FSP RAIDER S550 550W 80PLUS Silver
Cooler Master Elite RC-342
2x WD Red 2 To SATA 6Gb/s
2x Kingston SNA-BR2/35
2x SanDisk SSD Ultra II 480G

Cette combinaison de carte mère et de CPU a été sélectionnée du fait qu'elle est administrable à distance (reboot + VNC depuis l’écran de BIOS) grâce à la technologie Intel VPro.

Voir le bon de commande

8) Installation des machines avec Proxmox

Informations sur l'installation et le déploiement des deux machines physiques chez Tetaneutral (TTNN).

Accès :

La carte mère de chaque machine possèdent une fonction qui permettent d'avoir accès à la machine dès son démarrage (VPro).

Avec le VPro, on a un accès VNC au machine en passant par un rebond sur nagios.tetaneutral.net

```
ssh -p 2222 nagios.tetaneutral.net -l ttnn -N \
-L 8081:192.168.128.151:16992 -L 5911:192.168.128.151:5900 \
-L 8082:192.168.128.152:16992 -L 5912:192.168.128.152:5900
```

On a aussi une redirection vers le ssh des ip privée des machines en passant par

```
ssh -p 65055 root@nagios.tetaneutral.net pour alice (détail nohup
socat  tcp4-listen:65055,reuseaddr,fork tcp6:
[fe80::d250:99ff:fe8b:1b5d%eth0.128]:22 >& /dev/null < /dev/null&)
ssh -p 65056 root@nagios.tetaneutral.net pour bob
```

Avec un navigateur les URLs suivantes donnent accès a divers outils :

Interface web d'alice pour l'inventaire et le lancement des commandes de power on/off

```
http://localhost:8081
```

Même chose pour bob

```
http://localhost:8082
```

Accès VNC remote alice

```
remote-viewer vnc://localhost:5911
```

Même chose pour bob

```
remote-viewer vnc://localhost:5912
```

Procédure d'installation:

L'installation de la machine passe par PXE en utilisant https://netboot.xyz/ derrière.

Installation:

Choisir debian jessie, installeur mode graphique (en mode texte le vPro finit par planter et freezer la machine hard, les gens chez intel doivent tester uniquement les install windows graphique ...)

Pendant l'install la carte eth0 (I219LM) est celle qui a le DHCP qui donne accès au net temporairement.

Apres l'install la carte eth1 (82572EI) est celle qui portera les ipv4/v6 publiques, eth0 sera “down”, réservé uniquement vPro et pas sur le net.

Configuration réseau:

Alice :

```
A 91.224.148.84
AAAA 2a03:7220:8080:5400::1

for i in /proc/sys/net/ipv6/conf/*; do \
  for j in autoconf accept_ra; do echo 0 > $i/$j; done;done

ip link set eth1 up
ip addr add 91.224.148.84/32 dev eth1
ip route add default via 91.224.148.0 dev eth1 onlink
ip -6 addr add 2a03:7220:8080:5400::1/56 dev eth1
ip -6 addr add fe80::80:54/64 dev eth1
ip -6 route add default via fe80::31 dev eth1
echo nameserver 91.224.148.10 > /etc/resolv.conf
echo nameserver 91.224.149.254 >> /etc/resolv.conf
```

Bob :

```
A 91.224.148.85 bob
AAAA 2a03:7220:8080:5500::1

for i in /proc/sys/net/ipv6/conf/*; do \
  for j in autoconf accept_ra; do echo 0 > $i/$j; done;done

ip link set eth1 up
ip addr add 91.224.148.85/32 dev eth1
ip route add default via 91.224.148.0 dev eth1 onlink
ip -6 addr add 2a03:7220:8080:5500::1/56 dev eth1
ip -6 addr add fe80::80:55/64 dev eth1
ip -6 route add default via fe80::31 dev eth1
```

```
echo nameserver 91.224.148.10 > /etc/resolv.conf
echo nameserver 91.224.149.254 >> /etc/resolv.conf
```

IP supplémentaires:

Elles sont routées de la même manière que vos premières IP, en /32 et /56 via fe80::... :

```
    alice2 IN A 91.224.148.57
    alice2 IN AAAA 2a03:7220:8080:3900::1
    # fe80::80:39
    alice3 IN A 91.224.148.58
    alice3 IN AAAA 2a03:7220:8080:3a00::1
    # fe80::80:3a
    alice4 IN A 91.224.148.59
    alice4 IN AAAA 2a03:7220:8080:3b00::1
    # fe80::80:3b
    bob2 IN A 91.224.148.60
    bob2 IN AAAA 2a03:7220:8080:3c00::1
    # fe80::80:3c
    bob2 IN A 91.224.148.61
    bob2 IN AAAA 2a03:7220:8080:3d00::1
    # fe80::80:3d
```

Installation :

Au niveau de l'installation, on installe dans un premier temps un jessie “classique”. Pour les disques, on partitionne comme suit:

SSD1:

```
sda1 200M configurée en RAID
sda2 500G configurée en RAID
```

SSD2:

```
sdb1 200M configurée en RAID
sdb2 500G configurée en RAID
```

Installation RAID:

2 périphériques multidisques sda1/sdb1 et sda2/sdb2

```
Raid de 200M => utilisé comme ext2 et monté sur le /boot
Raid de 500G => utilisé comme périphérique lvm
```

Lvm:

```
Création d'un groupe logique nommé pve
Création de 3 volumes logique
  data => ext4 monté /var/lib/vz
  root => ext4 monté /
  swap => utilisé comme swap
```

Installation classique pour la suite. Ne pas oublier d'installer le serveur ssh Pour grub, on l'installe manuellement sur sda

Après le premier reboot, on install grub sur sdb grub-install /dev/sdb On modifie le /etc/default/grub en ajoutant i915.modset=0 à la ligne GRUB_CMDLINE_LINUX_DEFAULT

Pour la configuration réseau, on n'utilise pas le fichier /etc/network/interface et on passe tout ça dans le rc.local. En cas de pb sur une interface, on ne foire pas toutes les connexions de la machine. Montage des ip de production dans le rc.local

```
root@alice:~# cat /etc/rc.local

#!/bin/sh

sleep 3
for i in /proc/sys/net/ipv6/conf/*; do \
  for j in autoconf accept_ra; do echo 0 > $i/$j; done;done

ip link set eth0 up
ip link set eth1 up
ip addr add 91.224.148.84/32 dev eth1
ip route add default via 91.224.148.0 dev eth1 onlink
ip -6 addr add 2a03:7220:8080:5400::1/56 dev eth1
ip -6 addr add fe80::80:54/64 dev eth1
ip -6 route add default via fe80::31 dev eth1

echo nameserver 91.224.148.10 > /etc/resolv.conf
echo nameserver 91.224.149.254 >> /etc/resolv.conf

exit 0
```

Installation de Proxmox :

En suivant la doc d'install
https://pve.proxmox.com/wiki/Install_Proxmox_VE_on_Debian_Jessie

Installation du RAID sur les HDD:

Récupération des id des disques via un fdisk

```
root@alice:~# fdisk -l | grep s
    Disque /dev/sda : 447,1 GiB, 480103981056 octets, 937703088
secteurs
    /dev/sda1  *      2048    391167    389120  190M fd Linux raid
autodetect
    /dev/sda2       391168 937701375 937310208  447G fd Linux raid
autodetect

    Disque /dev/sdb : 447,1 GiB, 480103981056 octets, 937703088
secteurs
    /dev/sdb1         2048    391167    389120  190M fd Linux raid
autodetect
    /dev/sdb2       391168 937701375 937310208  447G fd Linux raid
autodetect

    Disque /dev/sdc : 1,8 TiB, 2000398934016 octets, 3907029168
secteurs

    Disque /dev/sdd : 1,8 TiB, 2000398934016 octets, 3907029168
secteurs

fdisk /dev/sdc

    Commande (m pour l'aide) : n
    Commande (m pour l'aide) : t
    Type de partition (taper L pour afficher tous les types) : 21
    Commande (m pour l'aide) : w

fdisk /dev/sdd

    Commande (m pour l'aide) : n
    Commande (m pour l'aide) : t
    Type de partition (taper L pour afficher tous les types) : 21
    Commande (m pour l'aide) : w

root@alice:~# fdisk -l | grep sd

    Disque /dev/sda : 447,1 GiB, 480103981056 octets, 937703088
secteurs
    /dev/sda1  *      2048    391167    389120  190M fd Linux raid
autodetect
    /dev/sda2       391168 937701375 937310208  447G fd Linux raid
autodetect

    Disque /dev/sdb : 447,1 GiB, 480103981056 octets, 937703088
secteurs
    /dev/sdb1         2048    391167    389120  190M fd Linux raid
```

```
autodetect
    /dev/sdb2       391168 937701375 937310208  447G fd Linux raid
autodetect

    Disque /dev/sdc : 1,8 TiB, 2000398934016 octets, 3907029168
secteurs
    /dev/sdc1   2048 3907029134 3907027087  1,8T Linux RAID

    Disque /dev/sdd : 1,8 TiB, 2000398934016 octets, 3907029168
secteurs
    /dev/sdd1   2048 3907029134 3907027087  1,8T Linux RAID
```

Création du raid :

```
mdadm --create --verbose /dev/md2 --level=mirror --raid-
devices=2 /dev/sdc1 /dev/sdd1
```

Infos sur la synchro du raid :

```
root@alice:~# cat /proc/mdstat

    Personalities : [raid1]

    md2 : active raid1 sdd1[1] sdc1[0]
          1953382464 blocks super 1.2 [2/2] [UU]
          [>.....................]  resync =  0.2%
(4435840/1953382464) finish=205.0min speed=158422K/sec
          bitmap: 15/15 pages [60KB], 65536KB chunk

    md1 : active raid1 sda2[0] sdb2[1]
          468524032 blocks super 1.2 [2/2] [UU]
          bitmap: 1/4 pages [4KB], 65536KB chunk

    md0 : active raid1 sda1[0] sdb1[1]
          194368 blocks super 1.2 [2/2] [UU]
```

Infos sur le raid :

```
    root@alice:~# mdadm --detail /dev/md2

    /dev/md2:
            Version : 1.2
      Creation Time : Fri Nov 11 16:45:03 2016
         Raid Level : raid1
         Array Size : 1953382464 (1862.89 GiB 2000.26 GB)
      Used Dev Size : 1953382464 (1862.89 GiB 2000.26 GB)
       Raid Devices : 2
```

```
      Total Devices : 2
        Persistence : Superblock is persistent
      Intent Bitmap : Internal
        Update Time : Fri Nov 11 16:46:43 2016
              State : clean, resyncing
     Active Devices : 2
    Working Devices : 2
     Failed Devices : 0
      Spare Devices : 0
      Resync Status : 0% complete
               Name : alice:2  (local to host alice)
               UUID : 2df4f5fb:b5c74d03:7ba276ba:42ea760d
             Events : 20
        Number   Major   Minor   RaidDevice State

           0       8       33        0      active sync
/dev/sdc1

           1       8       49        1      active sync
/dev/sdd1
```

Sauvegarde de la configuration:

```
mdadm --detail --scan => on récupère la ligne correspondante au
nouveau raid et on l'ajoute au fichier /etc/mdadm/mdadm.conf
```

9) Mise en place d’un cluster 2 nœuds sous Proxmox

La mise en place d'un cluster Proxmox peut poser quelques problèmes si l'on a jamais eu à travailler avec des outils tels que corosync ou heartbeat. Ces outils permettent de détecter les différentes machines et d'effectuer des actions si l'une d'entre elles venait à défaillir.

Je suppose qu'avant d'en arriver à ce stade, vous avez monté deux machines Proxmox indépendantes (voir article)

Une chose à savoir est que lorsque l'on met en place un cluster Proxmox, celui-ci n'est possible que par le biais d'IP de service dans un range privé. Corosync lors de la détection des différents noeuds va lancer des commandes de ping unicast à travers le réseau et il est préférable que cela se fasse sur un réseau privé.

La première chose à faire est de s'assurer que les fichiers etc/hosts sont correctement renseignés avec les différents IP privées des machines. Par exemple:

```
root@alice:~# cat /etc/hosts
127.0.0.1 localhost
192.168.10.1 alice
192.168.10.2 bob
```

On peut ensuite vérifier que le unicast fonctionne correctement entre les 2 noeuds sur le hostname Démarrer omping sur les 2 noeuds en inversant les noeuds

```
Sur le alice : # omping alice bob
Affiche :  alice : waiting for response msg
```

```
Sur le bob : # omping bob alice
Affiche sur les 2 noeuds, si tout fonctionne : unicast, seq=4,
size=69 bytes, dist=1, time=0.710ms
```

On peut ensuite créer notre cluster. Sur un noeud, on lance la commande:

```
pvecm create moncluster
```

Ensuite, on va modifier le fichier /etc/corosync/corosync.conf pour permettre un cluster à deux noeuds. En effet, en temps normal, les clusters sont composés de trois noeuds afin de permettre l’élection d'un master sans que cela ne pose de problème au niveau des votes (2 votes contre 1). Il faut donc spécifier ici ce type particulier de cluster:

```
logging {
  debug: off
```

```
  to_syslog: yes
}

nodelist {
  node {
    name: alice
    nodeid: 1
    quorum_votes: 1
    ring0_addr: alice
  }
  node {
    name: bob
    nodeid: 2
    quorum_votes: 1
    ring0_addr: bob
  }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

totem {
  cluster_name: picasoft
  config_version: 2016120703
  ip_version: ipv4
  secauth: on
  transport: udpu
  version: 2
  interface {
    ringnumber: 0
  }
}
```

Cette modification est à effectuer sur les deux noeuds. Attention à bien mettre à jour le champ config_version à chaque modification de ce fichier (sur les deux noeuds).

Une fois cette configuration appliquée, on peut relancer le processus corosync et voir l'état du cluster

```
systemctl restart corosync.service
root@alice:~# pvecm status
Quorum information
------------------
Date:             Wed Dec  7 23:12:08 2016
```

```
Quorum provider:  corosync_votequorum
Nodes:            2
Node ID:          0x00000001
Ring ID:          1/44
Quorate:          Yes

Votequorum information
----------------------
Expected votes:   2
Highest expected: 2
Total votes:      2
Quorum:           1
Flags:            2Node Quorate WaitForAll

Membership information
----------------------
    Nodeid      Votes Name
0x00000001          1 192.168.10.1 (local)
0x00000002          1 192.168.10.2
```

Problème rencontré:

Lors de la mise en place de notre cluster, nous avons été confrontés à un problème. Du fait que nous montons les Ip de la machine via le fichier /etc/rc.local, l'interface réseau n'est pas prête (avec une IP) au lancement de corosync au boot. Une manière simple de contourner ce problème est de relancer corosync lorsque celui-ci ne démarre pas. Ainsi dès que l'IP sera monté, le redémarrage de corosync sera fonctionnel et le cluster prêt à être utilisé.

Dans le fichier /etc/systemd/system/multi-user.target.wants/corosync.service, on ajoute deux directives

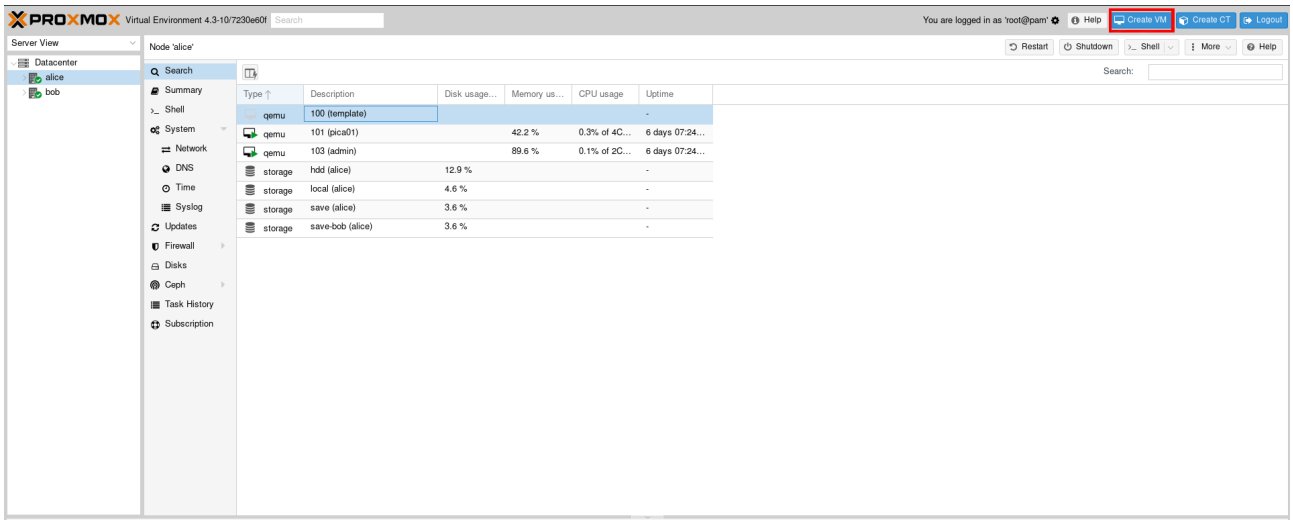
```
root@alice:~# cat /etc/systemd/system/multi-
user.target.wants/corosync.service
[Unit]
Description=Corosync Cluster Engine
ConditionKernelCommandLine=!nocluster
ConditionPathExists=/etc/corosync/corosync.conf
Requires=network-online.target
After=network-online.target
DefaultDependencies=no
Before=shutdown.target
Conflicts=shutdown.target

[Service]
ExecStart=/usr/share/corosync/corosync start
ExecStop=/usr/share/corosync/corosync stop
```

```
Restart=on-failure #À ajouter
RestartSec=5s #À ajouter
Type=forking

[Install]
WantedBy=multi-user.target
```

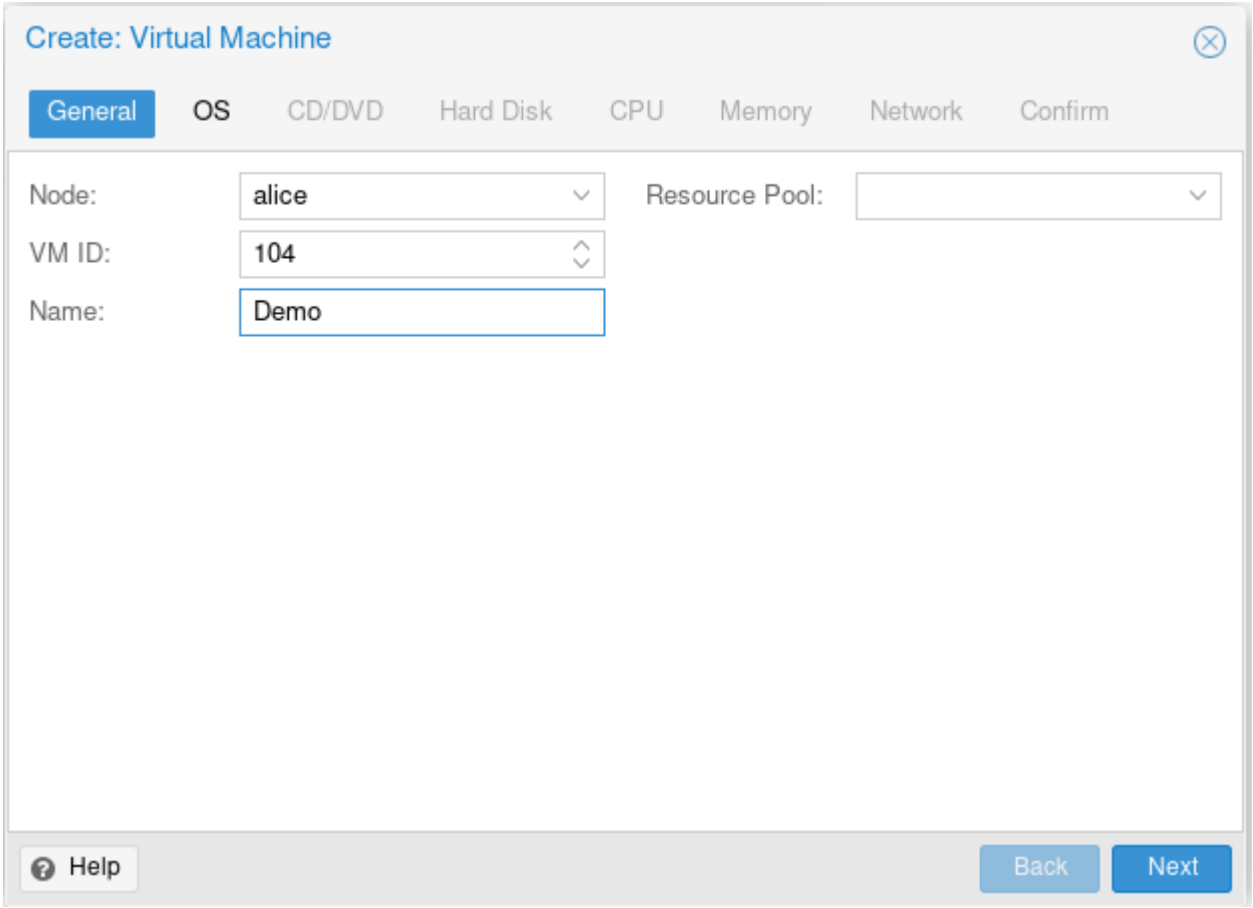
10) Création d'une machine virtuelle

Une fois Proxmox installé, il est possible de commencer certaines opérations, telle que la création d'une VM.

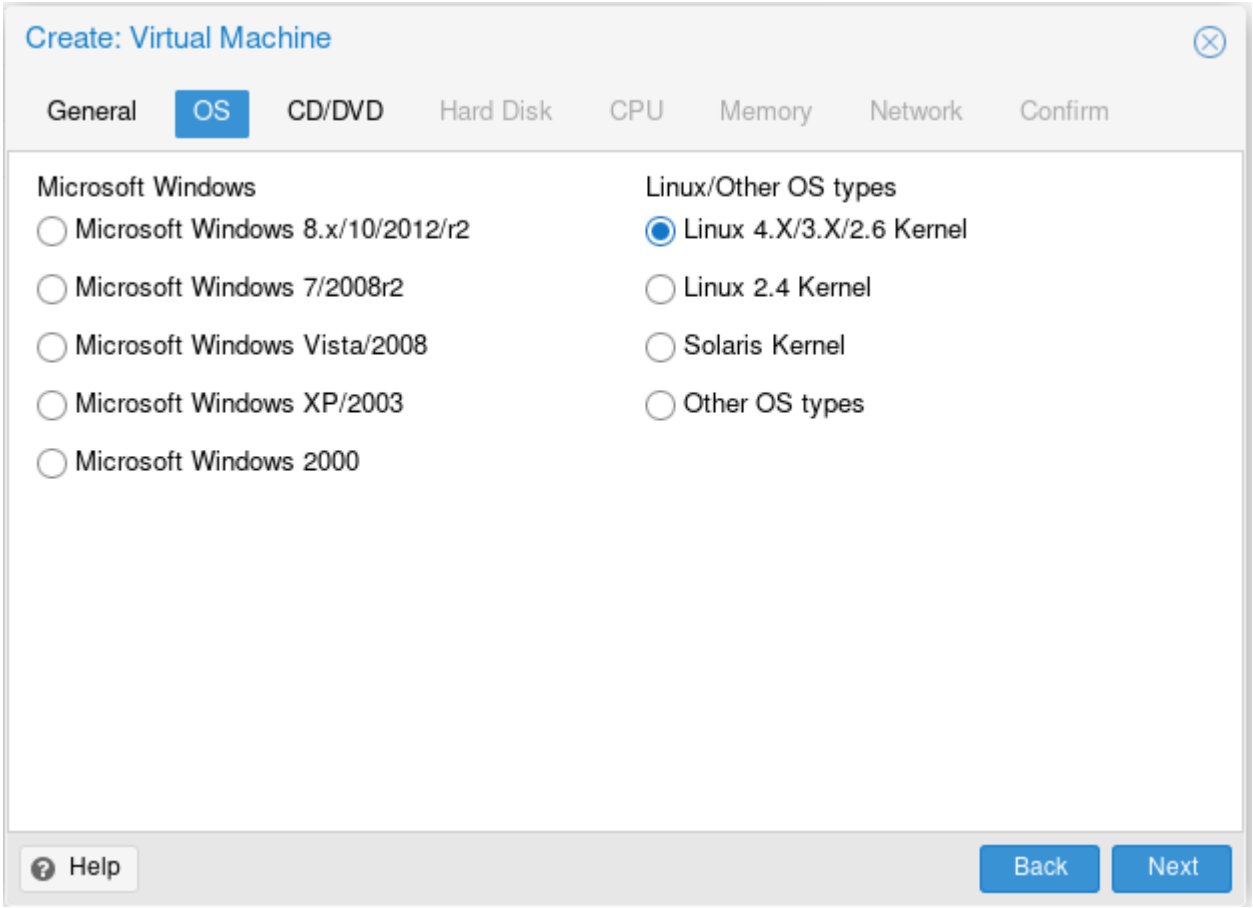
On commence d'abord par lancer l'utilitaire de création d'une machine virtuelle :



On choisit ensuite un nom de VM :

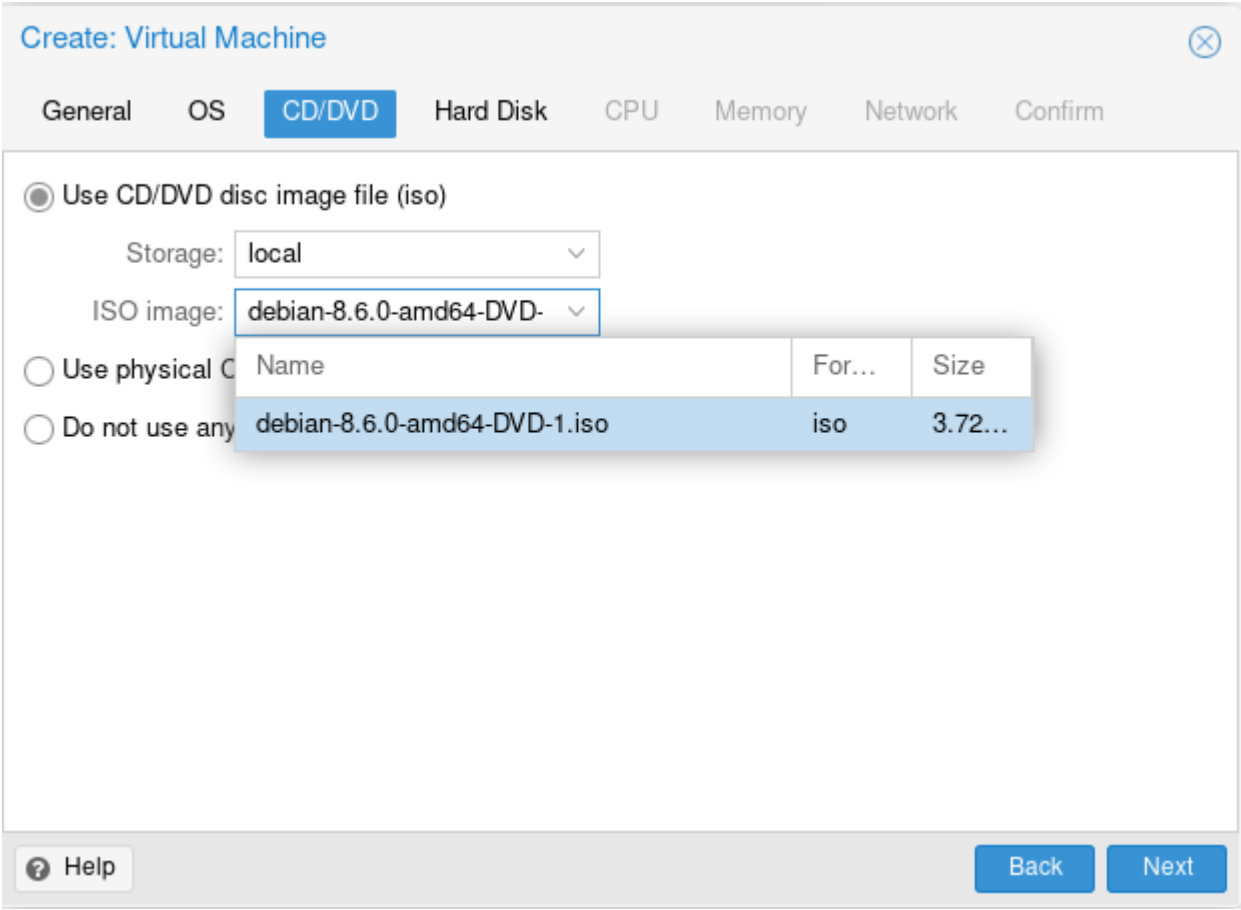


On choisit le bon kernel :

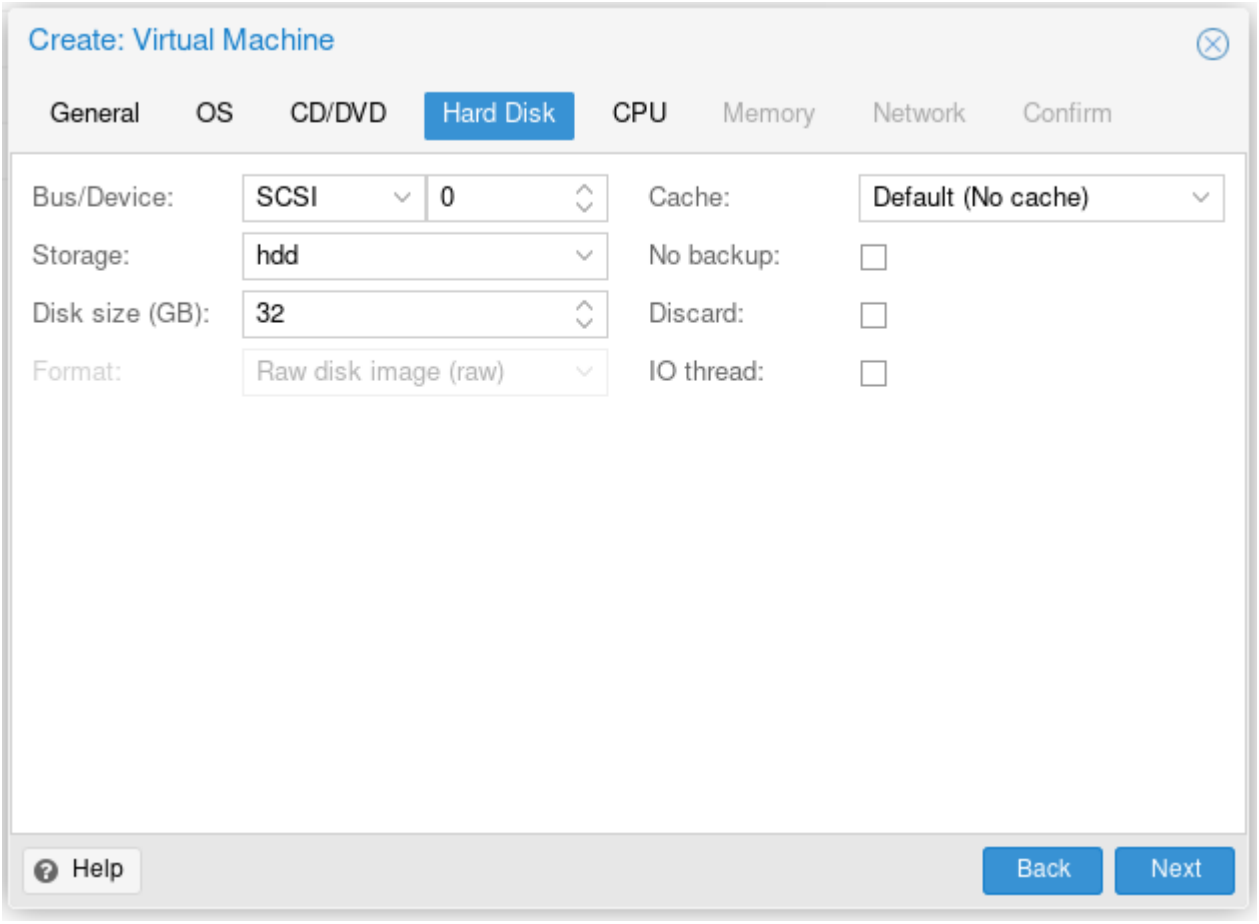


Et on prends l'image locale debian : Au besoin, il est possible de télécharger une nouvelle version de l'iso. Pour cela, il faut effectuer un wget de l'image dans le dossier

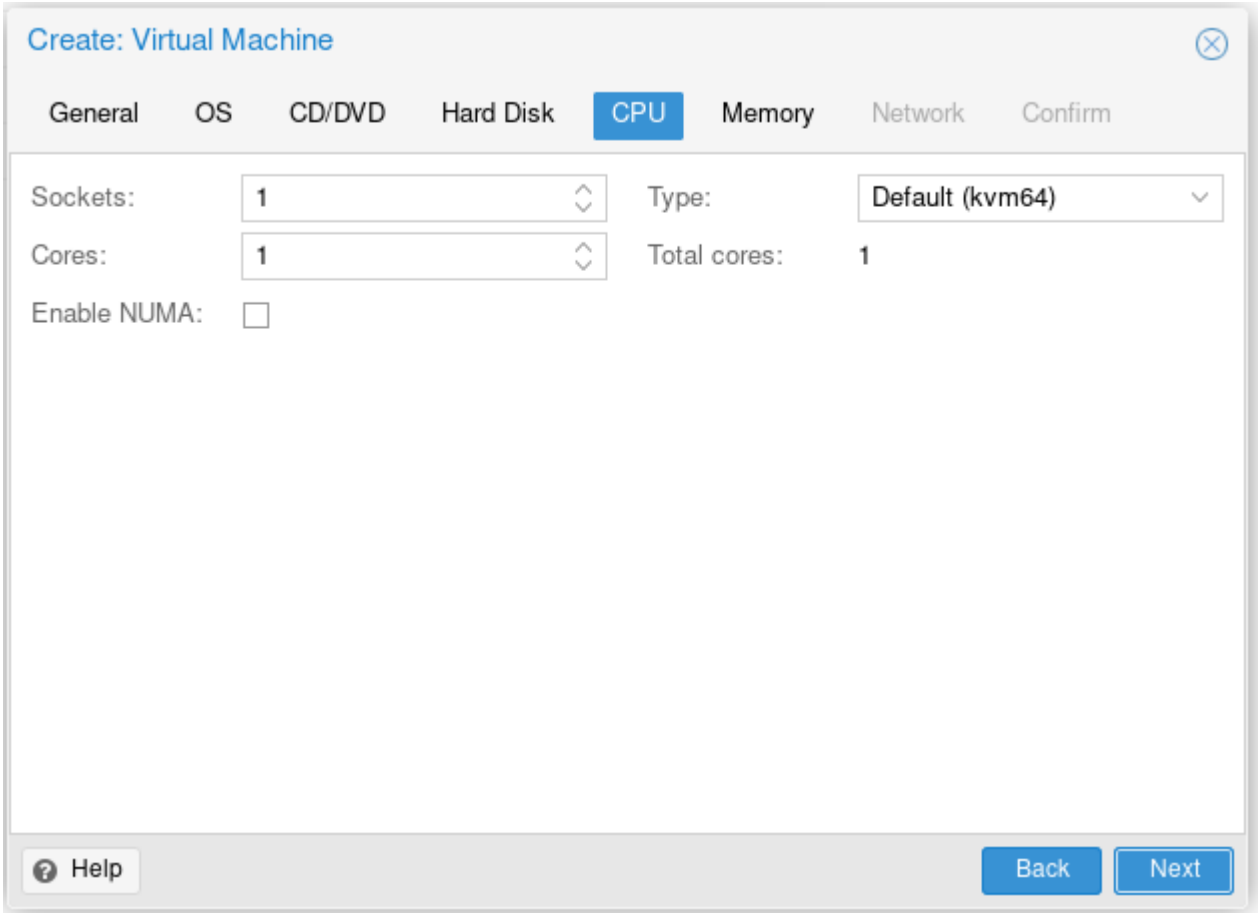
```
/var/lib/vz/template/iso/
```



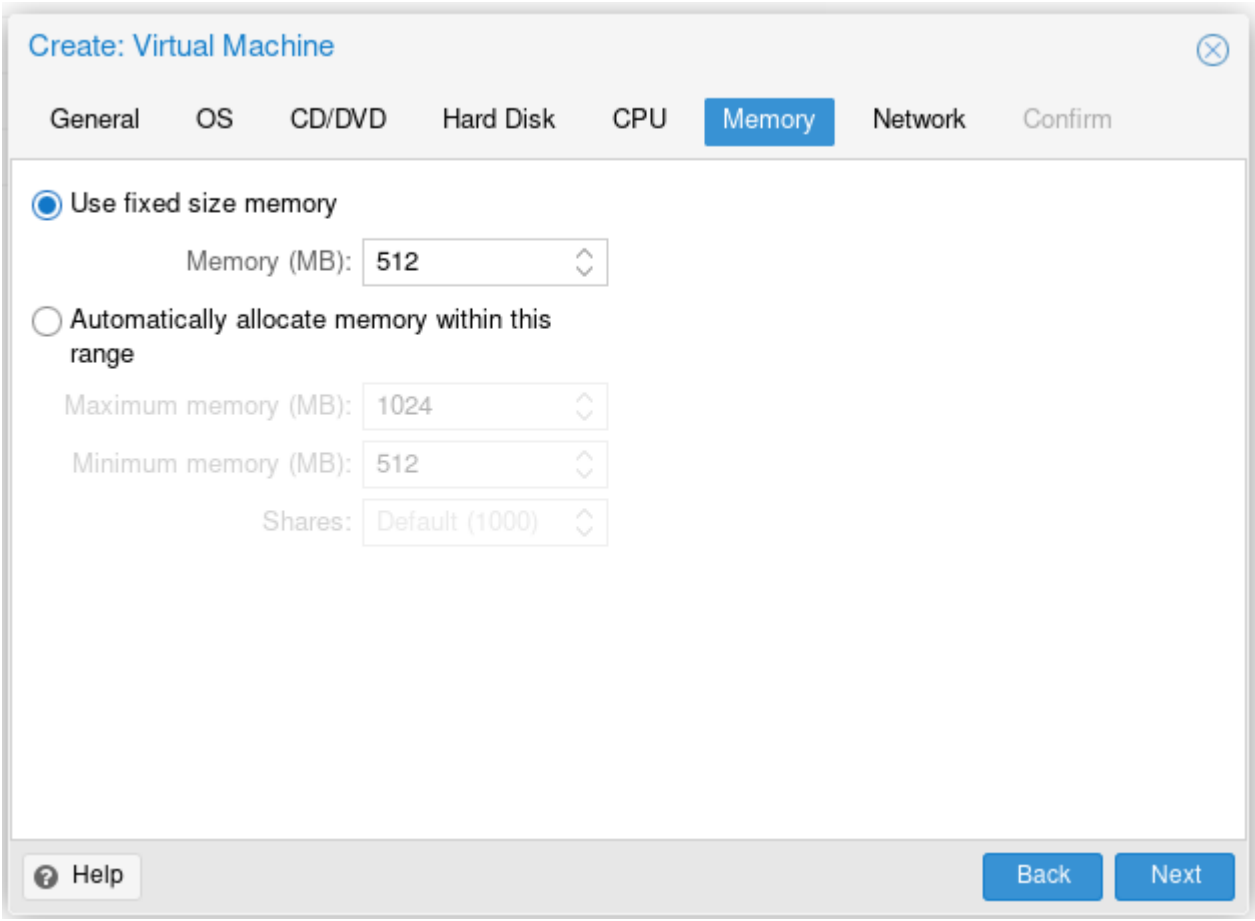
On choisit ensuite la taille du disque dur nécessaire. Le bus est à mettre en SATA. Par défaut, les machines sont buildées sur le disque local qui correspond au RAID de SSD sur les machines. Dans les paramètres de proxmox, il est possible d'autoriser l'installation de VM sur les disques dur. Dans ce cas, il faudra sélectionner l'option adaptée dans le menu déroulant.



On choisit le nombre de coeur alloué à la machine virtuelle: NB: Une configuration 1 processeur à 2 coeurs est équivalente à mettre 2 processeurs à 1 coeur.

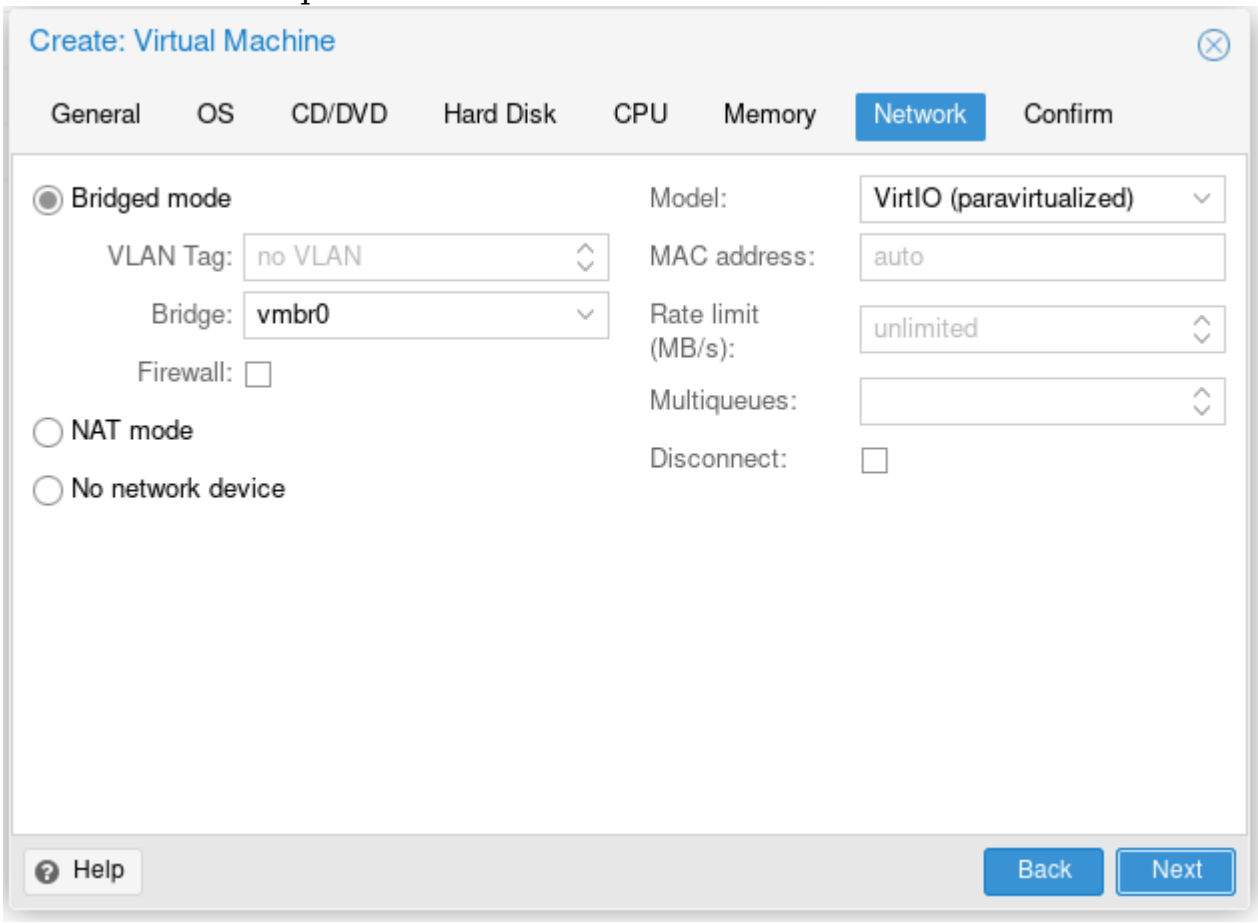


On choisit ensuite la mémoire vive allouée, encore une fois, le paramètre par défaut est souvent suffisant. Il est également possible d'allouer la RAM dynamiquement. Il faut alors fournir une borne supérieure et inférieure.

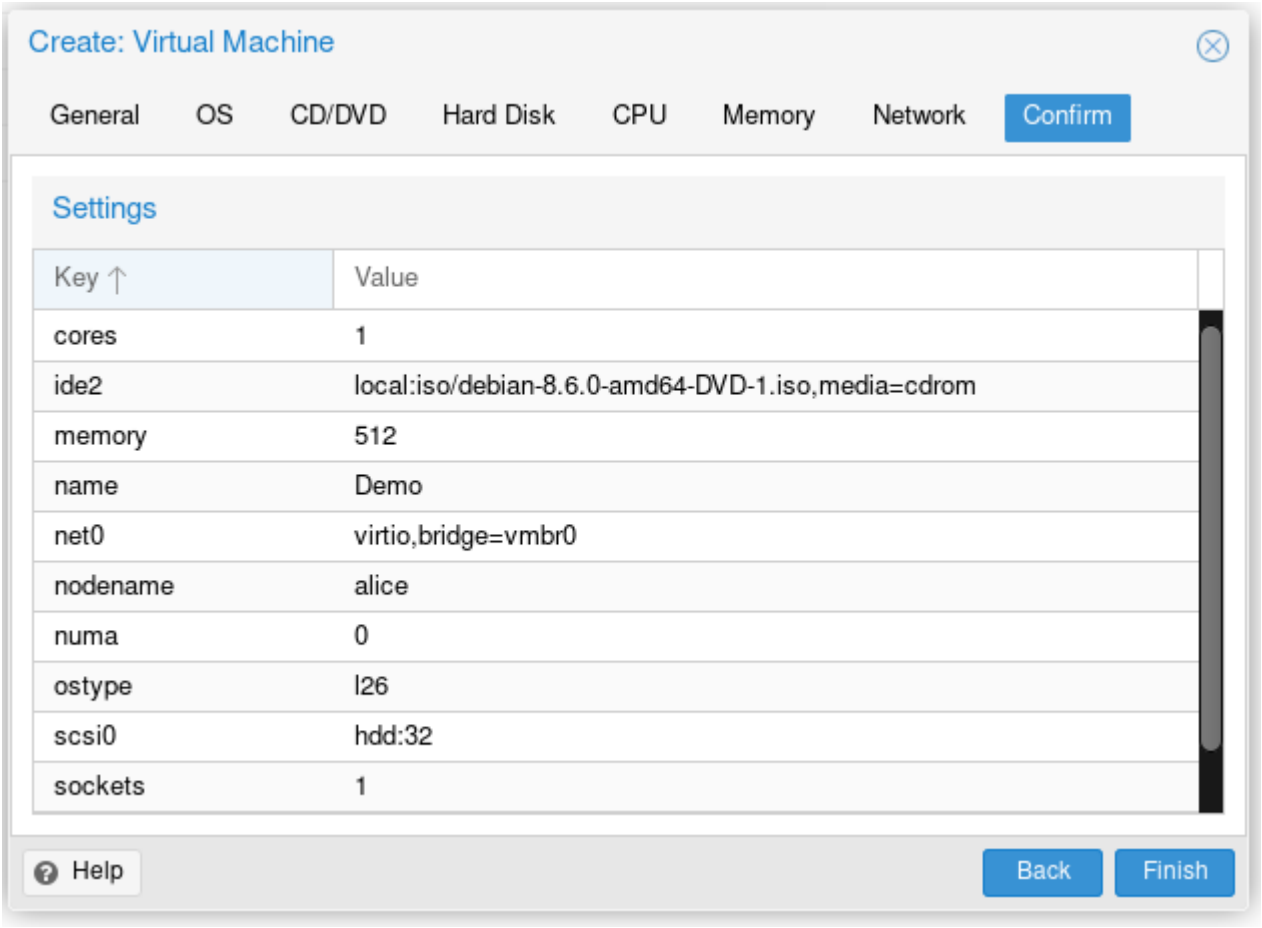


Bridge : Concernant le réseau, en fonction de la machine, il faut choisir le bridge 0 ou 1. Le bridge 0 est à utiliser si la VM à builder va recevoir une IP publique. Dans ce cas, les IP publiques sont forwardées directement à la VM. Le bridge 1 permet de builder des VM sur un réseau privé naté (dans ce cas de figure, c'est l'ip de l'hyperviseur qui est connu et les services sont exposés suivant leurs ports). Sur l'installation de picasoft, il n'y a pas de DHCP. Le réseau configuré sur le bridge 1 est sur le range suivant: 10.0.42.0/24. l'IP 10.0.42.1 est réservée au noeud physique et sert de passerelle de sortie pour les VM. A noter qu'il n'est pas possible de faire communiquer des VM qui serait sur le réseau privé entre alice et bob. Il s'agit ici de deux

environnements séparés.



On confirme les paramètres :

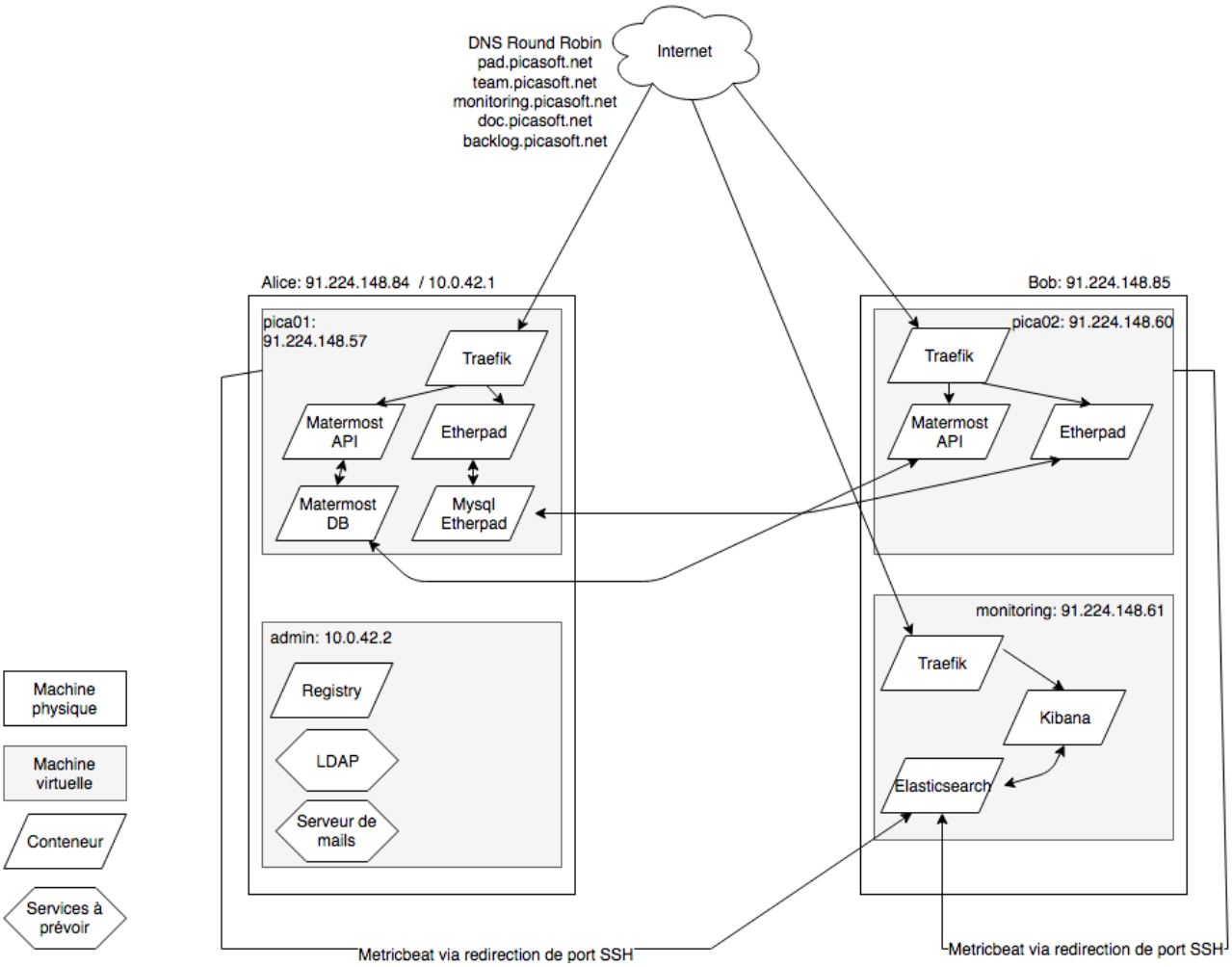


Il faut attendre la création de la machine virtuelle. Une fois celle-ci crée, il est possible de la démarrer via la barre latérale. Pour effectuer l'installation, on utilise ensuite la console (présente en haut à droite de l'interface).

11) Architecture globale

> **Alice** & **Bob** possède respectivement une machine virtuelle *pica01* et *pica02*. Ces machines virtuelles font tourner Docker/Swarm sur lequel tourne les différents services.

Cet article a pour but la présentation globale de l'architecture technique de picasoft.



2 machines physiques hébergent quatre VM avec des services

Services

[Pour avoir une vision détaillée des services]

Monitoring

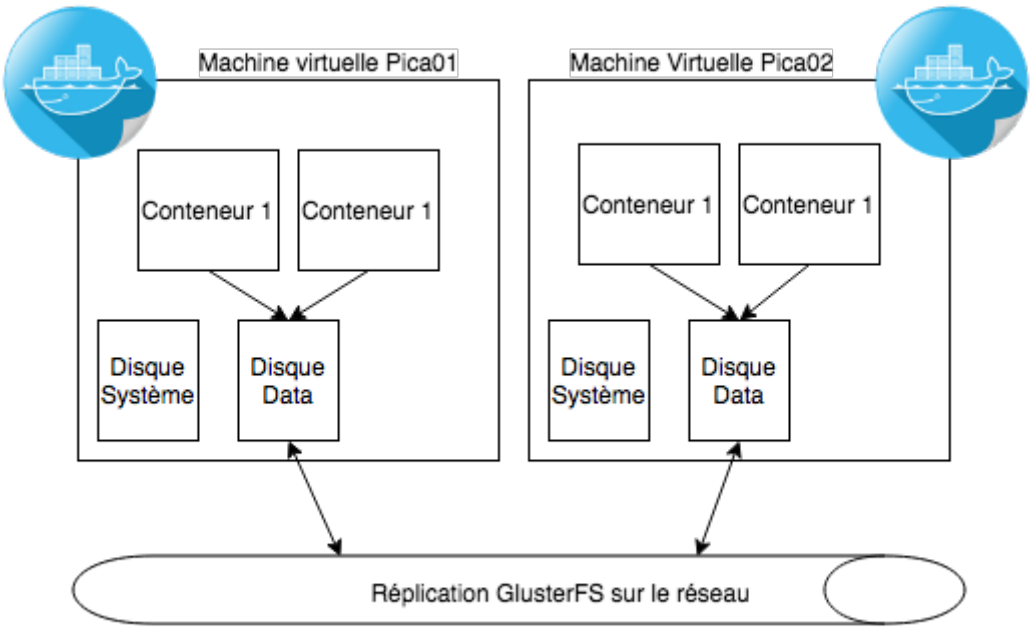
Une machine virtuelle dédiée au monitoring avec un IP publique a été mise en place. Dans le futur, il est tout à fait envisageable de la passer sur un réseau privé et de faire du NAT Monitoring des VM hébergeant les services Redirection de ports

Admin

La machine d'admin a pour vocation d'être la machine à tout faire. Actuellement elle sert pour héberger le registry docker. À l'avenir, celle-ci pourra héberger un LDAP ou encore un serveur mail.

12) Installation d'un cluster Docker Swarm

Cette page a pour but la mise en place d'un cluster Docker Swarm sur deux noeuds avec un partage des données via un volume partagé utilisant GlusterFS



Préparation d'un disque logique

Volume LVM

Afin de stocker les données, j'ai décidé de créer un disque dédié au volume GlusterFS. C'est ce volume qui sera ensuite partagé avec les différents serveurs sur le réseau.

```
$ vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  vg00   1   2   0 wz--n- 39,76g     0
  vg01   1   0   0 wz--n- 15,00g 15,00g
```

Sur la machine virtuelle, j'ai donc ajouté un nouveau disque d'une taille de 15G ici ajouté sur un groupe logique vg01. Je créé ensuite un volume logique (lv) à partir de ce vg et qui va prendre tout l'espace disponible

```
$ lvcreate -l 100%FREE -n data vg01
```

On peux maintenant formater ce disque et le monter sur la machine:

```
$ mkfs -t ext4 /dev/mapper/vg01-data.
$ echo "/dev/mapper/vg01-data /gluster-data ext4    defaults
0       2" >> /etc/fstab
$ mkdir /gluster-data && mount -a
```

Volume GlusterFS

Une fois le volume monté, il faut maintenant créer un volume de plus haut niveau à l'aide de glusterFS. Cela permettra aux différentes machines de notre cluster d'accéder à un volume de données partagé et répliqué automatiquement entre les nœuds. Cela signifie qu'en cas de panne, le nœud restant sera toujours en mesure de fournir le service évitant ainsi l'interruption de service.

La première chose à faire et de s'assurer que les différentes machines du cluster se connaissent les unes les autres:

```
$ cat /etc/hosts
   127.0.0.1   localhost
   91.224.148.57   pica01 pica01.picasoft.net
   91.224.148.60   pica02 pica02.picasoft.net
```

Il faut ajouter les clés du repo ainsi que le repo contenant les packages

```
$ wget -O -
http://download.gluster.org/pub/gluster/glusterfs/3.9/rsa.pub |
apt-key add -
$ echo deb
http://download.gluster.org/pub/gluster/glusterfs/3.9/LATEST/Debia
n/jessie/apt jessie main > /etc/apt/sources.list.d/gluster.list
$ apt-get update
$ apt-get -y install glusterfs-server
```

Ensuite, on ajoute le second serveur à partir du premier noeud du cluster. Ici, on lance donc sur pica01

```
$ gluster peer probe pica02
$ gluster peer status
  Number of Peers: 1
  Hostname: pica02
  Uuid: 36530258-860b-4403-85b3-3de1fd6bb47a
  State: Peer in Cluster (Connected)
```

On peut maintenant créer un volume répliqué que l'on va configurer en mirroring afin d'avoir une copie parfaite des données entre les noeuds.

```
$ gluster volume create gluster-data replica 2 transport tcp
pica01:/gluster-data pica02:/gluster-data force
$ gluster volume start gluster-data
```

Une fois le volume créé, on peut vérifier l'état de celui-ci via la commande:

```
$ gluster volume info
  Volume Name: gluster-data
  Type: Replicate
  Volume ID: 24ec2dcb-84fe-4981-9e01-1a5614cd209b
  Status: Started
  Snapshot Count: 0
  Number of Bricks: 1 x 2 = 2
  Transport-type: tcp
  Bricks:
  Brick1: pica01:/gluster-data
  Brick2: pica02:/gluster-data
```

Afin d'éviter que n'importe qui puisse accéder à notre volume et faire des écriture, on doit limiter les droits en écriture aux machines pica01/02

```
$ gluster volume set gluster-data auth.allow pica02
```

On peut maintenant ajouter une ligne au fichier fstab pour lancer le montage automatique du volume. À noter qu'il faut adapter le point de montage en fonction de la machine

```
$ echo "pica01:/gluster-data /DATA glusterfs defaults,_netdev 0 0"
>> /etc/fstab
$ mkdir /DATA && mount -a
```

Pour le moment, le montage automatique au boot de la machine ne fonctionne pas. Pour palier à ce problème, j'ai ajouté les lignes suivante au fichier /etc/rc.local

```
$ echo "systemctl start glusterfs-server" >> /etc/rc.local
$ echo "mount -a" >> /etc/rc.local
```

Installation de Docker

Docker Engine

Docker va nous servir pour mettre en place et faire tourner les conteneurs contenant les services. Avant de mettre en place un cluster Swarm permettant de répartir automatiquement les conteneurs sur les différentes machines, il faut installer le Docker engine sur chacune de nos machines. Pour cela:

```
$ apt-get update
$ apt-get install apt-transport-https ca-certificates
$ apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80
--recv-keys
58118E89F3A912897C070ADBF76221572C52609D
$ apt-get update && apt-get install docker-engine
$ systemctl start docker && systemctl enable docker
```

On peut vérifier que l'installation s'est bien déroulée à l'aide de la commande `docker version`

```
$ docker version
  Client:
   Version:      1.12.3
   API version:  1.24
   Go version:   go1.6.3
   Git commit:   6b644ec
   Built:        Wed Oct 26 21:39:14 2016
   OS/Arch:      linux/amd64
  Server:
   Version:      1.12.3
   API version:  1.24
   Go version:   go1.6.3
   Git commit:   6b644ec
   Built:        Wed Oct 26 21:39:14 2016
   OS/Arch:      linux/amd64
```

Déploiement de Swarm

On peut maintenant déployer le cluster Swarm à l'aide des commandes suivantes: Sur pica01

```
$ root@pica01:~# docker swarm init
  Swarm initialized: current node (alpfdvrzjrbiuvya7nyiwhbjg) is
now a manager.
  To add a worker to this swarm, run the following command:
      docker swarm join \
      --token SWMTKN-1-
0aieljefhgeirgjkzrjkbgerohzg69rnlj6why0dmaddseswnsmntldd9-
ci1ew93yu91p7dfl1gxlur1zh \
      91.224.148.57:2377
  To add a manager to this swarm, run 'docker swarm join-token
manager' and follow the instructions.
```

Sur pica02 on lance la commande proposée par pica01

```
$ root@pica02:~# docker swarm join \
>     --token SWMTKN-1-
0aieljefhgeirgjkzrjkbgerohzg69rnlj6why0dmaddseswnsmntldd9-
ci1ew93yu91p7dfl1gxlur1zh \
>     91.224.148.57:2377
This node joined a swarm as a worker.
```

Sur le nœud master, on peut voir les nœuds du cluster:

```
$ docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY
MANAGER STATUS
9c3xnjjo1u6mbol5qtbeh92cb    pica02    Ready   Active
```

```
alpfdvrzjrbiuvya7nyiwhbjg *  pica01    Ready   Active
Leader
```

Dans le cas d'un cluster à deux noeuds, la notion de manager et de client est difficile à mettre en place. On peut donc passer le second noeud comme manager. En cas de panne du premier, il sera possible de lancer des commande “service” sur le second noeud

```
$ docker node promote pica02
```

Le cluster est maintenant prêt à être utilisé. Avant de lancer les premiers services, il faut savoir que Docker lorsque est utilisé en mode Swarm ne permet plus de lier des conteneurs les uns aux autres comme c'est le cas avec Docker en standalone. Pour palier à ce problème, il faut créer une réseau overlay qui par la suite permettra de contacter n'importe quel conteneur sur ce réseau en utilisant son nom. La commande ci-dessous permet de créer un réseau overlay sur les différentes machines

```
$ docker network create --driver overlay pica-net
```

Maintenant, lorsque l'on lance différents services sur le réseau pica-net à l'aide du paramètre –network, ils seront en mesure de communiquer à partir du nom qui leur aura été attribué au lancement.

Lancement d'un service

Il est désormais possible de lancer les premiers services utilisant le volume partagé glusterfs pour la persistance des données. Par exemple, le lancement d'un conteneur dokuwiki peut se traduire avec la commande suivante:

```
$ docker service create --name dokuwiki --replicas 3 \
  --publish 3000:80 \
  --mount type=bind,source=/DATA/WEB/dokuwiki,target=/var/www/html
\
  --network pica-net registry.picasoft.net:5000/pica-dokuwiki
```

Cette commande va automatiquement déployer 3 instances dokuwiki sur les différents noeuds du cluster en fonction de la charge de ceux-ci. Le service est accessible sur le port 3000 sur chacune des machines du cluster et ce même si le noeud n'héberge pas le service. Il est par la suite possible d'augmenter ou de réduire le nombre d'instance à l'aide de la commande

```
$ docker service scale dokuwiki=1
  dokuwiki scaled to 1
$ docker service scale dokuwiki=4
  dokuwiki scaled to 4
```

Bon à savoir: Pour qu'une image puisse être lancée sur les deux serveurs, il faut au préalable avoir télécharger cette image sur les différents noeuds du cluster. Sans cela, Swarm va scheduller les conteneurs sur une unique machine qui possède les images.

13) Redirection de ports

Sur les hôtes physiques, lorsque l'on a une machine virtuelle sur un réseau privé et que l'on souhaite accéder à un service, il convient d'appliquer des règles de NAT (Network Address Translation) et de redirection de ports. Lorsque l'hyperviseur recevra une requête sur un port qui a du NAT en place, celui-ci va se charger de transmettre l'information directement à la machine virtuelle concernée. C'est également par ce biais que les machines virtuelles ont accès à internet. À la manière d'une box, le serveur physique va transmettre les requêtes réseau qui ne lui sont pas adressées sur sa passerelle par défaut.

Bridge pour les VM

la première étape consiste à avoir une interface réseau dédié au réseau privé. Celle-ci doit être de la forme vmbrX pour être reconnue par proxmox. Comme nous l'avons vu dans la page Installation, les configurations réseau sont au niveau du /etc/rc.local. On peut donc y ajouter les instructions suivantes:

```
#Interface de bridge privée pour les VM
brctl addbr vmbr1 #ajout d'une interface virtuelle
ip link set vmbr1 up #mise en ligne
ip addr add 10.0.42.1/24 dev vmbr1 #assignation d'une IP
echo 1 > /proc/sys/net/ipv4/conf/vmbr1/proxy_arp #Activation du
proxy arp qui permet à 2 machines du réseau privé de communiquer
entre elles
iptables -t nat -A POSTROUTING -s '10.0.42.0/24' -o vmbr0 -j
MASQUERADE #routage des requêtes vers la patte internet
```

Redirection de port

La redirection de port s'effectue avec des règles iptables:

```
iptables -t nat -A PREROUTING -i vmbr0 -p tcp --dport 80 -j DNAT
--to-destination 10.0.42.2:80
```

La règle ci-dessus redirige le trafic qui arrive de l'interface vmbr0 sur le port 80 vers la VM 10.0.42.2 sur le port 80

14) Sauvegarde

Afin de s'assurer de ne perdre aucune donnée, un système de sauvegarde est en place sur l'infrastructure de picasoft. Une première partie concerne les services plus particulièrement les bases de données et une autre les machines virtuelles. En cas de panne ou de problème logiciel, on doit être en mesure de redémarrer les services sans pertes de données ou alors une perte “minime”.

Sauvegarde des bases de données

Les bases de données sur l'architecture de picasoft ne sont pour le moment pas redondés. C'est-à-dire qu'en cas de crash d'une de nos bases de données, le service va tenter de redémarrer sur l'un des noeuds disponibles, mais si le système de fichier de la base est corrompu, il ne sera pas possible de la redémarrer. Pour pallier à ce problème, un script est en place et tourne en tâche de fond. Celui-ci se charge de faire une sauvegarde de la base de données toutes les 6h. Les fichiers sont stockés au niveau du /DATA/BACKUP/{mysql,postgresql}.

Mysql

Pour le mysql, nous avons utilisé une image du marché proposée par tutum tutum/mysql-backup Celle-ci exécute un script simple qui permet de faire un mysqldump à intervalle régulier. Elle intègre également un système de purge pour ne garder qu'un certain nombre de sauvegardes au delà de ce nombre, les sauvegardes les plus anciennes sont supprimés. (Actuellement, nous avons 7,5J de sauvegarde qui sont stockés).
Une image a été buildées sur le registry: registry.picasoft.net:5000/msql-backup

```
#!/bin/bash

if [ "${MYSQL_ENV_MYSQL_PASS}" == "**Random**" ]; then
        unset MYSQL_ENV_MYSQL_PASS
fi

MYSQL_HOST=${MYSQL_PORT_3306_TCP_ADDR:-${MYSQL_HOST}}
MYSQL_HOST=${MYSQL_PORT_1_3306_TCP_ADDR:-${MYSQL_HOST}}
MYSQL_PORT=${MYSQL_PORT_3306_TCP_PORT:-${MYSQL_PORT}}
MYSQL_PORT=${MYSQL_PORT_1_3306_TCP_PORT:-${MYSQL_PORT}}
MYSQL_USER=${MYSQL_USER:-${MYSQL_ENV_MYSQL_USER}}
MYSQL_PASS=${MYSQL_PASS:-${MYSQL_ENV_MYSQL_PASS}}

[ -z "${MYSQL_HOST}" ] && { echo "=> MYSQL_HOST cannot be empty"
&& exit 1; }
[ -z "${MYSQL_PORT}" ] && { echo "=> MYSQL_PORT cannot be empty"
```

```
&& exit 1; }
[ -z "${MYSQL_USER}" ] && { echo "=> MYSQL_USER cannot be empty"
&& exit 1; }
[ -z "${MYSQL_PASS}" ] && { echo "=> MYSQL_PASS cannot be empty"
&& exit 1; }

BACKUP_CMD="mysqldump -h${MYSQL_HOST} -P${MYSQL_PORT} -u$
{MYSQL_USER} -p${MYSQL_PASS} ${EXTRA_OPTS} ${MYSQL_DB} >
/backup/"'${BACKUP_NAME}'

echo "=> Creating backup script"
rm -f /backup.sh
cat <<EOF >> /backup.sh
#!/bin/bash
MAX_BACKUPS=${MAX_BACKUPS}
BACKUP_NAME=\$(date +\%Y.\%m.\%d.\%H\%M\%S).sql
echo "=> Backup started: \${BACKUP_NAME}"
if ${BACKUP_CMD} ;then
    echo "   Backup succeeded"
else
    echo "   Backup failed"
    rm -rf /backup/\${BACKUP_NAME}
fi
if [ -n "\${MAX_BACKUPS}" ]; then
    while [ \$(ls /backup -N1 | wc -l) -gt \${MAX_BACKUPS} ];
    do
        BACKUP_TO_BE_DELETED=\$(ls /backup -N1 | sort | head -n 1)
        echo "   Backup \${BACKUP_TO_BE_DELETED} is deleted"
        rm -rf /backup/\${BACKUP_TO_BE_DELETED}
    done
fi
echo "=> Backup done"
EOF
chmod +x /backup.sh

echo "=> Creating restore script"
rm -f /restore.sh
cat <<EOF >> /restore.sh
#!/bin/bash
echo "=> Restore database from \$1"
if mysql -h${MYSQL_HOST} -P${MYSQL_PORT} -u${MYSQL_USER} -p$
{MYSQL_PASS} < \$1 ;then
    echo "   Restore succeeded"
else
    echo "   Restore failed"
fi
echo "=> Done"
```

```
EOF
chmod +x /restore.sh

touch /mysql_backup.log
tail -F /mysql_backup.log &

if [ -n "${INIT_BACKUP}" ]; then
    echo "=> Create a backup on the startup"
    /backup.sh
elif [ -n "${INIT_RESTORE_LATEST}" ]; then
    echo "=> Restore lates backup"
    until nc -z $MYSQL_HOST $MYSQL_PORT
    do
        echo "waiting database container..."
        sleep 1
    done
    ls -d -1 /backup/* | tail -1 | xargs /restore.sh
fi

echo "${CRON_TIME} /backup.sh >> /mysql_backup.log 2>&1" >
/crontab.conf
crontab  /crontab.conf
echo "=> Running cron job"
exec cron -f
```

Postgresql

Il n'existait pas de solution toute prête permettant de sauvegarder une base de données postgresql. Nous avons donc décidés de nous baser sur l'image de tutum et de l'adapter à postgresql afin de sauvegarder la base de données et d'avoir une rotation des sauvegardes automatique.
Une image a été buildées sur le registry: registry.picasoft.net:5000/postgres-backup Le script de sauvegarde est le suivant:

```
#!/bin/bash

[ -z "${POSTGRES_HOST}" ] && { echo "=> POSTGRES_HOST cannot be
empty" && exit 1; }
[ -z "${POSTGRES_PORT}" ] && { echo "=> POSTGRES_PORT cannot be
empty" && exit 1; }
[ -z "${POSTGRES_USER}" ] && { echo "=> POSTGRES_USER cannot be
empty" && exit 1; }
[ -z "${POSTGRES_PASS}" ] && { echo "=> POSTGRES_PASS cannot be
empty" && exit 1; }
[ -z "${POSTGRES_DB}" ] && { echo "=> POSTGRES_DB cannot be empty"
&& exit 1; }
```

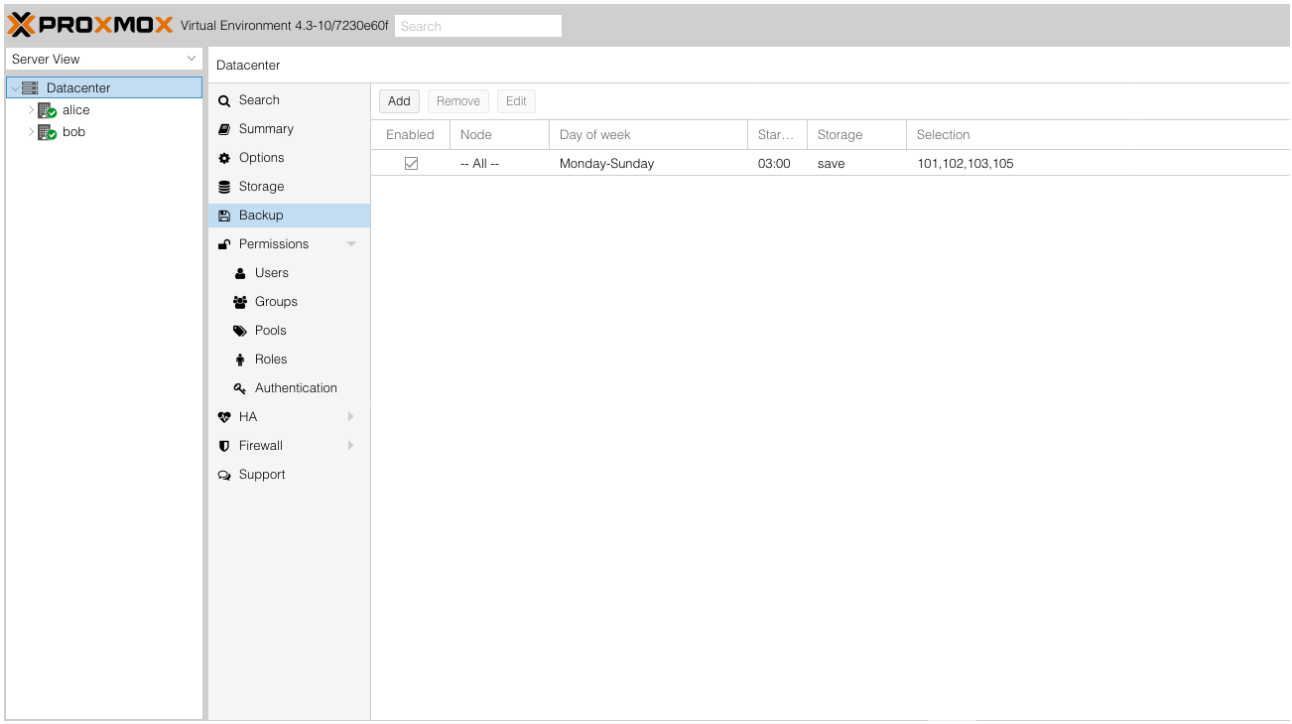
```
BACKUP_CMD="pg_dump -w -c > /backup/"'${BACKUP_NAME}'

echo "=> Creating backup script"
rm -f /backup.sh
cat <<EOF >> /backup.sh
#!/bin/bash
MAX_BACKUPS=${MAX_BACKUPS}
export PGHOST=$POSTGRES_HOST
export PGPORT=$POSTGRES_PORT
export PGDATABASE=$POSTGRES_DB
export PGUSER=$POSTGRES_USER
export PGPASSWORD=$POSTGRES_PASS

BACKUP_NAME=\$(date +\%Y.\%m.\%d.\%H\%M\%S).sql

echo "=> Backup started: \${BACKUP_NAME}"
if ${BACKUP_CMD} ;then
    echo "   Backup succeeded"
else
    echo "   Backup failed"
    rm -rf /backup/\${BACKUP_NAME}
fi

if [ -n "\${MAX_BACKUPS}" ]; then
    while [ \$(ls /backup -N1 | wc -l) -gt \${MAX_BACKUPS} ];
    do
        BACKUP_TO_BE_DELETED=\$(ls /backup -N1 | sort | head -n 1)
        echo "   Backup \${BACKUP_TO_BE_DELETED} is deleted"
        rm -rf /backup/\${BACKUP_TO_BE_DELETED}
    done
fi
echo "=> Backup done"
EOF
chmod +x /backup.sh

echo "=> Creating restore script"
rm -f /restore.sh
cat <<EOF >> /restore.sh
#!/bin/bash
export PGHOST=$POSTGRES_HOST
export PGPORT=$POSTGRES_PORT
export PGDATABASE=$POSTGRES_DB
export PGUSER=$POSTGRES_USER
export PGPASSWORD=$POSTGRES_PASS

echo "=> Restore database from \$1"
if pg_restore -w -d $POSTGRES_DB < \$1 ;then
```

```
    echo "   Restore succeeded"
else
    echo "   Restore failed"
fi
echo "=> Done"
EOF
chmod +x /restore.sh

touch /postgres_backup.log
tail -F /postgres_backup.log &

if [ -n "${INIT_BACKUP}" ]; then
    echo "=> Create a backup on the startup"
    /backup.sh
elif [ -n "${INIT_RESTORE_LATEST}" ]; then
    echo "=> Restore lates backup"
    until nc -z $POSTGRES_HOST $POSTGRES_PORT
    do
        echo "waiting database container..."
        sleep 1
    done
    ls -d -1 /backup/* | tail -1 | xargs /restore.sh
fi

echo "${CRON_TIME} /backup.sh >> /postgres_backup.log 2>&1" >
/crontab.conf
crontab  /crontab.conf
echo "=> Running cron job"
exec cron -f
```

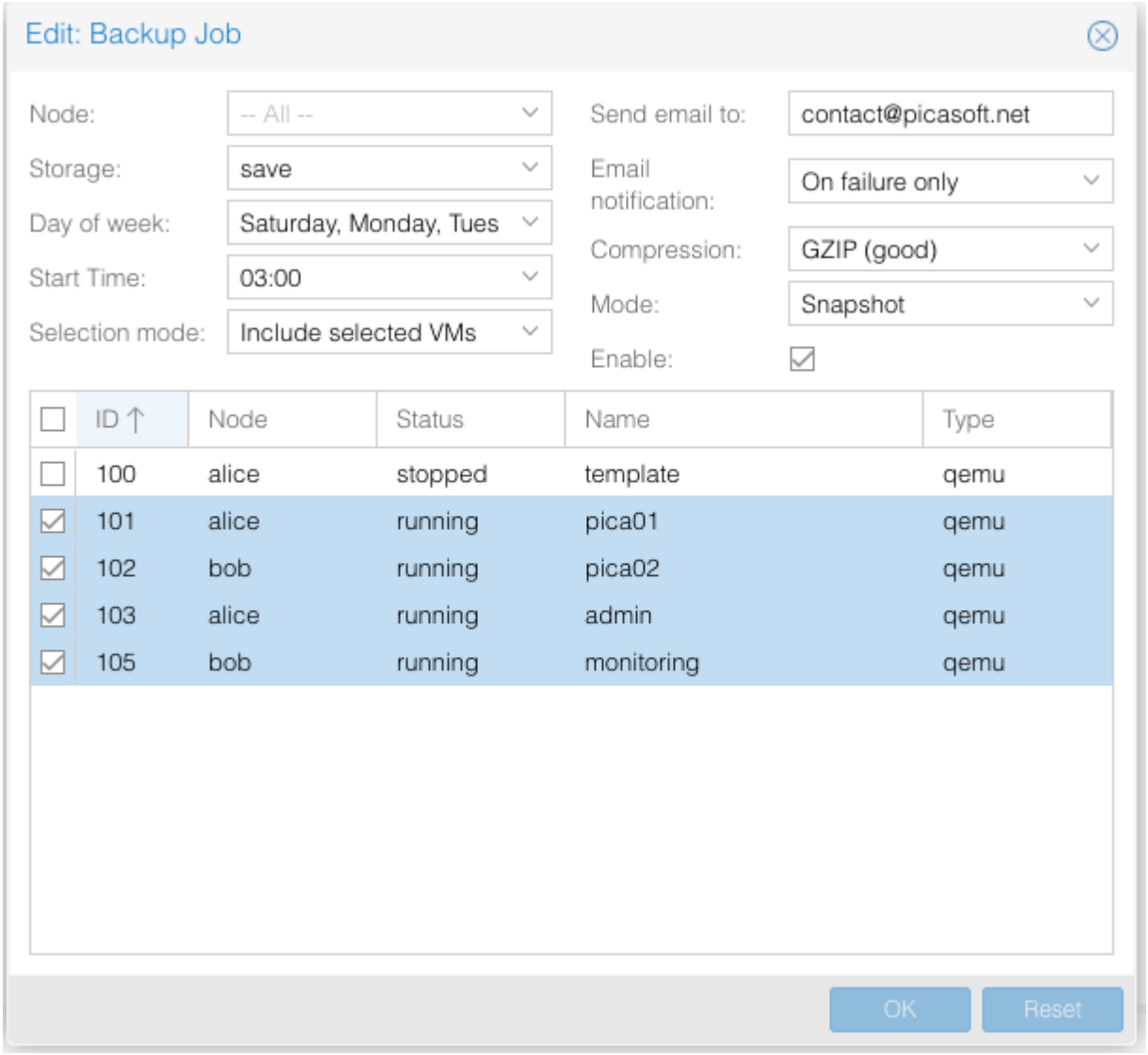
Sauvegarde des machines

Snapshot

Pour sauvegarder les machines virtuelles, Proxmox intègre un système de snapshot. Il est en mesure de sauvegarder le système de fichier complet de la machine virtuelle et de le restaurer en cas de besoin. Dans le cas ou une personne n'est pas sûr de ce qu'elle va faire sur une machine, il est possible de prendre un snapshot de celle-ci et de faire un rollback pour revenir à la situation initiale



Page de backup



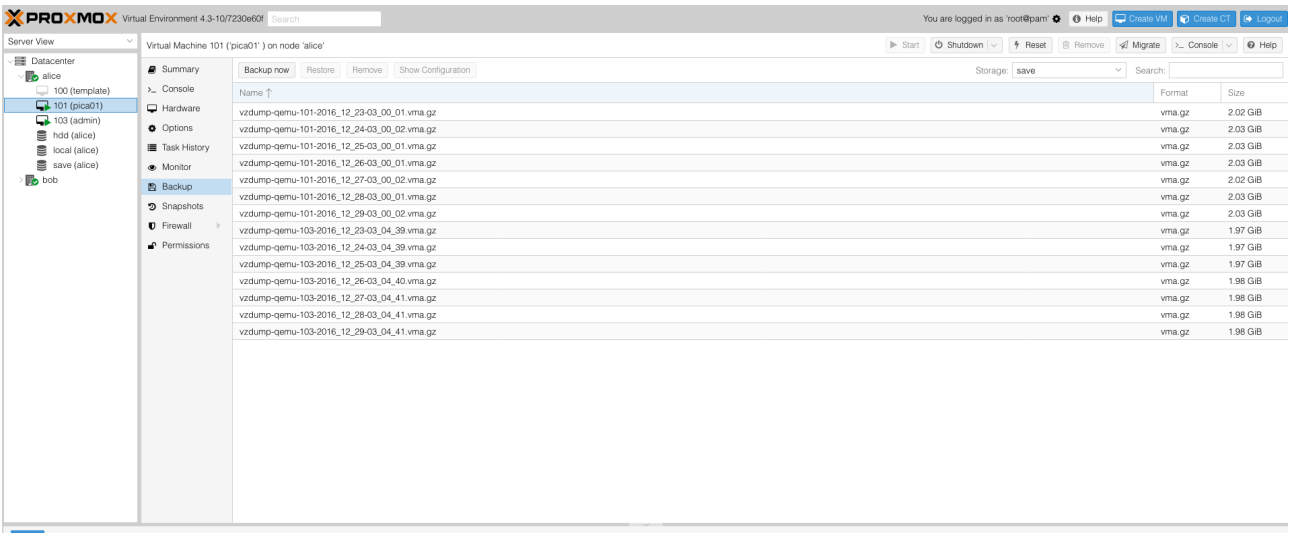
Règles de backup

Chaque noeud physique a une partition /SAVE de 200G. Par défaut, le noeud va sauvegarder les machines virtuelles sur cette partition chaque jour à 3h du matin. Pour assurer une plus grande redondance, un rsync est lancé entre les deux machines à 5h sur alice et 6h sur bob pour synchroniser les deux répertoires et ainsi avoir une copie complète du répertoire /SAVE sur chacun des noeuds. Pour ne pas saturer l'interface reliée à internet, ce rsync est effectué sur l'IP privée des noeuds physiques:

```
#Sur alice
rsync -avzh --progress 192.168.128.152:/SAVE/dump/* /SAVE/dump/

#Sur bob
rsync -avzh --progress 192.168.128.151:/SAVE/dump/* /SAVE/dump/
```

Restauration

la restauration d'une machine s'effectue en cliquant sur une machine virtuelle dans l'interface web. On se rend ensuite dans l'onglet backup. On a alors une liste des backup sur la machine. Lorsque l'on clique sur celui-ci, on a alors la possibilité de restaurer la machine virtuelle.



15) Documentation Docker

Apprendre docker :

https://docs.docker.com/
https://docs.docker.com/engine/installation/
http://putaindecode.io/fr/articles/docker/dockerfile/

Si vous n'avez rien contre les tutos vidéos :
https://www.youtube.com/watch?v=pGYAg7TMmp0

Quelques éléments de bases sur l'architure du docker :

```
Dockerfile tx-debian : Il s'agit du dockerfile dont hériterons les autres images. On y installe les outils de base pour ne pas avoir à les réinstaller dans chaque nouvelle image.

Dockerfile tx-apache : Serveur web basé sur l'image tx-debian.

Dockerfile tx-kanboard : Basé sur tx-apache, télécharge kanboard dans le dossier web (/var/www/html)
```

Note : Le dockerfile tx-etherpad n'est pas basé sur tx-apache car celui-ci fonctionne via NodeJS. L'image est donc basée sur tx-debian

Mise en place des containers :

```
  Actuellement, les images dockerfile sont stockées sur le registry installé sur la machine d'admin
```

Il faut donc construire les images :
https://docs.docker.com/engine/tutorials/dockerimages/

Idéalement, ajouter votre utilisateur au groupe docker pour ne pas avoir à les lancer en root. (Puis se déconnecter/reconnecter pour appliquer les changements).

Exemple avec tx-debian :

```
cd ./path/to/folder
docker build -t tx-debian .
```

NB : L'option t permet d'associer un nom à l'image

On construit les images suivantes sur le même principe en respectant l'ordre des dépendances.

Rappel :

- Lister les containers running : docker ps
- Lister tous les containers : docker ps -a

- Lister les images disponibles : docker images
- Lister les services swarm en cours : docker service ls
- Lister les nœuds d'un cluster swarm : docker node ls
- Lister les processus d'un service swarm : docker service ps mon_service
- Supprimer un service : docker service rm mon_service

Une fois les bases de docker acquises, vous devez comprendre le fonctionnement de Swarm. (Voir la suite du wiki).

16) Architecture des services

Haute disponibilité

Dans le but d'avoir une architecture avec une haute disponibilité, nous avons décidé d'avoir deux machines virtuelles hébergeant les services. Chaque machine se trouvant sur une machine physique différente. Dans un souci de facilité, nous avons décidé de gérer la haute disponibilité de nos services en nous basant sur du round robin DNS. Cette méthode consiste à ajouter plusieurs enregistrements DNS pour un même domaine. Ainsi, lorsque les clients requêtent le domaine, ils sont redirigés sur l'une ou l'autre des machines. Cela permet également de répartir la charge entre les deux serveurs.

Docker Swarm

Pour héberger les services, nous utilisons des conteneurs Docker en mode cluster avec la fonctionnalité Swarm de Docker 1.12. Cela permet de lancer des services et de les scaler en fonction de nos besoins. Si vous voulez plus d'informations, je vous invite à consulter cet article

Une chose à prendre en compte est que sur Swarm, les conteneurs sont placés sur un réseau privé (on parle d'overlay) qui est partagé entre les différents noeuds du cluster. Lors du lancement d'un service sur un noeud du cluster, on peut donc spécifier à quel réseau il peut avoir accès. Une fois lancés, les conteneurs sont capables de résoudre leurs ip privées via des entrées DNS basées sur leur nom. En d'autres termes, si on lance un conteneur avec le flag

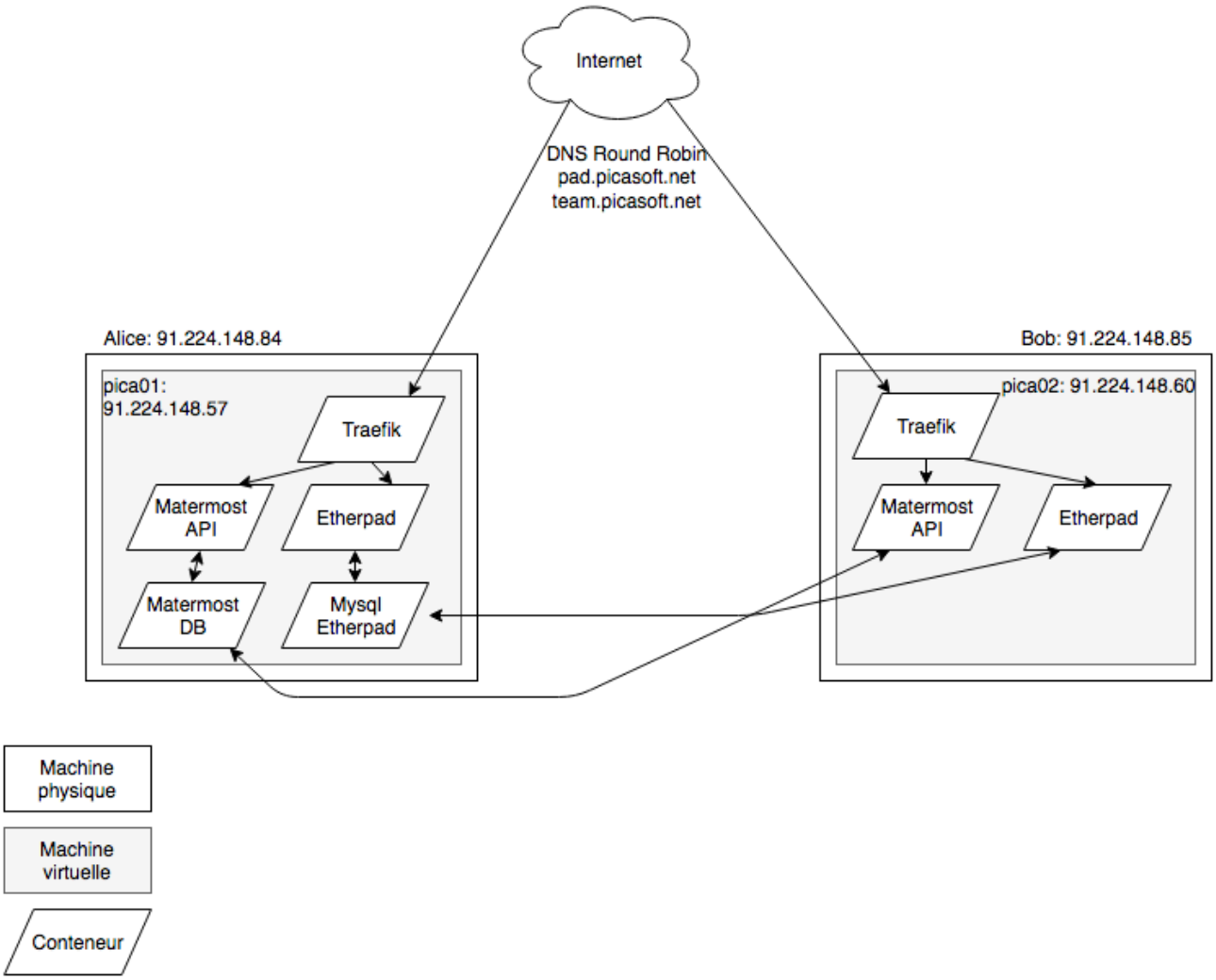
```
--name bdd
```

et un second avec le flag

```
--name apache
```

, cela signifie qu'à partir du conteneur apache, il sera possible d'avoir accès à l'IP et donc au service rattaché au conteneur bdd (un ping bdd pingera l'ip privée du conteneur bdd). l'inverse est aussi vrai.

Architecture



Comme je le disais, les services sont donc lancés sur les deux machines virtuelles. Pour rediriger le flux entrant vers le bon conteneur, nous utilisons un reverse proxy orienté micro service et qui est en mesure de fonctionner sous swarm. Traefik, c'est son nom écoute les actions réalisées sur le démon docker. Lorsqu'il détecte un conteneur lancé avec certaines variables d'environnement, il génère une configuration pour rediriger et répartir la charge vers l'un ou l'autre des conteneurs.

Pour le lancer, on créé le service suivant:

```
docker service create \
 --name traefik \
 --publish 80:80 --publish 443:443 --publish 8080:8080 \
 --mount
type=bind,source=/var/run/docker.sock,target=/var/run/docker.sock
\
 --mount
type=bind,source=/DATA/WEB/traefik/traefik.toml,target=/etc/traefi
```

```
k/traefik.toml \
 --mount type=bind,source=/DATA/WEB/traefik/certs,target=/certs \
 --network pica-net \
 --mode global \
 traefik \
 --docker \
 --docker.swarmmode \
 --docker.domain=swarm.picasoft.net \
 --docker.watch \
 --logLevel=INFO \
 --web
```

Ensuite, il suffit d'ajouter les étiquettes suivantes aux conteneurs pour générer la configuration adéquate

```
--label traefik.port=80
--label traefik.frontend.rule=Host:team.picasoft.net
```

Traefik.port correspond au port d'écoute de notre service dans le conteneur
Traefik.frontend.rule correspond au nom de domaine du service exposé. Il faut au préalable avoir créé les entrées DNS pour que cela fonctionne.
Traefik va donc être capable de configurer à la volée les différents services sans que l'on ait à écrire le moindre fichier de configuration. Il y a également une interface de configuration que l'on peut atteindre via le port 8080 et qui montre l'état des services et différentes statistiques.
Traefik peut aussi générer automatiquement une configuration letsencrypt. Pour cela, il faut utiliser le fichier de configuration suivant:

```
defaultEntryPoints = ["http", "https"]

[entryPoints]
 [entryPoints.http]
 address = ":80"
   [entryPoints.http.redirect]
      entryPoint = "https"
 [entryPoints.https]
 address = ":443"
 [entryPoints.https.tls]

[acme]
email = "picasoft@assos.utc.fr"
storageFile = "/certs/acme.json"
entryPoint = "https"
onDemand = true


[[acme.domains]]
```

```
 main = "swarm.picasoft.net"

[web]
address = ":8080"
```

Au premier lancement du conteneur, Traefik va aller générer un certificat SSL auprès de letsencrypt et ensuite rediriger le service en https.

Problème des bases de données

La plupart des systèmes de base de données ne gèrent pas les accès concurrents au niveau de leurs fichiers. Il faut comprendre par là qu'il n'est pas possible de lancer deux processus mysql qui pointent vers les mêmes fichiers de configuration (/var/lib/mysql). Pour outre passer ce problème, il faut obligatoirement utiliser une solution de plus haut niveau qui va effectuer des synchronisations de données manuellement. Ces solutions sont lourdes à mettre en place. Dans le cadre de cette TX, nous avons volontairement fait le choix de nous passer de ce type de solution et d'avoir des instances standalone de nos bases de données:

- Mysql pour le service etherpad
- PostgreSQL pour le service mattermost

Des sauvegardes sont réalisées toutes les nuits. Cela nous permet de restaurer l'état de la base de données en cas de problème en limitant les pertes de données applicatives à 24h maximum.

17) Mise en place d'un registry Docker

Lorsque l'on utilise Docker, il y a différentes possibilités pour utiliser nos propres images.

Build manuel

La première solution sans doute la plus simple consiste à mettre le Dockerfile sur les machines qui nous intéresse et de builder cette image localement à l'aide de la commande

```
$ docker build -t mon_image .
```

Cette solution fonctionne, mais est longue. En cas de changement sur l'image, il faut rebuilder toute l'image ce qui peut prendre quelques minutes.

Docker Hub

La seconde solution est d'utiliser un registry mis en place par docker. Il permet de stocker ses images de manière centralisée. En cas de modification, on ne rebuild qu'une seule fois l'image. On la pousse ensuite sur le registry et il suffit de tirer la dernière version sur nos machines pour mettre à jour nos images. Cette solution fonctionne bien, mais impose d'exposer publiquement ses images sur le Hub. À cela, on ne sait pas comment sont stockées ni même où sont stockées nos images.

Registry privé

La solution du registry privé permet d'avoir les avantages des deux solutions précédentes. Un registry privé est un entrepôt qui permet d'héberger des images Docker et qui est auto hébergé. Cela permet de savoir exactement où sont nos images et de centraliser celles-ci à un seul endroit.

Mise en place

La mise en place d'un registry est assez simple. La première chose à faire est d'avoir une machine avec Docker d'installée dessus. En effet, le registry va se présenter sous la forme d'une image à déployer sur les machines. En ce qui nous concerne, nous allons aussi utiliser Let's Encrypt pour nous assurer que la communication entre nos machines clientes et le registry s'effectue de manière chiffrée et donc sécurisée.

Let's Encrypt

Pour générer un certificat Let’s Encrypt pour votre registry, vous aurez besoin du client certbot.

- Clonez le repository :

```
git clone https://github.com/certbot/certbot.git
```

- Dans le répertoire certbot, utilisez certbot-auto pour générer un certificat :

```
./certbot-auto certonly --standalone --email admin@example.com
-d registry.picasoft.net
```

- Votre certificat sera stocké dans /etc/letsencrypt.

Installation du registry Docker

Pour s'assurer de la persistance des données, on prépare un dossier pour monter les différentes configurations du conteneur.

```
$ mkdir -p /DOCKER/registry/{data,certs,auth}
```

Dans le répertoire auth, on va générer un couple utilisateur/mot de passe au format htpasswd. Seuls les utilisateurs enregistrés pourront accéder aux images stockées sur le registry.

```
$ docker run --entrypoint htpasswd registry:2 -Bbn pica M0tDeP4sS&
>> /DOCKER/registry/auth/htpasswd
```

Dans le répertoire certs, on retrouve la chaine et la clé du certificat Let’s Encrypt :

```
$ cp
/etc/letsencrypt/live/registry.picasoft.net.net/privkey.pem
/DOCKER/registry/certs/domain.key
$ cp /etc/letsencrypt/live/registry.picasoft.net.net/fullchain.pem
/DOCKER/registry/certs/domain.crt
```

On peut maintenant lancer le registry. Pour cela, on va utiliser docker-compose. S'il n'est pas installé, on procède comme suit:

```
$ apt-get install -y python-pip
$ pip install docker-compose
```

```
$ cat /DOCKER/docker-compose.yml
registry:
  restart: always
  image: registry:2
  ports:
    - 5000:5000
  environment:
    REGISTRY_HTTP_TLS_CERTIFICATE: /certs/domain.crt
    REGISTRY_HTTP_TLS_KEY: /certs/domain.key
    REGISTRY_AUTH: htpasswd
```

```
    REGISTRY_AUTH_HTPASSWD_PATH: /auth/htpasswd
    REGISTRY_AUTH_HTPASSWD_REALM: Registry Realm
  volumes:
    - /DOCKER/registry/data:/var/lib/registry
    - /DOCKER/registry/certs:/certs
    - /DOCKER/registry/auth:/auth
```

On lance le conteneur:

```
$ docker-compose -f /DOCKER/docker-compose.yml up -d
```

Maintenant que le registy est prêt, on peut builder et lui pousser des images:

```
$ docker build -t monimage .
$ docker tag monimage registry.picasoft.net:50000/monimage:v1
$ docker push registry.picasoft.net:50000/monimage:v1
```

Sur les client, il faut au préalable se logger:

```
$ docker login registry.picasoft.net:5000
Username (pica): pica
Password:
Login Succeeded
```

```
$ docker pull registry.picasoft.net:50000/monimage:v1
```

Archi Picasoft

Picasoft possède son propre registry privé. Celui-ci se trouve sur la machine admin. (Voir architecture globale picasoft) La machine admin est accessible depuis alice

```
root@alice $ ssh admin
```

Comme ci-dessus, les configurations se trouvent dans le répertoire /DOCKER. Les images utilisées et celles que nous avons préparées sont poussées sur le registry. Les sources de ces images se trouvent dans le répertoire tx-chatons

```
root@admin:/DOCKER/tx-chatons# pwd
/DOCKER/tx-chatons
root@admin:/DOCKER/tx-chatons# ll
total 36
drwxr-xr-x 4 root root 4096 nov.  26 11:50 database
drwxr-xr-x 2 root root 4096 nov.  26 11:50 docker-compose
drwxr-xr-x 3 root root 4096 nov.  26 11:53 tx-apache
drwxr-xr-x 2 root root 4096 nov.  26 11:53 tx-debian
```

```
drwxr-xr-x 2 root root 4096 déc.   6 21:09 tx-dokuwiki
drwxr-xr-x 3 root root 4096 déc.  15 19:33 tx-etherpad
drwxr-xr-x 2 root root 4096 nov.  26 11:53 tx-kanboard
drwxr-xr-x 6 root root 4096 déc.   2 18:46 tx-mattermost
drwxr-xr-x 2 root root 4096 nov.  30 20:27 tx-nginx
```

La machine d'admin n'a pas d'IP publique. Elle est donc sur un réseau privé d'alice (Création d'une machine virtuelle cf partie réseau)
Pour accéder au registry exposé sur le port 5000 de la machine, il y a donc un redirection de port. Le port 5000 d'alice ainsi que les ports 80/443 d'alice sont redirigés vers la machine admin. Le port 5000 est utilisé par le registry et les ports 80/443 par letsencrypt pour la génération des certificats.

Dans notre exemple, le domaine registry.picasoft.net a donc un champ A qui pointe vers l'IP d'alice.

18) Rédaction des images Docker

Dans le cadre de cette TX, différentes images Docker ont été réalisées. Celles-ci sont stockées au sein du gitlab de l'UTC et sont également présente sur la machine admin qui héberge le registry (/DATA/docker/Dockerfiles)

Bonnes pratiques

Pour la réalisation des images de la TX, nous avons décidé de créer une première image basée sur l'image Debian officielle et sur laquelle nous avons installé des outils en plus comme l'éditeur de texte vim, curl ou encore git. Lorsque l'on doit éditer un fichier de configuration sur un conteneur en fonctionnement, il n'y a donc rien à installer en plus.
Les images client héritent de ce premier conteneur en spécifiant celui-ci comme étant l'image de base dans le FROM du Dockerfile.
Concernant les services, nous avons décidé d'utiliser le gestionnaire de processus supervisord pour lancer les commandes dans le conteneur. Il permet de gérer les processus avec une plus grande souplesse de même que les logs associés. Par exemple, il est possible de lancer un serveur web avec l'utilisateur www-data et de rediriger les logs sur la sortie standard. En parallèle, on peut imaginer faire fonctionner un processus cron avec l'utilisateur root sans redirection des logs.
Supervisord est vraiment très simple à mettre en place et se configure via un fichier de configuration yaml.

Réduire au maximum des couches

Un Dockerfile est composé de différentes couches. Une couche correspond à une instruction exécutée par le Dockerfile. Lorsqu'un conteneur est lancé, il va accéder aux commandes du conteneur en remontant les différents niveaux. À la manière de l'héritage en programmation, si un conteneur cherche a appeler une commande qu'il ne connaît pas, il va remonter à sa classe mère et ainsi de suite. Or, ces accès réduisent la vitesse d'exécution de la commande. Il faut donc veiller à réduire au maximum le nombre de couches d'un conteneur et privilégier les instructions avec plusieurs commandes:

```
#Exemple:
RUN echo "deb http://packages.dotdeb.org jessie all" >
/etc/apt/sources.list.d/dotdeb.list && \
wget -O- https://www.dotdeb.org/dotdeb.gpg | apt-key add - && \
apt-get update -y && apt-get install -y php7.0 php7.0-fpm php7.0-
gd php7.0-xml nginx supervisor curl tar
```

Attention aux volumes

Lors de la réalisation d'images, on peut être tenté d'exposer des volumes sur des images “temporaires”. Par exemple, un conteneur nginx auquel on va exposer le volume /var/www/html. Le problème de cette pratique est que si l'on veut créer une image dokuwiki basée sur cette image nginx, il ne sera pas possible d'écrire par dessus le volume /var/www/html. Même si au sein du Dockerfile on spécifie des commandes pour remplacer le contenu de l'image, au final, le contenu du dossier restera celui de l'image mère.

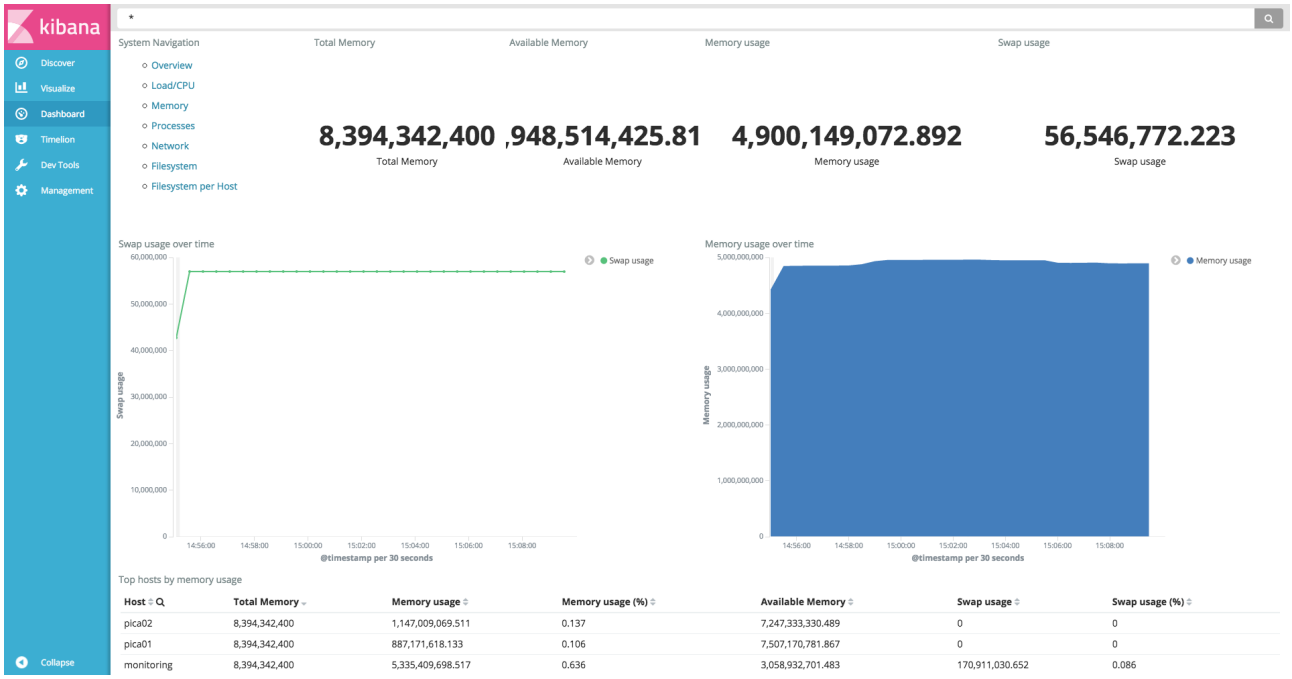
Documentation Docker

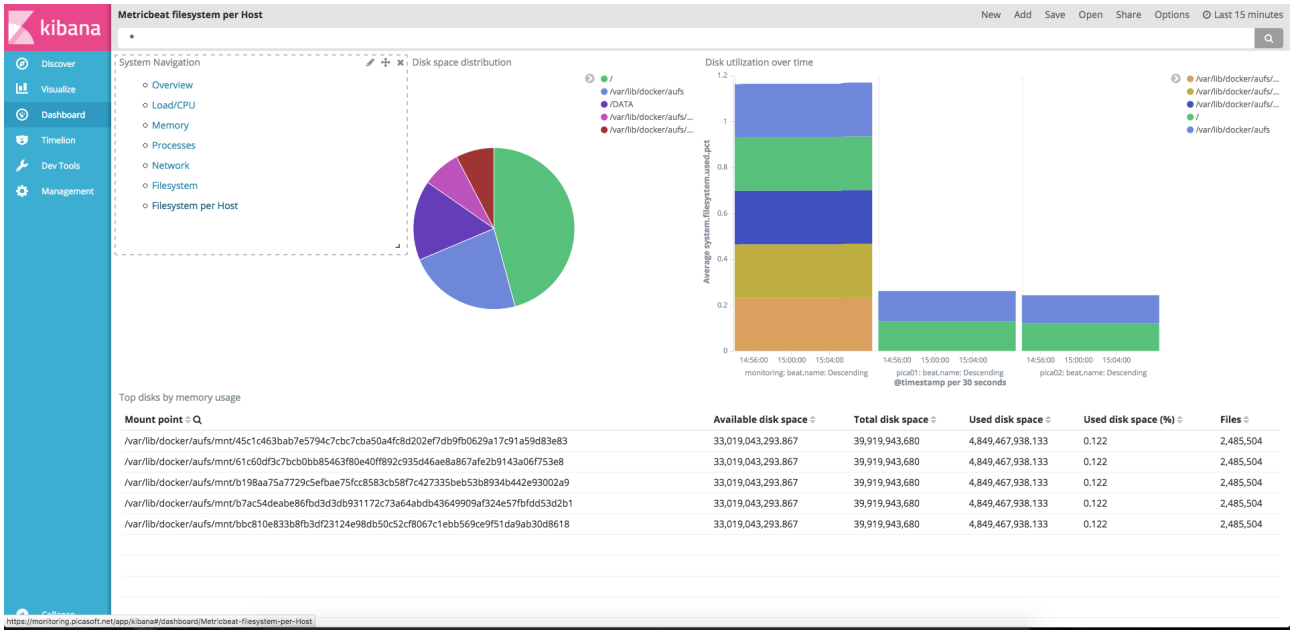
Best practices for writing Dockerfiles Dockerfile reference

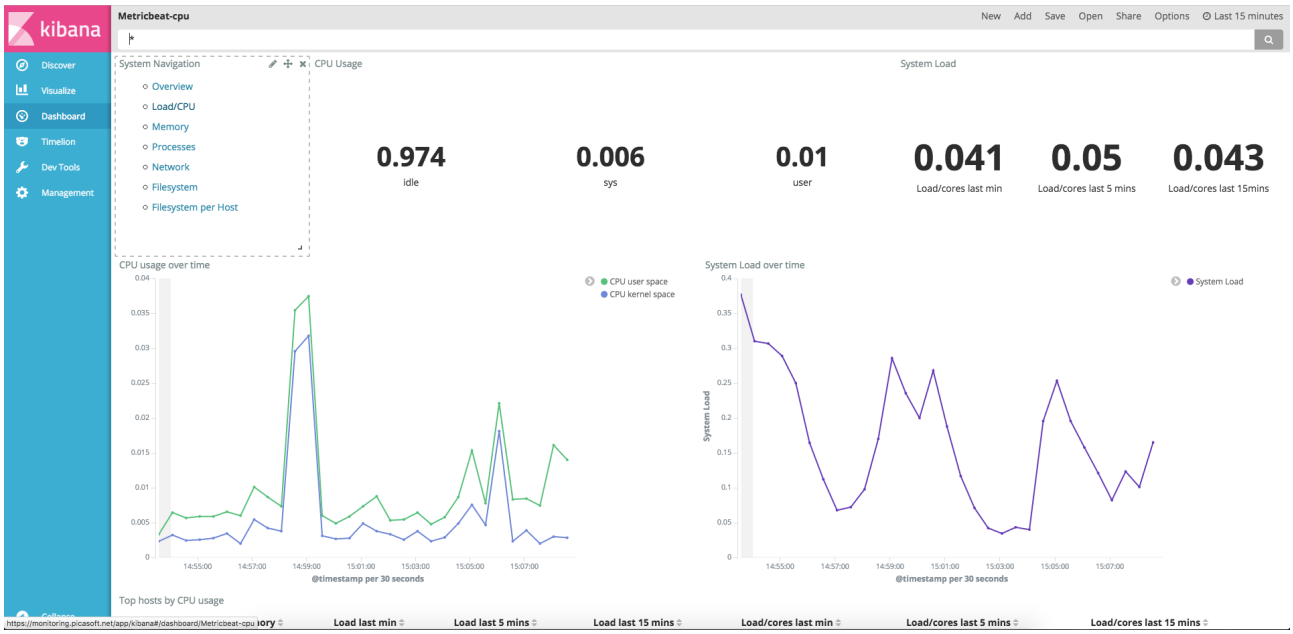
19) Monitoring des VM hébergeant les services

Lors de cette TX, nous voulions avoir une remontée d'information centralisée de l'état de nos machines afin de nous rendre compte de la charge ou non de celle-ci. Comme le temps nous a manqué, la solution retenue a été la mise en place d'une solution metricbeat, couplée à un moteur de recherche elasticsearch et au client kibana pour visualiser les données. En l'état actuel, cela fonctionne et il est possible d'avoir des informations sur l'état des machines via l'URL suivante.
Cette solution nécessiterait que l'on se penche dessus afin de sécuriser l'accès au Kibana par exemple. Une solution de véritable monitoring est aussi à envisager avec le déploiement d'une solution comme checkmk par exemple.

Aperçu de l'interface finale




kibana
Metricbeat filesystem per Host
New Add Save Open Share Options Last 15 minutes
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
System Navigation
Overview
Load/CPU
Memory
Processes
Network
Filesystem
Filesystem per Host
Disk space distribution
Disk utilization over time
Average system.filesystem.used.pct
@timestamp per 30 seconds
Top disks by memory usage
Mount point
Available disk space
Total disk space
Used disk space
Used disk space (%)
Files
/var/lib/docker/aufs/mnt/45c1c463bab7e5794c7cbc7cba50a4fc8d202ef7db9fb0629a17c91a59d83e83 33,019,043,293.867 39,919,943,680 4,849,467,938.133 0.122 2,485,504
/var/lib/docker/aufs/mnt/61c60df3c7bcb0bb85463f80e40ff892c935d46ae8a867afe2b9143a06f753e8 33,019,043,293.867 39,919,943,680 4,849,467,938.133 0.122 2,485,504
/var/lib/docker/aufs/mnt/b198aa75a7729c5efbae75fcc8583cb58f7c427335beb53b8934b442e93002a9 33,019,043,293.867 39,919,943,680 4,849,467,938.133 0.122 2,485,504
/var/lib/docker/aufs/mnt/b7ac54deabe86fbd3d3db931172c73a64abdb43649909af324e57fbfdd53d2b1 33,019,043,293.867 39,919,943,680 4,849,467,938.133 0.122 2,485,504
/var/lib/docker/aufs/mnt/bbc810e833b8fb3df23124e98db50c52cf8067c1ebb569ce9f51da9ab30d8618 33,019,043,293.867 39,919,943,680 4,849,467,938.133 0.122 2,485,504


kibana
Metricbeat-cpu
New Add Save Open Share Options Last 15 minutes
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
System Navigation
Overview
Load/CPU
Memory
Processes
Network
Filesystem
Filesystem per Host
CPU Usage
0.974
idle
0.006
sys
0.01
user
System Load
0.041
Load/cores last min
0.05
Load/cores last 5 mins
0.043
Load/cores last 15mins
CPU usage over time
CPU user space
CPU kernel space
System Load over time
System Load
@timestamp per 30 seconds
Top hosts by CPU usage
Load last min
Load last 5 mins
Load last 15 mins
Load/cores last min
Load/cores last 5 mins
Load/cores last 15 mins

MetricBeat

Afin de remonter les information, nous utilisons donc la solution MetricBeat. C'est un produit open source développé par la société Elastic et qui permet de remonter des informations sur l'état des machines (nombre de processus, mémoire utilisée, système de fichier ...). En sortie, ce MetricBeat est configuré pour envoyer ses données vers le moteur de recherche Elasticsearch. Il stocke et indexe les données chaque jour et permet de requêter ces données via une API REST ou via l'interface web Kibana.

Le fichier de configuration se trouve dans /etc/metricbeat/metricbeat.yml

```
root@hostname:~# cat /etc/metricbeat/metricbeat.yml  | grep -v '#'
| grep -v '^$'
metricbeat.modules:
- module: system
  metricsets:
    - cpu
    - load
    - diskio
    - filesystem
    - fsstat
    - memory
    - network
    - process
  enabled: true
  period: 10s
  processes: ['.*']
```

```
name: hostname
output.elasticsearch:
  hosts: ["localhost:9200"]
```

Comme on peut le voir ci-dessus, on remonte toutes les métrique et on les envoie en output vers un elasticsearch sur le port 9200. En effet, afin de sécuriser l'Elasticsearch et d'empêcher des écritures depuis n'importe où, nous avons effectué une redirection SSH depuis le serveur de monitoring vers les machines hébergeant les services.
Un utilisateur ssher à été instancié sur la machine de monitoring. Les clés de pica01 et pica02 ont été ajoutées à cet utilisateur ce qui permet de se connecter sans mot de passe. Ensuite, on lance la commande suivante.

```
ssh -nNT -f -L9200:127.0.0.1:9200 ssher@monitoring.picasoft.net
```

Cette commande ouvre une session distante sur le serveur de monitoring avec le user ssher et redirige le port 9200 de la machine de monitoring vers le port 9200 de la machine locale.

Pour s'assurer que cette connexion reste ouverte en permanence, nous avons installé autossh. Cela permet de rouvrir le pont ssh si jamais la connexion a été coupée. On le lance avec la commande:

```
autossh -M 0 -N -f -L9200:127.0.0.1:9200
ssher@monitoring.picasoft.net
```

Il ne faut pas oublier de l'ajouter dans le /etc/rc.local pour le lancer au démarrage du serveur.

Elasticsearch

Les données sont donc stockées dans le moteur de recherche Elasticsearch. Comme sur toute la plateforme, celui-ci est lancé dans un conteneur Docker. Au contraire de pica01/02, la machine de monitoring n'utilise pas swarm pour faire tourner ces conteneurs. Comme pour pica01/02, la configuration docker se trouve dans le /DATA.
Du fait que l'on utilise pas swarm, on peut utiliser docker-compose pour orchestrer et lancer nos conteneurs:

```
version: '2'

services:
  elasticsearch:
```

```
    image: elasticsearch
    ports:
      - "127.0.0.1:9200:9200"
      - "127.0.0.1:9300:9300"
    environment:
      ES_JAVA_OPTS: "-Xms4g -Xmx4g"
    volumes:
      - /DATA/elasticsearch/data:/DATA/elasticsearch/data
      -
/DATA/elasticsearch/config/elasticsearch.yml:/DATA/elasticsearch/c
onfig/elasticsearch.yml
    networks:
      - docker_elk
  kibana:
    image: kibana
    volumes:
      - /DATA/kibana/config/kibana.yml:/etc/kibana/kibana.yml
    networks:
      - docker_elk
    depends_on:
      - elasticsearch
      - traefik
    ports:
      - 5601:5601
networks:
  docker_elk:
    driver: bridge
```

Le service Elasticsearch est bien exposé sur le 127.0.0.1:9200/9300 pour éviter que celui-ci ne soit accessible depuis l'extérieur via l'IP publique. On ne peut donc attaquer l'Elasticsearch que depuis la machine locale ou depuis des redirections SSH.

Pour faire du nettoyage dans les données qui sont stockées au sein de l'Elasticsearch, nous utilisons l'outil curator lui aussi développé par Elastic. Il permet de lister et de faire des actions sur des index. Ce qui nous intéresse ici, c'est de supprimer les données plus vieilles de 3 jours afin de ne pas saturer l'espace de stockage sur la machine de monitoring. Pour cela, on créé une cron qui lance la commande suivante:

```
root@monitoring:/DATA/elasticsearch# cat /etc/cron.daily/remove-
elasticsearch-index
#!/bin/bash

/usr/bin/curator_cli --host localhost --port 9200 delete_indices
--filter_list
```

```
'[{"filtertype":"age","source":"name","direction":"older","unit":"
days","timestring": "%Y.%m.%d","unit_count":3},
{"filtertype":"pattern","kind":"prefix","value":"metricbeat"}]'
```

Cette commande permet donc à curator de lister les index plus vieux de 3 jours en se basant sur le nom de l'index et de les supprimer.

Ce qui peut être amélioré

Ce système fonctionne bien et il a l'avantage de se mettre en place très rapidement. En revanche, il n'est pas en mesure de lever des alertes en cas de problèmes. Une solution de monitoring complète permettrait d'avoir plus d'informations sur les machines et de notifier les administrateurs en cas d'anomalies.

Contacts en cas de questions :
Antoine Picasoft
Grégoire Picasoft

20) Conclusion

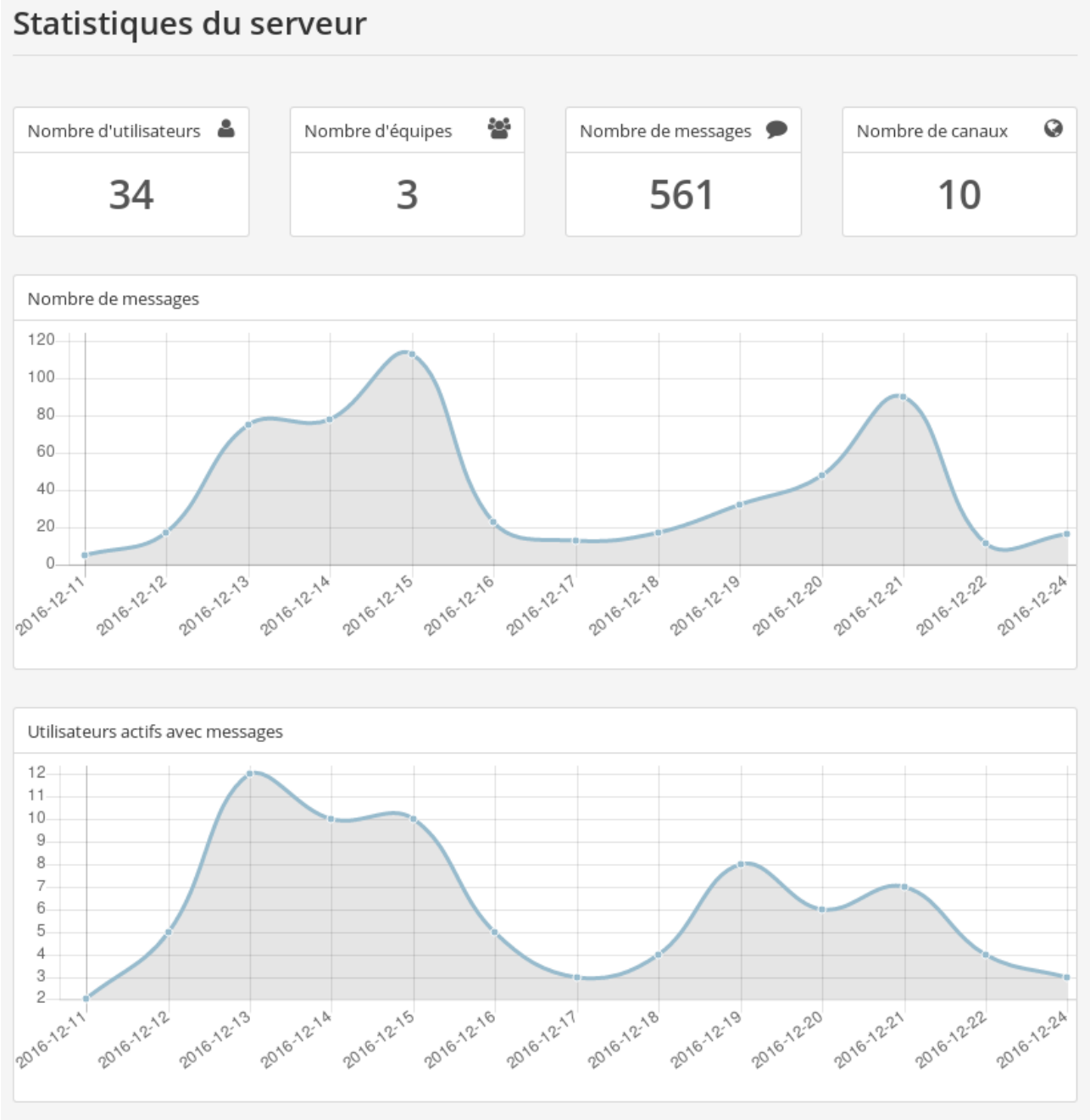
Critiques :

Critiques positives :

- La mise en place de l'architecture et des premiers services s’est correctement déroulée : deux services (Mattermost & Etherpad) sont disponibles au public avant la fin de la TX.

- L'amphi de présentation à permis de rassembler de nombreux intéressés dont certains souhaitent s'investir dans la poursuite de Picasoft : en tout, environ 80 personnes sont tenues informées des avancements de Picasoft par mail.

- Au 25 décembre, le Mattermost de Picasoft est déjà utilisés par plus de 30 utilisateurs (capture réalisée dans la console administrateur du Mattermost Picasoft).



Critiques négatives

- Utilisation d'instance standalone pour les bases de données (voir ici).
- La solution de monitoring (Metricbeat/ElasticSearch/Kibana) peut être remplacée par une solution de monitoring et d'alerting complète telle que Checkmk.
- Par faute de temps, nous n'avons pas pu mettre en place un test de charge sur les services.

Perspectives :

Il est important pour une initiale comme Picasoft de s'inscrire dans la durée. C'est pour cette raison que Picasoft est désormais un club au sein de l'UTC https://assos.utc.fr/asso/picasoft.

De nombreuses perspectives s'ouvrent maintenant pour Picasoft, notamment :

- La mise en place d'ateliers de formations et de sensibilisation;
- L'organisation de conférence au sein de l'UTC;
- La sensibilisation dans les écoles;

Organisation pour la suite

Les personnes souhaitant continuer le travail qui à été commencé dans cette TX seront amenés à :

1. Mettre en place de nouveaux services : Studs, Ethercalc...;
2. Corriger les critiques négatives qui ont étaient formulés :

- En mettant un place un système de monitoring complet,
- En réalisant un test de charge,
- En réalisation une base de données partagée (enlever l'instance standalone);

3. Améliorer la documentation technique pour assurer la pérennité;