

Compilatio

Présentation

Compilatio est une entreprise qui va s'occuper de vérifier s'il y a du plagiat dans un document. Le plagiat est le fait de copier les idées d'un autre sans le consentement de celui-ci. Elle va donc chercher les similitudes en analysant les documents, repérer et identifier les copier-coller : elle recherche les passages identiques entre le document et les ressources publiques disponibles sur Internet.



Elle propose une solution qui est généralement utilisée par les enseignants et les étudiants. Les enseignants vont s'en servir pour regarder si l'étudiant à fait des copier-coller sur Internet et elle renvoie les sources pour prouver que c'est le même contenu. Tandis que les étudiants peuvent s'en servir pour éviter de se faire sanctionner pour plagiat. Actuellement l'entreprise cherche à valider les normes de sécurité ISO 9001 pour la qualité organisationnelle et de sécurité du système d'organisation ISO 27001. Elle cherche également à faire face au fléau ChatGPT qui est une IA qui peut faire tout ce qu'on lui dit, par exemple : un étudiant va utiliser ChatGPT pour écrire sa dissertation en CGE. Compilatio met en œuvre un contre pour trouver un contre face à celui-ci.

Solutions

Compilatio Magister: Programme complet anti plagiat

Logiciels de détection de similitude en ligne, supports de sensibilisation, outils de formations, accompagnement personnalisé.



Logiciel anti-plagiat

- ✓ Pourcentage de similitudes et ajustement possible
- ✓ Liste, provenance et emplacements des sources
- ✓ Collecte des devoirs depuis votre compte
- ✓ Rapport d'analyse



Ressources pédagogiques

- ✓ Tutoriels d'utilisation du logiciel
- ✓ Ressources documentaires sur le plagiat, les normes de citation, le droit d'auteur...
- ✓ Modèles de documents réglementaires
- ✓ Supports de communication



Accompagnement institutionnel

- ✓ Interlocuteur dédié au suivi du projet anti-plagiat
- ✓ Formation à la mise en place du programme
- ✓ Webinars récurrents
- ✓ Support technique et guides



Services complémentaires

- ✓ Formations pour enseignants et étudiants
- ✓ Enquêtes pour état des lieux de votre projet
- ✓ Site internet personnalisé dédié au plagiat
- ✓ Intégration dans votre LMS

Comptez sur l'efficacité du logiciel Compilatio Magister

Performance et fiabilité des analyses de similitudes

- ✓ Comparaison de vos documents avec un **grand nombre de sources** : web mondial, publications scientifiques et documents des utilisateurs Magister du monde entier.
- ✓ Analyse de **tous les formats** de documents : Google Drive, Word, Excel, Acrobat...
- ✓ Interface et analyses dans **plusieurs langues**.

Appuyez-vous sur des résultats précis

Une analyse complète et objective

- ✓ **Pourcentage de similitudes** : affinage possible du score de similitudes, en ignorant les passages entre guillemets ou les sources correctement citées.
- ✓ **Emplacement des similitudes** : indication de chaque source dans le texte et sa proportion.
- ✓ **Provenance des sources** : origine des sources indiquée (web mondial, documents déjà analysés, devoirs d'un professeur de votre établissement ou d'un établissement partenaire...), comparaison du texte avec la source en face-à-face.
- ✓ **Rapport d'analyse** : synthèse des résultats en vue d'être présentés à l'étudiant ou à un jury.

Utilisez un outil pensé pour faciliter l'enseignant

Simplicité de déploiement et d'utilisation

- ✓ **En ligne, sans installation** (service SaaS).
- ✓ Intégration plateforme pédagogique : Moodle, Canvas, BlackBoard, Brightspace, Microsoft Teams...
- ✓ Notices et tutoriels d'utilisation.
- ✓ Webinars de découverte et d'accompagnement.

Autonomie et liberté de gestion de vos données

- ✓ Pilotage de l'indexation à votre **bibliothèque de références** (matériel de comparaison pour vos analyses).
- ✓ Choix pour **collecter les devoirs** numériques : téléchargement manuel ou dépôt par les étudiants.
- ✓ **Organisation personnelle** de votre espace et de vos actions : création de dossiers, paramétrage des seuils de similitudes, planification de vos analyses...

Partagez les ressources éducatives aux étudiants

Des supports prêts à intégrer vos cours

Gagnez du temps en vous appuyant sur des **ressources clés en main**, élaborées par des experts de l'intégrité académique.

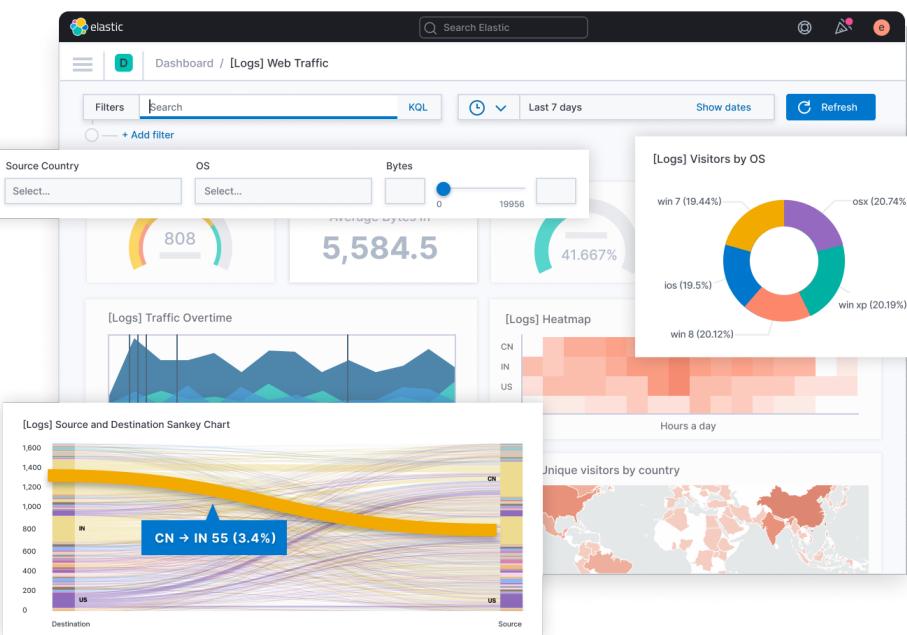
- ✓ **Ressources spécialisées et variées** autour de la sensibilisation au droit d'auteur.
- ✓ **Accès illimité** pour toutes les équipes pédagogiques, directement depuis chaque compte utilisateur Compilatio Magister.
- ✓ **Diffusion ou inspiration libres des ressources** dans le cadre de cours sur la recherche académique.

Afin de détecter les similitudes, le logiciel va faire une comparaison entre les documents, ElasticSearch (moteur de recherche et d'analyse) permet de faire des recherches et afficher tous les fichiers correspondant à la recherche, il va ensuite faire une boucle pour chercher dans tous les documents proposé par ElasticSearch et faire un score de comparaison pour déterminer la pertinence des fichiers. De plus, Anaframe fait la liaison entre tous les modules. Après avoir fait la comparaison, il fournit un rapport avec le taux de plagiat, les sources et les points d'intérêts.

Présentation des logiciels et des plateformes

ElasticSearch

ElasticSearch est un moteur de recherche et d'analyse. Il est ouvert à tous types de données tels que : les données textuelles, numériques, géospatiales, structurées et non structurées. Dans ElasticSearch, l'unité est le document. L'index comporte donc l'ensemble des documents qui sont stockés dans au format JSON dans le moteur. L'utilisation d'ElasticSearch comprend plusieurs avantages car il a été conçu pour gérer les montées de charges et il peut supporter de gros volumes de données. Kibana est un greffon de visualisation de données pour ElasticSearch, cela marche comme de la supervision.

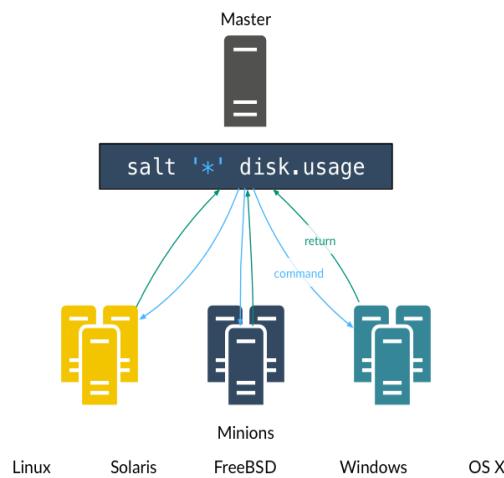


Le monitoring consiste à la surveillance d'un réseau, son but est d'assurer la disponibilité et la performance des infrastructures, des équipements, des logiciels et des processus supportant les données. Compilatio utilise actuellement Munin pour le monitoring et souhaite se diriger vers Grafana. L'alerting consiste lui, à prévenir les administrateurs réseau en cas de défaillance. Le logiciel d'alerting que Compilatio utilise est Shinken et tout comme Munin il souhaite changé et passé à Prometheus.

SaltStack

SaltStack est un logiciel de gestion de configuration fonctionnant sur le principe client-serveur. Il a pour but de rendre la gestion de configuration simple mais flexible.

On retrouve deux services: salt-master représentant le serveur et salt-minion représentant le client.



MongoDB

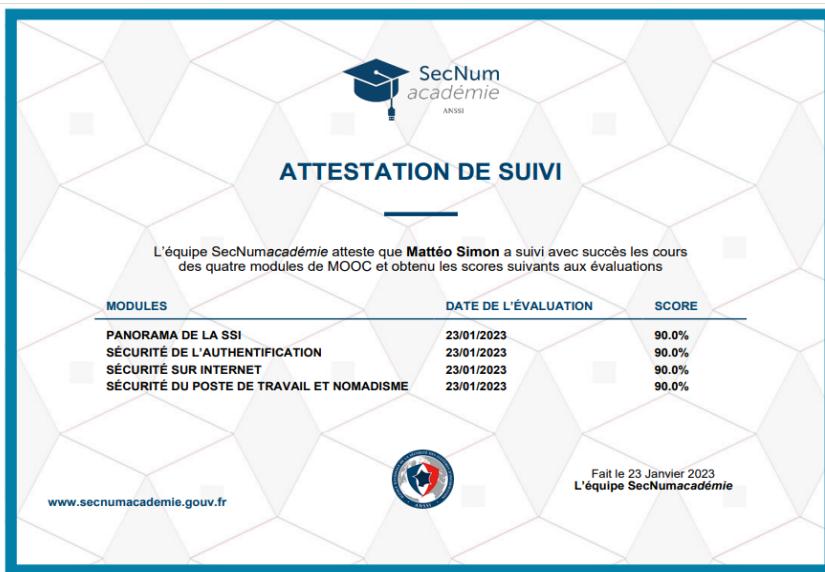
C'est une base de données NoSQL, elle peut traiter des données structurées, semi-structurées et non structurées. Elle utilise un modèle de données non relationnel, orienté document. Elle est utilisée pour le stockage de volumes massifs de données. Les avantages de MongoDB est qu'il est très flexible et permet de stocker plusieurs types de données. Elle utilise le sharding qui est un processus permettant de diviser les données et de les répartir sur plusieurs serveurs. Et pour finir, MongoDB stocke les données dans la RAM pour un accès plus rapide et une meilleure performance lors de l'exécution des requêtes donc elle nécessite moins de puissance de traitement pour rechercher et récupérer des données qu'une base de données relationnelle.

Certifications

La certification MOOC (Massive Open Online Courses)

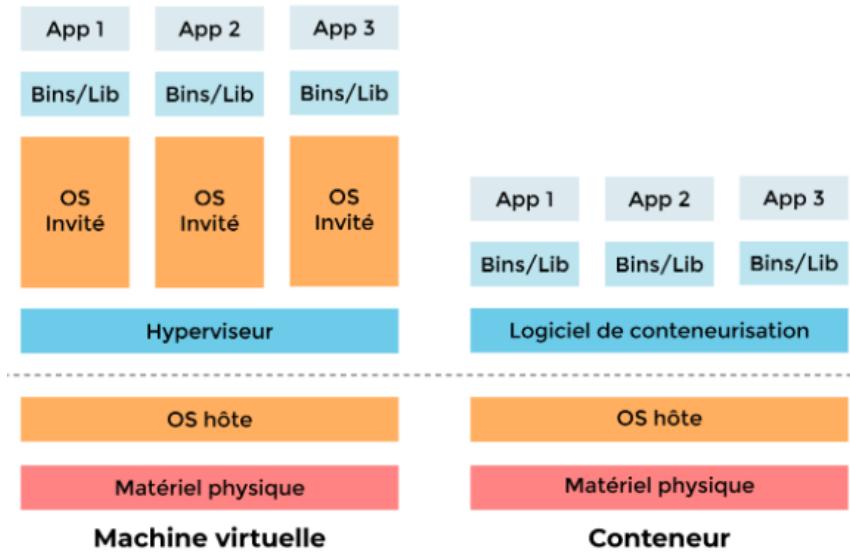


La Certification MOOC est une certification qui comporte 4 modules : Panorama de la SSI, Sécurité de l'authentification, Sécurité sur Internet et Sécurité du poste de travail et nomadisme.



Cours sur les Dockers (OpenClassroom)

Docker est une plateforme logicielle qui vous permet de concevoir, tester et déployer des applications rapidement à l'aide de conteneurs. Un conteneur Linux est un processus ou un ensemble de processus isolés du reste du système, tout en étant légers. Il permet de faire de la virtualisation légère, c'est-à-dire qu'il ne virtualise pas les ressources, il ne crée qu'une isolation des processus. Le conteneur partage donc les ressources avec le système hôte.



Le principal avantage est qu'il est plus sobre qu'une VM et donc permet d'être moins vulnérable et d'utiliser moins de puissance tout en répondant aux mêmes besoins. Il permet également d'installer des paquets sur leurs dernières versions. Prenons

l'exemple de PHP, dans Tags on voit les différentes versions de PHP et on peut remarquer qu'il y a la commande d'installation au dessus.

Docker Hub - php

OVERVIEW TAGS

Sort by Newest Filter Tags

TAG	OS/ARCH	COMPRESSED SIZE
latest	linux/386	164.21 MB
94897da7ce60	linux/amd64	162.02 MB
6631d939bcde	linux/arm/v5	140.61 MB
3e653b50c24d		
+5 more...		

docker pull php:latest

Il y a bien évidemment d'autres avantages à utiliser Docker comme la Standardisation des opérations, la Migration aisée et le Développement d'applications de façon plus efficiente, plus rapide et avec moins de ressources. Pour finir, si un pirate arrive à s'infiltrer dans le réseau et il arrivera dans un docker et l'avantage est qu'il restera coincé dans le docker, il devra trouver une solution et pour sortir du docker et pour passer le docker pour arriver à ses fins.

Cours sur le système d'exploitation Linux (OpenClassroom)

Cours

Administrez un système Linux	24/01/2023	<div style="width: 100%;">100 %</div>	Obtenir votre certificat
Gérez votre serveur Linux et ses services	24/01/2023	<div style="width: 100%;">100 %</div>	Obtenir votre certificat

Projet Compilatio : Monter un réseau virtuel

Contexte : Avec la grève du 31/01, je n'ai pas pu aller en stage alors on m'a donné pour mission de faire un réseau virtuel contenant un serveur passerelle, un serveur DHCP, un serveur DNS, un serveur NTP et un serveur LDAP.

Création et configuration d'un serveur DHCP

à finir : dire quel service on installe (isc-dhcp-server) + prendre les screens de la config (192.68.56.1) + montrer que les autres serveurs récupèrent bien les adresses IP + attribuer spécialement une adresse IP au serveur DNS (192.168.56.2).

Création et configuration d'un serveur DNS

à finir : dire quel paquet on installe (bind9) + prendre des screens de la configuration de chacun des fichiers + montrer qu'on ping bien Internet + attribuer le DNS aux serveurs du réseau et ping Internet.

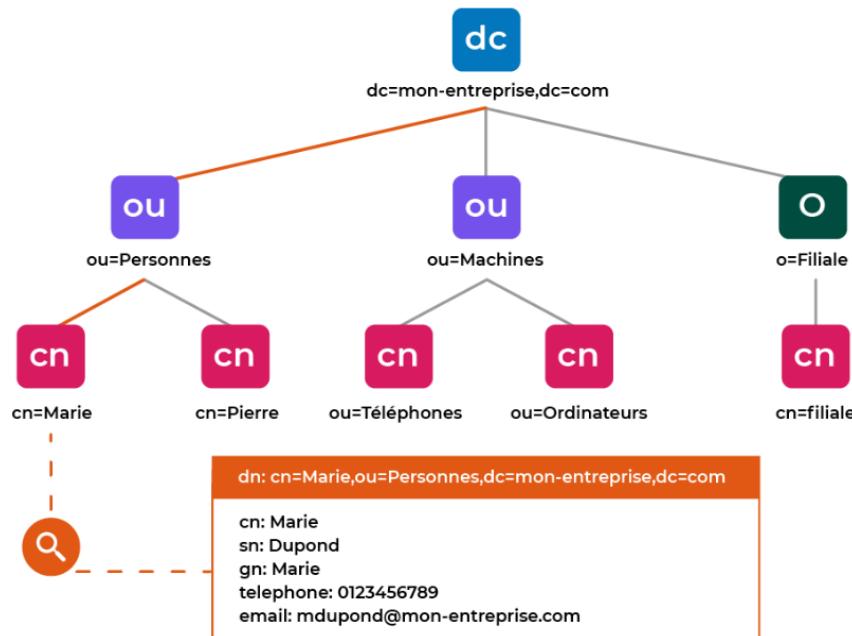
Création et configuration d'un serveur NTP

Network Time Protocol est un protocole qui permet de synchroniser l'horloge locale des ordinateurs sur une référence d'heure. Il est important d'avoir un serveur NTP car ce service s'arrête au déclenchement du seuil de panique, les journaux enregistrent un événement, ce qui peut alerter sur une possible activité malveillante. On peut conclure qu'un serveur NTP garantit une certaine sécurité dans un réseau.

à finir : dire quel service on installe (chrony) + prendre les screens de la configuration + montrer que les serveurs du réseau ont bien l'heure à jour.

Création et configuration d'un annuaire LDAP

LDAP est un protocole ouvert et multiplateforme utilisé pour l'authentification des services d'annuaire. Les services d'annuaires stockent les comptes et les mots de passe des utilisateurs et des ordinateurs. L'annuaire LDAP partage également les informations avec d'autres entités du réseau. Il est organisé de façon hiérarchique d'entrées. Cette organisation constitue un arbre DIT (Directory Information Tree).



à finir : dire quel service on installe (slapd ldap-utils) + prendre les screens de la configuration + LDIF + backup de l'annuaire LDAP.

A savoir sur l'environnement de Compilatio

```
root@ns3045309(trans-saltmaster-absol):~# cd /srv
root@ns3045309(trans-saltmaster-absol):/srv# ls
pillar  salt
root@ns3045309(trans-saltmaster-absol):/srv# █
```

On peut remarquer 2 répertoires dans /srv : pillars et salt. Pillars est un répertoire où l'on va définir des paramètres propres à des environnements, cela servira à dynamiser des configurations. Salt est un répertoire où l'on va retrouver les configurations des serveurs.

Nouvelles commandes UNIX

Dans cette installation, j'ai pu voir de nouvelles commandes. Certaines commandes sont propres à eux. Je ne vais pas expliquer ici les commandes qui leurs sont propres car elles seront expliquées dans l'installation.

grep = filtrer une commande

break = sortir d'une boucle

sed = remplacer

dig = collecter les informations DNS

wc = compter les lignes

sort = trier dans l'ordre alphabétique

awk = récupérer les informations, marche comme les tableau en bash

uptime = montre depuis combien de temps la machine est optionnelle

iptables = configurer le Firewall

ll = abréviation de ls -l

xargs va récupérer la liste des commandes et permet de l'appliquer

zcat = voir le contenu d'un fichier compressé

tail = aller à la fin d'un fichier

iostats = surveiller les statistiques d'entrées/sorties du système

fdisk = lister les partitions et les disques avec leurs places

df -h = lister les partitions et les disques avec leurs places

Installation de serveurs à l'aide de scripts sur OVH

Objectif

L'objectif ici est d'installer de nouveaux serveurs avec Ubuntu à jour. Ces nouveaux serveurs remplaceront les anciens et seront plus performant car ils seront plus récent donc plus de mémoire...

Les objectifs sont les suivants :

- Installation les nouveaux serveurs
- Mettre les serveurs dans la configuration
- Appliquer les modifications
- Vérification de la création des serveurs et de la réPLICATION des fichiers
- Suppression des anciens serveurs

Réinstallation d'un serveur prod-worker à l'aide de scripts

Pour cette installation nous allons commencer par créer une variable combos comportant le nom du serveur et le nom de la machine OVH + le reverse DNS. Le reverse DNS permet de faire une résolution de nom depuis une IP vers un nom d'hôte pleinement nommé (FQDN).

```
combos="prod-worker-zamazenta,ns3148766.ip-51-91-72.eu"
```

Ensute on entre dans une boucle. On va tout d'abord faire en sorte de prendre la valeur donnée dans la variable \$combos et l'implémenter dans OVH. On peut remarquer SoYouStart qui correspond à une gamme d'infrastructures pour les entreprises.

Cette boucle utilise la commande msrv (Manage Serveur) qui appelle plusieurs API (OVHApi et SoYouStartApi) qui elles mêmes appellent plusieurs fonctions différentes permettant de se connecter automatiquement à OVH avec compilatio.net et également à se diriger dans les fonctionnalités pour installer le serveur. Grâce à la commande POST on va pouvoir mettre toutes les valeurs qu'on veut pour la configuration du serveur.

Au niveau de la sécurité, il y a les commandes mfw (Manage Firewall) pour ajouter le nom du serveur, l'ajouter dans l'environnement prod et pour avoir les rôles que l'on a choisi.

Pour finir, on prend la template worker et on force la réinstallation (car il y avait déjà un serveur existant sinon nous n'avons pas besoin de faire --force_reinstall).

```
#  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'  
    soyoustart_name=`echo $combo | sed 's/.*,/'`  
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"  
    #msrv $compi_name $soyoustart_name --only_dns || break  
    #msrv $compi_name $soyoustart_name --only_reverse || break  
    #mfw server $soyoustart_name add || break  
    #mfw server $compi_name add envs prod || break  
    #mfw server $compi_name add input_roles nrpe munin_node ssh dont_backup_me gridfs-mongos mongodb-mongos mongoqueueuing-mongos mongoperf-mongos mongomajoperf-mongos worker captcha_proxy zeromq mongo_temp redis-cluster || break  
    #msrv $compi_name $soyoustart_name --only_install --template worker --force_reinstall || break  
    echo  
    echo  
    echo  
done
```

Ensute on va lister les clés et on va accepter le nouveau serveur.

```
salt-key -L  
salt-key --accept=prod-worker-*
```

```
-----  
Génération des certificats  
-----  
  
#  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'  
    echo "===== $(date) - compi_name[$compi_name]"  
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name  
done
```

On va lister le contenu du répertoire demandé et on va filtrer ce qu'on va lister.

```
# Contrôle présence des nouveaux serveurs :  
ll /srv/pillar/prod/certifs | grep -v morpheo | grep prod-worker
```

Dans cette boucle on va faire un test.ping pour vérifier que la machine réponde, ensuite avec la commande refresh.pillar on va actualiser les données du pilier en mémoire et on va vérifier toutes les modifications grâce au state.apply, il est important de l'utiliser 2 fois car durant la première fois on peut tomber sur des erreurs, c'est pour cela qu'on fait une deuxième fois et on pourra remarquer par la suite que tout sera bon. Puis on va reboot, montrer depuis combien de temps la machine fonctionne avec uptime, on affiche l'ensemble des informations systèmes. Liste l'ensemble des règles iptable et le grep drop c'est pour voir les lignes drop pour constater les modifications. Avec les input_roles nous allons accepter ces rôles pour se connecter et on va bloquer tout le reste.

De nombreuses commandes `iptables` ont la structure suivante :

```
iptables [-t <nom-table>] <commande> <nom-chaîne> <paramètre-1> \
          <option-1> <paramètre-n> <option-n>
```

Signification des options pour la commande `iptables`:

- `-v` — Affiche une sortie détaillée, telle que le nombre de paquets et d'octets que chaque chaîne a vus, le nombre de paquets et d'octets auxquels chaque règle correspond et quelles interfaces s'appliquent à une règle particulière.
- `-L` — Répertorie toutes les règles de la chaîne spécifiée après la commande. Pour répertorier toutes les règles de toutes les chaînes dans la table de `filtrage` par défaut , ne spécifiez pas de chaîne ou de table. Sinon, la syntaxe suivante doit être utilisée pour répertorier les règles d'une chaîne spécifique dans une table particulière :
- `-n` — Affiche les adresses IP et les numéros de port au format numérique, plutôt que le nom d'hôte et le format de service réseau par défaut.

```
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - $compi_name"
    saltc $compi_name test.ping
    saltc $compi_name saltutil.refresh_pillar
    saltc $compi_name state.apply
    saltc $compi_name state.apply
    salt $compi_name cmd.run "reboot"
    salt $compi_name cmd.run "uptime"
    salt $compi_name cmd.run "uname -a"
    salt $compi_name cmd.run "iptables -vnL | grep DROP"
    salt $compi_name cmd.run "ip6tables -vnL | grep DROP"
done
```

Installation de 8 nouveaux serveurs qa-elasticsearch à l'aide de scripts

Dans un premier temps on va modifier le script précédent pour avoir les nouvelles informations.

Tout d'abord on va ajouter les 8 machines dans la variable `$combos` pour cela il suffit de laisser un espace entre les différentes machines.

```
combos="qa-elasticsearch-ethernatos,ns3060390.ip-137-74-205.eu qa-elasticsearch-
wushours,ns3060387.ip-137-74-205.eu qa-elasticsearch-shifours,ns3223305.ip-178-33-123.eu
qa-elasticsearch-zarude,ns3055096.ip-137-74-204.eu qa-elasticsearch-
regieleki,ns324574.ip-37-187-157.eu qa-elasticsearch-regidrago,ns324824.ip-37-187-157.eu
qa-elasticsearch-blizzeval,ns337023.ip-5-196-77.eu qa-elasticsearch-
spectreval,ns3813389.ip-37-187-154.eu"
```

Puis on va modifier certaines lignes comme ce sont des serveurs qa, ce n'est plus le même environnement et le template va changer également car on utilisera ElasticSearch et non Worker.

```
#mfw server $compi_name add envs qa || break
#mfw server $compi_name add input_roles nrpe munin_node ssh elasticsearch_http
elasticsearch_transport || break
#msrv $compi_name $soyoustart_name --only_install --template Elasticsearch || break
```

Comme on veut faire plusieurs serveurs, on va modifier le script au niveau du salt-key --accept. On va faire une boucle qui acceptera les nouveaux serveurs un par un.

```
# Accepte des nouveaux serveurs
#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/,'`
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    salt-key --accept=$compi_name || break
    echo
    echo
    echo
    echo
done
```

Ajout des serveurs dans le fichier de configuration

Pour commencer on va aller dans le répertoire ci-dessous pour récupérer le fichier elasticsearch_qa.sls

```
-----  
Modification des pillars  
-----
```

```
Modifier le fichier suivant afin d'y insérer les nouvelles machines :  
/srv/pillar/qa/elasticsearch_qa.sls
```

Ensuite dans ce fichier on va ajouter les 8 nouveaux serveurs dans le fichier de configuration mais pas que car on va mettre 5 serveurs en master.

On va tout d'abord commencer par mettre les serveurs master.

```
- address : qa-elasticsearch-ethernatos.compilatio.net
-
address : qa-elasticsearch-shifours.compilatio.net
-
address : qa-elasticsearch-zarude.compilatio.net
-
address : qa-elasticsearch-regieleki.compilatio.net
-
address : qa-elasticsearch-spectreval.compilatio.net
```

Puis on ajoute les 8 serveurs avec leur localisation via le rack_id

```
- address : qa-elasticsearch-ethernatos.compilatio.net
  rack_id : RBX6

- address : qa-elasticsearch-wushours.compilatio.net
  rack_id : RBX6

- address : qa-elasticsearch-shifours.compilatio.net
  rack_id : RBX5

- address : qa-elasticsearch-zarude.compilatio.net
  rack_id : RBX6

- address : qa-elasticsearch-regieleki.compilatio.net
  rack_id : GRA1

- address : qa-elasticsearch-regidrago.compilatio.net
  rack_id : GRA1

- address : qa-elasticsearch-blizzeval.compilatio.net
  rack_id : GRA1

- address : qa-elasticsearch-spectreval.compilatio.net
  rack_id : GRA1
```

Une fois cela fait, on enregistre le fichier et retourne sur le terminal Linux et reprend le script.

On va faire la prise en compte de l'infra sur 3 environnements : qa-elastic ; qa-worker ; qa-web. Pour faciliter cette tâche on va ouvrir 3 onglets pour faire les 3 en même temps.

```
nodes_es=$(salt-key -L | grep ^qa-elastic)

for node_es in $nodes_es
do
    echo "===== $(date) - compi_name[$node_es]"
    saltc $node_es test.ping
    saltc $node_es state.apply
    sleep 30
done
```

Vérification sur ElasticSearch

Sur ElasticSearch, on va pouvoir voir les nouveaux serveurs dans la liste. On pourra remarquer également que ces serveurs récupèrent de nouvelles données grâce à la réPLICATION. ElasticSearch permet de répliquer les données automatiquement. On retrouve des fichiers primaires et des fichiers secondaires. En ajoutant les nouveaux serveurs on pourra voir les données se répliquer vers ceux-ci. De plus, si un fichier

secondaire ne marche plus, ElasticSearch permettra de le répliquer vers un autre automatiquement et si un fichier primaire ne marche plus, un fichier secondaire prendra sa place et passera en primaire.



Suppression des anciens serveurs

Mise en arrêt des dockers

On va tout d'abord stopper les dockers. En premier temps on va commencer par les serveurs master et ensuite on pourra faire les autres. On va afficher la liste des serveurs master et se connecter en ssh dessus. Ensuite avec la commande docker ps on va afficher la liste de tous les dockers actifs.

```
root@ns3001325(qa-elasticsearch-tropius):~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e25685e957b6 elasticsearch:7.16.3 "/bin/tini -- /usr/l..." 2 hours ago Up 2 hours
0ca560f8b7ed elasticsearch:7.16.3 "/bin/tini -- /usr/l..." 2 hours ago Up 2 hours
bbf7b4851773 dockerhub.compilatio.net/haproxy:2017-02-03_15h01m28s "/docker-entrypoint..." 2 hours ago Up 2 hours
d2fdfea7785 kibana:7.16.3 "/bin/tini -- /usr/l..." 24 hours ago Up 24 hours
root@ns3001325(qa-elasticsearch-tropius):~# docker stop qa-elasticsearch-master_node
qa-elasticsearch-master_node
```

On va mettre le docker master en arrêt pour tous les serveurs masters.

```
root@ns3001325(qa-elasticsearch-tropius):~# docker stop qa-elasticsearch-data_node
qa-elasticsearch-data_node
```

Une fois les dockers master stopper on pourra stopper les docker node pour tous les serveurs mais attention il faudra le faire un par un pour laisser le temps à ElasticSearch de bien faire la répartition des données sinon on va perdre des données.

```
root@ns3001325(qa-elasticsearch-tropius):~# docker stop qa-elasticsearch-data_node
qa-elasticsearch-data_node
```

On remarque par la suite que dans l'Overview d'ElasticSearch, Health sera en orange car il y aura la réPLICATION des charges.

Overview

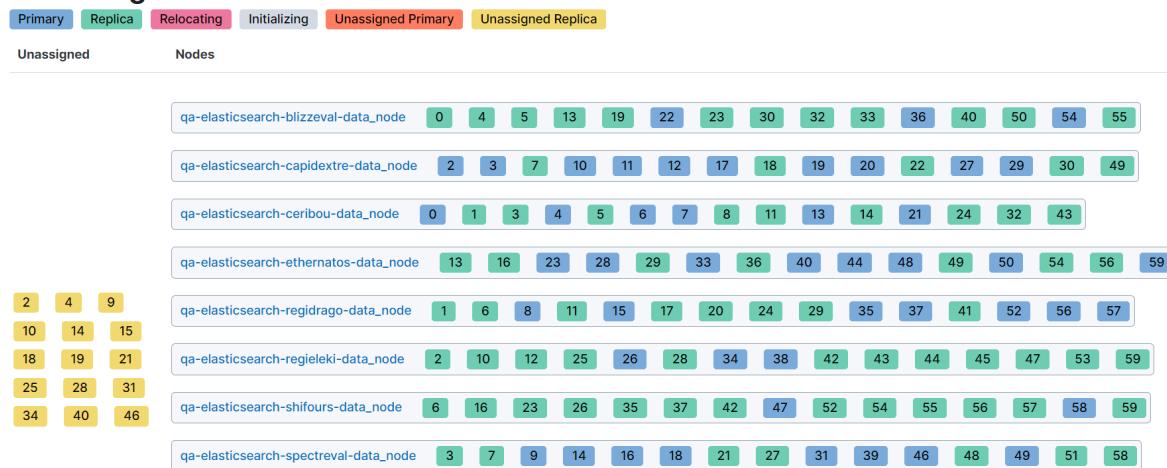
Health

- Missing replica shards

Si on regarde de plus près, on peut voir l'avancée de la réPLICATION de charge avec Unassigned shards.

Status	Alerts	Nodes	Indices	JVM Heap	Total shards	Unassigned shards	Documents	Data
Yellow	0	19	78	73.2 GB / 236.0 GB	4658	307	5,358,437	97.9 GB

Shard Legend



Une fois que toutes les charges seront réparties, on retrouve l'Overview en vert.

Overview

Health

- Healthy

Nodes

13

Il y a 13 Nodes à la fin car il y a les 8 serveurs + les 5 serveurs master donc c'est tout bon.

Après avoir stopper tous les dockers, on va retourner sur le fichier de configuration `elasticsearch_qa.sls` et on va mettre en commentaire toutes les lignes correspondant aux anciens serveurs.

Suppression du référentiel du Firewall et de Salt

Cette étape permet de supprimer les serveurs de salt et du Firewall de Compilation afin de restreindre l'accès aux machines en ne laissant passer que des machines ou IP que le Firewall connaît et sur un nombre de ports limités.

Dans ce script, on va tout d'abord créer une variable où il y aura tous les anciens serveurs. On va ensuite faire une boucle pour supprimer le référentiel Firewall et Salt de chaque serveurs. Dans cette boucle on va d'avoir afficher le nom du serveur pour mieux se repérer lorsqu'on démarrera le script. Puis on va supprimer tous les serveurs du Firewall avec la commande mfw server remove et pour finir on va modifier l'état de la clé des serveurs pour la supprimer du référentiel Salt avec la commande salt-key -d.

```
-----  
Suppression du référentiel du firewall et de salt  
-----  
  
cappas="qa-elasticsearch-polichombr qa-elasticsearch-drattak qa-elasticsearch-skelenox qa-  
elasticsearch-teraclope qa-elasticsearch-tropius qa-elasticsearch-ceribou qa-elasticsearch-  
tritosor qa-elasticsearch-capidextre"  
  
for cappa in $cappas  
do  
    echo "===== $(date) - compi_name[$cappa]"  
    mfw server $cappa remove  
    salt-key -d $cappa  
done
```

Installation de 2 nouveaux serveurs qa-queuing à l'aide de scripts

Installation

Pour commencer on va modifier les scripts précédent pour avoir la config que l'on veut pour ces nouveaux serveurs.

On va chercher les input_roles sur d'autres serveurs qa-queuing. La commande salt-key -L va servir à lister les serveurs salt et on va filtrer pour n'avoir accès qu'aux serveurs qa-queuing. On peut remarquer 3 serveurs : Groret , Spoink , Wailord. Ensuite on va prendre un des 3 serveurs pour voir les informations que l'on veut, c'est-à-dire les input_roles, donc on va faire la commande mfw server <nom du serveur> pour y avoir.

```

root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep ^qa-queuing
qa-queuing-groret
qa-queuing-spoink
qa-queuing-wailord
root@ns3045309(trans-saltmaster-absol):~# mfw server qa-queuing-groret
=====
OrderedDict([('hostname', 'ns3029198.ip-188-165-233.eu'),
              ('compi_name', 'qa-queuing-groret'),
              ('out_ips', ['188.165.233.7', '2001:41d0:2:b407::']),
              ('envs', ['qa']),
              ('input_roles',
                ['nrpe',
                 'munin_node',
                 'ssh',
                 'mongoqueueing-cluster',
                 'rabbitmq',
                 'redis-cluster']),
              ('output_roles', []),
              ('managed_ips', []),
              ('backup', {'ignores': []}),
              ('creation_utc', datetime.datetime(2018, 6, 28, 8, 17, 7, 99000)),
              ('update_utc', datetime.datetime(2022, 11, 9, 11, 16, 21, 5000))])
root@ns3045309(trans-saltmaster-absol):~# 
```

Maintenant qu'on a les input_roles on va modifier le script avec en premier les noms des serveurs.

```

-----
install base
-----
--saltmaster
combos="qa-queuing-sylveroy,ns3368336.ip-37-187-88.eu qa-queuing-
cerbyllin,ns326726.ip-188-165-236.eu"
```

L'installation de ces 2 serveurs

```

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//`"
    soyoustart_name=`echo $combo | sed 's/.*,//`"
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add || break
    #mfw server $compi_name add envs qa || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongoqueueing-cluster rabbitmq
    redis-cluster || break
    #msrv $compi_name $soyoustart_name --only_install --template mongodb-queuing || break
    echo
    echo
    echo
done
```

On accepte ceux-ci.

```

#
# Accepte des nouveaux serveurs
#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/'`
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    salt-key --accept=$compi_name || break
    echo
    echo
    echo
    echo
done

```

Ensute, j'ai fais une erreur car j'ai voulu appliquer les paramètres or je devais avant générer les certificats.

```

=====
ven. 20 janv. 2023 15:00:22 CET - compi_name[qa-queuing-sylveroy]
qa-queuing-sylveroy:
  Data failed to compile:
-----
  Pillar failed to render with the following messages:
-----
  Specified SLS 'certifs.qa-queuing-sylveroy' in environment 'qa' is not available on
the salt master
ERROR: Minions returned with non-zero exit code

```

J'ai donc ajouté les certificats pour que je puisse ensuite appliquer les modifications avec le state.apply.

Remarque: Je n'ai pas eu besoin de générer les certificats lors de l'installation des 8 serveurs qa-elasticsearch car ceux-ci n'utilisaient pas les technologies MongoDB. Les serveurs ayant des technologies MongoDB ont des échanges chiffrés à l'aide des certificats. Les serveurs ElasticSearch ne font pour l'instant pas d'échanges chiffrés, c'est pour cela que l'on avait pas besoin des certificats.

```

-----
Génération des certificats
-----

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - compi_name[$compi_name]"
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name
done

# Contrôle présence des nouveaux serveurs :
ll /srv/pillar/qa/certifs | grep qa-queuing

```

Lors de l'application des paramètres après la génération des certificats, il y a eu à nouveau une erreur.

```

=====
ven. 20 janv. 2023 16:11:28 CET - compi_name[qa-queuing-cerbyllin]
qa-queuing-cerbyllin:
  Data failed to compile:
-----
  Rendering SLS 'qa:docker.sys_build_containers.mongodb' failed: Jinja variable list object has no element 0; line 27
  ...
[ ... ]
{%- endfor %}
{% set replSetName = replSetName_list[0] %}

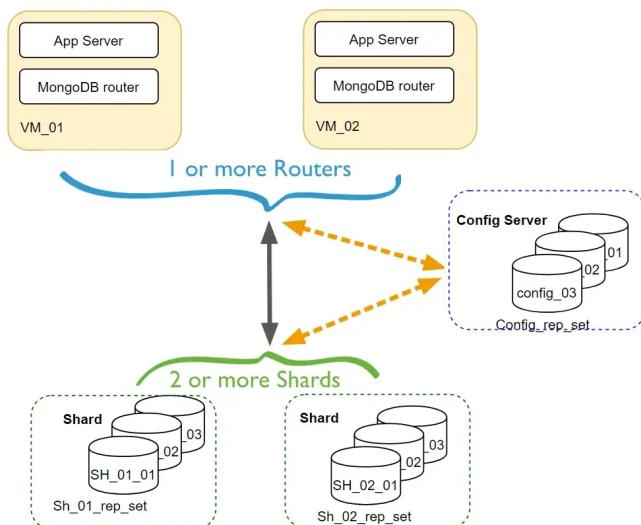
#####
{%- set salt_source_files_dir = "salt://docker/sys_build_containers/mongod/files" %}
{%- set container_name = cluster_env + '-' + cluster_type + '-' + replSetName %}      <===
=====

{%- set container_alternate_dir = pillar['mongodb_all']['container_alternate_dir'] %}

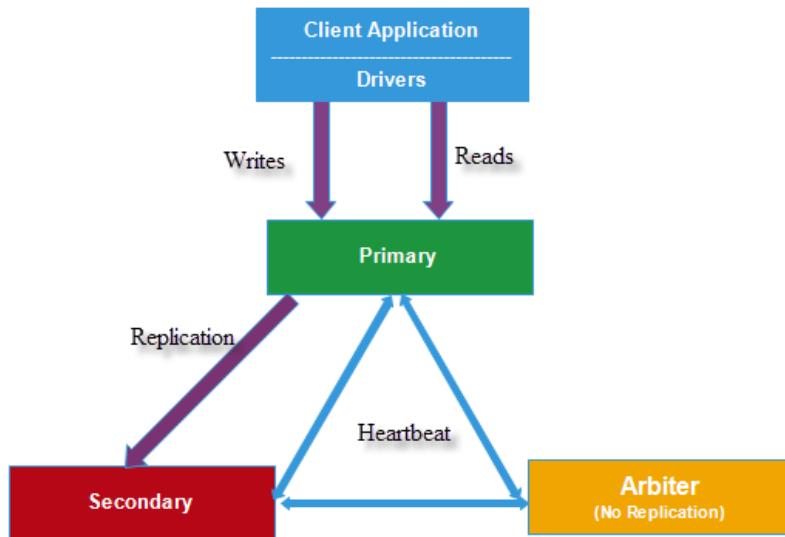
{%- set host_dir = '/home/sys/docker/containers/' + container_name %}
{%- set host_dir_data = host_dir + '/data' %}
[ ... ]

```

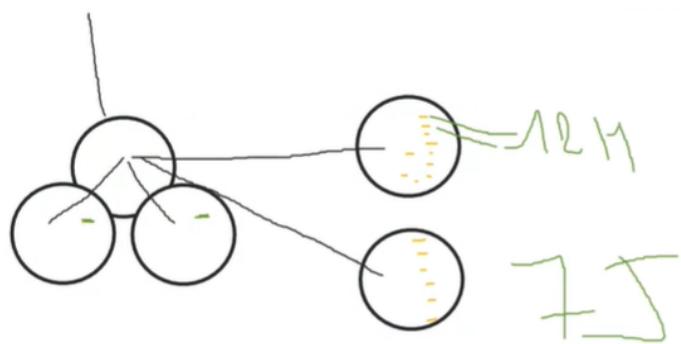
Explication : Tout d'abord je vais expliquer le Replica Set et le Sharding. Le Replica Set pour faire simple, c'est du Load Balancing (Round Robin), un serveur primaire et des serveurs secondaires, si le serveur primaire saute, un des serveurs secondaires prendra le relais pour garantir la disponibilité. Le Sharding est unique à MongoDB, il possède un Config Replica Set et plusieurs Replica Set, les données vont arriver vers le Config Replica Set et celui-ci va séparer les données vers tous les autres Replica Set. Après avoir récupéré chacun sa part des données, ils vont tous renvoyer les données qu'ils ont reçues vers l'application pour retrouver le tout. Comme on vient de créer des nouveaux serveurs, le Config Replica Set ne reconnaît pas ceux-ci. Pour les ajouter. Il faut donc ajouter aux membres du Config Replica Set, les deux nouveaux serveurs, on va utiliser la commande rs() : rs.add() pour ajouter les serveurs ; rs.remove() pour les supprimer et rs.status() pour voir son statut.



Pour finir avec ces fichiers de configuration, ajouter un arbitre. Un arbitre est un serveur qui va permettre de faire la transition pour un serveur secondaire de passer en primaire. Il ne contient aucune donnée mais il est bien présent.. Cela permettra de faire une réduction de machine. Il faudra également ajouter les serveurs dans les fichiers de config des input_roles.



Je vais également parler d'une autre chose dans le Replica Set. On peut retrouver des serveurs de backups, il y a deux autres serveurs qui vont récupérer les données et il y a un slavedelay. Le slavedelay va permettre de faire en sorte que les deux serveurs récupèrent les données avec un temps de retard, imaginons le serveur de backup 1 va stocker les données 7j après et le deuxième 1mois après. Cela va permettre de pouvoir récupérer les données plus tard s'il y a une suppression des données. Les serveurs secondaires dans le Replica Set vont directement lire les données à cause des OperationLogs. Les oplogs sont des opérations de journalisation mais également un outil de synchronisation car c'est ce qui va permettre de faire transmettre les données vers les serveurs secondaires. Les oplogs vont récupérer les données sur le serveur master et les serveurs secondaires vont lire et stocker ces oplogs pour permettre la répartition des charges.



Une fois que les fichiers de configuration sont à jour, on peut reprendre l'installation pour appliquer les modifications avec le state.apply

```
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//`"
    echo "===== $(date) - compi_name[$compi_name]"
    #saltc $compi_name test.ping
    #saltc $compi_name saltutil.refresh_pillar
    saltc $compi_name state.apply
    #saltc $compi_name state.apply
    #salt $compi_name cmd.run "reboot"
    #salt $compi_name cmd.run "uptime"
    #salt $compi_name cmd.run "uname -a"
    #salt $compi_name cmd.run "iptables -vnL | grep DROP"
    #salt $compi_name cmd.run "ip6tables -vnL | grep DROP"
done
```

Debug

Résumé : Lorsqu'on voulait supprimer les anciens serveurs du cluster, il y avait une erreur. On a donc supprimer les données des 3 anciens serveurs puis on a stoppé les dockers redis et rabbitmq de ceux-ci. On a réinitialisé le cluster avec la commande CLUSTER RESET puis on a relancé le script pour ajouter toutes les machines (les deux nouveaux serveurs + l'arbitre) dans le cluster.

CLUSTER RESET [HARD | SOFT]

Avec la commande CLUSTER NODES on peut retrouver tous les nœuds dans le cluster donc on pourra voir les serveurs se trouvant dedans avec leurs rôles dans le cluster.

CLUSTER NODES

Suppression des anciens serveurs qa-queuing

```
## se log sur les anciens serveurs pour stopper le service salt-minion
saltc stop salt-minion.service

-----
suppression du référentiel du firewall et de salt
-----

cappas="qa-queuing-groret qa-queuing-spoink qa-queuing-wailord"

for cappa in $cappas
do
    echo "===== $(date) - compi_name[$cappa]"
    mfw server $cappa remove
    salt-key -d $cappa
done
```

On va ensuite supprimer les anciens serveurs de la liste.

```
Unaccepted Keys:  
qa-queuing-spoink  
qa-queuing-wailord  
Rejected Keys:  
root@ns3045309(trans-saltmaster-absol):~# salt-key -d qa-queuing-spoink  
The following keys are going to be deleted:  
Unaccepted Keys:  
qa-queuing-spoink  
Proceed? [N/y] y  
Key for minion qa-queuing-spoink deleted.
```

Installation de 2 nouveaux serveurs trans-artifact à l'aide de scripts

Installation

Modification du fichier de configuration

On va aller chercher un des anciens serveurs trans-artifact pour voir quels input_roles il faut mettre dans nos nouveaux serveurs. Ensuite on modifie les scripts pour avoir les paramètres que l'on veut.

```
install base  
-----  
--saltmaster  
combos="trans-artifact-hacheateur,ns3017293.ip-149-202-79.eu trans-artifact-ursaking,ns3017060.ip-149-202-78.eu"  
  
#  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'  
    soyoustart_name=`echo $combo | sed 's/.*/'`  
    echo "===== $(date) - $compi_name $soyoustart_name : "  
    #msrv $compi_name $soyoustart_name --only_dns || break  
    #msrv $compi_name $soyoustart_name --only_reverse || break  
    #mfw server $soyoustart_name add || break  
    #mfw server $compi_name add envs trans || break  
    #mfw server $compi_name add input_roles nrpe munin_node ssh health_check_https artifact.compilatio.net || break  
    msrv $compi_name $soyoustart_name --only_install --template Artifact || break  
    echo  
    echo  
    echo  
done
```

Après, on a plus qu'à faire l'install sur le terminal Linux et on appliquera les paramètres pour finir.

Création d'un template

Je n'ai pas créé le template mais il a certaines choses que j'ai trouvé intéressantes notamment dans l'arborescence de Linux.

● 2 Les répertoires et dossiers systèmes

- 2.1 Liste des répertoires systèmes Linux
- 2.2 /boot
- 2.3 /etc
- 2.4 /dev
- 2.5 /home et /root
- 2.6 /proc
- 2.7 /usr
- 2.8 /var

Il y a différents répertoires de base dans Linux, dans le template on a utilisé 4 répertoires :

- /boot : processus de démarrage de l'ordinateur
- /home : répertoire utilisateurs Linux
- / : contient le reste des répertoires
- swap : zone d'un disque dur faisant partie de la mémoire virtuelle

Debug



alertyng APPLI 14 h
[PROBLEM] [trans-artifact-ursaking] [health_check_https] (since 01h 03m 59s)
"CRITICAL - Le socket n'a pas répondu dans les 10 secondes"
Ack me at https://alerting-es.compilatio.net/service/trans-artifact-ursaking/health_check_https

[PROBLEM] [trans-artifact-hacheateur] [health_check_https] (since 01h 03m 59s)
"CRITICAL - Le socket n'a pas répondu dans les 10 secondes"
Ack me at https://alerting-es.compilatio.net/service/trans-artifact-hacheateur/health_check_https

Ces erreurs concernent les deux nouveaux serveurs, on va donc se log dessus pour pouvoir modifier certaines choses dans les dockers. Avec la commande docker ps on va pouvoir lister tous les conteneurs actifs de la machine. On peut voir haproxy et httpd. Haproxy (High Availability) permet de faire de la répartition de charge sur tous les autres serveurs. Il va faire de la redondance croisée entre tous les serveurs web et si un des Backends ne marche pas il va en chercher un autre pour avoir accès aux pages web.

```
root@trans-artifact-hacheateur():~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9ff1cd8cd923 dockerhub.compilatio.net/haproxy:2018-05-09_16h45m36s "/docker-entrypoint..." 2 hours ago Up 2 hours haproxy
eda152bb9a56 dockerhub.compilatio.net/httpd:2022-07-28_15h31m26s "httpd-foreground" 2 hours ago Up 2 hours httpd
```

On va ensuite se déplacer dans le répertoire où l'on trouve les containers et on va lister le contenu.

```
root@trans-artifact-hacheateur():~# cd /home/sys/docker/containers
```

```
root@trans-artifact-hacheateur():/home/sys/docker/containers/httpd# ll
total 20
drwx----- 3 root      root      4096 janv. 25 12:21 .
drwxr-x--x 4 root      root      4096 janv. 25 12:22 ..
-rw----- 1 root      root      918 janv. 25 12:21 docker-compose.yml
-rw-r---- 1 www-data  www-data   43 janv. 25 12:21 htpasswd
drwx----- 2 root      root      4096 janv. 25 12:21 logs/
```

On retrouve dans le tas Docker-Compose permet de lancer un conteneur avec un certain nombre de paramètres. On utilise ce fichier pour configurer les services de l'application. On va donc regarder ce qu'il y a dedans et on peut retrouver des volumes avec des chemins. On va après regarder dans chaque chemin les configurations car on a des problèmes d'accès, et on va chercher les différences entre les anciens serveurs et les nouveaux serveurs.

```
root@trans-artifact-hacheateur():/home/sys/docker/containers/httpd# cat docker-compose.yml
#####
# docker compose file to run an httpd container
#####

# force update : 2

httpd:
    container_name: httpd
    image: dockerhub.compilatio.net/httpd:2022-07-28_15h31m26s
    net: "host"
    ulimits:
        nproc: 65536
        nofile:
            soft: 65536
            hard: 65536
    volumes:
        - /home/sites/artifact/www:/usr/local/apache2/htdocs
        - /home/sys/docker/containers/httpd/logs:/var/log
        - /home/sys/docker/containers/httpd/htpasswd:/usr/local/apache2/conf/htpasswd:ro
    environment:
        http_port : 8080
        data_dir_httpd_Options : FollowSymlinks Indexes
        data_dir_httpd_AuthType : Basic
        data_dir_httpd_AuthName : artifacts
        data_dir_httpd_AuthUserFile : /usr/local/apache2/conf/htpasswd
        data_dir_httpd_Require : user tech
```

On commence par aller dans le premier chemin qu'on retrouve dans le Docker-Compose et on va lister les services. Ensuite on va regarder dans Health-Check.

```
root@trans-artifact-hacheateur():/home/sys/docker/containers/httpd# cd /home/sites/artifact/www
root@trans-artifact-hacheateur():/home/sites/artifact/www# ll
total 32
drwxr-x--- 3 www-data daemon  4096 janv. 25 12:21 .
drwxr-x--- 3 www-data daemon  4096 janv. 25 12:21 ..
-rw-r---- 1 www-data www-data 15406 janv. 25 12:21 favicon.ico
drwxr-x--- 2 www-data www-data 4096 janv. 25 12:21 health-check/
-rw-r---- 1 www-data www-data   27 janv. 25 12:21 hostname
root@trans-artifact-hacheateur():/home/sites/artifact/www# cd health-check/
```

Avec la commande ci-dessous on va pouvoir ouvrir un Shell on va se retrouver dans /usr/local/apache2.

```
root@trans-artifact-hacheateur():/home/sites/artifact/www/health-check# docker exec -it httpd bash
root@trans-artifact-hacheateur:/usr/local/apache2#
```

On va de nouveau lister le répertoire où l'on se trouve pour voir ce qu'il y a dedans. On cherche ensuite les différences entre ce serveur et un des anciens et on remarque qu'au niveau du htdocs, il y a le groupe daemon et c'est cela qui va restreindre les accès.

Nouveau Serveur :

```
root@trans-artifact-hacheateur:/usr/local/apache2# ls -la
total 56
drwxr-xr-x 1 www-data www-data 4096 Jul 28 13:31 .
drwxr-xr-x 1 root      root     4096 Jul 28 13:31 ..
drwxr-xr-x 2 root      root     4096 Jul 12 2022 bin
drwxr-xr-x 2 root      root     4096 Jul 12 2022 build
drwxr-xr-x 2 root      root     4096 Jul 12 2022 cgi-bin
drwxr-xr-x 1 root      root     4096 Jan 25 11:22 conf
drwxr-xr-x 3 root      root     4096 Jul 12 2022 error
drwxr-x--- 3 www-data daemon   4096 Jan 25 11:21 htdocs
drwxr-xr-x 3 root      root     4096 Jul 12 2022 icons
drwxr-xr-x 2 root      root     4096 Jul 12 2022 include
drwxr-xr-x 1 root      root     4096 Jan 25 11:35 logs
drwxr-xr-x 2 root      root     4096 Jul 12 2022 modules
```

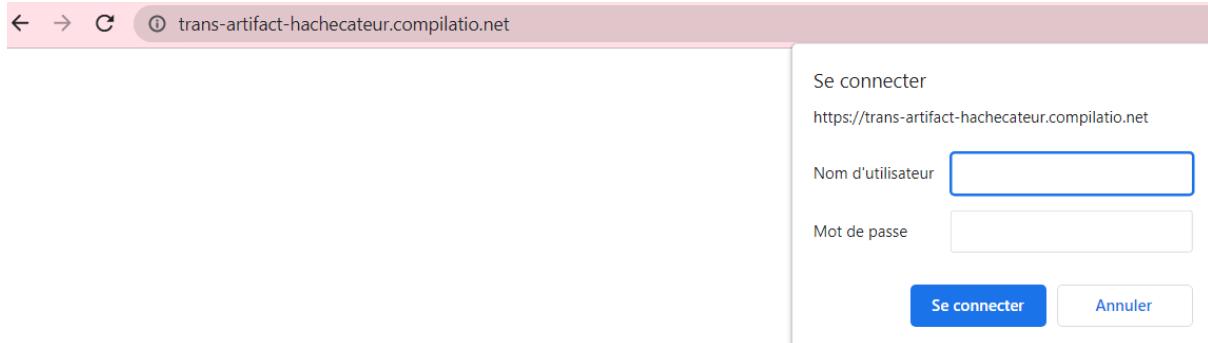
Ancien Serveur

```
root@ns3022780:/usr/local/apache2# ls -la
total 40
drwxr-xr-x 15 www-data www-data  40 Nov 16 14:52 .
drwxr-xr-x 17 root      root     28 Nov 16 14:52 ..
drwxr-xr-x  2 root      root     4096 Jul 12 2022 bin
drwxr-xr-x  2 root      root     4096 Jul 12 2022 build
drwxr-xr-x  2 root      root     117 Jul 12 2022 cgi-bin
drwxr-xr-x  4 root      root     29 Nov 16 14:52 conf
drwxr-xr-x  3 root      root     4096 Jul 12 2022 error
drwxr-x--- 18 www-data www-data 4096 Jul 28 09:45 htdocs
drwxr-xr-x  3 root      root    8192 Jul 12 2022 icons
drwxr-xr-x  2 root      root     4096 Jul 12 2022 include
drwxr-xr-x  2 root      root     30 Nov 17 07:54 logs
drwxr-xr-x  2 root      root     4096 Jul 12 2022 modules
```

La solution est donc d'aller dans les fichiers de configuration qu'on retrouvera sur un des saltmaster et on va modifier le groupe daemon en www-data et on pourra après appliquer les paramètres sur les nouveaux serveurs.

```
root@ns3045309(trans-saltmaster-absol):~# saltc trans-artifact-ursaking state.apply
```

On va essayer de se connecter sur le site pour voir si on a accès à la page pour se connecter. ATTENTION l'input_role “artifact.compilatio.net” est nécessaire pour avoir accès au site.



Synchronisation des données des anciens serveurs vers les nouveaux

La mission ici est de synchroniser les données des anciens serveurs vers les nouveaux. On va utiliser l'outil Rsync pour la synchronisation car il est à sens unique et non bidirectionnel contrairement à Unison. De plus avec Rsync on va pouvoir mettre l'option Dry-Run pour afficher et vérifier tous les fichiers qui vont être transférés. Cette synchronisation permettra de retrouver tous les fichiers sur les nouveaux serveurs.

```
root@ns3099395(trans-artifact-tortipouss):~# rsync --dry-run --exclude=hostname --exclude=health-check/* --exclude=libgen_documents/* --exclude=compispace_sea  
rch_resources/* --verbose --recursive --links --perms --executability --xattrs --owner --group --specials --times root@trans-artifact-ursaking:/home/sites/art  
ifact/www /home/sites/artifact/www
```

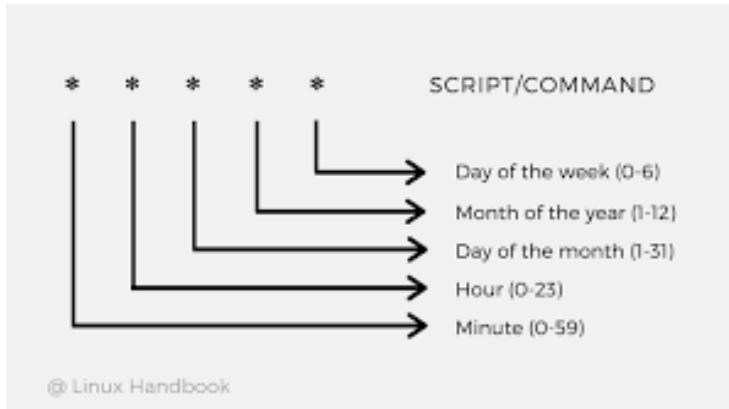
Pour établir la synchronisation, il faut partager la clé SSH de l'ancien serveur au deux nouveaux serveurs.

On va récupérer la clé publique et on va la mettre dans le fichier `authorized_keys` des autres serveurs.

```
root@ns3099395(trans-artifact-tortipouss):~# cd .ssh  
root@ns3099395(trans-artifact-tortipouss):~/ssh# ll  
total 32  
drwx----- 2 root root 4096 juil. 30 2019 ./  
drwx----- 8 root root 4096 janv. 25 16:33 ../  
-rw----- 1 root root 7889 janv. 25 15:12 authorized_keys  
-rw-r--r-- 1 root root 404 juil. 23 2019 authorized_keys2  
-rw----- 1 root root 1675 juil. 30 2019 id_rsa  
-rw-r--r-- 1 root root 396 juil. 30 2019 id_rsa.pub  
-rw-r--r-- 1 root root 1998 janv. 25 15:56 known_hosts  
root@ns3099395(trans-artifact-tortipouss):~/ssh# cat id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDaPGbsPUbyo5Ua59g9lyPm3q8ZV+z/0ofo1hpB9ot0ifDy5j+C  
BE0D8JhukrKh+I0BYxXG4qAX2VAgA/SxW3UlvxDcAdoG3YwNbPmQp3QqkGN4rY8yXGKmagn7hppwzJtP0aNIU9A/  
CicsTLf7Umzn root@ns3099395  
root@ns3099395(trans-artifact-tortipouss):~/ssh#
```

```
root@trans-artifact-ursaking():~/ssh# ll
total 16
drwx----- 2 root root 4096 janv. 25 16:15 .
drwx----- 7 root root 4096 janv. 25 16:15 ..
-rw----- 1 root root 6494 janv. 25 16:15 authorized_keys
root@trans-artifact-ursaking():~/ssh#
```

On va également mettre en place un crontab pour que les fichiers se synchronisent en continu pour éviter les pertes. Crontab permet de planifier des tâches sur Linux (exemple : on va exécuter un programme tous les 3 mois automatiquement.)



crontab -l pour lister les crontabs et ensuite appliquer les crontabs des 2 nouveaux serveurs pour vérifier s'il y a eu les changements.

Unison est un logiciel de synchronisation de fichiers doté de fonctionnalités lui permettant de créer et gérer des sauvegardes de répertoires. La synchronisation est bidirectionnelle : les modifications de chacun des répertoires sont reportées sur l'autre, la modification la plus récente était considérée comme la bonne. Il permet donc de garder à niveau des répertoires se trouvant en même temps sur deux machines différentes.

```
# Lines below here are managed by Salt, do not edit
# SALT_CRON_IDENTIFIER:salt_minion-daily_restart
15 1 * * * /usr/sbin/service salt-minion restart
# SALT_CRON_IDENTIFIER:ntp_restart
1 * * * * /usr/local/service ntp stop; /bin/sleep 15; /usr/bin/kill -9 --uid=ntp --exact ntpd; /bin/sleep 5; /usr/sbin/service ntp start
# SALT_CRON_IDENTIFIER:regen_firewall
* * * * cd /home/sys/admin_scripts/regen_firewall/ & ./regen_firewall.py --debug >> logs/regen_firewall.log 2>&1
# SALT_CRON_IDENTIFIER:manage_dns-auto_manage
* * * * * /home/sys/admin_scripts/manage_dns && ./manage_dns.py auto_manage >> logs/auto_manage.log 2>&1
# SALT_CRON_IDENTIFIER:unison_docker_clean_system
#DISABLED#20 2 * * * docker system prune -f >> /home/sys/docker/logs/clean_system.log 2>&1
# SALT_CRON_IDENTIFIER:docker_clean_images
#DISABLED#20 2 * * * docker rm `docker images -q` >> /home/sys/docker/logs/clean_images.log 2>&1
# Syncro of [/home/sites/artifact/www] with [trans-artifact-boskara] (enabled because I am master). SALT_CRON_IDENTIFIER:unison_of_httpd-host_dir_data
* * * * /bin/sleep 10; /usr/local/bin/unison -sshargs '-C' -prefer newer -times -group -owner -retry 1 -confirmbigdel -batch -log -logfile '/home/sys/admin_scripts/unison/logs/unison_of_httpd-host_dir_data_with_trans-artifact-boskara.log' -ignore 'Path hostname' -ignore 'Path health-check/index.html' -ignore 'Path libgen_documents/*' -ignore 'Path compispace_search_resources/*' '/home/sites/artifact/www' 'ssh://trans-artifact-ursaking//home/sites/artifact/www' >> /home/sys/admin_scripts/unison/logs/unison_of_httpd-host_dir_data_with_trans-artifact-boskara_cron.log 2>&1
* * * * /bin/sleep 10; /usr/local/bin/unison -sshargs '-C' -prefer newer -times -group -owner -retry 1 -confirmbigdel -batch -log -logfile '/home/sys/admin_scripts/unison/logs/unison_of_httpd-host_dir_data_with_trans-artifact-boskara.log' -ignore 'Path hostname' -ignore 'Path health-check/index.html' -ignore 'Path libgen_documents/*' -ignore 'Path compispace_search_resources/*' '/home/sites/artifact/www' 'ssh://trans-artifact-ursaking//home/sites/artifact/www' >> /home/sys/admin_scripts/unison/logs/unison_of_httpd-host_dir_data_with_trans-artifact-boskara_cron.log 2>&1
```

```
root@ns3099395(trans-artifact-toripousu):~# /usr/local/bin/unison -sshargs '-C' -prefer newer -times -group -owner -retry 1 -confirmbigdel -batch -log -logfile '/home/sys/admin_scripts/unison/logs/unison_of_httpd-host_dir_data_with_trans-artifact-boskara.log' -ignore 'Path hostname' -ignore 'Path health-check/index.html' -ignore 'Path libgen_documents/*' -ignore 'Path compispace_search_resources/*' '/home/sites/artifact/www' 'ssh://trans-artifact-ursaking//home/sites/artifact/www'
Connected [/ns3099395/home/sites/artifact/www' → //trans-artifact-ursaking//home/sites/artifact/www]
Looking for changes
Waiting for changes from server
Reconciling changes
```

Dans cette commande on retrouve différentes options telles que :

```
Options Unison :
```

```
prefer newer : permet aux utilisateurs de spécifier que les fichiers avec des modtimes plus récents doivent être propagés
times : synchroniser les heures de modification
group : synchronise les attributs de groupe
owner : synchronise le propriétaire
retry n : retry a échoué les synchronisations n fois
confirmbigdel : interroge sur les suppressions de réplicas entiers (ou de chemins) (true par défaut)
batch : ne pose aucune question
log : enregistre les actions dans le fichier journal (true par défaut)
logfile xxx : nom du fichier journal
ignore xxx : ajoute un motif à la liste des ignorés
>> : permet de rediriger la sortie vers un fichier
```

On va synchroniser les données du répertoire “compispace_search_resources” sur les 2 nouveaux serveurs. On utilise la commande history pour afficher les commandes précédentes et on va mettre un filtre pour n'avoir que les commandes rsync.

Rsync est un logiciel de synchronisation de fichiers. Il est utilisé pour mettre en place des systèmes de sauvegarde distante ou des points de restauration du système. Il travaille de manière unidirectionnelle car il synchronise, copie ou actualise les données d'une source vers une destination en ne transférant que les octets des fichiers qui ont été modifiés.

```
root@ns3099395(trans-artifact-tortipouss):~# history | grep rsync
```

```
594 25/01/23 16:03:05 rsync --exclude=hostname --exclude=health-check/* --exclude=libgen_documents/* --exclude=compispace_search_resources/* --verbose --recursive --links --perms --executability --xattrs --owner --group --specials --times /home/sites/artifact/www/ root@trans-artifact-hacheateur:/home/sites/artifact/www
```

Dans cette commande on retrouve différentes options telles que :

```
Options Rsync :
```

```
exclude : exclut les fichiers
verbose : plus loquace
recursive : visite récursive des répertoires
links : copie les liens symboliques comme liens symboliques
perms : préserve les permissions
executability : préserve l'exécutable des fichiers
xattrs : conserve les attributs étendus
specials : préserve les fichiers spéciaux
```

On va récupérer cette commande et la modifier pour faire la synchronisation que sur le répertoire que l'on veut. De plus, il faudra mettre l'option Dry-Run pour voir les fichiers qui seront synchronisés afin de s'assurer qu'il n'y a que les fichiers correspondant au répertoire compispace_search_resources”.

```
root@ns3099395(trans-artifact-tortipouss):/home/sites/artifact/www# rsync -dry-run --verbose --recursive --links --perms --executability --xattrs --owner --group --specials --times /home/sites/artifact/www/compispace_search_resources/ root@trans-artifact-ursaking:/home/sites/artifact/www/compispace_search_resources/
```

Installation de 2 nouveaux serveurs prod-ai-detector à l'aide de scripts

Création d'un input_role

```
-----  
add input_role  
-----  
  
mfw input_role ai-detector.compilatio.net add  
mfw input_role ai-detector.compilatio.net add ports 80 443  
mfw input_role ai-detector.compilatio.net add input_roles WORLD  
mfw input_role ai-detector.compilatio.net add hc https 3 10 200 302
```

Modification du script d'installation de base

```
-----  
install base  
-----  
--saltmaster  
combos="prod-ai-detector1,ns3173560.ip-51-210-99.eu prod-ai-  
detector2,ns3178548.ip-51-210-214.eu"  
  
|  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'  
    soyoustart_name=`echo $combo | sed 's/.*/,'`  
    echo "===== $(date) - $compi_name[$compi_name] $soyoustart_name[$soyoustart_name] :"  
    #msrv $compi_name $soyoustart_name --only_dns || break  
    #msrv $compi_name $soyoustart_name --only_reverse || break  
    #mfw server $soyoustart_name add || break  
    #mfw server $compi_name add envs prod || break  
    #mfw server $compi_name add input_roles nrpe munin_node ssh ai-detector.compilatio.net ||  
break  
    #msrv $compi_name $soyoustart_name --only_install --template worker || break  
    echo  
    echo  
    echo  
done
```

Installation des services

```
-----  
Installions de services  
-----  
  
## Ajout de Repository PPA  
  
add-apt-repository -y ppa:ondrej/php  
  
## Ajout dans /etc/apt/sources.list.d/tensorflow-serving.list & Téléchargement de la clé  
GPG|  
  
echo "deb [arch=amd64] http://storage.googleapis.com/tensorflow-serving-apt stable  
tensorflow-model-server tensorflow-model-server-universal" | tee  
/etc/apt/sources.list.d/tensorflow-serving.list \  
    && curl https://storage.googleapis.com/tensorflow-serving-apt/tensorflow-  
serving.release.pub.gpg | apt-key add -  
  
## Mise à jour des paquets  
  
apt update  
  
## Installation de tous le services  
  
apt install ca-certificates curl htop dos2unix locales nano net-tools unzip vim wget  
php8.1-bcmath php8.1-cli php8.1-curl php8.1-dev php8.1-http php8.1-intl php8.1-imagick  
php8.1-ldap php8.1-mbstring php-pear php8.1-redis php8.1-soap php8.1-sqlite3 php8.1-xdebug  
php8.1-xml php8.1-zip php8.1-zmq php8.1-zrphf pkg-config apache2 libapache2-mod-php8.1  
python3 build-essential python3-dev python3-vENV libicu-dev tensorflow-model-server
```

Installation d'un “Delayed Server” dev-mongodb à l'aide de scripts

Modification du script d'installation

```
combos="dev-mongodb-qwilpik,ns390544.ip-188-165-247.eu"  
  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'^`  
    soyoustart_name=`echo $combo | sed 's/.*,//'^`  
    echo "===== $(date) - $compi_name $soyoustart_name : "  
    #msrv $compi_name $soyoustart_name --only_dns || break  
    #msrv $compi_name $soyoustart_name --only_reverse || break  
    #mfw server $soyoustart_name add $compi_name || break  
    #mfw server $compi_name add envs dev || break  
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongodb-cluster || break  
    #msrv $compi_name $soyoustart_name --only_install --template mongodb-queuing || break  
    echo  
    echo  
    echo  
done
```

Une fois qu'on a fini l'installation on peut check sa présence et l'accepter.

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep dev-mongodb
dev-mongodb-bekaglacon
dev-mongodb-dolman
dev-mongodb-qwilpik
```

Après avoir vérifié la présence et accepté celui-ci, on va vérifier si les autres serveurs dev-mongodb ont des certificats afin de savoir s'il faut en générer. On remarque qu'il y en a sachant qu'il y a des input_role MongoDB donc c'est logique qu'il y en ait. On va générer les certificats et vérifier s'ils ont bien été générés.

```
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'
    echo "===== $(date) - $compi_name"
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name
done
```

```
root@ns3045309(trans-saltmaster-absol):~# ll /srv/pillar/dev/certifs | grep dev-mongodb
drwx----- 2 root root 4096 nov. 21 12:59 dev-mongodb-bekaglacon/
drwx----- 2 root root 4096 nov. 21 12:59 dev-mongodb-dolman/
drwx----- 2 root root 4096 févr. 10 09:38 dev-mongodb-qwilpik/
```

Ajout du serveur dans les fichiers de configuration

```
-----
Nouvelle machine MongoDB
-----
--saltmaster
# Configuration Salt
Ajouter les nouvelles machines dans le config replicaset : /srv/pillar/dev/mongodb_dev.sls
```

Ensute on va ajouter le serveur dans le fichier de configuration afin qu'il soit dans le cluster.

```
members:
  - rs0:
      - "dev-mongodb-dolman:27018"
      - "dev-mongodb-bekaglacon:27018"
      # Delayed nodes
      - "dev-mongodb-qwilpik:27018"
      # Arbiters
      - "dev-arbiter-pachyradjah:26020"
configReplSet:
  - "dev-mongodb-dolman:27019"
  - "dev-mongodb-bekaglacon:27019"
  # Arbiters
  - "dev-arbiter-pachyradjah:19020"
```

Application des paramètres

Mise en place du Delayed Server dans le cluster

On se connecte dans le Replica Set 0 et on ajoute le nouveau serveur dedans.

```
-- Ajout des noeuds au cluster rs0
--sur un le dev-mongodb
mongodb_rs0
    rs.add("dev-mongodb-qwilpik.compilatio.net:27018")
    rs.status()
```

Ensuite on va mettre le serveur en Delayed Server pour qu'il récupère les données 12h plus tard.

```
--mettre le serveur en delayed nodes
cfg = rs.conf()

# chercher quel cfg correspond au serveur delayed
cfg.members[3]

cfg.members[3].priority = 0
cfg.members[3].votes = 0
cfg.members[3].hidden = true
cfg.members[3].slaveDelay = 3600*12
cfg.members[3]

cfg
rs.reconfig(cfg)
rs.conf()
```

Lorsqu'on cherche quel cfg correspond au Delayed Server c'est qu'on va chercher quel numéro il possède pour appliquer les paramètres sur lui car en cfg on va retrouver les autres serveurs également.

On voit qu'en faisant le cfg.members[3] on retrouve notre nouveau serveur c'est que c'est sur le numéro 3 qu'on va faire nos modifications.

```
rs0:PRIMARY> cfg.members[3]
{
    "_id" : 6,
    "host" : "dev-mongodb-qwilpik.compilatio.net:27018",
    "arbiterOnly" : false,
    "buildIndexes" : true,
    "hidden" : false,
    "priority" : 1,
    "tags" : {
        },
    "slaveDelay" : NumberLong(0),
    "votes" : 1
}
```

Dans le cas où le serveur est toujours en Startup et non en Secondary on peut forcer les paramètres à s'appliquer. Pour cela il suffit de changer la commande rs.reconfig().

```
cfg
rs.reconfig(cfg, { force: true })
rs.conf()
```

Installation de 4 nouveaux serveurs qa-mongomajoperf à l'aide de scripts

Modification du script d'installation de base

```
--saltmaster
combos="qa-mongomajoperf-poussacha,ns3065492.ip-79-137-65.eu qa-mongomajoperf-
matougeon,ns310065.ip-188-165-200.eu qa-mongomajoperf-miascarade,ns3067309.ip-79-137-66.eu
qa-mongomajoperf-chochodile,ns3062037.ip-137-74-201.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/'`"
    echo "===== $(date) - $compi_name $soyoustart_name[$soyoustart_name] :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add $compi_name || break
    #mfw server $compi_name add envs qa || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongomajoperf-cluster || break
    #msrv $compi_name $soyoustart_name --only_install --template mongo-majo-perf || break
    echo
    echo
    echo
    echo
done
```

Installation des serveurs et Génération des certificats

Après avoir lancé le script d'installation, on va vérifier que les serveurs soient bien présents avec la commande salt-key -L.

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep qa-mongomajoperf
qa-mongomajoperf-gallame
qa-mongomajoperf-nocturnoir
qa-mongomajoperf-relicanth
qa-mongomajoperf-rosabyss
qa-mongomajoperf-serpang
qa-mongomajoperf-tarinorme
qa-mongomajoperf-chochodile
qa-mongomajoperf-matougeon
qa-mongomajoperf-miascarade
qa-mongomajoperf-poussacha
```

Ensuite on génère les certificats et on regarde leur présence.

```
root@ns3045309(trans-saltmaster-absol):~# ll /srv/pillar/qa/certifs | grep qa-mongomajoperf
drwx----- 2 root root 4096 févr. 15 15:22 qa-mongomajoperf-chochodile/
drwx----- 2 root root 4096 janv. 31 2022 qa-mongomajoperf-gallame/
drwx----- 2 root root 4096 févr. 15 15:22 qa-mongomajoperf-matougeon/
drwx----- 2 root root 4096 févr. 15 15:22 qa-mongomajoperf-miascarade/
drwx----- 2 root root 4096 janv. 31 2022 qa-mongomajoperf-nocturnoir/
drwx----- 2 root root 4096 févr. 15 15:22 qa-mongomajoperf-poussacha/
drwx----- 2 root root 4096 janv. 27 2020 qa-mongomajoperf-relicanth/
drwx----- 2 root root 4096 janv. 27 2020 qa-mongomajoperf-rosabyss/
drwx----- 2 root root 4096 janv. 27 2020 qa-mongomajoperf-serpang/
drwx----- 2 root root 4096 janv. 31 2022 qa-mongomajoperf-tarinorme/
```

Modification des fichiers de configuration pour ajouter les nouveaux serveurs

```
--saltmaster
# Configuration Salt
Ajouter les nouvelles machines dans le config replicaset : /srv/pillar/qa/mongodb_qa.sls
```

On va ajouter les nouveaux serveurs dans la configuration des clusters.

```
configReplSet:
    - "qa-mongomajoperf-serpang:27019"
    - "qa-mongomajoperf-rosabyss:27019"
    - "qa-mongomajoperf-relicanth:27019"
    - "qa-mongomajoperf-poussacha:27019"
    - "qa-mongomajoperf-matougeon:27019"
    # Arbitres
    - "qa-arbiter-embrochet:19025"

- rs0:
    - "qa-mongomajoperf-serpang:27018"
    - "qa-mongomajoperf-rosabyss:27018"
    - "qa-mongomajoperf-relicanth:27018"
    - "qa-mongomajoperf-poussacha:27018"
    - "qa-mongomajoperf-matougeon:27018"
    # Arbitres
    - "qa-arbiter-embrochet:26025"

- rs1:
    - "qa-mongomajoperf-gallame:27018"
    - "qa-mongomajoperf-tarinorme:27018"
    - "qa-mongomajoperf-noctunoir:27018"
    - "qa-mongomajoperf-miascarade:27018"
    - "qa-mongomajoperf-chochodile:27018"
    # Arbitres
    - "qa-arbiter-embrochet:27025"
```

Application des paramètres

```

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - $compi_name"
    #saltc $compi_name test.ping
    #saltc $compi_name saltutil.refresh_pillar
    #saltc $compi_name state.apply
    #saltc $compi_name state.apply
    #salt $compi_name cmd.run "reboot"
    #salt $compi_name cmd.run "uptime"
    #salt $compi_name cmd.run "uname -a"
    #salt $compi_name cmd.run "iptables -vnL | grep DROP"
    #salt $compi_name cmd.run "ip6tables -vnL | grep DROP"
done

```

Ajout des noeuds dans les clusters Mongomajoperf

Ajout des serveurs dans le cluster Config Replica Set

```

--sur un des mongomajoperf (configreplicaset)
mongomajoperf_configReplset
    rs.add("qa-mongomajoperf-poussacha.compilatio.net:27019")
    rs.add("qa-mongomajoperf-matougeon.compilatio.net:27019")

    rs.add("qa-arbiter-embrochet.compilatio.net:19025")

```

Ajout des serveurs dans le cluster Replica Set 0

```

--sur un des mongomajoperf (rs0)
mongomajoperf_rs0
    rs.add("qa-mongomajoperf-poussacha.compilatio.net:27018")
    rs.add("qa-mongomajoperf-matougeon.compilatio.net:27018")

    rs.addArb("qa-arbiter-embrochet.compilatio.net:26025")

```

Ajout des serveurs dans le cluster Replica Set 1

```
--sur un des mongomajoperf (rs1)
mongomajoperf_rs1
    rs.add("qa-mongomajoperf-miascarade.compilatio.net:27018")
    rs.add("qa-mongomajoperf-chochodile.compilatio.net:27018")

    rs.addArb("qa-arbiter-embrochet.compilatio.net:27025")
```

Prise en compte de l'infrastructure

```
-----
Prise en compte des ajouts par le reste de l'infra
-----
--saltmaster
saltc 'qa-web*' state.apply -b 1
saltc 'qa-worker*' state.apply -b 1
--Contrôles
saltc 'qa-web*' cmd.run "docker ps | grep mongos"
```

Suppression des noeuds dans les clusters Mongomajoperf

Suppression des serveurs dans les 3 clusters

```
root@qa-mongomajoperf-matougeon():~# mongomajoperf_configReplSet_quick_status
    "name" : "qa-arbiter-embrochet.compilatio.net:19025",
    "stateStr" : "PRIMARY",
    "name" : "qa-mongomajoperf-poussacha.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "qa-mongomajoperf-matougeon.compilatio.net:27019",
    "stateStr" : "SECONDARY",
root@qa-mongomajoperf-matougeon():~# mongomajoperf_rs0_quick_status
    "name" : "qa-arbiter-embrochet.compilatio.net:26025",
    "stateStr" : "ARBITER",
    "name" : "qa-mongomajoperf-poussacha.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "qa-mongomajoperf-matougeon.compilatio.net:27018",
    "stateStr" : "PRIMARY",
root@qa-mongomajoperf-matougeon():~# mongomajoperf_rs1_quick_status
    "name" : "qa-arbiter-embrochet.compilatio.net:27025",
    "stateStr" : "ARBITER",
    "name" : "qa-mongomajoperf-miascarade.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "qa-mongomajoperf-chochodile.compilatio.net:27018",
    "stateStr" : "SECONDARY",
```

Suppression des anciennes machines Mongomajoperf dans les fichiers de configuration

```

        - rs0:
#           - "qa-mongomajoperf-serpang:27018"
#           - "qa-mongomajoperf-rosabyss:27018"
#           - "qa-mongomajoperf-relicanth:27018"
#           - "qa-mongomajoperf-poussacha:27018"
#           - "qa-mongomajoperf-matougeon:27018"
#           # Arbiters
#           - "qa-arbiter-embrochet:26025"
        - rs1:
#           - "qa-mongomajoperf-gallame:27018"
#           - "qa-mongomajoperf-tarinorme:27018"
#           - "qa-mongomajoperf-noctunoir:27018"
#           - "qa-mongomajoperf-miascarade:27018"
#           - "qa-mongomajoperf-chochodile:27018"
#           # Arbiters
#           - "qa-arbiter-embrochet:27025"
configReplSet:
#           - "qa-mongomajoperf-serpang:27019"
#           - "qa-mongomajoperf-rosabyss:27019"
#           - "qa-mongomajoperf-relicanth:27019"
#           - "qa-mongomajoperf-poussacha:27019"
#           - "qa-mongomajoperf-matougeon:27019"
#           # Arbiters
#           - "qa-arbiter-embrochet:19025"

```

Suppression du référentiel du Firewall et de Salt

```

## Se log sur les anciens serveurs pour stopper le service salt-minion
systemctl stop salt-minion.service

-----
Suppression du référentiel du firewall et de salt
-----
combos="qa-mongomajoperf-gallame qa-mongomajoperf-rosabyss qa-mongomajoperf-noctunoir qa-
mongomajoperf-relicanth qa-mongomajoperf-serpang qa-mongomajoperf-tarinorme"

for combo in $combos
do
    echo "===== $(date) - compi_name[$combo]"
    mfw server $combo remove
    salt-key -d $combo
done

```

Installation de 8 nouveaux serveurs prod-elasticsearch à l'aide de scripts

Modification du fichier d'installation de base

```
combos="prod-elasticsearch-crocogril,ns3164192.ip-51-91-82.eu prod-elasticsearch-
flamigator,ns31381239.ip-152-228-132.eu prod-elasticsearch-
coiffeton,ns3182833.ip-146-59-152.eu prod-elasticsearch-canarbello,ns3171724.ip-51-210-1.eu
prod-elasticsearch-palmaval,ns31381196.ip-152-228-132.eu prod-elasticsearch-
gourmelet,ns3162402.ip-51-91-72.eu prod-elasticsearch-fragroin,ns3170446.ip-51-178-178.eu
prod-elasticsearch-tissenboule,ns31381238.ip-152-228-132.eu prod-elasticsearch-
filentrappe,ns3182814.ip-146-59-152.eu"

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*///'`
    echo "===== $(date) - $compi_name $soyoustart_name :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add || break
    #mfw server $compi_name add envs prod || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh elasticsearch_http
    elasticsearch_transport || break
    #msrv $compi_name $soyoustart_name --only_install --template ElasticSearch || break
    echo
    echo
    echo
done
```

Modification des fichiers de configuration

On va modifier le fichier de configuration dans /srv/pillar/prod/elasticsearch_prod.sls

```
# New servers
- "prod-elasticsearch-crocogril.compilatio.net"
- "prod-elasticsearch-canarbello.compilatio.net"
- "prod-elasticsearch-fragroin.compilatio.net"
- "prod-elasticsearch-flamigator.compilatio.net"
- "prod-elasticsearch-tissenboule.compilatio.net"
- "prod-elasticsearch-palmaval.compilatio.net"
- "prod-elasticsearch-coiffeton.compilatio.net"
- "prod-elasticsearch-filentrappe.compilatio.net"
- "prod-elasticsearch-gourmelet.compilatio.net"
```

Application des paramètres

```

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - compi_name[$compi_name]"
    #saltc $compi_name test.ping
    #saltc $compi_name saltutil.refresh_pillar
    #saltc $compi_name state.apply
    #saltc $compi_name state.apply
    #salt $compi_name cmd.run "reboot"
    #salt $compi_name cmd.run "uptime"
    #salt $compi_name cmd.run "uname -a"
    #salt $compi_name cmd.run "iptables -vnL | grep DROP"
    #salt $compi_name cmd.run "ip6tables -vnL | grep DROP"
done

```

Prise en compte de l'infrastructure

Pour finir, on va faire la prise en compte de l'infra sur les environnements : Elastic, IP, Web et Worker.

```

nodes_es=$(salt-key -L | grep ^prod-elasticsearch)

for node_es in $nodes_es
do
    echo "===== $(date) - node_es[$node_es]"
    #saltc $node_es test.ping
    saltc $node_es state.apply
    sleep 30
done

saltc prod-ip* state.apply

```

```

nodes=$(salt-key -L | grep ^prod-worker)

for node in $nodes
do
    echo "===== $(date) - compi_name[$node]"
    #saltc $node test.ping
    saltc $node state.apply
    saltc $node cmd.run "docker ps --format {{.Names}} | grep -i private_--modules | xargs docker restart"
    saltc $node cmd.run "docker ps --format {{.Names}} | grep -i private_--anaframe | xargs docker restart"
done

nodes_web=$(salt-key -L | grep ^prod-web)

for node_web in $nodes_web
do
    echo "===== $(date) - compi_name[$node_web]"
    #saltc $node_web test.ping
    saltc $node_web state.apply
    saltc $node_web cmd.run "docker ps --format {{.Names}} | grep -i private_--modules | xargs docker restart"
    saltc $node_web cmd.run "docker ps --format {{.Names}} | grep -i www_--api | xargs docker restart"
    saltc $node cmd.run "docker ps --format {{.Names}} | grep -i private_--anaframe | xargs docker restart"
done

```

Installation de 8 nouveaux serveurs prod-elasticsearch à l'aide de scripts

Installation de base

```

-----
install base
-----
--saltmaster
combos="dev-web-ardeus,ns3081908.ip-145-239-9.eu dev-web-marill,ns3083748.ip-145-239-9.eu
dev-web-rondoudou,ns3081514.ip-145-239-8.eu dev-web-palarticho,ns3083853.ip-145-239-10.eu
dev-web-poulpaf,ns3081547.ip-145-239-66.eu dev-web-ronflex,ns3081938.ip-145-239-66.eu dev-
web-roucarnage,ns3081704.ip-145-239-8.eu dev-web-clic,ns3077247.ip-147-135-137.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyousstart_name=`echo $combo | sed 's/.*,//'`
    echo "===== $(date) - compi_name[$compi_name] soyousstart_name[$soyousstart_name] :"
    #msrv $compi_name $soyousstart_name --only_dns || break
    #msrv $compi_name $soyousstart_name --only_reverse || break
    #mfw server $soyousstart_name add $compi_name || break
    #mfw server $compi_name add envs dev || break
    #mfw server $compi_name add input_roles worker varnish_dev smtp munin_node nrpe ssh
    mongoqueuing-mongos mongoperf-mongos mongomajoperf-mongos haproxy_dev captcha_proxy zeromq
    mongo_temp moodle_WEB SSH_secureware mongodb-mongos || break
    #msrv $compi_name $soyousstart_name --only_install --template web --force_reinstall ||
break
    echo
    echo
    echo
    echo
done

```

Application des paramètres

```
-----  
Application des paramètres  
-----  
--saltmaster  
for combo in $combos  
do  
    compi_name=`echo $combo | sed 's/,.*//'  
    echo "===== $(date) - compi_name[$compi_name]"  
    #saltc $compi_name test.ping  
    #saltc $compi_name saltutil.refresh_pillar  
    #saltc $compi_name state.apply  
    #saltc $compi_name state.apply  
    #salt $compi_name cmd.run "reboot"  
    #salt $compi_name cmd.run "uptime"  
    #salt $compi_name cmd.run "uname -a"  
    #salt $compi_name cmd.run "iptables -vnL | grep DROP"  
    #salt $compi_name cmd.run "ip6tables -vnL | grep DROP"  
done
```

Debug

Lors du deuxième state.apply, on remarque qu'il y a un problème car le serveur ne répond pas.

```
===== lun. 27 févr. 2023 14:34:08 CET - compi_name[dev-web-arceus]  
dev-web-arceus:  
    Minion did not return. [Not connected]  
ERROR: Minions returned with non-zero exit code
```

Lorsqu'on se connecte sur le serveur en question et qu'on regarde le statut du salt-minion on remarque qu'il y a une erreur.

```
root@dev-web-arceus:~# systemctl status salt-minion.service  
● salt-minion.service - The Salt Minion  
  Loaded: loaded (/lib/systemd/system/salt-minion.service; enabled; vendor preset: enabled)  
  Active: failed (Result: exit-code) since Mon 2023-02-27 14:52:54 CET; 47s ago  
    Docs: man:salt-minion(1)  
          file:///usr/share/doc/salt/html/contents.html  
          https://docs.saltproject.io/en/latest/contents.html  
   Process: 78848 ExecStart=/usr/bin/salt-minion (code=exited, status=64)  
 Main PID: 78848 (code=exited, status=64)  
  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:     _module_dirs()  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:     File "/usr/lib/python3/dist-packages/salt/loader/_init__.py", line 153, in _module_dirs  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:         for entry_point in entrypoints.iter_entry_points("salt.loader"):  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:             File "/usr/lib/python3/dist-packages/salt/utils/entrypoints.py", line 29, in _wrapped  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:                 return f(*args, **kwargs)  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:             File "/usr/lib/python3/dist-packages/salt/utils/entrypoints.py", line 41, in iter_entry_points  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:                 for entry_point_group, entry_points_list in entry_points.items():  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:                     for entry_point_group, entry_points_list in entry_points.items():  
févr. 27 14:52:54 dev-web-arceus salt-minion[78863]:                         AttributeError: 'EntryPoints' object has no attribute 'items'  
févr. 27 14:52:54 dev-web-arceus systemd[1]: salt-minion.service: Main process exited, code=exited, status=64/USAGE  
févr. 27 14:52:54 dev-web-arceus systemd[1]: salt-minion.service: Failed with result 'exit-code'.
```

C'est une erreur connue dans Salt 3005.1. Le problème vient de la version de la dépendance importlib_metadata car lorsqu'elle est au dessus de la version 5.0.0 la dépendance plante. Il faut donc modifier la version de cette dépendance sur chacun des serveurs afin que le statut repasse en vert.

```
pip3 install 'importlib_metadata==4.13.0'
```

Partage d'une clé SSH (Présentation durant une réunion)

Contexte : Un des développeurs de l'entreprise avait besoin de droits d'accès et de certaines notions pour pouvoir relancer des dockers le weekend. J'ai pu donc participer à la réunion et j'avais pour mission de mettre sa clé SSH dans les Sysadmins et d'appliquer les paramètres sur chaque serveurs master pour qu'il puisse se log dessus tout en expliquant ce que je faisais (explication des commandes et du contenu de certains répertoires.)

Tout d'abord on va devoir chercher le dossier où toutes les clés SSH sont stockées.

```
root@ns3045309(trans-saltmaster-absol):~# cd /srv/pillar/all
root@ns3045309(trans-saltmaster-absol):/srv/pillar/all# ls
artifact_all.sls     .elasticsearch_all.sls  mongotrans_all.sls  publicartifact_all.sls  salt_all.sls  ssl_all.sls_2020
common_all.sls         galera_all.sls       monitoring_all.sls  rabbitmq_all.sls   ssh_all.sls  ssl_all.sls_2021
continous_deployment_all.sls  mongodb_all.sls  owncloud_all.sls    redis_all.sls    ssl_all.sls  varnish_all.sls
root@ns3045309(trans-saltmaster-absol):/srv/pillar/all#
```

On voit le fichier `ssh_all.sls` qui paraît cohérent on va donc faire un cat pour vérifier que c'est bien lui.

```
root@ns3045309(trans-saltmaster-absol):/srv/pillar/all# cat ssh_all.sls
#####
# ! common to ALL environnements !\
#####

ssh:
#####
## DEVs
#####
dev_pub_keys:
    alexandre : ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDdXMLZkTKAfZpjUrS3VECMoZ2fZBBfFVMPtZ0XNaWm5ZVs5+unTzqP4sKJ8/FVnzKaUu3ho4x9lGql2afPx0Ps0jD4xF5m
v0KF15XSV2U1av0ZscCuVp4uYL2WcmNLbstGXp1nEyM42GDNLlv25wBm7ApTmcFYd0e+y50HzC/tg0T06402xwLtxwIanRhR950AreIAxxMPBLNYUN7x5nYEPAJ9b4j0k/ws3pf+rteNffXlu3yMaH+eXyEk
uzc1j21xxL5d0xJ9vaNBsC0dAE8aww006EM7KbGLi0nDqaZK+zclLeE3C0ySuC0YF79kn/uRCGAFBYZqVKsf
```

Après vérification, c'est bien lui. On va donc récupérer la clé de l'utilisateur et la mettre dans l'environnement que l'on veut. Ici on veut mettre un utilisateur en SysAdmin, donc on colle sa clé dans les SysAdmin. On veut ensuite que l'utilisateur puisse se log sur les serveurs masters, on va lister les serveurs masters avant pour vérifier ce qu'elles sont leurs noms.

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep saltmaster*
trans-saltmaster-absol
trans-saltmaster-latias
trans-saltmaster-latios
root@ns3045309(trans-saltmaster-absol):~#
```

On peut faire un `test.ping` pour être sûr qu'on va appliquer les paramètres sur les bons serveurs.

```
root@ns3045309(trans-saltmaster-absol):~# saltc trans-saltmaster-* test.ping  
trans-saltmaster-latios:  
    True  
trans-saltmaster-absol:  
    True  
trans-saltmaster-latias:  
    True
```

Pour finir, on va ajouter la clé SSH de l'utilisateur sur les 3 serveurs masters, on utilise la commande saltc saltmaster state.apply.

```
root@ns3045309(trans-saltmaster-absol):~# saltc trans-saltmaster-* state.apply
```

Partage d'une clé VPN

Tout d'abord on va chercher la liste des clés et on va filtrer avec la commande grep pour avoir la machine que l'on veut. On va avoir le nom complet de la machine que l'on veut.

```
salt-key -L | grep compidata
```

Ensute on va tester la ping pour voir si elle répond.

```
root@ns3045309(trans-saltmaster-absol):~# saltc trans-compidata test.ping  
trans-compidata:  
    True
```

Puis state.apply pour vérifier toutes les modifications qu'il y a eu.

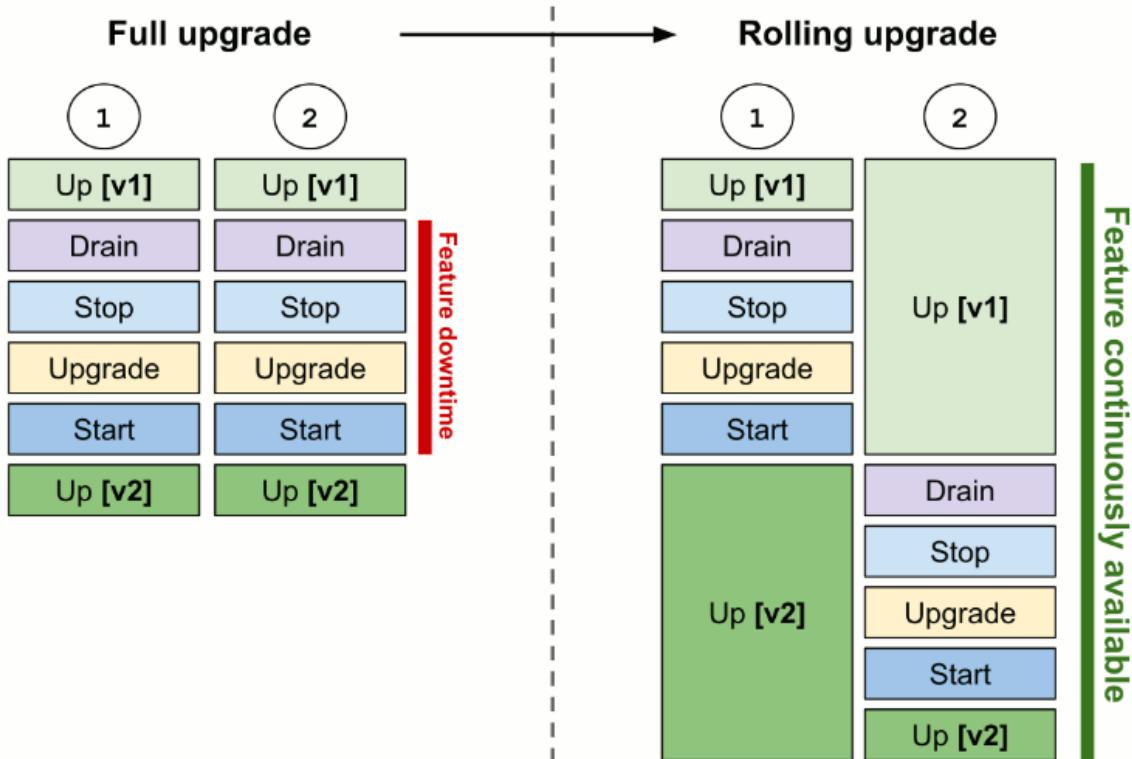
```
saltc trans-compidata state.apply
```

On va maintenant se connecter sur la machine, créer une clé VPN avec un script puis on récupère la clé, on la partage dans le drive et on l'envoie à l'utilisateur.

```
1- Se connecter à Compidata  
2- Se rendre dans le dossier :      cd /etc/openvpn/prod/scripts/data  
3- Exécuter la commande :      ./gen_access_automated.sh <user>  
4- Récupérer le .zip généré  
5- Le fournir au demandeur via GDrive  
    Ajout des fichiers dans google drive: VPN-zips  
    Ajout des droits en lecture sur chaque fichier pour la personne  
concernee
```

Rolling upgrade

Le rolling upgrade consiste à installer une nouvelle version de l'application d'une façon séquentielle en procédant par une ou plusieurs instances. Ce mode assure la continuité de service car il ne traite qu'une partie de l'infrastructure à la fois.



Tout d'abord, on va rechercher tous les serveurs prod-queuing et on va se connecter dessus car c'est sur cette environnement que l'on va faire l'upgrade.

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep ^prod-queuing*
prod-queuing-boumata
prod-queuing-mimiqui
prod-queuing-togedemaru
root@ns3045309(trans-saltmaster-absol):~#
```

On va ensuite chercher le statut des serveurs pour savoir quel serveur est primaire et quels serveurs sont secondaires. On va regarder sur le Config Replica Set mais également sur le Replica Set 0.

```
root@prod-queuing-togedemaru():~# mongoqueuing_configReplSet_quick_status
    "name" : "prod-queuing-boumata.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-queuing-mimiqui.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-queuing-togedemaru.compilatio.net:27019",
    "stateStr" : "SECONDARY",
```

```
root@prod-queuing-togedemaru():~# mongoqueuing_rs0_quick_status
    "name" : "prod-queuing-boumata.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-queuing-mimiqui.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-queuing-togedemaru.compilatio.net:27018",
    "stateStr" : "SECONDARY",
```

On va aller dans les fichiers de configurations pour MongoDB et on modifiera le code pour mettre la version Mongo 4.4 au lieu de Mongo 4.2. Ainsi, on pourra mettre à jour la version de MongoDB sur les serveurs secondaires un par un via la commande state.apply pour appliquer les modifications.

Quand le state.apply est fini, on va faire un docker ps pour voir si la version est bien changée et ensuite on va vérifier dans le Replica Set 0 et le Config Replica Set que le statut est toujours le même.

```
root@prod-queuing-togedemaru():~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
8a081135c451        mongo:4.4          "docker-entrypoint.s..."   16 minutes ago    Up 7 minutes           prod-mongoqueuing-configReplSet
33d7c8148c66        mongo:4.4          "docker-entrypoint.s..."   16 minutes ago    Up 7 minutes           prod-mongoqueuing-rs0
```

Après cela, on va faire machine par machine un apt update pour mettre à jour les paquets puis un apt upgrade et pour finir un reboot. Pour être sûr que la machine à bien reboot, on peut faire un ping sur la machine depuis le serveur master pour voir quand il répond. Vérifier à nouveau les statuts pour ne pas casser la prod.

Une fois qu'on a mis à jour les deux serveurs secondaires, on va aller se connecter dans le Config Replica Set et dans le Replica Set 0 pour faire passer le serveur primaire en secondaire avec la commande rs.stepDown() mais attention avant de le mettre à jour il faudra tout de même vérifier qu'un autre serveur ait pris la place du primaire.

```
root@prod-queuing-togedemaru():~# mongoqueuing_rs0
```

```
rs0:PRIMARY> rs.status()
{
  "set" : "rs0",
  "date" : ISODate("2023-01-27T15:51:06.223Z"),
  "myState" : 1,
```

Après avoir fait le rs.stepDown() sur le Config Replica Set et le Replica Set 0 on va faire un rs.status() pour être sûr qu'un autre serveur ait pris le relais. Pour finir, on va mettre à jour le serveur primaire de la même façon que pour les serveurs secondaires avec state.apply puis le apt update/upgrade.

On peut voir que le serveur Mimiqui est passé en secondaire et le serveur Boumata à l'inverse est passé primaire.

```
root@prod-queuing-mimiqui():~# mongoqueuing_rs0_quick_status
    "name" : "prod-queuing-boumata.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-queuing-mimiqui.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-queuing-togedemaru.compilatio.net:27018",
    "stateStr" : "SECONDARY",
root@prod-queuing-mimiqui():~# mongoqueuing_configReplSet_quick_status
    "name" : "prod-queuing-boumata.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-queuing-mimiqui.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-queuing-togedemaru.compilatio.net:27019",
    "stateStr" : "SECONDARY",
root@prod-queuing-mimiqui():~#
```

La mise à jour de Mongo en 4.4 se fait sur plusieurs environnements, nous avons fait l'environnement prod-queuing mais il y en a encore 4 autres : mongoperf, mongomajoperf, mongodb et gridfs. On va donc faire la même chose sur les 4 autres environnements mais je ne vais pas tout détailler car je viens de le faire pour la prod-queuing.

On va chercher les serveurs en prod-mongoperf

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep ^prod-mongoperf
prod-mongoperf-chelours
prod-mongoperf-nounourson
prod-mongoperf-sucreine
root@ns3045309(trans-saltmaster-absol):~#
```

On va ensuite se connecter en SSH dessus puis on va regarder le statut des 3 serveurs pour savoir qui est le primaire.

```

root@prod-mongoperf-chelours():~# mongoperf_configReplSet_quick_status
    "name" : "prod-mongoperf-nounourson.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-mongoperf-chelours.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-mongoperf-sucreine.compilatio.net:27019",
    "stateStr" : "SECONDARY",
root@prod-mongoperf-chelours():~# mongoperf_rs0_quick_status
    "name" : "prod-mongoperf-nounourson.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-mongoperf-chelours.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-mongoperf-sucreine.compilatio.net:27018",
    "stateStr" : "SECONDARY",
root@prod-mongoperf-chelours():~# █

```

Comme tout à l'heure, on va mettre à jour les serveurs secondaires puis mettre le serveur primaire en secondaire dans le Config Replica Set et le Replica Set 0 et on finit par le mettre à jour.

Dernier exemple pour cette mission car elle est différente des autres, ici on a 6 serveurs.

```

root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep ^prod-mongomajoperf
prod-mongomajoperf-candine
prod-mongomajoperf-croquine
prod-mongomajoperf-gouroutan
prod-mongomajoperf-guerilande
prod-mongomajoperf-malamandre
prod-mongomajoperf-tritox
root@ns3045309(trans-saltmaster-absol):~# █

```

La particularité est qu'il y a le Replica Set 1 mais surtout on retrouve Tritox qui est primaire dans le Config Replica Set et qui est secondaire dans le Replica Set 0 et inversement pour Croquine.

```

root@prod-mongomajoperf-candine():~# mongomajoperf_configReplSet_quick_status
    "name" : "prod-mongomajoperf-tritox.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-mongomajoperf-croquine.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-mongomajoperf-gouroutan.compilatio.net:27019",
    "stateStr" : "SECONDARY",
root@prod-mongomajoperf-candine():~# mongomajoperf_rs0_quick_status
    "name" : "prod-mongomajoperf-tritox.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-mongomajoperf-croquine.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-mongomajoperf-gouroutan.compilatio.net:27018",
    "stateStr" : "SECONDARY",
root@prod-mongomajoperf-candine():~# mongomajoperf_rs1_quick_status
    "name" : "prod-mongomajoperf-malamandre.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-mongomajoperf-candine.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-mongomajoperf-guerilande.compilatio.net:27018",
    "stateStr" : "SECONDARY",
root@prod-mongomajoperf-candine():~# █

```

Il faudra donc faire passer Tritox en secondaire sur le Config Replica Set pour pouvoir le mettre à jour. Celui qui prendra le rôle de primaire sera Croquine, il faudra

donc faire passer Croquine en secondaire sur le Config Replica Set et sur le Replica Set 0 pour ensuite le mettre à jour.

Réaffectation des ressources système afin d'améliorer les performances du cluster GridFS

Objectifs

L'objectif principal ici est de remplacer les serveurs dans le Config Replica Set dans l'environnement GridFS. Comme il y a énormément de serveurs ElasticSearch, on va en prendre 3 (Tournicoton, Torgamord et Scocolendre) pour forcer une réinstallation avec les paramètres que l'on veut . Avant de modifier le Config Replica Set on a des étapes à faire :

- Stopper le conteneur des anciens serveurs elasticsearch
- Supprimer les anciens serveurs de la centrale ElasticSearch
- Modifier le script d'installation pour les nouveaux serveurs GridFS
- Réinstaller les 3 nouveaux serveurs prod-gridfs
- Appliquer les paramètres
- Ajout des nouveaux serveurs dans le Config Replica Set
- Stopper les conteneurs Config Replica Set sur les anciens serveurs

Explication : On va modifier le Config Replica Set car on peut voir que les serveurs Araqua, Mimantis et Spododo sont dans le Config Replica Set mais également dans le Replica Set 0. On va donc remplacer ces serveurs par de nouveaux afin que les nouveaux soient dans le Config Replica Set et les anciens ne soient que dans le Replica Set 0.

```
root@prod-gridfs-araqua():~# gridfs_configReplSet_quick_status
    "name" : "prod-gridfs-araqua.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-gridfs-mimantis.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-gridfs-spododo.compilatio.net:27019",
    "stateStr" : "SECONDARY",
root@prod-gridfs-araqua():~# gridfs_rs0_quick_status
    "name" : "prod-gridfs-araqua.compilatio.net:27018",
    "stateStr" : "PRIMARY",
    "name" : "prod-gridfs-mimantis.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-gridfs-spododo.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-gridfs-lunala.compilatio.net:27018",
    "stateStr" : "SECONDARY",
    "name" : "prod-gridfs-solgaleo.compilatio.net:27018",
    "stateStr" : "SECONDARY",
```

Mise en arrêt des conteneurs sur les anciens serveurs ElasticSearch

On va se connecter sur un des 3 serveurs ElasticSearch pour stopper tous les dockers. Il faudra faire la même chose sur les 2 autres serveurs mais attention à bien faire un par un pour laisser le temps à ElasticSearch pour répartir toutes les charges afin de ne pas perdre des données.

```
root@prod-elasticsearch-scolocendre():~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
56ca90bb1fc dockerhub.compilatio.net/haproxy:2017-02-03_15h01m28s "/docker-entrypoint..." 2 weeks ago Up 2 weeks
445c93c70a84 kibana:7.16.3 "/bin/tini -- /usr/l..." 2 months ago Up 2 months
b39fb7e67416 elasticsearch:7.16.3 "/bin/tini -- /usr/l..." 2 months ago Up 2 months
root@prod-elasticsearch-scolocendre():~# docker stop elasticsearch-http_proxy
elasticsearch-http_proxy
root@prod-elasticsearch-scolocendre():~# docker stop kibana
kibana
root@prod-elasticsearch-scolocendre():~# docker stop prod-elasticsearch-data_node
prod-elasticsearch-data_node
```

On peut avoir la vue de la répartition des charges sur ElasticSearch. +
Shard Activity

Completed recoveries							
Index	Stage	Total Time	Source / Destination	Files	Bytes	Translog	
common_users_docs_v1	Index	0:16:52	prod-elasticsearch-rongrigou-data_node > prod-elasticsearch-chapotus-data_node	95.4% 333 / 349	29.8% 25.6 GB / 86.0 GB	0.0% 16 / 16	
common_users_docs_v1	Index	0:16:52	prod-elasticsearch-fourbelin-data_node > prod-elasticsearch-engloutyran-data_node	95.5% 316 / 331	34.0% 28.9 GB / 85.0 GB	0.0% 18 / 18	
Status	Nodes	Indices	JVM Heap	Total shards	Unassigned shards	Documents	Data
● Yellow	64	93	319.7 GB / 906.0 GB	3438	56	156,492,289	12.5 TB

Suppression des machines ElasticSearch des configurations

On va aller dans le fichier de configuration pour supprimer les machines du cluster.

```
#           - "prod-elasticsearch-tournicoton.compilatio.net"
#           - "prod-elasticsearch-blancoton.compilatio.net"
#           - "prod-elasticsearch-moumouton.compilatio.net"
#           - "prod-elasticsearch-moumouflon.compilatio.net"
#           - "prod-elasticsearch-khelocrok.compilatio.net"
#
#           - "prod-elasticsearch-torgamord.compilatio.net"
#           - "prod-elasticsearch-volttoutou.compilatio.net"
#           - "prod-elasticsearch-fulgudog.compilatio.net"
#           - "prod-elasticsearch-angoliath.compilatio.net"
#           - "prod-elasticsearch-bibichut.compilatio.net"
#           - "prod-elasticsearch-chapotus.compilatio.net"
#           - "prod-elasticsearch-fourbelin.compilatio.net"
#           - "prod-elasticsearch-grillepattes.compilatio.net"
#           - "prod-elasticsearch-grimalin.compilatio.net"
#           - "prod-elasticsearch-hastacuda.compilatio.net"
#           - "prod-elasticsearch-ixon.compilatio.net"
#           - "prod-elasticsearch-krakos.compilatio.net"
#           - "prod-elasticsearch-polthegeist.compilatio.net"
#           - "prod-elasticsearch-poulpaf.compilatio.net"
#           - "prod-elasticsearch-salarsen.compilatio.net"
#
#           - "prod-elasticsearch-scolocendre.compilatio.net"
```

Pour finir, on va supprimer les machines des référentiels Firewall et Salt.

```
# Suppression du Firewall et de salt

combos="prod-elasticsearch-tournicoton,ns3067890.ip-217-182-174.eu prod-
elasticsearch-torgamord,ns3128398.ip-54-38-92.eu prod-elasticsearch-
scolocendre,ns3081544.ip-145-239-66.eu"
#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/,,/`'
    echo "===== $(date) - compi_name[$compi_name]"
    soyoustart_name[$soyoustart_name] :"
    mfw server $compi_name remove || break
    salt-key -d $compi_name || break
    echo
    echo
    echo
    echo
done
```

Modification du script d'installation pour les nouveaux serveurs GridFS

```
combos="prod-gridfs-tournicoton,ns3067890.ip-217-182-174.eu prod-gridfs-
torgamord,ns3128398.ip-54-38-92.eu prod-gridfs-scolocendre,ns3081544.ip-145-239-66.eu"

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/,,/`"
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add || break
    #mfw server $compi_name add envs prod || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh gridfs-cluster || break
    #msrv $compi_name $soyoustart_name --only_install --template GridFS --force_reinstall ||
break
    echo
    echo
    echo
    echo
done
```

Pour savoir si l'on a besoin de générer des certificats il suffit de contrôler la présence de ceux-ci sur les serveurs en prod-grids.

```
root@ns3045309(trans-saltmaster-absol):~# ll /srv/pillar/prod/certifs | grep -v morpheo
| grep prod-gridfs
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-araqua/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-cancrelove/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-cosmovum/
drwx----- 2 root root 4096 juin 1 2022 prod-gridfs-dodoala/
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-floramantis/
drwx----- 2 root root 4096 mai 19 2022 prod-gridfs-keunotor/
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-lampignon/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-lunala/
drwx----- 2 root root 4096 juin 1 2022 prod-gridfs-meteno/
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-mimantis/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-mouscoto/
drwx----- 2 root root 4096 juin 1 2022 prod-gridfs-silvallie/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-solgaleo/
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-spododo/
drwx----- 2 root root 4096 mai 16 2022 prod-gridfs-tarenbulle/
drwx----- 2 root root 4096 août 1 2022 prod-gridfs-zerooid/
```

Comme on peut le voir, la commande à lister tous les serveurs en prod-grids cela veut donc dire que les nouveaux serveurs devront avoir eux aussi un certificat.

```
-----
Génération des certificats
-----

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - compi_name[$compi_name]"
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name
done
|
=> [OK]

# Contrôle présence des nouveaux serveurs :
ll /srv/pillar/prod/certifs | grep -v morpheo | grep prod-gridfs
```

On va faire l'installation depuis le saltmaster en lançant les scripts, puis on va générer les certificats et vérifier qu'il soit bien présent. Pour finir on va appliquer les paramètres et regarder via la commande docker ps que la machine soit bien en Mongo 4.4.

On va maintenant modifier les pillars pour qu'on puisse ajouter les 3 nouveaux serveurs dans la config de MongoDB.

```
configReplSet:
  - "prod-gridfs-araqua:27019"
  - "prod-gridfs-mimantis:27019"
  - "prod-gridfs-spododo:27019"
  - "prod-gridfs-tournicoton:27019"
  - "prod-gridfs-torgamord:27019"
  - "prod-gridfs-scolocendre:27019"
```

Ensute on va prendre en compte les changements par l'infrastructure. On va lancer ce script sur environnements : prod-elastic, prod-web et prod-worker.

```
nodes_es=$(salt-key -L | grep ^prod-elastic)

for node_es in $nodes_es
do
    echo "===== $(date) - compi_name[$node_es]"
    #saltc $node_es test.ping
    saltc $node_es state.apply
    sleep 15
done
```

Ajout des machines dans le cluster Config Replica Set de GridFS

On ajoute les 3 serveurs en premier pour ne pas casser le cluster car si on supprime les 3 serveurs actuels le Config Replica Set cessera de fonctionner.

```
-----
Ajout des noeuds au gridfs_configReplSet
-----

--sur un des gridfs (configreplicaset)

gridfs_configReplSet
rs.add("prod-gridfs-tournicoton.compilatio.net:27019")
rs.add("prod-gridfs-torgamord.compilatio.net:27019")
rs.add("prod-gridfs-scolocendre.compilatio.net:27019")
```

Puis on va à nouveau reprendre en compte les changements par l'infrastructure. On va contrôler dans les dockers que mongos est présent.

```
--saltmaster
saltc 'prod-web*' state.apply -b 1
saltc 'prod-worker*' state.apply -b 1
saltc 'prod-ip*' state.apply

--Contrôles
saltc 'prod-web*' cmd.run "docker ps | grep mongos"
saltc 'prod-worker*' "docker ps | grep mongos"
saltc 'prod-ip*' "docker ps | grep mongos"
```

Maintenant on va pouvoir supprimer les 2 serveurs secondaires du cluster, puis faire la commande rs.stepDown() pour faire passer le serveur primaire en secondaire et on va regarder le statut afin de s'assurer que le rôle de primaire soit redistribué. Pour finir on va supprimer le serveur devenu secondaire du cluster pour qu'il ne reste que les 3 nouveaux serveurs que nous venons d'ajouter.

```
-----
Suppression des noeuds du gridfs_configReplSet
-----
--sur un des gridfs (configreplicaset)
gridfs_configReplSet
    rs.status()

Lorsque les 3 nouveaux sont en "SECONDARY" :
rs.remove("prod-gridfs-araqua.compilatio.net:27019")
rs.remove("prod-gridfs-spododo.compilatio.net:27019")
rs.stepDown()
rs.remove("prod-gridfs-mimantis.compilatio.net:27019")

rs.status()
```

```
root@prod-gridfs-tournicoton():~# gridfs_configReplSet_quick_status
    "name" : "prod-gridfs-tournicoton.compilatio.net:27019",
    "stateStr" : "PRIMARY",
    "name" : "prod-gridfs-torgamord.compilatio.net:27019",
    "stateStr" : "SECONDARY",
    "name" : "prod-gridfs-scolocendre.compilatio.net:27019",
    "stateStr" : "SECONDARY",
```

Suppression des conteneurs et des données sur les anciens serveurs

Dans cette partie, il suffit juste de stopper le docker correspondant au Config Replica Set puisqu'ils ne sont plus dedans et on va supprimer les données qui se trouvent dans les conteneurs de celui-ci.

```
--araqua
docker stop prod-gridfs-configReplSet
rm -r /home/sys/docker/containers/prod-gridfs-configReplSet/

--mimantis
docker stop prod-gridfs-configReplSet
rm -r /home/sys/docker/containers/prod-gridfs-configReplSet/

--spododo
docker stop prod-gridfs-configReplSet
rm -r /home/sys/docker/containers/prod-gridfs-configReplSet/
```

Industrialiser un serveur web à destination des développeurs pour les plugins LMS

Inventaire software d'un serveur oldprod-lti afin de lister tous les services nécessaires

Le but ici est de lister tous les services, les configurations... pour savoir ce qui est nécessaire pour installer des nouveaux serveurs.

On va commencer avec la commande service –status-all pour lister tous les services avec leurs statuts (les services actifs ont un + et les inactifs ont un -).

```
root@oldprod-lti-venalgue():~# service --status-all
[ - ]  apache-htcacheload
[ + ]  apache2
[ + ]  apparmor
[ + ]  apport
[ + ]  atd
[ + ]  auditd
[ - ]  console-setup.sh
[ + ]  cron
[ - ]  cryptdisks
[ - ]  cryptdisks-early
[ + ]  dbus
[ + ]  docker
[ + ]  fail2ban
[ + ]  firewall
[ - ]  grub-common
[ - ]  hwclock.sh
[ + ]  irqbalance
[ - ]  iscsid
[ - ]  keyboard-setup.sh
[ + ]  kmod
[ + ]  lm-sensors
[ - ]  lvm2
[ - ]  lvm2-lvmpolld
[ + ]  multipath-tools
[ + ]  munin-node
[ + ]  mysql
[ + ]  nagios-nrpe-server
[ + ]  ntp
[ - ]  open-iscsi
[ - ]  open-vm-tools
[ - ]  plymouth
[ - ]  plymouth-log
[ + ]  procps
[ + ]  qemu-guest-agent
[ - ]  rsync
[ + ]  rsyslog
[ + ]  salt-minion
[ - ]  screen-cleanup
[ + ]  ssh
[ + ]  sysstat
[ + ]  udev
[ + ]  ufw
[ + ]  unattended-upgrades
[ - ]  uuidd
```

On retrouve également la commande `systemctl list-units` qui va lister toutes les unités actives que `systemd` reconnaît et pour plus de précision on peut ajouter un filtre pour n'avoir que les services.

```
accounts-daemon.service
apache2.service
apparmor.service
apport.service
atd.service
auditd.service
blk-availability.service
cloud-config.service
cloud-final.service
cloud-init-local.service
cloud-init.service
console-setup.service
containerd.service
cron.service
dbus.service
docker.service
fail2ban.service
finalrd.service
firewall.service
getty@tty1.service
irqbalance.service
keyboard-setup.service
kmod-static-nodes.service
lm-sensors.service
logrotate.service
lvm2-monitor.service
ModemManager.service
```

```
multipathd.service
munin-node.service
mysql.service
nagios-nrpe-server.service
networkd-dispatcher.service
ntp.service
polkit.service
qemu-guest-agent.service
rsyslog.service
salt-minion.service
serial-getty@ttyS0.service
setvtrgb.service
ssh.service
sysstat.service
systemd-fsck@dev-disk-by\x2dlabel-UEFI.service
systemd-journal-flush.service
systemd-journald.service
systemd-logind.service
systemd-modules-load.service
systemd-networkd-wait-online.service
systemd-networkd.service
systemd-random-seed.service
systemd-remount-fs.service
systemd-resolved.service
systemd-sysctl.service
systemd-sysusers.service
systemd-tmpfiles-setup-dev.service
systemd-tmpfiles-setup.service
systemd-udev-settle.service
systemd-udev-trigger.service
systemd-udevd.service
systemd-update-utmp.service
systemd-user-sessions.service
udisks2.service
ufw.service
unattended-upgrades.service
user-runtime-dir@0.service
user@0.service
```

Ensuite il y a la commande dpkg –l pour afficher la liste des paquets.

		Architecture	Description
ii	accountservic	0.6.55-0ubuntu12~20.04.5	query and manipulate user account information
ii	aduser	3.118ubuntu2	add and remove users and groups
ii	alsa-topology-conf	1.2.2-1	ALSA topology configuration files
ii	alsa-ucm-conf	1.2.2-1ubuntu0.12	ALSA Use Case Manager configuration files
ii	apache2	2.4.41-4ubuntu3.12	Apache HTTP Server
ii	apache2-bin	2.4.41-4ubuntu3.12	Apache HTTP Server (modules and other binary files)
ii	apache2-data	2.4.41-4ubuntu3.12	Apache HTTP Server (common files)
ii	apache2-utils	2.4.41-4ubuntu3.12	Apache HTTP Server (utility programs for web servers)
ii	apparmor	2.13.3-7ubuntu5.1	user-space parser utility for AppArmor
ii	apport	2.26.11-0ubuntu27.24	automatically generate crash reports for debugging
ii	apport-symptoms	0.23	symptom scripts for apport
ii	apt	2.0.6	commandline package manager
ii	apt-transport-https	2.0.6	transitional package for https support
ii	apt-utils	2.0.6	package management related utility programs
ii	at	1.5.22-23.2ubuntu1	Delayed job execution and batch processing
ii	autodt	2.69-11.1	User space tools for security auditing
ii	automake	1:1.16.1-4ubuntu6	automatic configuration script builder
ii	autopoint	0.19.8.1-10build1	automake program from GNU Standard-compliant Makefiles
ii	autoools-dev	20180224.1	Update infrastructure for config.{guess,sub} files
ii	base-files	11ubuntu5.5	Debian base system miscellaneous files
ii	base-passwd	3.5.47	Debian base system master password and group files
ii	bash	5.0-6ubuntu1.2	GNU Bourne Again Shell
ii	bash-completion	1:2.10-1ubuntu1	programmable completion for the bash shell
ii	bc	1.07.1-2build1	GNU bc arbitrary precision calculator language
ii	bcache-tools	1.0.8-3ubuntu0.1	bcache userspace tools
ii	bind9-dnsutils	1:9.16.1-0ubuntu2.12	Clients provided with BIND 9
ii	bind9-host	1:9.16.1-0ubuntu2.12	DNS lookup Utility
ii	bind9-libs:amd64	1:9.16.1-0ubuntu2.12	Shared libraries used by BIND 9
ii	binutils	2.34-6ubuntu1.4	GNU assembler, linker and binary utilities
ii	binutils-common:amd64	2.34-6ubuntu1.4	Common files for the GNU assembler, linker and binary utilities
ii	binutils-x86-64-linux-gnu	2.34-6ubuntu1.4	GNU binary utilities, for x86-64-linux-gnu target
ii	bolt	0.8-4ubuntu1	system daemon to manage thunderbolt 3 devices
ii	bsdmainutils	11.1.2ubuntu3	collection of more utilities from FreeBSD

On peut regarder aussi dans les configurations globales des serveurs oldprod dans /srv/salt/oldprod/lti/init.sls et on peut également aller voir dans le top.sls.

```
#####
# packages
#####
lti-pkgs:
    pkg.installed:
        - pkgs:
            - apache2
            - libapache2-mod-php
            - php-apcu
            - php-bcmath
            - php-cli
            - php-curl
            - php-intl
            - php-mbstring

oldprod:
    'oldprod-*':
        - common_all
        - common_oldprod
    '^oldprod-(solrwrapper|rdsmaster)*':
        - match : pcre
        - smtp_oldprod
        - mongodb_oldprod
        - mysql_oldprod
    '^oldprod-(elasticsearch|solrwrapper)*':
        - match : pcre
        - elasticsearch_oldprod
        - elasticsearch_all
    'oldprod-lti-*':
        - smtp_oldprod
        - mysql_oldprod
    'oldprod-shibboleth-*':
        - shibboleth_oldprod
```

Pour finir, récupérer les input_roles avec la commande mfw server.

```
root@ns3045309(trans-saltmaster-absol):~# mfw server oldprod-lti-venalgue
=====
OrderedDict([('hostname', 'oldprod-lti-venalgue'),
 ('compi_name', 'oldprod-lti-venalgue'),
 ('out_ips', ['51.68.114.146']),
 ('envs', ['oldprod']),
 ('input_roles',
  ['nrpe', 'munin_node', 'ssh', 'noswap', 'lti.compilatio.net']),
 ('output_roles', []),
 ('managed_ips', []),
 ('backup', {'ignores': []}),
 ('creation_utc',
  datetime.datetime(2022, 3, 15, 9, 25, 54, 120000)),
 ('update_utc', datetime.datetime(2022, 3, 15, 14, 30, 4, 891000))])
```

Vérifier la version de PHP.

```
root@oldprod-lti-venalgue():~# php -v
PHP 8.1.3 (cli) (built: Feb 21 2022 14:48:42) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.3, Copyright (c) Zend Technologies
    with Zend OPcache v8.1.3, Copyright (c), by Zend Technologies
```

Création d'une image docker personnalisée contenant tous le requis pour les plugins MLS ainsi que sa configuration

Pour ce qui est de la suite, je vais résumer ce que l'on a fait avec mon maître de stage. Nous avons déterminé tous les éléments nécessaires pour le fonctionnement de la machine et nous avons décidé de créer un conteneur spécifique pour automatiser l'installation des serveurs Iti.

Dans les fichiers de configuration, on retrouve un Dockerfile. Ce Dockerfile permet de lister toutes les instructions à exécuter pour build une image. On va faire en sorte que Apache2 soit démarré automatiquement lors du lancement de la machine.

```
ADD global_compi.conf /etc/apache2/conf-available/
ADD default_vhost_compi.conf /etc/apache2/sites-available/
ADD mpm_prefork.conf /etc/apache2/mods-available/
ADD favicon.ico /home/sites/default/www/
RUN a2enconf global_compi && \
    a2dismod status && \
    a2enmod deflate rewrite && \
    a2dissite 000-default && \
    a2ensite default_vhost_compi && \
    phpenmod mongodb yaml && \
    touch /etc/apache2/custom_compi.conf && \
    chown -R www-data:www-data /home/sites/default/www/favicon.ico && \
    chmod 700 /etc/apache2

#####
# Start apache2
#####
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

comme on démarrait le conteneur avec bash et avec docker on ne peut pas faire plusieurs cmd donc on ne peut lancer qu'un seul script.

docker run -d, pour lancer le conteneur comme une tache de fond
docker exec -it pour interactif pour lancer bash dans le conteneur
<https://docs.docker.com/engine/reference/commandline/run/>
<http://supervisord.org/introduction.html>

Réinstallation de 4 serveurs ElasticSearch en serveurs LTI

Suppression des machines ElasticSearch

Suppression dockers des anciennes machines ElasticSearch

Comme il y a énormément de serveurs ElasticSearch on va en choisir 4 et on va ensuite les réinstaller pour les utiliser d'une autre façon. Une fois qu'on a choisi ces 4 serveurs, on va se connecter sur chacun d'entre eux et on va stopper tous les dockers mais attention il faudra faire serveur par serveur et attendre que la répartition des charges se termine pour éviter la perte de données.

```
root@prod-elasticsearch-moumouflon():~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
04e7c7e3ea99 dockerhub.compilatio.net/haproxy:2017-02-03_15h01m28s "/docker-entrypoint..." 2 weeks ago Up 2 weeks
f24a1847796a kibana:7.16.3 "/bin/tini - /usr/l..." 3 months ago Up 3 months
b426e5a93f30 elasticsearch:7.16.3 "/bin/tini - /usr/l..." 3 months ago Up 3 months
root@prod-elasticsearch-moumouflon():~# docker stop elasticsearch-http_proxy
elasticsearch-http_proxy
root@prod-elasticsearch-moumouflon():~# docker stop kibana
kibana
root@prod-elasticsearch-moumouflon():~# docker stop prod-elasticsearch-data_node
prod-elasticsearch-data_node
```

On peut vérifier sur ElasticSearch la répartition des charges pour supprimer les dockers des prochains serveurs. On remarque 56 fragments de données non assignés, il suffit donc d'attendre qu'ils soient assignés.

Status	Alerts	Nodes	Indices	JVM Heap	Total shards	Unassigned shards
● Yellow	● 0	63	93	367.6 GB / 892.0 GB	3320	56

Suppression des anciennes machines ElasticSearch de leur fichier de configuration

On va maintenant supprimer les serveurs ElasticSearch de leurs fichiers de configuration pour qu'ils ne soient plus considérés par le cluster.

```
# Configuration Salt
Editer et supprimer les références aux machines sélectionner de
/srv/pillar/prod/elasticsearch_prod.sls
```

On va simplement mettre toutes les lignes qui mentionnent les 4 serveurs en commentaire.

```
#           - "prod-elasticsearch-tournicoton.compilatio.net"
#           - "prod-elasticsearch-blancoton.compilatio.net"
#           - "prod-elasticsearch-moumouton.compilatio.net"
#           - "prod-elasticsearch-moumouflon.compilatio.net"
#           - "prod-elasticsearch-khelocrok.compilatio.net"
#           - "prod-elasticsearch-torgamord.compilatio.net"
#           - "prod-elasticsearch-volttoutou.compilatio.net"
#           - "prod-elasticsearch-fulgudog.compilatio.net"
```

Suppression du référentiel Firewall et Salt

```
# Suppression du Firewall et de salt
combos="prod-elasticsearch-moumouflon,ns3075071.ip-217-182-174.eu prod-elasticsearch-
moumouton,ns3075060.ip-217-182-174.eu prod-elasticsearch-khelocrok,ns3101653.ip-51-38-52.eu
prod-elasticsearch-fulgudog,ns3128436.ip-54-38-92.eu"

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*,//'`
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    mfw server $compi_name remove || break
    salt-key -d $compi_name || break
    echo
    echo
    echo
    echo
done
```

Pour les input_role on veut avoir accès à l'environnement MongoDB on va aller chercher quel input-role va nous permettre de faire ceci. On regarde sur un prod-mongodb quels input_role il possède. On remarque qu'il y a mongodb-cluster donc on va regarder ce que contient cet input_role. De la même façon, on voit qu'il y a mongos et que si l'on va voir dedans.

```

root@ns3045309(trans-saltmaster-absol):~# mfw server prod-mongodb-cosmog
=====
OrderedDict([('hostname', 'ns3076850.ip-147-135-137.eu'),
    ('compi_name', 'prod-mongodb-cosmog'),
    ('out_ips', ['147.135.137.113']),
    ('envs', ['prod']),
    ('input_roles', ['nrpe', 'munin_node', 'ssh', 'mongodb-cluster']),
    ('output_roles', []),
    ('managed_ips', []),
    ('backup', {'ignores': []}),
    ('creation_utc',
        datetime.datetime(2022, 7, 20, 14, 36, 16, 944000)),
    ('update_utc',
        datetime.datetime(2022, 7, 20, 14, 36, 18, 503000))])
root@ns3045309(trans-saltmaster-absol):~# mfw input_role mongodb-cluster
=====
OrderedDict([('role_name', 'mongodb-cluster'),
    ('ports',
        [{'desc': 'mongod shardsvr', 'num': 27018, 'proto': 'tcp'},
         {'desc': 'config srv', 'num': 27019, 'proto': 'tcp'}]),
    ('allowed_envs', ['SAME']),
    ('allowed_roles',
        ['mongodb-cluster',
         'mongodb-mongos',
         'adsl',
         'alerting',
         'metrology'])])

```

```

root@ns3045309(trans-saltmaster-absol):~# mfw input_role mongodb-mongos
=====
OrderedDict([('role_name', 'mongodb-mongos'),
    ('ports', [{'num': 27020, 'proto': 'tcp'}]),
    ('allowed_envs', ['SAME']),
    ('allowed_roles',
        ['mongodb-mongos',
         'mongodb-cluster',
         'apache',
         'worker',
         'adsl',
         'alerting',
         'metrology'])])

```

On va maintenant regarder les `input_role` des serveurs `prod-web` et des serveurs `dev-web` afin de s'assurer que c'est le même `input_role` sur les 2 environnements.

Modification des fichiers `top.sls` dans les répertoires `pillar` et `salt`

Modification sur le fichier `/srv/pillar/top.sls` pour les serveurs `prod` et `dev`

```

'^dev-(arbiter|mongodb|db|queuing|mongoperf|mongomajperf|web|worker|ip|gpu|lti)': 
    - match : pcre
    - mongodb_all
    - mongodb_dev

```

```

'dev-lti-*':
  - artifact_all
  - continius_deployment_all
  - smtp_dev

'^prod-(mongodb|queuing|gridfs|mongoperf|mongomajoperf|web|worker|ip|gpu|lti)-':
  - match : pcre
  - mongodb_all
  - mongodb_prod

'prod-lti-*':
  - artifact_all
  - continius_deployment_all
  - smtp_prod

```

Modification sur le fichier /srv/salt/top.sls pour les serveurs prod et salt

```

'dev-lti-*':
  - helpers.memory_tuning
  - sites.user_www-data
  - access.developpers
  - continius_deployment.shared_resources
  - docker.sys_build_containers.haproxy
  - docker.sys_build_containers.mongos.mongodb
  - docker.sys_build_containers.php_apache_lti

'prod-lti-*':
  - helpers.memory_tuning
  - sites.user_www-data
  - access.developpers
  - continius_deployment.shared_resources
  - docker.sys_build_containers.haproxy
  - docker.sys_build_containers.mongos.mongodb
  - docker.sys_build_containers.php_apache_lti

```

Prise en compte de l'infrastructure

On va prendre en compte différentes infrastructures telles que la prod de Elastic, IP, Worker et Web. La prise en compte de l'infra consiste à state.apply et on va redémarrer les dockers pour certains environnements.

```
-- Depuis un Saltmaster

nodes_es=$(salt-key -L | grep ^prod-elastic)
for node_es in $nodes_es
do
    echo "===== $(date) - compi_name[$node_es]"
    saltc $node_es test.ping
    saltc $node_es state.apply
    sleep 15
done
```

Installation des serveurs dev-lti et prod-lti

Modification du script pour l'installation du serveur dev-lti

```
--saltmaster
combos="dev-lti-moumouflon,ns3075071.ip-217-182-174.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyousstart_name=`echo $combo | sed 's/.*,//'`
    echo "===== $(date) - compi_name[$compi_name] soyousstart_name[$soyousstart_name] :"
    #msrv $compi_name $soyousstart_name --only_dns || break
    #msrv $compi_name $soyousstart_name --only_reverse || break
    #mfw server $soyousstart_name add $compi_name || break
    #mfw server $compi_name add envs dev || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongodb-mongos || break
    #msrv $compi_name $soyousstart_name --only_install --template web --force_reinstall ||
break
echo
echo
echo
echo
done
```

Modification du script pour l'installation des serveurs prod-lti

```
combos="prod-lti-moumouton,ns3075060.ip-217-182-174.eu prod-lti-
khelocrok,ns3101653.ip-51-38-52.eu prod-lti-fulgudog,ns3128436.ip-54-38-92.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyousstart_name=`echo $combo | sed 's/.*,//'`
    echo "===== $(date) - compi_name[$compi_name] soyousstart_name[$soyousstart_name] :"
    #msrv $compi_name $soyousstart_name --only_dns || break
    #msrv $compi_name $soyousstart_name --only_reverse || break
    #mfw server $soyousstart_name add $compi_name || break
    #mfw server $compi_name add envs prod || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongodb-mongos || break
    #msrv $compi_name $soyousstart_name --only_install --template web --force_reinstall ||
break
echo
echo
echo
echo
done
```

On va vérifier que les serveurs se sont bien installés avec la commande salt-key puis on va accepté les clés.

```
root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep lti
dev-lti-moumouflon
olddev-lti-golgopathe
oldprod-lti-venalgue
prod-lti-fulgudog
prod-lti-khelocrok
prod-lti-moumouton
```

```
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'^
    soyoustart_name=`echo $combo | sed 's/.*/^'^
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    salt-key --accept=$compi_name || break
    echo
    echo
    echo
    echo
done
```

Ensute on va générer les certificats et vérifier leurs présences

```
-----
Génération des certificats
-----
combos="prod-lti-moumouton,ns3075060.ip-217-182-174.eu prod-lti-
khelocrok,ns3101653.ip-51-38-52.eu prod-lti-fulgudog,ns3128436.ip-54-38-92.eu dev-lti-
moumouflon,ns3075071.ip-217-182-174.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'^
    echo "===== $(date) - compi_name[$compi_name]"
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name
done

# Contrôle présence des nouveaux serveurs :
ll /srv/pillar/prod/certifs | grep lti
ll /srv/pillar/dev/certifs | grep lti
```

Application des paramètres

On va ensuite appliquer les paramètres afin de voir les modifications et s'il n'y a pas d'erreurs. On pourra remarquer qu'après le 2ème state.apply il y a toujours des erreurs, cela veut dire qu'il faut faire du debug et grâce aux erreurs on peut se douter vers où il faut aller pour corriger.

```

combos="prod-lti-moumouton,ns3075060.ip-217-182-174.eu prod-lti-
khelocrok,ns3101653.ip-51-38-52.eu prod-lti-fulgudog,ns3128436.ip-54-38-92.eu dev-lti-
moumouflon,ns3075071.ip-217-182-174.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - ${compi_name}"
    #saltc ${compi_name} test.ping
    #saltc ${compi_name} saltutil.refresh_pillar
    #saltc ${compi_name} state.apply
    #saltc ${compi_name} state.apply
    #salt ${compi_name} cmd.run "reboot"
    #salt ${compi_name} cmd.run "uptime"
    #salt ${compi_name} cmd.run "uname -a"
    #salt ${compi_name} cmd.run "iptables -vnL | grep DROP"
    #salt ${compi_name} cmd.run "ip6tables -vnL | grep DROP"
done

```

Mise en place de différents besoins pour le développeur

Contexte : Après avoir livré les serveurs au développeur pour qu'ils puissent travailler dessus, il nous a fait un rapport de quelques fonctionnalités afin qu'il ait tout ce dont il ait besoin et qu'il puisse avoir accès à tout.

De plus, l'utilisateur ne peut pas avoir accès au conteneur or les fichiers de logs sont dedans. C'est pour cela que nous allons déplacer des fichiers de log en dehors du conteneur puis on va mettre en place un console_to_container pour qu'il puisse avoir accès à celui-ci avec root ou www-data

Pour rappel : Nous avons utilisé des scripts qui étaient déjà faits et nous les avons modifiés selon les besoins du développeur.

Création de plusieurs crontab Canvas, Brightspace et Teams

Pour commencer, il avait besoin de plusieurs crontab donc il nous les a donnés puis on les a intégrés dans les fichiers de configuration pour que lorsqu'il se connecte tout sera déjà présent. On va modifier dans le répertoire /srv/salt/prod/docker/sys_build_containers/php_apache_lti.

```

#####
# Script
#####
{{ container_name }}-console_to_container:
    file.managed:
        - name : {{ scripts_dir }}/{{ container_name }}-console_to_container.sh
        - source : {{ salt_source_files_dir }}/{{ container_name }}-console_to_container.jinja.sh
        - mode : 700
        - user : root
        - template : jinja
        - context :
            allowed_users : {{ allowed_users }}
            container_name : {{ container_name }}

```

```

#####
# aliases
#####
continius_deployment-aliases:
    file.managed:
        - name : /etc/profile.d/continius_deployment_aliases.sh
        - source : {{ salt_source_files_dir }}/lti_continius_deployment_aliases.jinja.sh
        - template : jinja
        - context :
            scripts_dir : {{ scripts_dir }}

#####
# sudoers
#####
continius_deployment-sudoers:
    file.managed:
        - name : /etc/sudoers.d/continius_deployment
        - source : {{ salt_source_files_dir }}/lti_continius_deployment_sudoers.jinja
        - template : jinja
        - context :
            scripts_dir : {{ scripts_dir }}
        - mode : 440

```

Après les avoir mis dans la config pour qu'on puisse les voir il suffit de juste faire un state.apply. On fait la commande crontab - l pour lister toutes les crontab actives.

```

# SALT_CRON_IDENTIFIER:canvas_updateStatus
* * * * * docker exec -it php_apache_1tt sh -c "php /home/sites/lti/www/canvas/app/cron/updateStatus.php" >> /home/sites/logs/logs_appli/canvas_cron.log 2>&1
# SALT_CRON_IDENTIFIER:canvas_cleanTmp
0 * * * * docker exec -it php_apache_1tt sh -c "find /home/sites/lti/www/canvas/tmp -maxdepth 1 -minin +60 -exec rm {} \;" 
# SALT_CRON_IDENTIFIER:canvas_dailyCleanLog
0 6 * * * docker exec -it php_apache_1tt sh -c "rm /home/sites/logs/logs_appli/canvas_cron.log"
# SALT_CRON_IDENTIFIER:brightspace_getStatus
* * * * * docker exec -it php_apache_1tt sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/getStatus.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
# SALT_CRON_IDENTIFIER:brightspace_cleanTmp
0 * * * * docker exec -it php_apache_1tt sh -c "find /home/sites/lti/www/brightspace/tmp -maxdepth 1 -minin +60 -exec rm {} \;" 
# SALT_CRON_IDENTIFIER:brightspace_dailyCleanLog
0 6 * * * docker exec -it php_apache_1tt sh -c "rm /home/sites/logs/logs_appli/brightspace_cron.log"
# SALT_CRON_IDENTIFIER:brightspace_sendFiles
*/5 * * * * docker exec -it php_apache_1tt sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/sendFiles.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
# SALT_CRON_IDENTIFIER:teams_sendFiles
*/5 * * * * docker exec -it php_apache_1tt sh -c "php /home/sites/lti/www/teams/sendFiles.php" >> /home/sites/logs/logs_appli/teams_cron.log 2>&1

```

Changement des accès de certains fichiers

On veut maintenant que les logs se fassent hors conteneur. On crée donc des variables possédant des chemins d'accès pour diriger les informations fait ces variables.

```

#####
## POST-conf
#####
{% set container_image = 'dockerhub.compilatio.net/' + container_name + ':' + wanted_container_version %}

{% set host_dir = '/home/sites/' %}
{% set host_dir_data = host_dir + '/lti' %}
{% set host_dir_logs = host_dir + '/logs' %}
{% set host_dir_logs_apache = host_dir_logs + '/logs_apache' %}
{% set host_dir_logs_appli = host_dir_logs + '/logs_appli' %}
{% set host_dir_access = host_dir + '/_access' %}

{% set container_dir_data = '/home/sites/lti/' %}
{% set container_dir_logs = '/home/sites/logs/' %}
{% set container_dir_access = '/home/sites/_access/' %}

{% set container_host_dir = '/home/sys/docker/containers/' + container_name %}
{% set compose_file = container_host_dir + '/docker-compose.yml' %}
```

```
#####
## dirs
#####
{{ container_name }}-host_dir:
    file.directory:
        - name : {{ host_dir }}
        - user : {{ data_dir_owner }}
        - group : {{ data_dir_owner }}
        - dir_mode : 700
        - makedirs : True

{{ container_name }}-host_dir_data:
    file.directory:
        - name : {{ host_dir_data }}
        - dir_mode : 750
        - user : {{ data_dir_owner }}
        - group : {{ data_dir_owner }}
        - makedirs : True

{{ container_name }}-host_dir_logs:
    file.directory:
        - name : {{ host_dir_logs }}
        - user : {{ data_dir_owner }}
        - group : {{ data_dir_owner }}
        - dir_mode : 700

{{ container_name }}-host_dir_logs_apache:
    file.directory:
        - name : {{ host_dir_logs_apache }}
        - user : {{ data_dir_owner }}
        - group : {{ data_dir_owner }}
        - dir_mode : 700

{{ container_name }}-host_dir_logs_appli:
    file.directory:
        - name : {{ host_dir_logs_appli }}
        - user : {{ data_dir_owner }}
        - group : {{ data_dir_owner }}
        - dir_mode : 700

{{ container_name }}-container_host_dir:
    file.directory:
        - name : {{ container_host_dir }}
        - dir_mode : 700
```

Grâce à ce script, on peut regarder sur les serveurs-lti voir si les scripts sont dans leurs bons répertoires.

On retrouve bien dans /home/sites : _access, lti et logs. Si on regarde dans les logs on devrait avoir accès aux répertoires logs_apache et logs_appli.

```
root@dev-lti-moumouflon():/home/sites# ll
total 8
drwx----- 8 www-data www-data 129 févr. 9 12:14 .
drwxr-xr-x 4 root      root      30 févr. 7 15:27 ..
drwxr-x--- 2 root      www-data 275 févr. 7 15:46 _access/
-rw------- 1 www-data www-data 1928 févr. 9 15:48 .bash_history
drwx----- 2 www-data www-data  34 févr. 7 16:47 .cache/
drwxrwxr-x 3 www-data www-data  19 févr. 7 16:48 .local/
drwx----- 4 www-data www-data  43 févr. 9 12:43 logs/
drwxr-x--- 5 www-data www-data  62 févr. 7 17:08 lti/
-rw-rw-r-- 1 www-data www-data  66 févr. 7 17:08 .selected_editor
drwx----- 2 www-data www-data  29 févr. 7 15:27 .ssh/
```

```
root@dev-lti-moumouflon():/home/sites/logs# ll
total 0
drwx----- 4 www-data www-data 43 févr. 9 12:43 .
drwx----- 8 www-data www-data 129 févr. 9 12:14 ..
drwx----- 2 www-data www-data 41 févr. 9 15:45 logs_apache/
drwx----- 2 www-data www-data 79 févr. 9 16:10 logs_appli/
root@dev-lti-moumouflon():/home/sites/logs#
```

On peut voir les logs dans le répertoire d'Apache car il est actuellement actif.

```
root@dev-lti-moumouflon():/home/sites/logs/logs_apache# ll
total 836
drwx----- 2 www-data www-data 41 févr. 9 15:45 .
drwx----- 4 www-data www-data 43 févr. 9 12:43 ..
-rw-r--r-- 1 root root 547009 févr. 9 17:03 access.log
-rw-r--r-- 1 root root 5 févr. 9 15:45 error.log
root@dev-lti-moumouflon():/home/sites/logs/logs_apache#
```

Mise en place d'un console_to_container pour avoir accès au conteneur

Tout d'abord, on a mis 3 nouveaux scripts importants pour la mise en place de la console sur le conteneur.

```
root@ns3045309(trans-saltmaster-absol):/srv/salt/prod/docker/sys_build_containers/php_apache_lti/files# ll
total 24
drwxr-xr-x 2 root root 4096 févr. 9 15:40 .
drwxr-xr-x 4 root root 4096 févr. 9 16:06 ..
-rw-r--r-- 1 root root 580 févr. 9 11:58 docker-compose.jinja.yml
-rw-r--r-- 1 root root 3543 févr. 9 15:29 lti_console_to_container.jinja.sh
-rw-r--r-- 1 root root 312 févr. 9 15:22 lti_continus_deployment_aliases.jinja.sh
-rw-r--r-- 1 root root 163 févr. 9 15:22 lti_continus_deployment_sudoers.jinja
```

Dans le fichier de configuration init.sls il faudra faire le lien entre les 3 fichiers et la configuration.

```
#####
# Script
#####
{{ container_name }}-console_to_container:
    file.managed:
        - name : {{ scripts_dir }}/console_to_container.sh
        - source : {{ salt_source_files_dir }}/lti_console_to_container.jinja.sh
        - mode : 700
        - user : root
        - template : jinja
        - context :
            allowed_users : {{ allowed_users }}
            container_name : {{ container_name }}
```

```
#####
# aliases
#####
continius_deployment-aliases:
    file.managed:
        - name : /etc/profile.d/continius_deployment_aliases.sh
        - source : {{ salt_source_files_dir }}/lti_continius_deployment_aliases.jinja.sh
        - template : jinja
        - context :
            scripts_dir : {{ scripts_dir }}

#####
# sudoers
#####
continius_deployment-sudoers:
    file.managed:
        - name : /etc/sudoers.d/continius_deployment
        - source : {{ salt_source_files_dir }}/lti_continius_deployment_sudoers.jinja
        - template : jinja
        - context :
            scripts_dir : {{ scripts_dir }}
        - mode : 440
```

Les sudoers sont tous les comptes utilisateurs permettant d'exécuter des commandes sudo donc dans cette partie, on crée un alias pour www-data puis on fait un lien avec le console_to_container pour qu'il puisse utiliser le sudo.

```
root@dev-lti-moumouflon():~# cat /etc/sudoers.d/continius_deployment
User_Alias      DEVS = www-data

Cmnd_Alias      CONSOLETOCONTAINER =
/home/sys/admin_scripts/continius_deployment/console_to_container.sh
DEVS           ALL = NOPASSWD: CONSOLETOCONTAINER
```

Dans cette alias si un compte root exécute le script il se fera normalement or si c'est un compte www-data il utilisera le sudo avant la commande.

```
# If not running interactively, don't do anything
[ -z "$PS1" ] && return
[ "$SHELL" = "/bin/bash" ] || return

function console_to_container {
    cmd={{ scripts_dir }}/console_to_container.sh
    whoami | egrep '^root$' > /dev/null
    if [ $? -eq 0 ]
    then
        $cmd
    else
        sudo $cmd
    fi
```

On peut désormais mettre à jour les paramètres sur le saltmaster et vérifier que l'on puisse se connecter dans le conteneur en tant qu'utilisateur www-data

```
root@dev-lti-moumouflon():~# su - www-data
www-data@dev-lti-moumouflon:~$ console_to_container
user ?
1) www-data
2) root
#? 2
[Info] - docker exec into container [php_apache_lti] as user [root]...
root@dev-lti-moumouflon:/home/sites/lti#
```

Ajout de nouvelles crontab Canvas, Brightspace, Teams et TMP

Nous allons devoir faire de nouvelles crontab pour le développeur :

- Pour TMP

```
#clean tmp folder
0 */1 * * * docker exec -i php_apache_lti sh -c "find /home/sites/lti/www/tmp -maxdepth 1 -mmin +60 -exec rm {} \;"
```

- Pour Teams

```
#Send files teams
#Server 1
0,15,30,45 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/teams/sendFiles.php" >> /home/sites/logs/logs_appli/teams_cron.log 2>&1
#Server 2
5,20,35,50 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/teams/sendFiles.php" >> /home/sites/logs/logs_appli/teams_cron.log 2>&1
#Server 3
10,25,40,55 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/teams/sendFiles.php" >> /home/sites/logs/logs_appli/teams_cron.log 2>&1
```

- Pour Brightspace

```
#Send files brightspace
#Server 1
0,9,18,27,36,45,54 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/sendFiles.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
#Server 2
5,20,35,50 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/sendFiles.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
#Server 3
10,25,40,55 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/sendFiles.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1

#get status brightspace
#Server 1
0,9,18,27,36,45,54 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/getStatus.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
#Server 2
3,12,21,30,39,48,57 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/getStatus.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
#Server 3
6,15,24,33,42,51,60 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/getStatus.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
```

- Pour Canvas

```
#Update status canvas
#Server 1
0,9,18,27,36,45,54 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/canvas/app/cron/updateStatus.php" >> /home/sites/logs/logs_appli/canvas_cron.log 2>&1
#Server 2
3,12,21,30,39,48,57 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/canvas/app/cron/updateStatus.php" >> /home/sites/logs/logs_appli/canvas_cron.log 2>&1
#Server 3
6,15,24,33,42,51,60 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti/www/canvas/app/cron/updateStatus.php" >> /home/sites/logs/logs_appli/canvas_cron.log 2>&1
```

Pour cela, nous allons aller dans le conteneur de php_apache_lti pour modifier le fichier init.sls comportant les paramètres sur les crontab.

Comme on peut le voir sur les crontab ci-dessus, on remarque qu'il faut faire une crontab avec des temps différents pour chaque serveur. On va faire des variables de temps afin de spécifier le temps pour les 4 serveurs.

```

{% if grains["id"].startswith("dev-lti-moumouflon") %}
    {% set sendFiles = '0,5,10,15,20,25,30,35,40,45,50,55' %}
    {% set Status = '0,3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48,51,54,57' %}
{% endif %}

{% if grains["id"].startswith("prod-lti-fulgudog") %}
    {% set sendFiles = '0,15,30,45' %}
    {% set Status = '0,9,18,27,36,45,54' %}
{% endif %}

{% if grains["id"].startswith("prod-lti-khelocrok") %}
    {% set sendFiles = '5,20,35,50' %}
    {% set Status = '3,12,21,30,39,48,57' %}
{% endif %}

{% if grains["id"].startswith("prod-lti-moumouton") %}
    {% set sendFiles = '10,25,40,55' %}
    {% set Status = '6,15,24,33,42,51' %}
{% endif %}

```

Explication : on commence par une boucle if pour dire que si le serveur il commence avec “???-lti-???” on va appliquer la boucle. Dans la boucle on retrouve 2 variables : sendFiles et Status ont l'on va retrouver les temps demandés. On termine avec la fin de la boucle if

Ensuite on va faire nos cron jobs pour chacune des demandes.

```

# BRIGHTSPACE
{{ container_name }}-cron-sendFiles_brightspace:
  cron.present:
    - identifier : sendFiles_brightspace
    - name : docker exec -i {{ container_name }} sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/sendFiles.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
    - minute : {{ sendfiles }}

{{ container_name }}-cron-getStatus_brightspace:
  cron.present:
    - identifier : getStatus_brightspace
    - name : docker exec -i {{ container_name }} sh -c "php /home/sites/lti/www/brightspace/app/cronTasks/getStatus.php" >> /home/sites/logs/logs_appli/brightspace_cron.log 2>&1
    - minute : {{ status }}

# CANVAS
{{ container_name }}-cron-updateStatus_canvas:
  cron.present:
    - identifier : updateStatus_canvas
    - name : docker exec -i {{ container_name }} sh -c "php /home/sites/lti/www/canvas/app/cron/updateStatus.php" >> /home/sites/logs/logs_appli/canvas_cron.log 2>&1
    - minute : {{ status }}

# TEAMS
{{ container_name }}-cron-sendFiles_teams:
  cron.present:
    - identifier : sendFiles_teams
    - name : docker exec -i {{ container_name }} sh -c "php /home/sites/lti/www/teams/sendFiles.php" >> /home/sites/logs/logs_appli/teams_cron.log 2>&1
    - minute : {{ sendfiles }}

# TMP
{{ container_name }}-cron-tmp_folder:
  cron.present:
    - identifier : tmp_folder
    - name : docker exec -i {{ container_name }} sh -c "find /home/sites/lti/www/tmp -maxdepth 1 -mmin +60 -exec rm {} \;" >> /home/sites/logs/logs_appli/tmp_cron.log 2>&1
    - minute : 0
    - hour : */1

```

Explication : On commence par spécifier la crontab puis on utilise cron.present afin de vérifier que la variable d'environnement spécifiée est présente dans la crontab pour l'utilisateur voulu. On retrouve 4 paramètres dans nos crontab :

1. Identifier : Il s'agit de l'identifiant de la crontab.
2. Name : Il s'agit de la commande qui doit être exécutée par la tâche cron.
3. Minute : Il peut s'agir de n'importe quelle chaîne prise en charge par le champ minute de votre système cron.

- Hour : Il peut s'agir de n'importe quelle chaîne prise en charge par le champ heure de votre système cron.

Pour finir, on va state.apply sur les 4 serveurs et regarder sur les 4 serveurs les crontab avec la commande crontab -l.

```
# SALT_CRON_IDENTIFIER:getStatus_brightspace
3,12,21,30,39,48,57 * * * * docker exec -i php_apache_lti sh -c "php /home/li/brightspace_cron.log 2>&1
# SALT_CRON_IDENTIFIER:updateStatus_canvas
3,12,21,30,39,48,57 * * * * docker exec -i php_apache_lti sh -c "php /home/as_cron.log 2>&1
# SALT_CRON_IDENTIFIER:sendFiles_teams
5,20,35,50 * * * * docker exec -i php_apache_lti sh -c "php /home/sites/lti
```

Maintenant qu'on a fait les nouvelles crontab, on va pouvoir supprimer les anciennes car elles sont plus utiles. On fait un state.apply ensuite pour réafficher les crontab.

Tests fonctionnels

Playwright permet de faire des tests fonctionnels et d'automatiser des navigateurs avec une seule API. Ces tests se feront grâce à un code qui va être exécuté sur le navigateur pour automatiser les tests et on va faire en sorte que toutes les x minutes, les tests s'appliquent et alertent s'il y a un problème. La configuration sera faite par le développeur mais la mise en place pour les sysadmins. De plus, les tests se feront sur Chromium, Firefox et Webkit, les erreurs peuvent différer selon le navigateur, c'est-à-dire qu'on peut se retrouver avec un problème sur Firefox mais ne pas l'avoir sur Chromium.

Installation d'un serveur spécifique pour le logiciel

Tout d'abord, on doit préparer l'installation de Playwright, on va donc modifier des fichiers de configurations et en créer également pour Playwright.

Dans le top.sls de Salt on va ajouter la configuration de Playwright.

```
'trans-(monitoring|playwright)-':
    - match: pcre
    - sites.user_www-data
    - access.developpers
```

On va ensuite créer son répertoire dans le common et modifier le fichier init.sls de celui-ci.

```
{% if grains["id"].startswith("trans-playwright") %}
    - common.playwright
{% endif %}
```

Pour finir avec les fichiers de configuration, on va créer un fichier init.sls dans le répertoire de Playwright et on va le configurer de façon à ce qu'il installe nodejs et npm qui sont deux services pré-requis pour l'installation de Playwright.

```
#####
# install pré-requis
#####

{% set dist = grains["oscodename"] %}
{% set version_nodejs = "node_18.x" %}

{% set repos = {
    "playwright": "deb https://deb.nodesource.com/" + version_nodejs + " " + dist + " main" ,
    "playwright-src": "deb-src https://deb.nodesource.com/" + version_nodejs + " " + dist + " main" } %}

{% for name, repo in repos.items() %}
playwright-repo-{{ name }}:
  pkgrepo.managed:
    - name: {{ repo }}
    - file: /etc/apt/sources.list.d/{{ name }}.list
    - key_url: https://deb.nodesource.com/gpgkey/nodesource.gpg.key
{% endfor %}

#####
# install
#####

playwright-nodejs_install:
  pkg.latest:
    - name: nodejs
    - refresh: True

playwright-npm_configuration:
  cmd.run:
    - name: npm config set cache /tmp/ --global
    - runas: root
```

On va créer un nouveau serveur spécifique pour Playwright. On va donc modifier le script d'installation de base et appliquer les paramètres une fois que ce sera fait.

```

combos="trans-playwright-amovenus,ns3368398.ip-37-187-88.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//`"
    soyoustart_name=`echo $combo | sed 's/.*,//`"
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add || break
    #mfw server $compi_name add envs trans || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh || break
    msrv $compi_name $soyoustart_name --only_install --template mongodb-queuing || break
    echo
    echo
    echo
done

```

Lors de l'application des paramètres, on va faire un state.apply et je me suis retrouvé avec une erreur.

```

=====mer. 15 févr. 2023 10:27:35 CET - compi_name[trans-playwright-amovenus]
trans-playwright-amovenus:
  Data failed to compile:
-----
  Rendering SLS 'trans:common.playwright' failed: Jinja syntax error: expected token ',', got ':'; line 12

[...]
{% set dist = grains["oscodename"] %}
{% set version_nodejs = "node_18.x" %}

{% set repos = {
    "playwright" : "deb [signed-by=$key_url] https://deb.nodesource.com/" + version_nodejs + " " + dist + " main > /etc/apt/sources.list.d/nodesource.list"
    "playwright-src" : "deb-src [signed-by=$key_url] https://deb.nodesource.com/" + version_nodejs + " " + dist + " main >> /etc/apt/sources.list.d/nodesource.list"}      =====
=====
  %}

{% for name, repo in repos.items() %}
playwright-repo-{{ name }}:
  pkgrepo.managed:
[...]
-----
ERROR: Minions returned with non-zero exit code

```

La solution était qu'il fallait mettre une virgule dans le set repos car j'en ai mis deux et cela marche comme un tableau donc il fallait mettre la virgule après le premier repos.

On peut désormais poursuivre l'application des paramètres. Ensuite on va installer Playwright sur la machine en question.

```
npm init playwright@latest
```

Supprimer le serveur olddev de lti-dev.compilatio.net

Pour cela, il suffit juste de remove son input_role lti-dev.compilatio.net

```
root@ns3045309(trans-saltmaster-absol):~# mfw server olddev-lti-golgopathe remove input_roles lti-dev.compilatio.net
=====
OrderedDict([('hostname', 'olddev-lti-golgopathe'),
 ('compi_name', 'olddev-lti-golgopathe'),
 ('out_ips', ['141.94.173.68']),
 ('envs', ['olddev']),
 ('input_roles', ['nrpe', 'munin_node', 'ssh', 'noswap']),
 ('output_roles', []),
 ('managed_ips', []),
 ('backup', {'ignores': []}),
 ('creation_utc',
  datetime.datetime(2022, 2, 18, 14, 11, 46, 37000)),
 ('update_utc', datetime.datetime(2022, 2, 21, 9, 49, 21, 333000))])
```

Avec la commande nlookup on peut regarder s'il a bien été enlevé.

```
root@ns3045309(trans-saltmaster-absol):~# nslookup lti-dev.compilatio.net
Server:      8.8.8.8
Address:     8.8.8.8#53

Non-authoritative answer:
Name:   lti-dev.compilatio.net
Address: 141.94.173.68
Name:   lti-dev.compilatio.net
Address: 217.182.174.138
```

On retrouve des adresses IP on va pouvoir faire la même commande pour savoir le nom des serveurs ayant pour IP celles qui sont apparues.

```
root@ns3045309(trans-saltmaster-absol):~# nslookup 141.94.173.68
68.173.94.141.in-addr.arpa      name = olddev-lti-golgopathe.compilatio.net.

Authoritative answers can be found from:
```

Il faut attendre quelques minutes afin que ça disparaisse.

Participation à un Planning Poker

Le Planning Poker est une technique que les équipes de développement utilisent pour deviner l'effort des tâches de gestion de projet. Ces estimations sont basées sur la contribution et le consensus de l'ensemble du groupe, ce qui les rend plus engageantes et précises. Pour aider à évaluer le nombre de points pour les tâches pertinentes, les équipes utilisent des cartes de poker de planification. Le fonctionnement est simple : une personne lit à haute voix une user story, nous avons tous des cartes avec des points (prenons par exemple 50 ça va être 3 semaines environ) et donc selon la demande dans la user story on attribue tous la carte correspondant au temps que l'on estime pour finaliser cette tâche. On prendra le nombre de points sur la carte qui ressortira le plus de fois. De plus, une personne qui

met beaucoup plus ou beaucoup moins de points pourra se justifier pour éventuellement changer l'avis des autres. Debug sur des problèmes de création de VMs

Problème de création de VMs lié à OVH (Incendie + Ransomware + DDOS)

 Le Monde Informatique

[Incendie SGB2 Strasbourg : OVH condamné à verser plus de 100 000 €](#)

Il y a 2 jours

 Le Monde Informatique

[Un ransomware attaque les clients ESXi des hébergeurs français \(MAJ\)](#)

Il y a 2 jours

OVH a subi un incendie et un ransomware le weekend du 03/02 au 06/02. Cela a provoqué beaucoup de problèmes au niveau des entreprises qui sont chez OVH comme Compilatio. Le problème ici est qu'on utilise pas mal de VM pour apporter plus d'efficacité dans les analyses. Avec les incidents d'OVH, la création de VMs était difficile car l'API avait des soucis et il avait des maintenances. Il était impossible de voir l'avancée de la création des machines et de voir d'où venait le problème lors de la création de celui-ci. On ne pouvait également pas faire un redémarrage hard sur les machines.

Enquêter

Heure de début : 06/02/2023 10:53 UTC

Impact sur le service : Le panel client ou les utilisateurs de api.ovh.com peuvent avoir des lenteurs ou des erreurs lors de la navigation.

Actions en cours : Enquête

Nous avons détecté une remontée d'erreurs sur api.ovh.com et le panel client. Notre équipe travaille à restaurer le service La mise à jour sera publiée au fur et à mesure que des progrès significatifs seront réalisés.

Posté il y a 1h . 06 février 2023 - 11:26 UTC _

Cet incident affecte : Panneau de configuration et API || api.ovh.com (UE).

Après que OVH ait résolu le problème avec l'API, on a pu reprendre correctement les création des VMs manuellement en forçant la création de celles-ci.

[Global][API] - Instabilités de l'API

Rapport d'incident pour le service client

[Abonnez-vous à ce rapport](#)

Surveillance

Plus aucune instabilité n'est signalée.

Nous surveillons la situation en ce moment pour une enquête plus approfondie.

Posté il y a 1h . 06 février 2023 - 11:46 UTC _

Pour finir, OVH a subi des DDOS qui nous empêchait de créer des VMs rapidement donc même si le problème de l'API était réglé, la création des VMs restait tout de même super lente surtout quand on faisait le state.apply (apt update et apt install).

Upgrade vers Ubuntu 20.04 du cluster qa MongoPerf

Installation des nouveaux serveurs qa-mongoperf

Modification du script d'installation + installation

```
combos="qa-mongoperf-paragruel,ns3063319.ip-37-187-135.eu qa-mongoperf-farfurex,ns3019263.ip-178-33-123.eu"

#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//`'
    soyoustart_name=`echo $combo | sed 's/.*/`'
    echo "===== $(date) - $compi_name $soyoustart_name :"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add $compi_name || break
    #mfw server $compi_name add envs qa || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongoperf-cluster redis-cluster || break
    #msrv $compi_name $soyoustart_name --only_install --template mongo-majo-perf || break
    echo
    echo
    echo
done
```

Une fois cela fait, on va attendre que l'installation se fasse et pour vérifier que les serveurs soient bien présents, on va utiliser la commande salt-key.

```

root@ns3045309(trans-saltmaster-absol):~# salt-key -L | grep qa-mongoperf
qa-mongoperf-cacnea
qa-mongoperf-cacturne
qa-mongoperf-tylton
qa-mongoperf-farfurex
qa-mongoperf-paragruel

#
# Accepte des nouveaux serveurs
#
for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/$/'`"
    echo "===== $(date) - compi_name[$compi_name] soyoustart_name[$soyoustart_name] :"
    salt-key --accept=$compi_name || break
    echo
    echo
    echo
    echo
done

```

Génération des certificats

```

-----
Génération des certificats
-----

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - compi_name[$compi_name]"
    /home/sys/admin_scripts/gen_certif_to_pillar/gen_certif_to_pillar.sh $compi_name
done

```

On va vérifier que les certificats sont bien présents.

```

root@ns3045309(trans-saltmaster-absol):~# ll /srv/pillar/qa/certifs | grep qa-mongoperf
drwx----- 2 root root 4096 janv. 27 2020 qa-mongoperf-cacnea/
drwx----- 2 root root 4096 janv. 27 2020 qa-mongoperf-cacturne/
drwx----- 2 root root 4096 févr. 8 11:50 qa-mongoperf-farfurex/
drwx----- 2 root root 4096 févr. 8 11:50 qa-mongoperf-paragruel/
drwx----- 2 root root 4096 janv. 27 2020 qa-mongoperf-tylton/

```

Ajout des nouvelles machines dans les fichiers de configuration

Ajout dans le fichier de configuration MongoDB

```

members:
  - rs0:
      - "qa-mongoperf-cacnea:27018"
      - "qa-mongoperf-cacturne:27018"
      - "qa-mongoperf-tylton:27018"
      - "qa-mongoperf-farfurex:27018"
      - "qa-mongoperf-paragruel:27018"
    # Arbiters
      - "qa-arbiter-embrochet:26022"
configReplSet:
  - "qa-mongoperf-cacnea:27019"
  - "qa-mongoperf-cacturne:27019"
  - "qa-mongoperf-tylton:27019"
  - "qa-mongoperf-farfurex:27019"
  - "qa-mongoperf-paragruel:27019"
  # Arbiters
  - "qa-arbiter-embrochet:19022"

```

Ajout dans le fichier de configuration Redis

```

addresses:
main:
  - "qa-queuing-cerbyllin:6379"
  - "qa-queuing-sylveroy:6379"
  - "qa-arbiter-embrochet:6379"
  - "qa-mongoperf-cacnea:6379"
  - "qa-mongoperf-cacturne:6379"
  - "qa-mongoperf-tylton:6379"
  - "qa-mongoperf-farfurex:6379"
  - "qa-mongoperf-paragruel:6379"

```

Application des paramètres

```

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    echo "===== $(date) - $compi_name[$compi_name]"
    #saltc $compi_name test.ping
    #saltc $compi_name saltutil.refresh_pillar
    #saltc $compi_name state.apply
    #saltc $compi_name state.apply
    #salt $compi_name cmd.run "reboot"
    #salt $compi_name cmd.run "uptime"
    #salt $compi_name cmd.run "uname -a"
    #salt $compi_name cmd.run "iptables -vnL | grep DROP"
    #salt $compi_name cmd.run "ip6tables -vnL | grep DROP"
done

```

Ajout des nouveaux serveurs et suppression des anciens dans le cluster MongoPerf

Ajout des noeuds pour les nouveaux serveurs dans les clusters Config Replica Set et Replica Set 0

Avant de commencer, on va devoir vérifier le statut des serveurs car il est important de savoir quel serveur est primaire dans les 2 clusters.

```

"name" : "qa-mongoperf-cacnea.compilatio.net:27019",
"stateStr" : "PRIMARY",
"name" : "qa-mongoperf-cacturne.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-tylton.compilatio.net:27019",
"stateStr" : "SECONDARY",

```

On va dorénavant ajouter les 3 serveurs dans le Config Replica Set un par un et en même temps vérifier le statut pour voir si le serveur ajouter dans le cluster à bien récupéré le rôle de Secondary.

```

--sur un des mongoperf (configreplicaset)
mongoperf_configReplSet
    rs.add("qa-mongoperf-paragruel.compilatio.net:27019")
    rs.add("qa-mongoperf-farfurex.compilatio.net:27019")
    rs.addArb("qa-arbiter-embrochet.compilatio.net:19022")

```

Lorsque j'ai voulu ajouter le serveur Arbitre dans le cluster Config Replica Set j'ai fait une erreur.

```
configReplSet:PRIMARY> rs.addArb("qa-arbiter-embrochet.compilatio.net:19022")
{
    "operationTime" : Timestamp(1675854826, 1),
    "ok" : 0,
    "errmsg" : "Arbiters are not allowed in replica set configurations being used fo
r config servers",
    "errInfo" : null
}
```

La solution est que pour le cluster Config Replica Set on utilise la commande rs.add.

```
configReplSet:PRIMARY> rs.add("qa-arbiter-embrochet.compilatio.net:19022")
```

On peut maintenant vérifier qu'ils y sont bien.

```
"name" : "qa-mongoperf-cacnea.compilatio.net:27019",
"stateStr" : "PRIMARY",
"name" : "qa-mongoperf-cacturne.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-tylton.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-paragruel.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-farfurex.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-arbiter-embrochet.compilatio.net:19022",
"stateStr" : "SECONDARY",
```

On vient d'ajouter les serveurs dans le Config Replica Set maintenant on va répéter la même chose pour le Replica Set 0.

```
--sur un des mongoperf (rs0)
mongoperf_rs0
    rs.add("qa-mongoperf-paragruel.compilatio.net:27018")
    rs.add("qa-mongoperf-farfurex.compilatio.net:27018")
    rs.addArb("qa-arbiter-embrochet.compilatio.net:26022")
```

Puis la vérification de leur statut.

```
"name" : "qa-mongoperf-cacnea.compilatio.net:27018",
"stateStr" : "PRIMARY",
"name" : "qa-mongoperf-cacturne.compilatio.net:27018",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-tylton.compilatio.net:27018",
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-paragruel.compilatio.net:27018"
"stateStr" : "SECONDARY",
"name" : "qa-mongoperf-farfurex.compilatio.net:27018",
"stateStr" : "SECONDARY",
"name" : "qa-arbiter-embrochet.compilatio.net:26022",
"stateStr" : "ARBITER",
```

Prise en compte des ajouts par le reste de l'infrastructure

```
-----
Prise en compte des ajouts par le reste de l'infra
-----

--saltmaster
saltc 'qa-web*' state.apply -b 1
saltc 'qa-worker*' state.apply -b 1
saltc 'qa-ip*' state.apply
--Contrôles
saltc 'qa-web*' cmd.run "docker ps | grep mongos"
saltc 'qa-worker*' "docker ps | grep mongos"
saltc 'qa-ip*' "docker ps | grep mongos"
```

Suppression des noeuds pour les anciens serveurs dans le Config Replica Set et dans le Replica Set 0

On va ensuite supprimer les anciens serveurs du Config Replica Set et du Replica Set 0. Pour cela il faudra être vigilant car on doit tout d'abord enlève tous les serveurs secondaires puis utilise la commande rs.stepDown() pour faire passer le primaire en secondaire et avec un rs.status() on regarde si le rôle de primaire a été redistribué. Pour finir, on enlève le dernier serveur.

```
--sur un des mongoperf (configreplicaset)
mongoperf_configReplSet
    rs.status()

Lorsque les 3 nouveaux sont en "SECONDARY" :
rs.remove("qa-mongoperf-cacturne.compilatio.net:27019")
rs.remove("qa-mongoperf-tylton.compilatio.net:27019")
rs.stepDown()
rs.remove("qa-mongoperf-cacnea.compilatio.net:27019")

rs.status()

--sur un des mongoperf (rs0)
mongoperf_rs0
    rs.status()

Lorsque les 3 nouveaux sont en "SECONDARY" :
rs.remove("qa-mongoperf-cacturne.compilatio.net:27018")
rs.remove("qa-mongoperf-tylton.compilatio.net:27018")
rs.stepDown()
rs.remove("qa-mongoperf-cacnea.compilatio.net:27018")

rs.status()
```

On peut voir notre cluster final avec nos nouvelles machines et l'arbitre qui servira de relais.

```
"name" : "qa-mongoperf-paragruel.compilatio.net:27019",
"stateStr" : "PRIMARY",
"name" : "qa-mongoperf-farfurex.compilatio.net:27019",
"stateStr" : "SECONDARY",
"name" : "qa-arbiter-embrochet.compilatio.net:19022",
"stateStr" : "SECONDARY",

"name" : "qa-mongoperf-paragruel.compilatio.net:27018",
"stateStr" : "PRIMARY",
"name" : "qa-mongoperf-farfurex.compilatio.net:27018",
"stateStr" : "SECONDARY",
"name" : "qa-arbiter-embrochet.compilatio.net:26022",
"stateStr" : "ARBITER",
```

Suppression des anciennes machines MongoPerf

On va supprimer les anciens serveurs des fichiers de configuration car ils seront prochainement supprimés.

Sur le fichier de configuration de MongoDB

```
- rs0:
#           - "qa-mongoperf-cacnea:27018"
#           - "qa-mongoperf-cacturne:27018"
#           - "qa-mongoperf-tylton:27018"
#           - "qa-mongoperf-farfurex:27018"
#           - "qa-mongoperf-paragruel:27018"
# Arbiters
#           - "qa-arbiter-embrochet:26022"
configReplSet:
#           - "qa-mongoperf-cacnea:27019"
#           - "qa-mongoperf-cacturne:27019"
#           - "qa-mongoperf-tylton:27019"
#           - "qa-mongoperf-farfurex:27019"
#           - "qa-mongoperf-paragruel:27019"
# Arbiters
#           - "qa-arbiter-embrochet:19022"
```

Sur le fichier de configuration de Redis

```
addresses:
main:
- "qa-queuing-cerbyllin:6379"
- "qa-queuing-silveroy:6379"
- "qa-arbiter-embrochet:6379"
#
#           - "qa-mongoperf-cacnea:6379"
#           - "qa-mongoperf-cacturne:6379"
#           - "qa-mongoperf-tylton:6379"
#           - "qa-mongoperf-farfurex:6379"
#           - "qa-mongoperf-paragruel:6379"
```

Mise en arrêt des dockers ES

Pour les oldprod ElasticSearch

On s'est connecté sur une machine pour accéder à Kopf pour voir l'ancienne version qu'il utilisait à la place d'ElasticSearch.

The screenshot shows the Kopf interface with the following statistics:

- 42 nodes
- 3 indices
- 630 shards
- 30,413,197 docs

Indices listed:

- common_customer_v2 (shards: 10 * 3 | docs: 25,312 | size: 4.90GB)
- common_magister_v4 (shards: 100 * 3 | docs: 30,257,062 | size: 379.23GB)
- common_stadium_v3 (shards: 100 * 3 | docs: 130,823 | size: 5.06GB)

Filtering options include "filter indices by name", "closed (0)", "special (0)", and "data".

On a listé les machines qui ne servaient à rien (ancienneté + prix) et on a stoppé les dockers sur celles-ci.

```
root@s3044748:~# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
0984ca16bd64 dockerhub.compilatio.net/haproxy:2017-02-03_15h01m28s "/docker-entrypoint..." 4 minutes ago Up 4 minutes 80/tcp elasticsearch-http_proxy
361a87a5ff0b elasticsearch:5.6.16 "/docker-entrypoint..." 2 months ago Up 2 months 9200/tcp oldprod-elasticsearch-data_node
0d339c53b65 dockerhub.compilatio.net/elasticsearch/kopf:2017-08-28_14h15m08s "/root/start.sh" 2 years ago Up 10 months 9200/tcp elasticsearch_kopf
root@s3044748:~# docker stop elasticsearch
Error: No such container: elasticsearch
root@s3044748:~# docker stop elasticsearch-http_proxy
root@s3044748:~# docker stop oldprod-elasticsearch-data_node
oldprod-elasticsearch-data_node
root@s3044748:~# docker stop elasticsearch_kopf
elasticsearch_kopf
root@s3044748:~#
```

Pour les prod ElasticSearch

Comme on a acheté et installé des nouveaux serveurs plus performants et avec plus de stockage on va pouvoir supprimer des serveurs ES dans la prod pour faire plus d'économie.

Les tâches sont les mêmes que lorsque l'on prend des serveurs ES pour faire l'installation de nouveaux serveurs dans d'autres environnements : mise en arrêt des dockers, modification du fichiers de configuration ElasticSearch afin de les enlever puis suppression des référentiels Salt et Firewall.

Pour commencer, on va stopper tous les dockers de toutes les machines en prenant en compte la répartition des charges sur Kibana donc on attend que ce soit Healthy. On va également enlever l elasticsearch_http dans le Firewall afin de ne pas avoir d'alertes inutiles.

```

root@ns3045309(trans-saltmaster-absol):~# mfw server prod-elasticsearch-bazoucan remove
input_roles elasticsearch_http
=====
OrderedDict([('hostname', 'ns3133103.ip-51-75-241.eu'),
 ('compi_name', 'prod-elasticsearch-bazoucan'),
 ('out_ips', ['51.75.241.16']),
 ('envs', ['prod']),
 ('input_roles',
  ['nrpe', 'munin_node', 'ssh', 'elasticsearch_transport']),
 ('output_roles', []),
 ('managed_ips', []),
 ('backup', {'ignores': []}),
 ('creation_utc',
  datetime.datetime(2022, 4, 14, 8, 28, 58, 861000)),
 ('update_utc', datetime.datetime(2022, 4, 14, 8, 29, 0, 418000))])

```

Ensuite on va modifier sur le fichier de configuration et on va mettre toutes les machines dont on a stopper les dockers en commentaire.

Pour finir, on va faire une variable comprenant tous les serveurs que l'on souhaite enlever puis on va faire un boucle avec les commandes permettant de supprimer les référentiels Salt et Firewall.

Réaffectation des ressources système afin d'améliorer les performances du cluster MongoDB

Suppression des anciennes machines ElasticSearch

Comme on l'a déjà vu, on va faire prendre quelques serveurs ElasticSearch car il y en a beaucoup, on va les supprimer et on va installer des nouveaux serveurs dans l'environnement MongoDB.

A commencer par la sortie du cluster en stoppant les dockers tout en vérifiant sur Kibana que les données se répartissent bien avant de stopper un autre serveur.

```

root@prod-elasticsearch-blancoton():~# docker stop prod-elasticsearch-data_node
prod-elasticsearch-data_node
root@prod-elasticsearch-blancoton():~# docker stop kibana
kibana
root@prod-elasticsearch-blancoton():~# docker stop elasticsearch-http_proxy
elasticsearch-http_proxy

```

S...	Nodes	Indices	JVM Heap	Total shards	Unassigned shards
●	68	95	360.2 GB / 962.0 GB	3402	49

Puis on termine cette partie par la suppression du référentiel Salt et Firewall.

Réinstallation des serveurs ElasticSearch en serveurs MongoDB

On commence tout d'abord par l'installation de base : installation des machines, vérification de leurs présences, modification des fichiers de configuration, applications des paramètres.

```
combos="prod-mongodb-blancoton,ns3067901.ip-217-182-174.eu prod-mongodb-grimalin,ns3081961.ip-145-239-66.eu prod-mongodb-tutetekri,ns3083855.ip-145-239-10.eu prod-mongodb-vorasterie,ns3207138.ip-37-187-134.eu prod-mongodb-manglouton,ns3133106.ip-51-75-241.eu prod-mongodb-hastacuda,ns3075034.ip-217-182-174.eu prod-mongodb-krakos,ns3081549.ip-145-239-66.eu prod-mongodb-volttoutou,ns3128403.ip-54-38-92.eu prod-mongodb-rubombelle,ns3621730.ip-149-202-84.eu prod-mongodb-minisange,ns315846.ip-37-187-29.eu"

for combo in $combos
do
    compi_name=`echo $combo | sed 's/,.*//'`
    soyoustart_name=`echo $combo | sed 's/.*/'`"
    echo "===== $(date) - $compi_name $soyoustart_name"
    #msrv $compi_name $soyoustart_name --only_dns || break
    #msrv $compi_name $soyoustart_name --only_reverse || break
    #mfw server $soyoustart_name add $compi_name || break
    #mfw server $compi_name add envs prod || break
    #mfw server $compi_name add input_roles nrpe munin_node ssh mongodb-cluster || break
    #msrv $compi_name $soyoustart_name --only_install --template mongodb-queuing --
    force_reinstall || break
    echo
    echo
    echo
    echo
done
```

```
members:
  - rs0:
      - "prod-mongodb-sovkipou:27018"
      - "prod-mongodb-sarmurai:27018"
      - "prod-mongodb-bacabouh:27018"
      - "prod-mongodb-blancoton:27018"
      - "prod-mongodb-grimalin:27018"
      - "prod-mongodb-voltoutou:27018"
      # Delayed nodes
      - "prod-mongodb-tokopiyon:27018"
      - "prod-mongodb-tokotoro:27018"
      - "prod-mongodb-vorasterie:27018"
      - "prod-mongodb-manglouton:27018"
    - rs1:
        - "prod-mongodb-quartermac:27018"
        - "prod-mongodb-concombaffe:27018"
        - "prod-mongodb-trepassable:27018"
        - "prod-mongodb-hastacuda:27018"
        - "prod-mongodb-krakos:27018"
        - "prod-mongodb-tutetekri:27018"
        # Delayed nodes
        - "prod-mongodb-tokopisco:27018"
        - "prod-mongodb-cosmog:27018"
        - "prod-mongodb-rubombelle:27018"
        - "prod-mongodb-minisange:27018"
  configReplSet:
    - "prod-mongodb-sovkipou:27019"
    - "prod-mongodb-sarmurai:27019"
    - "prod-mongodb-bacabouh:27019"
    - "prod-mongodb-blancoton:27019"
    - "prod-mongodb-grimalin:27019"
    - "prod-mongodb-voltoutou:27019"
```

Mise en place des clusters de MongoDB

Ajout des serveurs dans les clusters

Ensuite on va ajouter les serveurs dans le cluster Config Replica Set, Replica Set 0 et Replica Set 1.

On va commencer par ajouter les serveurs dans le Config Replica Set.

```
-----  
Ajout des noeuds au mongodb_configReplSet  
-----  
--sur un des mongodb (configreplicaset)  
mongodb_configReplSet  
    rs.add("prod-mongodb-blancoton.compilatio.net:27019")  
    rs.add("prod-mongodb-grimalin.compilatio.net:27019")  
    rs.add("prod-mongodb-volttoutou.compilatio.net:27019")
```

Ensuite on ajoute dans les 2 autres clusters.

```
-----  
Ajout des noeuds au mongodb_rs0 + Delayed server  
-----  
--sur un des mongodb (rs0)  
mongodb_rs0  
    rs.add("prod-mongodb-blancoton.compilatio.net:27018")  
    rs.add("prod-mongodb-grimalin.compilatio.net:27018")  
    rs.add("prod-mongodb-volttoutou.compilatio.net:27018")  
  
## Delayed server  
  
    rs.add("prod-mongodb-vorasterie.compilatio.net:27018")  
    rs.add("prod-mongodb-manglouton.compilatio.net:27018")  
    rs.status()
```

```
-----  
Ajout des noeuds au mongodb_rs1 + Delayed server  
-----  
--sur un des mongodb (rs1)  
mongodb_rs1  
    rs.add("prod-mongodb-hastacuda.compilatio.net:27018")  
    rs.add("prod-mongodb-krakos.compilatio.net:27018")  
    rs.add("prod-mongodb-tutetekri.compilatio.net:27018")  
  
## Delayed server  
  
    rs.add("prod-mongodb-rubombelle.compilatio.net:27018")  
    rs.add("prod-mongodb-minisange.compilatio.net:27018")  
    rs.status()
```

Ajout des Delayed Server dans les clusters Replica Set 0 et Replica Set 1

Par la même occasion, on va ajouter 2 Delayed Server : un Delayed Server de 12h et un de 7j. On a déjà ajouté les serveurs, il nous suffit juste de les configurer.

On va chercher sur les Delayed Server le cfg qui leur correspond.

```
--mettre le serveur en delayed nodes
cfg = rs.conf()

# Chercher quel cfg correspond au serveur delayed n°1 pour 12h
cfg.members[]
```

Ensuite on va modifier leurs configuration à partir de cfg.

```
# Chercher quel cfg correspond au serveur delayed n°1 pour 12h
cfg.members[]

cfg.members[4].priority = 0
cfg.members[4].votes = 0
cfg.members[4].hidden = true
cfg.members[4].slaveDelay = 3600*12
cfg.members[4]

cfg
rs.reconfig(cfg)
rs.conf()

# Chercher quel cfg correspond au serveur delayed n°2 pour 7j
cfg.members[5]

cfg.members[5].priority = 0
cfg.members[5].votes = 0
cfg.members[5].hidden = true
cfg.members[5].slaveDelay = 3600*24*7
cfg.members[5]

cfg
rs.reconfig(cfg)
rs.conf()
```

Avant de faire la suppression des anciens serveurs, il ne faudra pas oublier de faire la prise en compte de l'infrastructure.

Suppression des anciens serveurs MongoDB

On va désormais supprimer les anciens serveurs MongoDB. Premièrement, on va regarder quel serveur est primaire avec la commande rs.status(). Ensuite on va supprimer les serveurs secondaires et faire un rs.stepDown() afin de supprimer le dernier serveur.

Warning : Les autres serveurs doivent être enlevés quelques jours après à cause de l'Oplog et les serveurs delayed plus tard encore car ils stockent des données après 12h et 7j.

Création de permalink court pour les principales informations légales

L'objectif ici est de faire des redirections afin de simplifier le partage et l'utilisation de l'accès à ces informations. On va devoir faire plusieurs permalink pour répondre aux besoins de la User Story.

Mise en place des règles de redirection

Tout d'abord, on va chercher le fichier qui correspond aux redirections. Pour cela, on va se connecter sur un des 2 Proxys puis on va regarder dans /etc/apache2 afin de retrouver les configurations du service.

Dans le répertoire /sites-enabled on retrouve le fichier front.conf et c'est dans celui-ci que l'on va mettre nos redirections.

On va reprendre les commandes que l'on trouve dans le fichier et on va les modifier de façon à ce qu'il réponde à notre besoin.

```
# Redirections US 18427
ProxyPass /legal https://www.compilatio.net/en/legal-info
ProxyPassReverse /legal https://www.compilatio.net/en/legal-info

ProxyPass /terms-of-service-magister https://content.compilatio.net/documents/EN_COMPILATIO_MAGISTER-COPYRIGHT_Terms-of-Services_2021-27-10.pdf
ProxyPassReverse /terms-of-service-magister https://content.compilatio.net/documents/EN_COMPILATIO_MAGISTER-COPYRIGHT_Terms-of-Services_2021-27-10.pdf

ProxyPass /terms-of-service-copyright https://content.compilatio.net/documents/EN_COMPILATIO_MAGISTER-COPYRIGHT_Terms-of-Services_2021-27-10.pdf
ProxyPassReverse /terms-of-service-copyright https://content.compilatio.net/documents/EN_COMPILATIO_MAGISTER-COPYRIGHT_Terms-of-Services_2021-27-10.pdf

ProxyPass /terms-of-service-studium https://content.compilatio.net/documents/EN_STUDIUM_Terms-and-Conditions.pdf
ProxyPassReverse /terms-of-service-studium https://content.compilatio.net/documents/EN_STUDIUM_Terms-and-Conditions.pdf
```

Etant donné que certaines redirections ne marchaient pas dû au changement de nom de domaine, on a dû modifier les redirections pour rediriger vers la page directement. Le problème avec les liens ayant un autre nom de domaine est que Zendesk n'autorise qu'un seul domaine customisé.

```
# Redirections US 18427
RewriteEngine On

RewriteCond %{REQUEST_URI} /privacy [NC]
RewriteRule (.*) https://support.compilatio.net/hc/en/articles/4416953631505 [R=301,L]

RewriteCond %{REQUEST_URI} /security [NC]
RewriteRule (.*) https://support.compilatio.net/hc/en-us/articles/360001766858 [R=301,L]

RewriteCond %{REQUEST_URI} /gdpr [NC]
RewriteRule (.*) https://support.compilatio.net/hc/en/articles/360010397497 [R=301,L]
```

Les éléments à savoir pour comprendre les commandes :

On utilise tout d'abord le RewriteEngine pour autoriser les commandes Rewrite. Ensuite, on met le RewriteCond afin de donner notre condition ; le flag [NC] "NoCase" correspond au fait qu'il soit insensible à la casse. La commande RewriteRule permet de stipuler notre règle ; le flag [R=301, L] signifie que R (Redirect) va rediriger vers le navigateur de façon permanente (301) et L (Last) va arrêter le traitement du jeu de règles.

Pour finir, on va faire un systemctl reload apache2 sur les deux serveurs pour mettre à jour Apache sans avoir à le relancer.

 compilatio.net/privacy

 Politique de protection des données personnelles – Centre d'aide **Compilatio.net** - [compilatio.net/privacy](https://support.compilatio.net/hc/en/articles/4416953631505)

Nettoyage des disques sur des vieilles machines

Mise en place d'une clé USB Bootable

Pour faire la clé USB bootable, on va utiliser Rufus. c'est un utilitaire fiable de formatage USB. Rufus vous permet de formater n'importe quel support de stockage pour en modifier le format de fichier. Mais sa toute première utilité c'est la création de périphérique Bootable

On va brancher une clé USB vide, télécharger l'ISO de Shredos qui est un outil permettant de nettoyer les disques durs de façon clean afin que personne ne puisse récupérer des données dans les disques.

Dans l'interface Rufus, on va retrouver notre périphérique, on va choisir l'ISO dans SELECTION : Ce PC/Utilisateurs/Mattéo/Bureau/Shredos.img et on va démarrer.

Démarrer la vieille machine et brancher la clé USB dessus. On remarque que la machine est en train de booter avec Shredos puis on a plus qu'à attendre que le chargement se finisse et c'est bon.

