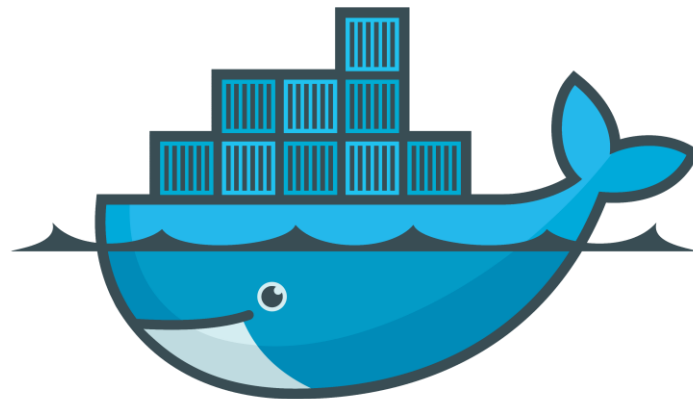
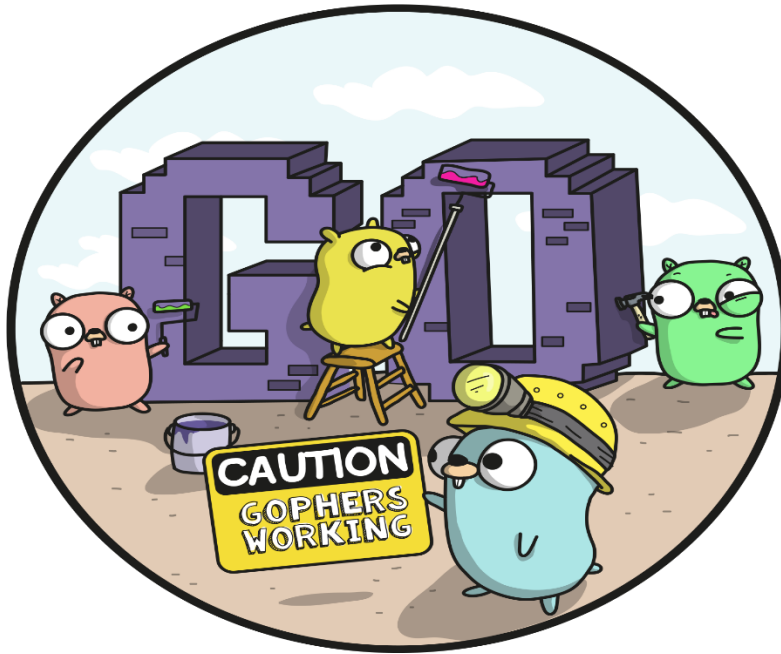


Projet BLOG (partie Docker) :



docker

Tables des matières

Mise en place de Vault afin de garantir la sécurité de nos variables.....	3
Mise en place de Traefik pour le LoadBalancing de nos serveurs web.....	4
Création et structure des différents Dockerfiles (Serveurs Web / Bases de données).....	5
Packages + dépendances des serveurs web.....	5
Éléments de configuration des serveurs web	5
Packages + dépendances des bases de données	5
Éléments de configuration des serveurs de base de données.....	6
Mise en place du registry privée	6
Authentification au registry	6
Mise en place de galera pour une réplication triples masters des bases de données	6
Mise en place à l'aide du plugin hyperdb pour assurer le FailOver	8
Connexion au Blog avril.....	8
Exposition des difficultés rencontrées durant le projet.....	9

Mise en place de Vault afin de garantir la sécurité de nos variables

Nous avons choisi Vault afin de permettre l'ajout de variables qui seront récupérées par les différents containers.

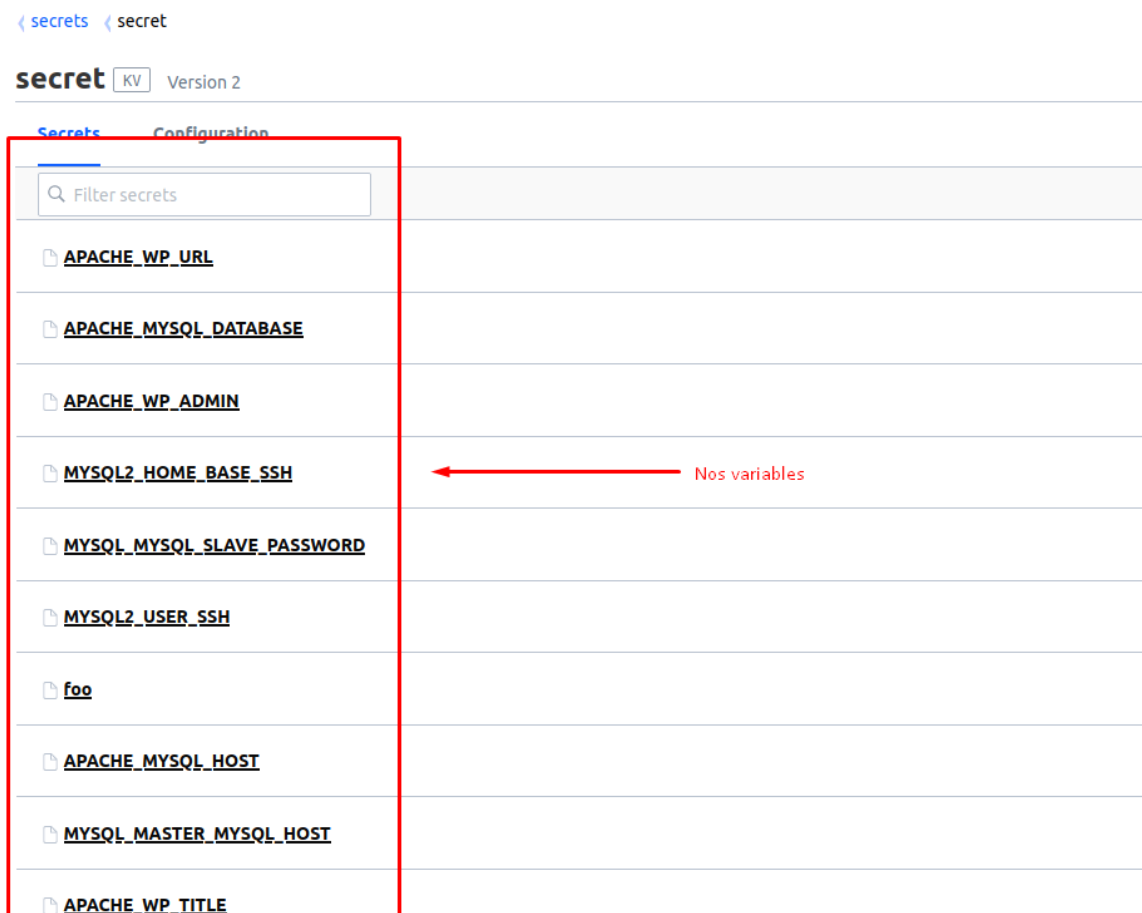
Variables dites « sensibles » et qui concernera tout ce qui est mot de passe de base de données avec des données de configuration pour notre site Wordpress.

Ces variables seront récupérées à l'aide de requêtes curl couplées à des filtres afin de lire les données en format .json sur Vault.

Afin de pouvoir récupérer ces données de manière automatique, il a été mis en place un script permettant l'export de la clé root de Vault afin que les requêtes d'interrogation des serveurs web vers Vault puissent se faire sans trop de difficultés.

On peut accéder à Vault au lien suivant : <http://0.0.0.0:8200/ui/vault/auth> et le mot de passe sera la clé root présente au dossier /scripts de chaque conteneurs web ou /scripts-conf des serveurs de bases de données.

Les variables seront ainsi contenues au sein de Vault et visible comme sur la capture ci-dessous une fois connecté à l'interface de Vault :

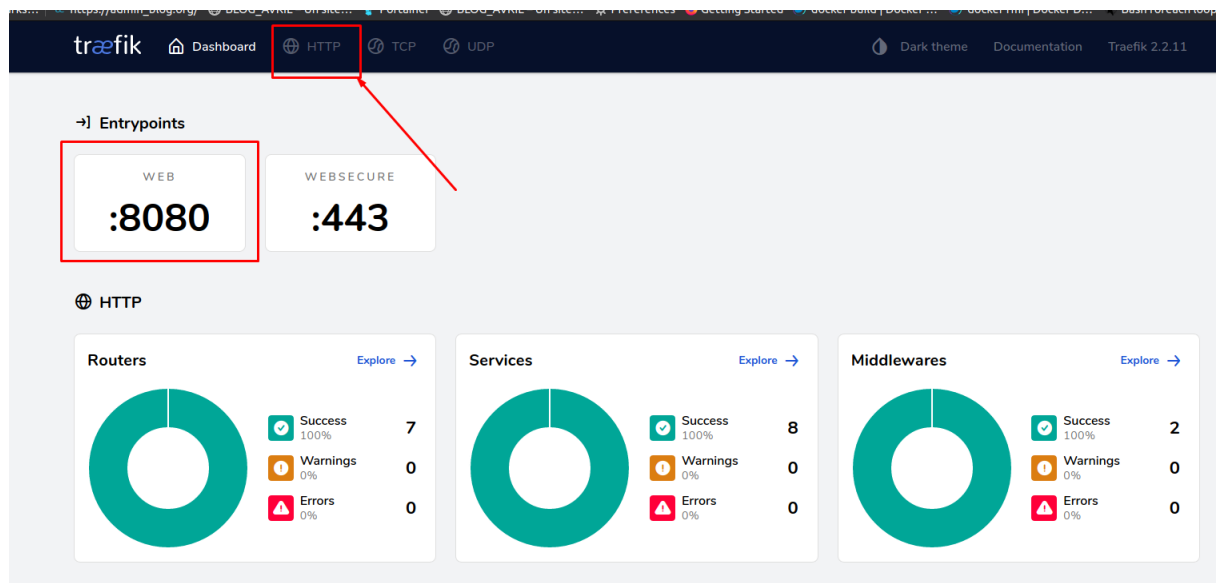


Mise en place de Traefik pour le LoadBalancing de nos serveurs web

Comme solution de LoadBalancing, nous avons choisi de nous tourner vers une solution de reverse proxy et par conséquent, nous avons opté pour Traefik qui permet d'effectuer du load balancing de nos serveurs web vers notre site Wordpress afin de fournir de la haute disponibilité en cas ou 1 voire 2 serveurs web venaient à être indisponible.

On peut accéder à Traefik avec le lien suivant : https://admin_blog.org/ avec comme login **admin** et password **wordpress**.

Une fois sur l'interface nous pouvons voir que nos trois serveurs apaches sont bien configurés et disponible pour être gérés en load balancing par Traefik sur le port 8080 :



The screenshot shows the 'HTTP Routers' section of the Traefik dashboard. It displays a table with 7 rows of routers. The first three rows are highlighted with a red box, showing the configuration for the three web servers.

Status	TLS	Rule	Entrypoints	Name	Service	Provider
✓	✓	Host('www.wordpressblogavril.org')	web websecure	apache-php_host1@docker	apache-php-host1-docker	🔗
✓	✓	Host('www.wordpressblogavril.org')	web websecure	apache-php_host2@docker	apache-php-host2-docker	🔗
✓	✓	Host('www.wordpressblogavril.org')	web websecure	apache-php_host3@docker	apache-php-host3-docker	🔗
✓	✓	Host('admin_blog.org')	websecure	api@file	api@internal	🔗
✓	✓	Host('portainer')	web websecure	portainer@docker	portainer	🔗
✓	✓	Host('traefik-traefik')	web websecure	traefik-traefik@docker	traefik-traefik	🔗
✓	✓	HostRegexp('{host:.+}')	web	web-to-websecure@internal	noop@internal	🔗

Si nous venons à arrêter un de nos serveurs web, il disparaîtra automatiquement de Traefik comme montré sur la capture ci-dessous :

Status	TLS	Rule	Entrypoints	Name	Service	Provider
✓	●	Host('www.wordpressblogavril.org')	web websecure	apache-php_host2@docker	apache-php-host2-docker	☁
✓	●	Host('www.wordpressblogavril.org')	web websecure	apache-php_host3@docker	apache-php-host3-docker	☁
✓	●	Host('admin_blog.org')	websecure	api@file	api@internal	🏠
✓		Host('portainer')	web websecure	portainer@docker	portainer	☁
✓		Host('traefik-traefik')	web websecure	traefik-traefik@docker	traefik-traefik	☁
✓		HostRegexp('host.+')	web	web-to-websecure@internal	noop@internal	∞

Même chose avec deux serveurs web.

Avec cette solution, nous assurons ainsi la haute disponibilité au niveau des services web.

Création et structure des différents Dockerfiles (Serveurs Web / Bases de données)

Afin de mettre nos serveurs web et bases de données en place, nous avons créés un dockerfile avec toutes les dépendances requises + éléments de configuration afin que le blog soit au maximum opérationnel.

Packages + dépendances des serveurs web

- **apache2** (serveurs web)
- **php libapache2-mod-php php-mysql** (modules php)
- **wget** (permet d'effectuer le téléchargement de sources)
- **unzip** (permet de décompresser les archives téléchargées)
- **iputils-ping** (permet de faire des tests de connectivités)
- **nano** (permet d'éditer certains fichiers de configuration)
- **curl** (permet d'effectuer des requêtes sur nos différents services, utiles pour vault notamment)
- **mariadb-client** (permettra d'interagir avec notre base de données sous mariadb)
- **jq** (permet d'interagir avec vault en format .json)

Éléments de configuration des serveurs web

Il était nécessaire de copier au sein du Dockerfile des scripts en bash afin d'automatiser certaines actions voulant limiter au maximum les interactions humaines, s'en suit également les éléments de configuration telles que :

- La clé d'authentification Vault permettant de récupérer certaines variables
- L'écoute des serveurs web du port 8080 nécessaire pour le load balancing et pour le site Wordpress
- La copie + configuration du fichier de configuration permettant la redirection automatique vers wordpress lorsque l'on saisi l'adresse <http://www.wordpressblogavril.org:8080/> dans le navigateur.
- Le script de démarrage en fonction du serveur web avec le lancement d'Apache en tant que daemon de sorte que le service soit par défaut chargé lors du démarrage du container.

Packages + dépendances des bases de données

- **iputils-ping**
- **curl**

- **jq**
- **nano**
- **openssh-client**
- **ssh**
- **sshpas**

Les modules ssh ont été installés dans le cadre d'un transfert de backup .sql de la base de données sur nos deux autres hosts d'où la nécessité d'installer ces derniers.

Éléments de configuration des serveurs de base de données

Afin de limiter les interactions humaines, des scripts en bash ont été requis.

Les éléments de configuration ont dû être mis en place :

- La clé d'authentification Vault permettant de récupérer certaines variables lors du paramétrage de la base de données Wordpress
- L'écoute des bases de données en fonction des requêtes qu'elles reçoivent
- Le script de configuration qui permettra le paramétrage (création comprise) de la base de données avec la configuration nécessaire afin de créer le cluster Galera permettant d'accueillir une réplication 3 maitres de la base de données Wordpress

Mise en place du registry privée

Afin de pouvoir démarrer nos services depuis le docker-compose.yml, nous avons mis en place un registry privé basé sur une image officiel afin de permettre le tag de nos images en local basés sur le port 5000 donnant comme nom à nos images : **localhost :5000/NomDeL'image** avec par la suite le push de ces images sur le registre (avec une authentification au registre permettant le push des images) et la suppression des précédentes images en local, ainsi, grâce à cette solution, nous pouvons récupérer nos images directement depuis le registry privée.

Nous avons référencés ces actions depuis un script bash qui s'occupe :

- De se logger à notre registry
- De tagger nos image en local
- De push nos images local sur le registry
- De supprimer les images locales
- De récupérer les images depuis le registry

Authentification au registry

Afin de garantir un minimum de sécurité, il a été mis en place un script bash qui permettra de générer un certificat SSL en local qui sera par la suite copié sur le registry lors du docker-compose up de notre fichier. yml de notre registry.

La création d'un utilisateur a été nécessaire afin de pouvoir s'identifier lors de la phase de login à notre registry cité précédemment.

Mise en place de galera pour une réplication triples masters des bases de données

Afin d'être dans une thématique de haute disponibilité des bases de données avec Hyperdb, il a été nécessaire de mettre en place une réplication dites « 3 masters » afin que nos bases de données soient répliquées et qu'elles puissent prendre le relais en cas d'indisponibilité de l'une d'entre elle.

La configuration de Galera (module présent avec l'installation de mariadb) se fait au démarrage de chaque conteneur de base données via un script, l'ordre de configuration est le suivant :


- On prépare notre service db au sein de notre docker-compose
- On monte ensuite notre service db2
- Puis on termine par notre service db3

Une fois ces actions effectuées, avec la commande **mysql -u root -p -e "SHOW STATUS LIKE 'wsrep%';"**,

Nous avons les informations liées à notre cluster composé de nos 3 bases de données :

```
root@f33dba685199:/# mysql -u root -p -e "SHOW STATUS LIKE 'wsrep%';"
Enter password:
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_applier_thread_count | 1 |
| wsrep_apply_oooe | 0.000000 |
| wsrep_apply_oool | 0.000000 |
| wsrep_apply_window | 1.000000 |
| wsrep_causal_reads | 0 |
| wsrep_cert_deps_distance | 20.823188 |
| wsrep_cert_index_size | 86 |
| wsrep_cert_interval | 0.000000 |
| wsrep_cluster_conf_id | 3 |
| wsrep_cluster_size | 3 |
| wsrep_cluster_state_uuid | 7b8c37b0-b099-11eb-99e5-5759dfd044ad |
| wsrep_cluster_status | Primary |
| wsrep_cluster_weight | 3 |
| wsrep_commit_oooe | 0.000000 |
| wsrep_commit_oool | 0.000000 |
| wsrep_commit_window | 1.000000 |
| wsrep_connected | ON |
| wsrep_desync_count | 0 |
| wsrep_evs_delayed | |
| wsrep_evs_evict_list | |
| wsrep_evs_repl_latency | 0/0/0/0/0 |
| wsrep_evs_state | OPERATIONAL |
| wsrep_flow_control_paused | 0.000000 |
| wsrep_flow_control_paused_ns | 0 |
| wsrep_flow_control_rcv | 0 |
| wsrep_flow_control_sent | 0 |
| wsrep_gcomm_uuid | 7b8bd081-b099-11eb-a862-6776d2dfeab8 |
| wsrep_incoming_addresses | 172.25.0.2:3306,172.25.0.4:3306,172.25.0.12:3306 |
| wsrep_last_committed | 345 |
| wsrep_local_bf_aborts | 0 |
| wsrep_local_cached_downto | 1 |
| wsrep_local_cert_failures | 0 |
| wsrep_local_commits | 246 |
| wsrep_local_index | 0 |
| wsrep_local_recv_queue | 0 |
+-----+-----+
```


Nos 3 serveurs de BDD



Si une des bases de données tombe, nous voyons ainsi que le cluster reste actif mais qu'il ne reste plus que 2 bases de données en son sein :

```
root@f33dba685199:/# mysql -u root -p -e "SHOW STATUS LIKE 'wsrep%';"
Enter password:
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_applier_thread_count | 1 |
| wsrep_apply_oooe | 0.000000 |
| wsrep_apply_ool | 0.000000 |
| wsrep_apply_window | 1.000000 |
| wsrep_causal_reads | 0 |
| wsrep_cert_deps_distance | 20.604585 |
| wsrep_cert_index_size | 6 |
| wsrep_cert_interval | 0.000000 |
| wsrep_cluster_conf_id | 4 |
| wsrep_cluster_size | 2 |
| wsrep_cluster_state_uuid | 7b8c37b0-b099-11eb-99e5-5759dfd044ad |
| wsrep_cluster_status | Primary |
| wsrep_cluster_weight | 2 |
| wsrep_commit_oooe | 0.000000 |
| wsrep_commit_ool | 0.000000 |
| wsrep_commit_window | 1.000000 |
| wsrep_connected | ON |
| wsrep_desync_count | 0 |
| wsrep_evns_delayed | 983730d3-b099-11eb-9bab-a2eb255f72f0:tcp://172.25.0.4:4567:1 |
| wsrep_evns_evict_list | 0/0/0/0/0 |
| wsrep_evns_repl_latency | 0/0/0/0/0 |
| wsrep_evns_state | OPERATIONAL |
| wsrep_flow_control_paused | 0.000000 |
| wsrep_flow_control_paused_ns | 0 |
| wsrep_flow_control_rcv | 0 |
| wsrep_flow_control_sent | 0 |
| wsrep_gcomm_uuid | 7b8bd081-b099-11eb-a862-6776d2dfeab8 |
| wsrep_incoming_addresses | 172.25.0.2:3306,172.25.0.12:3306 |
| wsrep_last_committed | 349 |
| wsrep_local_bf_aborts | 0 |
+-----+-----+
```

Nos deux bases de données toujours actives



A savoir que si deux bases de données tombent, le cluster n'est plus opérationnel bien que la base de données soit toujours là, ainsi, afin d'augmenter l'efficacité du dispositif, il peut être judicieux de mettre en place d'avantage de base de données afin de garantir une meilleure haute disponibilité et par conséquent fournir une meilleure tolérance aux pannes.

Mise en place à l'aide du plugin hyperdb pour assurer le FailOver

Afin que Wordpress puisse toujours être opérationnel, il a été nécessaire d'ajouter le plugin et de le configurer de sorte que lorsqu'une base de données tombe, il puisse récupérer les informations et se connecter aux autres bases de données encore active ayant une réplication maître – maître chacune.

La configuration de hyperdb se fait à l'aide d'un script (qui sera joué sur le serveur web host1) qui effectuera l'ajout des paramètres des deux autres bases avec l'ajout du fichier de configuration dans le dossier web de Wordpress, les modifications peuvent se faire à chaud sans qu'il y ait une interruption de service.

Connexion au Blog avril

Pour se connecter à notre blog avril, il faudra entrer le lien suivant dans le navigateur web :

<http://www.wordpressblogavril.org:8080/> et vous arriverez sur notre blog :

Bonjour tout le monde !

Bienvenue sur WordPress. Ceci est votre premier article. Modifiez-le ou supprimez-le, puis commencez à écrire !

Publié le 11 mai 2021
Par [GlobalNet](#)

Catégorisé comme [Non classé](#)

Exposition des difficultés rencontrées durant le projet

Les principales difficultés rencontrées durant ce projet furent :

- Un manque de connaissances sur les différentes technologies
- Des compétences sur Linux
- Rendre chaque technologie compatible entre elles
- Des délais pouvant être assez juste