{EPITECH}

KILOUPABOCOU_

< DÉCOUVRIR LE MONDE DE DOCKER />

KILOUPABOCOU

Lors de ce projet vous devrez utiliser et maitriser les outils suivants:

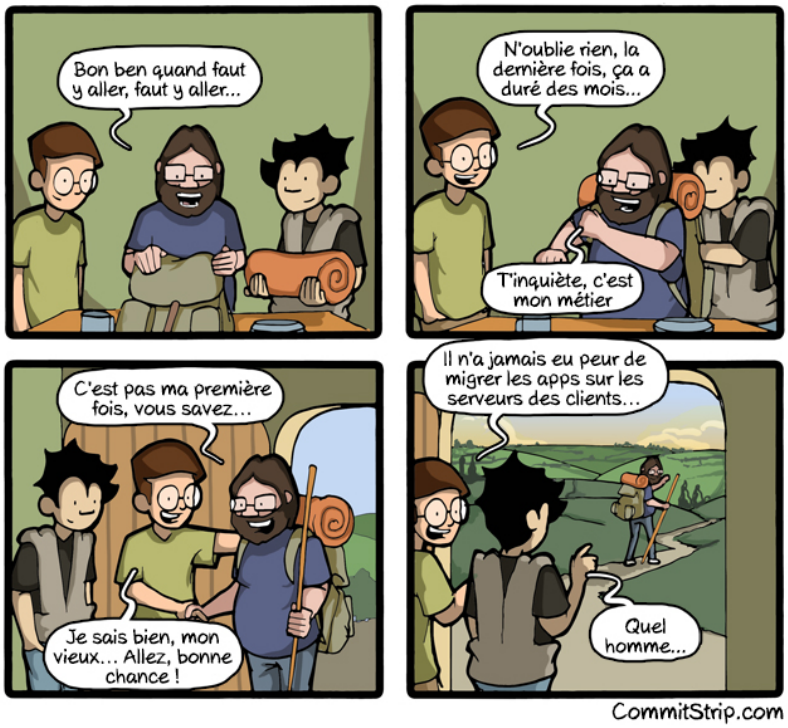
- ✓ Docker
- ✓ Dockerfile
- ✓ Docker-compose

La Dockerization de Kiloupabocou

Vous travaillez dans la start-up **Kiloupabocou** depuis maintenant 2 jours en tant qu'*administrateur système*.

Vous prenez tranquillement vos marques avec le système en place et vos collègues, c'est à ce moment là que Patrick, le patron décide de venir vous voir, des étoiles pleins les yeux et crie :
> Hey toi, le p'tit nouveau, on est plus en 2014 là ! Cette année on dockerize TOUT ! Alors je te laisse une semaine pour me changer notre infra et me rajouter quelques petits trucs en plus !

Et c'est ainsi que votre nouvelle mission pour dockerizer **Kiloupabocou** commence.



Migration de l'essentiel (partie obligatoire)

Patrick vous propose donc de faire cette migration progressivement, avec l'essentiel dans un premier temps, puis vous verrez après.

Attention : Patrick, malgré son niveau technique douteux, aime comprendre ce qu'il se passe dans son infrastructure, donc n'essayez pas de faire quelque chose que vous ne pourrez pas comprendre, sinon vous devrez assumer les conséquences et écouter Patrick vous crier dessus pendant 5 minutes !

Remarque: Il n'y a pas qu'une seule façon de faire, vous pouvez par exemple utiliser des `Dockerfile` avec les commandes nécessaires ou bien d'utiliser des `docker-compose.yml`.

Conteneurs pour le projet symfony

Vous devez créer ici au moins 3 conteneurs.

- ✓ **MySQL**
  Le conteneur pour la base de données de votre (ou vos) projet(s) Symfony.

  Vous devrez modifier le *mot de passe* `root` par défaut, créer une 'base de donnée' avec *un utilisateur* et *un mot de passe*

  Le **port devra être exposé** et un **volume créé** pour garder les données.
- ✓ **PHP**
  Ce conteneur vous permettra de déployer votre projet Symfony.

  Pour que tout fonctionne correctement, vous devrez **créer un volume** pour votre projet, **exposer le bon port** et si vous voulez bien faire les choses, créer un **volume pour les logs**.

  Remarque : Vous pouvez utiliser ce projet comme exemple, attention a bien modifier le `.env` pour que le projet utilise MySQL.
- ✓ **Nginx**
  Ce conteneur sera lié avec votre projet PHP, et exposera votre application Symfony.

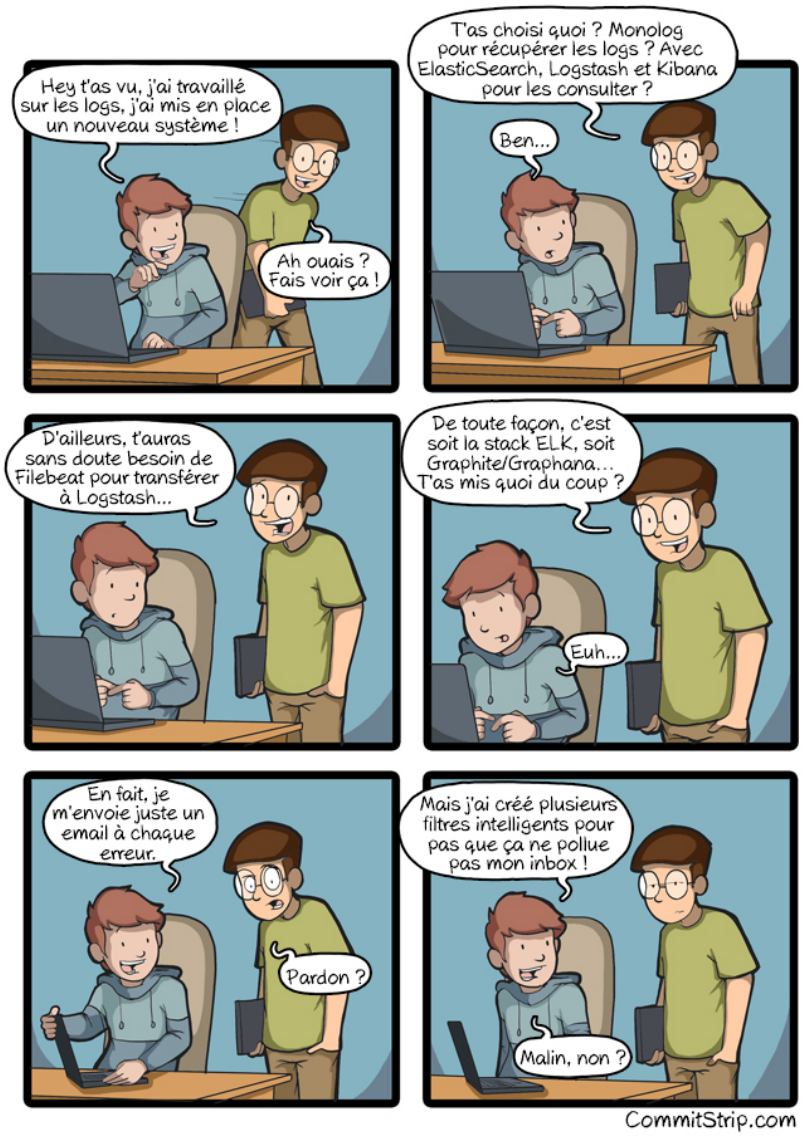
  Vous devrez **exposer le port 80**, créer un **volume qui contiendra les informations comme le nom de domaine** et **un autre pour les logs** nginx.

Conteneurs pour visualiser les logs

Génial, Patrick est aux anges car il découvre, tout comme vous comment fonctionne le monde de Docker !

Maintenant il aimerait garder un oeil sur les logs de vos volumes, et pour cela quoi de mieux que la stack **ELK** qui comprend *ElasticSearch, LogStash et Kibana*.

Pour comprendre comment ça fonctionne rapidement, vous pouvez regarder ici, ou alors apprendre par la pratique !



✓ **ElasticSearch**
Vous aurez besoin d'exposer des ports et de créer un volume qui vous permettra de garder les données d'ElasticSearch.

✓ **LogStash**
Ce conteneur devra être lié à ElasticSearch, vous devrez exposer un port et créer un volume.

Attention, contrairement à ElasticSearch où vous n'aurez pas grand chose à faire si ce n'est le conteneur; là vous devrez en plus créer un fichier de configuration pour LogStash.

De plus, pour que vous puissiez récupérer les fichiers de logs et les envoyer à LogStash, vous devrez installer et configurer Filebeat (vous pouvez passer par un conteneur ou bien directement sur votre système).

Bonus : Vous pouvez rajouter si vous le souhaitez d'autres produits Beats comme Packetbeat pour récupérer les données qui transitent sur votre réseau.

✓ **Kibana**
Dernier élément de la stack ELK, ce conteneur vous permettra de visualiser vos logs !

Vous devrez lier le conteneur ElasticSearch et exposer le port nécessaire pour atteindre l'application web.

Bonus : Normalement le conteneur Nginx est déjà configuré, vous pourrez donc en plus rajouter un nouveau fichier de config pour ajouter Kibana et son nom de domaine (exemple `kibana.mydomain.lan`).

Pour lier les conteneurs, vous pouvez vous référer a la documentation `docker --link` qui est deprecié, ou vous pouvez vous documenter sur les `networks` de Docker.

Et c'est parti pour le fun ! (Les bonus)

Maintenant que vous avez fait l'essentiel, Patrick a quelques demandes et aimerait que vous trouviez des idées pour enrichir l'infrastructure.

Rester à jour avec les séries

Patrick à une demande personnelle pour vous, tous les soirs quand il rentre, il perd du temps à vérifier les nouveaux épisodes des séries qu'il aime qui sont sortis. Il aimerait donc que vous mettiez en place un système qui vérifiera automatiquement les nouveaux épisodes qui sont sortis et qui les récupère automatiquement.

Pour se faire, rien de plus simple grâce à Flexget, en plus il y a une interface WEB, la possibilité de recevoir des notifications, de préciser les séries, les formats, les tailles etc...

Pour pouvoir télécharger automatiquement, il ne vous manque que Transmission ou deluge.

Vous devrez donc créer les conteneurs nécessaires avec les bonnes informations pour que tout fonctionne correctement.

Gérer le wiki de Kiloupabocou

Il est temps pour votre start-up d'avoir son propre wiki, pour cela rien de mieux que de dockerizer MediaWiki.

A vous de jouer !

Si vous avez d'autres idées pour rendre Kiloupabocou plus agréable, lancez-vous !

v 2