

Modalités

Groupe

Mettez-vous en groupe de deux ou de trois, un seul rendu par groupe.

Pré-requis

- 6 machine AWS par groupe : Ubuntu dernière version (n'oubliez pas de les résilier en fin de TP)

Rendu

Format

- Un mail par groupe avec une URL Github et les membres du groupe. Le sujet du mail devra contenir "[2017B3A] TP DevOps" ou "[2017B3B] TP DevOps". **Vous devez créer le dépôt Github et m'envoyer le lien avant même de commencer.**
- L'intégralité de votre rendu devra être sur un dépôt Github

Contenu

- Un fichier `README.md` doit être présent à la racine, qui détaille :
 - Comment faire fonctionner les différents scripts
 - Les membres du groupe (Prénom + Initial du nom, ça me va)
- Un fichier `.gitignore` qui permet de ne pas envoyer certains fichiers sur le git, mettez au moins `**/inventory.yml` pour éviter que votre inventory se retrouve sur Internet
- Trois dossiers `A`, `B` et `C` qui contiendront les réponses ou fichiers des exercices demandés

Important

Rendu

Chaque fois qu'un rendu est attendu sur votre dépôt il sera explicité par un libellé comme suit :

RENDU

Et il pourra éventuellement vous être indiqué un emplacement particulier dans un bloc comme suit (sinon mettez directement le ou les fichiers dans le dossier correspondant à l'exercice : A/xxx)

Mettez XXX à l'emplacement...

x/...

Screenshot

Si vous trouvez ce tag bleu sur un exercice, alors un screenshot est demandé :

SCREENSHOT

Stocker le screenshot sur votre dépôt dans le dossier réponse de l'exercice en cours, et nommé avec le numéro de question.

Ex: Pour l'exercice A, question 3, le screenshot doit être enregistré à l'emplacement

A/A3.png

Si vous avez plusieurs screenshots à faire pour une même question alors nommez les de cette manière :

A/A3-1.png, A/A3-2.png, ...

Triche

ATTENTION : J'ai une très bonne mémoire du code et des captures que j'ai déjà vu passer, j'en ai grillé quelques-uns l'an dernier qui ont eu 0. C'est un travail en binôme, pas un travail de classe, rien n'empêche de vous entraider, mais au sein d'un même binôme, vous devez avoir fait fonctionner ce que vous fournissez.

Notation

- Vous recevrez le détail de la note ainsi qu'un commentaire personnalisé par mail dès la correction

Conseil

- Ne paniquez pas, si vous n'arrivez pas à faire une partie, montrez-moi ce que vous savez faire et ce que vous avez retenu !

Mise en situation

Contexte

Une société de transport de appelée HomeColis vient de faire appel à vous. La livraison de colis est un milieu en plein essor et les sociétés de transports ont des besoins de plus en immédiats.

Actuellement, les coursiers récupèrent des colis aux entrepôts et font la livraison selon un chemin optimisé et prédéfini. Dans les grandes villes, les livraisons par camion en centre-ville sont toujours compliquées. C'est pourquoi ils désirent mettre en place un système de livraison express, par des coursiers à vélo, externe à la société.

Les coursiers à vélo, peuvent ouvrir une application sur leur téléphone, et indiquer qu'ils sont prêt à faire des livraisons. Ils reçoivent et peuvent accepter ou refuser des missions proches de leur localisation actuelle.

Leurs missions peuvent être d'aller chercher un colis chez un commerçant, dans un entrepôt ou en point relais. Puis de livrer ce colis à un particulier.

HomeColis a donc fait développer plusieurs applications :

- Une application Mobile pour les coursiers.
- Une site Web permettant de suivre la livraison d'un colis à partir d'un numéro de suivi
- Un back-office de gestion qui permet de rentrer de nouvelles courses à effectuer
- Une API (backend) qui permet au 3 applications de faire des requêtes pour récupérer les données.
L'API peut-être intégrée au BackOffice.

Mission

HomeColis décide de faire appel à vous pour fluidifier les flux, et améliorer leurs process.

Actuellement, les développeurs de **HomeColis** se partagent leur code source au travers d'un GIT.

Chaque fois qu'un nouveau développeur arrive, la mise en route est très compliquée, et à chaque fois deux jours sont perdus pour l'installation et le lancement du projet en local.

De plus HomeColis estime que leur service a un besoin important de haute disponibilité, et veulent donc pouvoir agrandir leur parc à la demande.

Votre mission va comporter trois aspects :

- Proposer un flux CI/CD
- Améliorer le lancement du projet sur des nouvelles machines pour le développement
- Préparer leur parc de machine, et prévoir qu'ils doivent pouvoir ajouter de nouvelles machine de façon simple

API

Vous travaillerez principalement sur la partie opérationnelle de l'API, elle est développée avec nodeJS (version 8) et se connecte sur une base de donnée MongoDB.

L'API est fournie ici : <https://github.com/Tronix117/wik-devops-3tp>

Pour les besoins de ce TP, l'API expose une seule route : `http://{url}/mongo` qui permet de tester si la connexion à MongoDB fonctionne ou pas.

Énoncé

A. Faciliter les nouveaux développement

1. Récupérer l'API en local

Le dépôt de l'API est : <https://github.com/Tronix117/wik-devops-3tp>

Récupérez ce dépôt sur votre ordinateur

2. Écrire un Dockerfile

Le Dockerfile doit créer une image qui permet de lancer le serveur et qui contient toutes les dépendances de l'API (lisez le README sur le dépôt)

RENDU

```
A/Dockerfile
```

3. Lancez la génération d'une image

Le nom de l'image doit être `homecolis-api@latest`

SCREENSHOT

Le(s) screenshot(s) doi(ven)t comporter la ligne de commande + la sortie

4. Écrire un docker-compose.yml

Le docker-compose doit lancer deux containers :

- un container applicatif avec l'API
- un container mongo

RENDU

```
A/docker-compose.yml
```

5. Lancez le compose

Lancez en mode NON-DÉTACHÉ

SCREENSHOT

Le(s) screenshot(s) doi(ven)t comporter la ligne de commande + la sortie

6. Testez la bonne inter-connection entre mongo et votre serveur

L'API fourni permet de faire cela, vous devez simplement visiter l'url : `http://127.0.0.1:{port}/mongo` (remplacez {port} par le port que vous avez bindé)

Prenez un screenshot de votre navigateur qui prouve la bonne inter-connection (url visible)

SCREENSHOT

Conclusion

Bien joué, désormais les nouveaux développeurs n'ont plus besoin de s'embêter à installer node ou mongodb, ils ont juste à lancer la commande du compose.

B. Automatisation

1. Écrivez votre inventory.yml

a. Listez vos 6 machines

Astuce

Au début du fichier vous pouvez donner des alias à vos machines de cette façon:

```
docker-manager1 ansible_ssh_host=xxx.xxx.xxx.xxx
```

- 3 machines devront être des docker-manager
- 1 machine devra être un docker-worker
- 1 machine devra être une machine maître mongo
- 1 machine devra être une machine esclave mongo

On va réaliser un cluster avec Docker Swarm, mais on va séparer nos mongos du swarm, car mongo est déjà prévu pour tourner en cluster

b. Listez vos machines dans plusieurs groupes : `docker_engine`, `docker_swarm_manager`, `docker_swarm_worker`, `mongo_master`, `mongo_slave`, `mongo`

SCREENSHOT

Floutez vos IPs sur le screenshot

2. Écrivez un fichier playbook-provisionnement.yml

Ce fichier devra réaliser les actions suivantes :

- Installez les serveurs mongo en cluster
- Installez les serveurs Docker
- Créer le Swarm et lier les instances entres-elles

a. Commencez par écrire le provisionning des machines MongoDB


Vous êtes libres d'utiliser un rôle sur ansible galaxy (personnellement, j'utilise greendayonfire.mongo)

Astuce

N'hésitez pas à créer un dossier `host_vars` avec dedans un fichier `mongo-master.yml` (nom-de-la-machine-dans-inventory.yml) avec des variables spécifiques à cette machine.

Vous pouvez aussi créer un dossier `group_vars` de la même façon, qui contient par exemple un fichier `mongo.yml` avec les variables communes des machines mongo.

Une machine doit être maître, une machine doit être esclave. (débloquez les ports mongodb sur les machines AWS)

Lancez le playbook.  du lancement de la commande + début de sortie

Lancez votre docker-compose en ayant pris soin de changer les variables d'environnement pour vous connecter au maître à distance.

 de votre navigateur web sur `/mongo`



La connexion au mongo distant et provisionnés depuis Ansible doit être en succès.

b. Complétez votre playbook-provisionnement.yml avec le provisionning du Docker Swarm

J'utilise personnellement atosatto.docker-swarm (mais il y a peut-être mieux)

Ajoutez des swarms labels sur vos node :

- manager1, manager2, manager3 : env=production, loadbalanced=true
- worker1: env=development

 Si vous n'arrivez pas à effectuer cette étape, déployer manuellement le cluster Docker Swarm. Prenez autant de  que nécessaire

Connectez-vous sur un docker manager, et listez les nodes, et prenez un screenshot



Fichier playbook-provisionnement.yml complet

Enregistrez le fichier sur votre dépôt à l'emplacement ``B/playbook-provisionnement.yml``

Si vous avez continué, mais que ce n'est pas fonctionnel, mettez le fichier qui ne fonctionne pas dans ``B/playbook-provisionnement.wip.yml`` (wip = Work In Progress)

RENDU

3-PREREQUIS. IMPORTANT

Simuler des noms de domaines

Modifiez le fichier host de votre ordinateur `/etc/hosts` (UNIX) ou `C:\Windows\System32\Drivers\etc` (WINDOWS) et complétez le avec :

```
xxx.xxx.xxx.xxx docker-manager1.homecolis.jt hello.homecolis.jt api.homecolis.jt dev.api.homecolis.jt
xxx.xxx.xxx.yyy docker-manager2.homecolis.jt
xxx.xxx.xxx.zzz docker-manager2.homecolis.jt
```

IMPORTANT: Remplacez le .jt par vos initiales, exemple Jean Dupont + Marc Toto : docker-manager.jdmt

Désormais sur votre machine UNIQUEMENT, vous pourrez accéder aux machines correspondantes en utilisant les noms de domaine. N'oubliez pas de remplacer par vos initiales lorsque vous les utiliserez.

3. Docker Stacks

Créez un dossier ``stacks`` au même endroit que votre `playbook-provisionnement.yml` (logiquement ``B/stacks`` si vous avez suivi la bonne nomenclature)

Un Proxy

Créez un fichier ``stacks/proxy.yml`` qui est un fichier Docker Stack, et complétez le de façon à déployer une stack qui contiendra :

un service haproxy basé sur l'image ``dockercloud/haproxy``

- Dans la clé ``deploy``, utilisez ``mode`` pour déterminer qu'il devra se lancer sur toutes les machines compatibles (recherchez)
- Chaque container devra disposer d'au moins 1% de processeur et d'au moins 20M de RAM, et devra utiliser au maximum 5% de processeur et 64M de RAM
- Le proxy ne doit redémarrer qu'un container à la fois lors du déploiement
- Le proxy doit pouvoir se relancer automatiquement s'il crash
- Le proxy doit être lancé uniquement sur les machines qui possèdent le label "loadbalanced=true"
- Le proxy doit être bindé sur le port 80 en mode "host" (le mode host permet de ne pas passer par le loadbalancer du manager)

Dans le même fichier de stack, lancez un service dockercloud/hello-world en 4 réplicas. Ce service Hello-World doit avoir un Virtual Host "http://hello.homecolis.jt" (remplacez .jt par ce que vous avez mis dans le fichier host)

Déployez la stack de façon manuelle sur un des manager

SCREENSHOT de la commande ``docker stack ps proxy`` sur un manager

SCREENSHOT de deux fenêtres de navigateur qui pointent sur "http://hello.homecolis.jt" avec un identifiant de container différent

RENDU

i Il est possible que vous deviez créer le network ``proxy`` de façon manuelle sur un de vos manager, ce n'est pas grave si pour faire cela vous ne passez pas par Ansible

! **Supprimez maintenant la stack avec ``docker stack rm proxy``**

Le registry

Pour envoyer l'image de votre API vous allez avoir besoin d'un registry.

Créez un fichier stacks/registry.yml qui permet de lancer un registry et déployez le manuellement.

Testez le bon fonctionnement du registry en envoyant l'image dessus ``docker push docker-manager1.homecolis.jt:5000/homecolis-api@latest``

SCREENSHOT de l'upload

RENDU

L'application

Créez un fichier ``stacks/api.yml`` et complétez le de façon à déployer une stack qui contiendra :

- 3 réplicas de l'API en production sur les nodes qui ont le label `env=production` (Virtual Host: `api.homecolis.jt`)
- 2 réplicas de l'API de développement sur les nodes qui ont le label `env=development` (Virtual Host: `dev.api.homecolis.jt`)

Vous êtes libre de définir les autres options de configuration comme vous le désirez, du moment que cela ait du sens.

Utilisez l'image que vous avez envoyé sur votre registry, attention, pour simplifiez, utiliser l'IP interne de amazon dans le nom de l'image, car votre fichier hosts est uniquement modifié sur votre ordinateur et pas sur les machines distantes. Utilisez donc ``image: 172.xxx.xxx.xxx:5000/homecolis-api@latest``

Lancez la stack.

Faites 4 screenshot :

SCREENSHOT 1 pour l'API de production sur le navigateur a l'url ``http://api.homecolis.jt/mongo``

SCREENSHOT 1 pour l'API de développement sur le navigateur a l'url `http://dev.api.homecolis.jt/mongo`

RENDU

❗ **Supprimez maintenant la stack avec `docker stack rm api`**

4. Écrivez un rôle Ansible qui permet de déployer des stacks

L'idée est de créer un playbook-deploy.yml qui ressemble à ceci.

playbook-deploy.yml

```
- name: "Deploy Docker Stack"
  hosts: docker-manager1
  remote_user: ubuntu
  become: yes
  gather_facts: no
  roles:
    - role: deploy-docker-stacks
      registry:
        host: "172.xxx.xxx.xxx:5000"
        username: xxxx
        password: yyyy
      stacks:
        - proxy
        - api
```

Le rôle `deploy-docker-stacks` est un rôle que vous devez écrire.

Il servira à :

- Envoyer les fichier de stack sur le host distant à un endroit connu
- Se connecter à votre registry (facultatif)
- Exécuter les commandes pour déployer les stacks

SCREENSHOT du lancement du script et des résultats

RENDU

Bonus: Load Balancer AWS

Utilisez l'Elastic Load Balancer de AWS (si c'est possible avec votre compte bloqué) pour avoir un seul point d'entrée qui pointe sur les 3 machines docker-manager à tour de rôle

SCREENSHOT

C. Proposition

1. Proposez une solution de Continuous Integration

- Fournir des pistes d'améliorations du code existant pour augmenter sa qualité
- Citez des exemples d'organisation du Git pour travailler à plusieurs
- Que devrait faire concrètement la pipeline de CI ?

Il est à noter qu'il est possible de lancer un script ou un ensemble de script à chaque fois qu'un développeur envoie son code.

RENDU

Dans un fichier Markdown

C/C1.md

2. Proposez une solution de Continuous Deployment

- Qu'utiliseriez-vous depuis une pipeline CI pour lancer des déploiements
- Comment utiliser le CD avec plusieurs environnements, proposez une solution
- Comment contrôler ce qui est déployé en production en pré-production ou en développement

RENDU

Dans un fichier Markdown

C/C2.md

3. Proposez au client d'autres pistes intéressantes et DevOps pour ses process

Vraie mise en situation imaginez-vous à la place d'un consultant DevOps, et rédigez un rapport tel que vous pourriez fournir à votre client.

Vous avez peu d'infos, donc plus de liberté. Vous avez tout à fait le droit d'imaginer des mauvais process en place et d'y apporter des solutions (ex: Nous avons remarqué que vous faisiez xxx, nous vous proposons ...").

Rendu d'une page A4 en PDF. (Je note pas la présentation, mais les tournures de phrase, ainsi que l'orthographe et la grammaire seront notés)

RENDU

C/C3.pdf