DOCKER - PORTAINER

Jolan Noirot BTS SIO 2

SOMMAIRE

INTRODUCTION

QU'EST-CE QUE DOCKER

Docker est une plateforme open-source permettant de créer, déployer et gérer des conteneurs, qui sont des environnements légers et isolés pour exécuter des applications. Il a été créé en 2013 par Solomon Hykes au sein de la startup **dotCloud**, avant de devenir un projet indépendant sous l'organisation **Docker Inc.**. Docker facilite le déploiement d'applications en garantissant leur compatibilité entre différents environnements.

QU'EST-CE QUE PORTAINER

Portainer est une interface web simplifiant la gestion des conteneurs Docker, Kubernetes et Swarm. Créé en 2017 par **Neil Cresswell** et **Anthony Lapenna**, il permet d'administrer facilement des environnements Docker sans passer par la ligne de commande. Portainer offre une gestion intuitive des images, volumes, réseaux et services, rendant l'orchestration des conteneurs accessible même aux débutants.

QU'EST-CE QUE DOCKER SWARM

Docker Swarm est une fonctionnalité d'orchestration de conteneurs développée par Docker pour permettre la gestion de clusters de machines Docker. Il permet de déployer, gérer et faire évoluer des applications conteneurisées sur plusieurs hôtes en les regroupant sous un seul cluster. Swarm offre des fonctionnalités telles que la haute disponibilité, le load balancing et la mise à l'échelle automatique des services, ce qui permet de gérer des applications complexes dans des environnements de production avec plusieurs nœuds.
Docker Swarm a été introduit en **2014** après l'acquisition de la société **Swarm** (une startup spécialisée dans l'orchestration des conteneurs), et il a été intégré dans Docker en tant que fonctionnalité native d'orchestration, directement dans la commande Docker sans nécessiter un outil tiers comme Kubernetes. Cela a simplifié la gestion des applications en cluster pour les utilisateurs de Docker, en offrant une solution d'orchestration légère et intégrée.

QU'EST-CE QUE DOCKER COMPOSE

Docker Compose est un outil qui permet de définir et de gérer des applications multi-conteneurs Docker sur une seule machine. À l'aide d'un fichier YAML (docker-compose.yml), il permet de spécifier les services, réseaux et volumes nécessaires à l'application, simplifiant ainsi le déploiement et la gestion de plusieurs conteneurs. Une fois configuré, tu peux démarrer, arrêter ou reconstruire l'ensemble de l'application avec des commandes simples comme docker-compose up et docker-compose down, ce qui rend la gestion des environnements complexes plus facile et reproductible.

QU'EST-CE QU'UN MIDDLEWARE

Un **middleware** est un logiciel qui agit comme un intermédiaire entre différentes applications ou couches d'une application. Dans le contexte des serveurs web et des applications, il s'agit souvent d'un ensemble de fonctions qui traitent les requêtes HTTP avant qu'elles n'atteignent l'application principale ou après que la réponse ait été générée. Les middlewares sont utilisés pour des tâches comme l'authentification, la gestion des sessions, la journalisation, la compression des réponses ou encore la gestion des erreurs. Ils permettent de séparer des préoccupations transverses et de rendre le code de l'application plus modulaire.

QU'EST-CE QU'UN WEBHOOK

Un **webhook** est une méthode permettant à une application d'envoyer des données en temps réel à une autre application dès qu'un événement spécifique se produit. Contrairement aux API traditionnelles qui nécessitent une requête pour récupérer des informations, un webhook "pousse" automatiquement les données vers une URL prédéfinie lorsque l'événement se déclenche, comme la création d'un nouvel utilisateur ou la mise à jour d'un fichier. Les webhooks sont souvent utilisés pour des intégrations en temps réel entre différents services, comme l'envoi de notifications, la synchronisation de données ou l'activation de processus.

INSTALLATION

COMMENT INSTALLER DOCKER

Commandes	
`sudo apt update`	Met à jour les repos de la machine (-y pour accepter automatiquement tout ce que demandera la machine)
`wget https://get.docker.com/`	Télécharge la page https://get.docker.com/
`sudo bash index.html`	Exécute la page télécharger en tant que script bash
`sudo usermod -aG docker $USER`	Permet l'exécution des commandes liées a docker par l'utilisateur sans avoir à utiliser sudo

COMMENT CRÉER UN CONTAINER

Commande	
`docker run IMAGE`	Permet de démarrer un conteneur à partir d'une image (si l'image n'est pas trouver sur le poste il l'a télécharge automatiquement)
Options	
`-d`	Permet de lancer un conteneur en arrière-plan
`-p PORT_EXT:PORT_INT`	Permet d'exposer un port du container à l'extérieur
`--name NOM`	Donne un nom au container
`--restart=POLICY`	Permet de définir la règle de démarrage du container, les options sont : - `always` (redémarre automatiquement même si il échoue a démarrer) - `unless-stopped` (redémarre automatiquement sauf

	si stopper par l'utilisateur) - `on-failure[:MAX-RETRIES]` (redémarre uniquement en car de d'échec, remplacer MAX-RETRIES par le nombre de tentatives)
`--network POLICY`	Permet de spécifier le réseau auquel ton conteneur doit être connecté, utile pour des container permettant de réveiller des appareils sur le réseau du host par exemple, les options sont : - `bridge` (par défaut) - `host` (permet au conteneur de partager le réseau de l'hôte) - `none` (désactive toute connexion réseau pour le conteneur)
`-e VARIABLE`	Définit une variable d'environnement pour le container
`-v CHEMIN_VOLUME`	Attribue un volume au container pour qu'il sauvegarde ses données

`bcast enp0s3`
`deadtime 5`
`keepalive 1`
`node web1 web2`
	bcast enp0s3 : Utilise le mode broadcast sur l'interface réseau enp0s3 pour envoyer des paquets Heartbeat.
	deadtime 5 : Définit le délai en secondes avant de considérer un nœud comme inactif.
	keepalive 1 : Spécifie l'intervalle en secondes entre les messages Heartbeat envoyés aux nœuds.
	node web1 web2 : Liste les nœuds participants au cluster Heartbeat, ici web1 et web2.

COMMENT INSTALLER PORTAINER

Commandes	
`docker volume create portainer_data`	Créer un volume de stockage pour le container
`docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:lts`	Créer le container de portainer

COMMENT CRÉER UNE STACK

Commandes	
`docker swarm init`	Afin de modifier l'adresse IP de la machine
`nano docker-compose.yml`	Créer le fichier docker-compose.yml et le modifie

`version: '3.7'`
 	
`services:`
`  web:`
`  image: nginx`
`  ports:`
`  - "80:80"`
`  db:`
`  image: mysql`
`  environment:`
`  MYSQL_ROOT_PASSWORD: example`
	Valeurs saisies dans le fichier docker-compose.yml afin de définir quelle container déployer et avec quelle image

`docker stack deploy -c docker-compose.yml NOM`	Déploie la stack
`docker stack services NOM`	Permet de voir l'état de la stack
`docker stack rm NOM`	Permet de supprimer la stack créer

SERVICES DÉPLOYÉS

SERVICES CHOISIS

La première stack contient un service nommé OpenCVE, se service utilise un ensemble de 8 containers pour fonctionner. Il permet de regrouper toutes les failles de sécurités connue/découvertes et de pouvoir filtrer par équipement afin de n'avoir que les failles concernant nos équipements

La deuxième stack est un middleware que j'ai personnellement développé comme étant une extension de OpenCVE. Pourquoi ne pas l'avoir directement intégré à la stack de OpenCVE ? Tout simplement pour faciliter sa désactivation. Son utilité ? Le service de notification d'OpenCVE est très efficace pour envoyer par mail mais malheureusement pour envoyer sur des services telle que Teams, Slack ou encore Discord il n'est pas fonctionnelle et facile d'utilisation. Le but de ce middleware est de permettre l'envoie de notification d'OpenCVE via webhooks sur différentes plateformes. Comment il fait ? il reçoit le payload (le contenu de la notification d'OpenCVE), le convertit en un format interprétable par le service souhaiter (Teams, Slack ou Discord, une version différente par services est proposée mais une solution universelle est en cours de développement), puis envoie le payload convertit au service souhaiter. Le deuxième container a pour but d'aider l'utilisateur à voir si le middleware fonctionne correctement, en affichant via une interface web si le middleware est en ligne et les journaux d'évènement du middleware.

POURQUOI AVOIR CHOISI DE CES SERVICES

J'ai découvert OpenCVE durant mon stage aux Restos du Cœur, et je l'ai mis en place, ils souhaitaient recevoir les notifications sur un canal Slack c'est ainsi que j'ai développé ce middleware en plus. La cybersécurité devient un élément clé dans la société d'aujourd'hui, ayant plusieurs services et serveurs dans mon infrastructure j'ai pensé utile de le mettre également en place dans mon infrastructure, j'ai également mis en place le service de middleware pour être informé sur discord dans mon serveur de notification directement.

COMMENT INSTALLER LES DEUX SERVICES

OPENCVE

Pour commencer clonez simplement ce répository GitHub :
https://github.com/opencve/opencve
Ensuite allez dans opencve/docker et faite la commande :

```
./install.sh
```

Et pour finir exécutez la commande :

```
./install.sh start
```

OpenCVE est désormais installé !

MIDDLEWARE

Pour commencer, clonez ce repository GitHub :
https://github.com/Furtif2005Sno/OpenCVE-To-Discord/
Allez ensuite dans OpenCVE-To-Discord modifiez les paramètres demander dans index.html, webui.py et middleware.py puis faite la commande :

```
sudo docker-compose up -d –build
```

Une fois cela fait allez sur votre service OpenCVE et dans l'onglet notification, ajoutez une notification de type Webhook, comme url mettez http://IP_DU_SERVEUR_MIDDLEWARE:5000/webhook et dans Headers rentrez cette valeur {"Content-Type": "application/json"}
Sauvegardez et... Le middleware est installé et lié à OpenCVE et Discord !

PS : La procédure est similaire pour la version Slack