**DevOps**

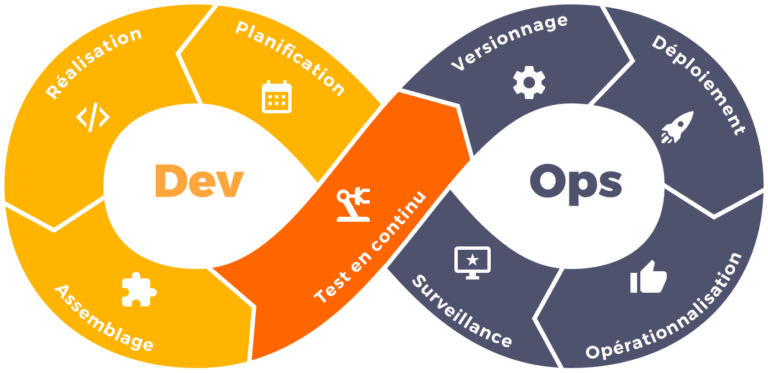


Table des matières

[Introduction 3](#_Toc30519392)

[Docker : 3](#_Toc30519393)

[Compose : 3](#_Toc30519394)

[Nginx : 3](#_Toc30519395)

[Quelques commandes utiles 3](#_Toc30519396)

[Installation 4](#_Toc30519397)

[Docker + Nginx 4](#_Toc30519398)

[Installation Docker Compose 4](#_Toc30519399)

[Installation Git 4](#_Toc30519400)

[Configuration 5](#_Toc30519401)

[Configuration Docker-Compose 5](#_Toc30519402)

[Dockerfile 5](#_Toc30519403)

[Liste de commande : 5](#_Toc30519404)

[Création Certificats 6](#_Toc30519405)

[Récupération du depo 7](#_Toc30519406)

[Convertion du depo en format Docker 7](#_Toc30519407)

# Introduction

Le sujet qui nous à été demandé était graçe à l’utilisation de docker pouvoir mettre en place l’hébergement ainsi que l’infrastructure necessaire.

## Docker :

Afin de pouvoir mettre à bien notre projet, nous aurons besoin d’utiliser docker, qui est un logiciel libre permettant facilement de lancer des applications dans des conteneurs (un conteneur Docker, à l’opposé de machine virtuelle, ne requiert aucun système d’exploitation séparé et n’en fournit aucun. Utiliser Docker pour créer et gérer des conteneurs peut simplifier la mise en œuvre de système distribués en permettant à de multiples application, tâches de fond et autres processus de s’éxecuter de façon autonome sur des machines physique.)

Avant toute chose, nous aurons besoin d’avoir Docker CE sur notre machine. Pour se faire, nous allons l’installer à partir d’un script qui est l’installation la plus simple, ainsi que la plus rapide.

## 

## Compose :

C’est un outil très intéressant vis-à-vis de la gestion de package docker. Cet outil va lancer vos conteneurs et leurs éventuels liens à partir d’un fichier de configuration écrit en yaml. Cet outil vient

## Nginx :

Nginx est un serveur web open-source, qui peut également être utilisé comme reverse proxy, cache http ou même comme load balancer. De plus Nginx est un serveur web très populaire qu’y supporte le node.js.

# Quelques commandes utiles

**docker ps (-a)** affiche toutes les instances de docker qui tournent actuellement sur votre environnement. Avec l’option -a, vous verrez mêmes les containers stoppés.

**docker images** montre les images que vous avez construites.

**docker-compose up** démarre les services décrits dans mon docker-compose.yml et ne me rend pas la main.

**docker-compose up -d** fait la même chose mais me rend la main une fois que les services sont démarrés.

**docker-compose up –build** reconstruit les services avant de les lancer.

**docker-compose down** stoppe les services.

**docker-compose logs** me retourne l’ensemble des logs des services depuis le dernier démarrage et me rend la main.

**docker-compose logs -f** affiche les logs des services et continue à les « écouter » sans me rendre la main.

# Installation

## Docker + Nginx

Dans un premier temps, nous aurons besoin d’installer docker sur notre machine sous Linux, ici la distribution est un Debian, du fait de la familiarisation avec l’OS. Nous pouvons soit installer docker à partir des nombreuses lignes de commande, soit à partir du script disponible sur la même page, à savoir la documentation officielle de [Docker](https://docs.docker.com/install/linux/docker-ce/ubuntu/) :

Voici les commandes à suivre :

su apt install curl

curl -fsSL https://get.docker.com -o get-docker.sh

su sh get-docker.sh

Une fois Docker installé, nous pouvons télécharger et lancer notre serveur Web via la commande suivante :

docker container run -–name nginx -p 80:80 -d nginx

Nous pouvons maintenant vérifier que le conteneur a bien été créé et qu’il tourne bien avec la commande :

docker ps



## Installation Docker Compose

Afin de gérer les différentes images Docker, nous allons donc installer [docker-compose](https://docs.docker.com/compose/install/)

su curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Application des permissions d’execution :

sudo chmod +x /usr/local/bin/docker-compose

## Installation Git

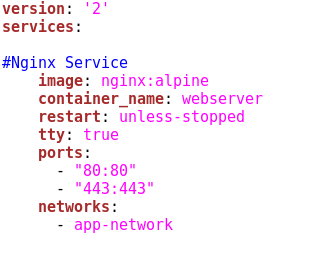
Afin de povoir utiliser le code de nos développeurs, nous allons également, installer le service de git.

apt install git

# Configuration

## Configuration Docker-Compose

Dans notre docker-compose.yml, nous allons décrire les services à utiliser, à savoir un serveur Web, avec le port 80 d’ouvert, ainsi que la version.



## Dockerfile

Un dockerfile défini le contenu de l’image de l’application via une suite de commande de construction qui configurent cette image.

## Liste de commande :

*FROM* nous permet de définir notre image de base, nous pouvons l’utiliser seulement une fois dans un dockerfile.

*RUN* nous permet d’exécuter une commande à l’intérieur de notre image

*ADD* nous permet d’ajouter des fichiers locaux ou distants à l’intérieur de notre image, il est le plus souvent uilisé pour importer les sources de notre projet ou des fichiers de configuration.

*WORKDIR* nous permet de changer le répertoire courant de notre image, toutes les commandes qui suivront seront exécutées à partir de ce répertoire.

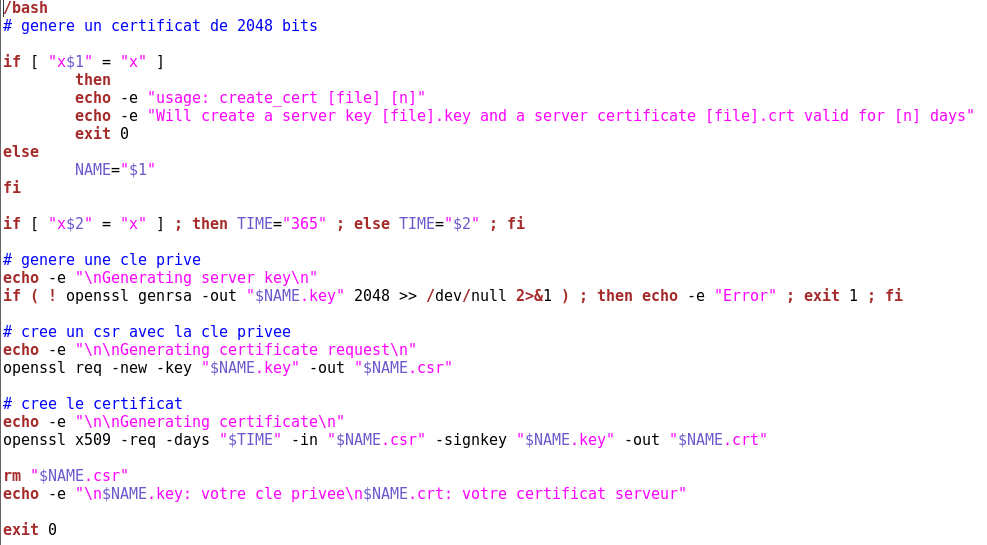
Afin de pouvoir utiliser notre Dockerfile, nous aurons donc besoin de créer dans un premier temps le dossier qui sera utiliser par notre projet, et par la suite de créer un document du nom de Dockerfile, et mettre à l’intérieur notre liste de commande, à savoir :

Nous allons donc inclure l’installation de Curl ainsi que Node JS.

# Image de base :  
FROM mynginx1  
  
# Installation de curl  
RUN apt update \ && apt install -y curl   
  
# Installation node.js  
RUN curl -LO "<https://nodejs.org/dist/v0.12.5/node-v0.12.5-linux-x64.tar.gz>" \  
&& tar -xzf node-v0.12.5-linux-x64.tar.gz -C /usr/local --strip-components=1 \  
&& rm node-v0.12.5-linux-x64.tar.gz  
  
# Installation dépendance :  
RUN npm install  
  
# Exposition du port 80  
EXPOSE 80

## Création Certificats

Dans un premier temps nous allons créer une clé de 2048 bits qui est le minimum recommandé à la création d’un certificat SSL.



## Récupération du depo

Et nous pourrons par la suite faire un simple git clone afin de disposer de ce code et ainsi pouvoir l’afficher.

git clone <https://github.com/YaMsanda/devops_armorAPI.git>

## Convertion du depo en format Docker

docker build -t manda217/devops-armor-api .