TD : Docker LAMP Multisite

Nous avons 2 applications :

- **Esport** constitué d'une API en NODEJS et d'un front en PHP/JS. L'API utilise une base de donnée Mongodb.
- **Biblio** une application PHP/MYSQLI.

Nous souhaitons utiliser Docker pour déployer les deux applications dont les noms de domaine seront inscrit dans un serveur DNS local BIND :

- http://esport.docker && http://www.esport.docker
- http://biblio.docker && http://biblio.esport.docker

1. Exécuter un serveur DNS avec Docker

Outre la notation standard des adresses Internet, composée de chiffres et de points, vous pouvez également faire référence à un hôte par un nom symbolique.

L'avantage d'un nom symbolique est qu'il est généralement plus facile à retenir.

Par exemple, la machine avec l'adresse Internet `158.121.106.19` est également connue sous le nom de `alpha.gnu.org`; et les autres machines du domaine `gnu.org` peuvent s'y référer simplement sous le nom de `alpha`. Le système de noms de domaine (DNS) est un service qui traduit les noms de domaine en adresses IP.

BIND 9 est un système open source transparent, sous licence MPL 2.0. BIND 9 a évolué pour devenir un système DNS très flexible et complet. BIND est utilisé avec succès pour toutes les applications, de la publication de la zone racine DNS (signée DNSSEC) et de nombreux domaines de premier niveau, aux fournisseurs d'hébergement qui publient de très gros fichiers de zone avec de nombreuses petites zones, aux entreprises avec des zones internes (privées) et externes, aux fournisseurs de services avec de grandes fermes de résolveurs.

Préparation

Configuration du réseau

Pour pouvoir utiliser le conteneur DNS avec des adresses IP statiques, je vais d'abord créer un réseau Docker. La commande suivante crée un réseau arbitraire appelé `lamp_network` avec la plage `172.24.0.0/16`:

Nous voulons :

- DNS : `172.24.0.254`
- Serveur Web : `172.24.0.200`

```
docker network create --subnet=172.24.0.0/16 lamp_network
```

Configuration du serveur DNS

Tout d'abord, je crée un fichier pour commencer à configurer le serveur Bind9 `named.conf.options`:

Cela garantira que BIND écoute sur toutes les interfaces et utilisera les serveurs DNS de Google comme transitaires :

/etc/bind/named.conf.options

```
options {
    directory "/var/cache/bind";

    recursion yes;
    listen-on { any; };

    forwarders {
            8.8.8.8;
            4.4.4.4;
    };
};
```

Ensuite, je vais définir une zone appelée `esport.docker` dans le fichier `named.conf.local`, qui pointe vers `/etc/bind/zones/db.esport.docker` le fichier de zone :

/etc/bind/named.conf.local

```
zone "esport.docker" {
    type master;
    file "/etc/bind/zones/db.esport.docker";
};

zone "0.24.172.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/db.172.24.0";
};
```

Le fichier de zone appelé `db.esport.docker` répertorie tous les services qui doivent être gérés (par exemple, le conteneur **Docker** sur le réseau Docker) et leur attribue un nom d'hôte et une adresse IP :

Ici, `172.24.0.200` serait l'IP du conteneur **apache**.

/etc/bind/zones/db.esport.docker

```
;
; BIND data file for local loopback interface
```

```
;
$TTL    604800
@       IN      SOA     esport.docker. admin.my3wa.ac. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.esport.docker.
ns      IN      A       172.24.0.200
@       IN      A       172.24.0.200
www     IN      A       172.24.0.200
```

Créez le fichier de zone inverse :

/etc/bind/zones/db.172.24.0

```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     esport.docker. admin.esport.docker. (
1         ; Serial
604800         ; Refresh
86400         ; Retry
2419200         ; Expire
604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.esport.docker.
200     IN      PTR     ns.esport.docker.
200     IN      PTR     www.esport.docker.
```

Création de l'image Docker

Je souhaite utiliser l'image Docker officielle de BIND 9 avec ubuntu, mais aussi installer quelques dépendances supplémentaires et y ajouter directement mes fichiers de configuration au lieu de les monter dans le conteneur :

Dockerfile

```
FROM  internetsystemsconsortium/bind9:9.19

RUN apt update \
    && apt install -y \
    bind9 \
    bind9utils \
    bind9-doc \
    geoip-bin \
    nano \
```

```
    dnsutils \
    net-tools

# On copie les fichiers de configuration

COPY configuration/named.conf.local /etc/bind
COPY configuration/named.conf.options /etc/bind
COPY configuration/db.172.24.0 /etc/bind/zones/
COPY configuration/db.esport.docker /etc/bind/zones/

# Expose les ports
EXPOSE 53/tcp
EXPOSE 53/udp
EXPOSE 953/tcp

# Start the Name Service
CMD ["/usr/sbin/named", "-g", "-c", "/etc/bind/named.conf", "-u", "bind"]
```

Je peux maintenant créer et étiqueter l'image BIND :

```
docker build -t dns-master .
```

Exécutez le conteneur

Le conteneur doit maintenant être créé à l'intérieur du réseau Docker `lamp_network` avec l'adresse IP qui lui est attribuée à l'intérieur `db.esport.docker`, à savoir : `172.24.0.1` :

```
docker run -d --rm --name=dns-master --network=lamp_network --ip=172.24.0.254 dns-
master
```

Je peux maintenant vérifier la configuration de mon serveur :

```
docker exec -ti dns-master /bin/bash
named-checkconf
named-checkzone esport.docker /etc/bind/zones/db.esport.docker
```

Service de connexion

Il est désormais possible d'exécuter les deux conteneurs de services en utilisant le conteneur `dns-server` comme serveur DNS comme sur un conteneur `httpd` :

```
docker run -d --rm --name=apache --net=lamp_network --ip=172.24.0.200
--dns=172.24.0.254 httpd
```

Tous les conteneurs fonctionnent désormais sur le même réseau :

```
docker network inspect lamp_network
```


```
[
    {
        "Name": "lamp_network",
        "Id": "c9aa78ade702ad0a73ae58a459da9f8844ca855c4292bfd5acc3751ce9164d58",
        "Created": "2024-07-04T00:37:05.927719293Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.24.0.0/16"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "944d7931b8d0c0e8b55511577736b715c2ac21d26bb96882d79d54ebca5a85a0": {
                "Name": "dns-master",
                "EndpointID":
"fb77fb360a5f29d672726f7cb0a9ae4a3951a6d9e8616ec9754c18bd262a1331",
                "MacAddress": "02:42:ac:18:00:fe",
                "IPv4Address": "172.24.0.254/16",
                "IPv6Address": ""
            },
            "aef79684a1b1a87fde34ec3e82721184c6f1a2cb79b53ad21c69332ea7cd8fb8": {
                "Name": "apache",
                "EndpointID":
"c2fdf28727994d266c9d2ac987659987d8503f60a77589b07ec97c2fe170c80a",
                "MacAddress": "02:42:ac:18:00:c8",
                "IPv4Address": "172.24.0.200/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

Je peux tester le service DNS en me connectant à l'un des services clients et en envoyant un ping à l'autre :

```
docker exec -it apache sh
```

installez l'utilitaire `nslookup` :

```
# apt update
# apt install dnsutils
# nslookup esport.docker

Server:         127.0.0.11
Address:        127.0.0.11#53

Name:   esport.docker
Address: 172.24.0.200
```

De plus, le transitaire fait son travail en me permettant de résoudre les domaines en dehors de la zone définie :

```
PS C:\Users\baptiste> docker exec -it apache nslookup google.com
Server:         127.0.0.11
Address:        127.0.0.11#53

Non-authoritative answer:
Name:   google.com
Address: 216.58.214.174
Name:   google.com
Address: 2a00:1450:4007:80d::200e
```

Ici, nous pouvons voir que la réponse n'est pas autoritaire , ce qui signifie que le serveur DNS que nous avons interrogé a dû transmettre la demande car il ne connaissait pas la réponse.

IMPORTANT

J'espère que ce guide vous a été utile, et vous permettra de comprendre la logique de fonctionnement de Docker et des fichiers Dockerfile.

Essayez maintenant, de reprendre vos notes concernant l'installation d'un serveur apache/php/mysql/phpmyadmin et de reproduire les mécanismes d'automatisation des tâches.

APACHE

- Un seul serveur **apache/php** pour le frontend. Les deux applications clientes partagent la même adresse IP, celle du serveur apache. Il faudra donc configurer Apache (cf: Virtual Host) pour que selon que l'on provient de du domaine `esport.docker` ou `biblio.docker` le dossier d'application

html cible ne soit pas le même. (exemple : `/var/www/html/esport` ou `/var/www/html/biblio`).