BTS Services Informatiques aux Organisations (SIO)

Option SISR – Épreuve E6 : Parcours de professionnalisation

Ressources documentaires – Réalisation Professionnelle n°1

Sujet : Mise en place d'une plateforme de déploiement applicatif conteneurisée, isolée et sécurisée à l'aide de Docker Compose

Candidat : Vandesquille Axel
Établissement : Lycée Saint-Exupéry – Saint-Raphaël

---

1. Architecture Réseau et Infrastructure Inter-VLAN

1.1 Rôle et Utilisation des Équipements

- **Le Routeur Cisco :** Il agit comme la passerelle par défaut (Gateway) pour l'ensemble des machines virtuelles. Son interface physique liée au switch est découpée en sous-interfaces virtuelles (ex: g0/0/0.10 pour le VLAN 10 et g0/0/0.20 pour le VLAN 20) selon le principe du *Router-on-a-Stick*. Chaque sous-interface porte l'adresse IP de la passerelle du VLAN correspondant, permettant le routage inter-VLAN.
- **Le Switch Cisco :** Il assure le cloisonnement de niveau 2. Le port connecté au routeur est configuré en mode **Trunk** pour acheminer les paquets taggués de tous les VLANs. De même, le port connecté au PC physique (hébergeant l'hyperviseur des VMs) est configuré en mode **Trunk** car la carte réseau de la machine virtuelle Debian doit pouvoir émettre et recevoir sur plusieurs VLANs (VLAN de production et VLAN d'administration/maison).

1.2 Plan d'adressage Réseau

Équipement / VM	VLAN	Réseau IP	Adresse IP	Passerelle (IP Routeur)
VM Client de test	VLAN 10 (Clients)	192.168.10.0/24	192.168.10.50	192.168.10.254
VM Serveur Docker (Prod)	VLAN 20 (Serveurs)	192.168.20.0/24	192.168.20.10	192.168.20.254

Équipement / VM	VLAN	Réseau IP	Adresse IP	Passerelle (IP Routeur)
VM Serveur Docker (Admin)	Réseau LAN Physique	192.168.1.0/24	192.168.1.23 (DHCP)	192.168.1.1

1.3 Configuration du Switch Cisco (SW1)

```
!
version 15.2
!
hostname SW1
!
vlan 10
 name VLAN-CLIENTS
!
vlan 20
 name VLAN-SERVEURS-DOCKER
!
interface FastEthernet0/1
 description === Lien vers PC Physique (Hyperviseur VMs) ===
 switchport mode trunk
!
interface FastEthernet0/24
 description === Lien vers Routeur (Inter-VLAN) ===
 switchport mode trunk
!
interface Vlan1
 no ip address
 shutdown
!
end
```

---

2. Architecture Applicative Conteneurisée (Docker Compose)

2.1 Justification de la solution

Plusieurs solutions ont été étudiées, notamment l’installation manuelle sur le système d'exploitation et la virtualisation lourde via des machines virtuelles dédiées. La solution basée sur **Docker Compose** a été retenue car elle permet de déployer rapidement une architecture complète et multiconteneur à partir d’un fichier de configuration unique (docker-compose.yml),

tout en garantissant la légèreté des instances, l'isolation des processus et une parfaite reproductibilité de l'environnement.

2.2 Sécurisation par le Cloisonnement Réseau (Frontend / Backend)

Afin de renforcer la sécurité de l'application et de respecter les bonnes pratiques professionnelles, l'infrastructure applicative est découpée en deux réseaux virtuels Docker distincts (utilisant le driver *bridge*) :

- **Réseau Frontend :** Ce réseau est exposé vers l'extérieur. Le conteneur Web WordPress possède une interface dans ce réseau afin d'écouter sur le port standard HTTP (80) et répondre aux requêtes des clients.
- **Réseau Backend :** Ce réseau est totalement isolé et inaccessible depuis l'extérieur de l'hôte Docker. Le conteneur de base de données MySQL est placé exclusivement dans ce réseau. Le conteneur WordPress possède également une seconde interface dans ce réseau pour communiquer avec la base de données. Ainsi, la base de données est protégée contre toute tentative d'accès direct depuis le réseau externe.

2.3 Fichier de configuration docker-compose.yml

```
version: '3.8'

services:
  mysql:
    image: mysql:8.0
    container_name: mysql_db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: [SECRET_ROOT_PASSWORD]
      MYSQL_DATABASE: wordpress_db
      MYSQL_USER: wordpress_user
      MYSQL_PASSWORD: [SECRET_USER_PASSWORD]
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - backend

  wordpress:
    image: wordpress:latest
    container_name: wordpress_app
    restart: always
    ports:
      - "80:80"
    environment:
      WORDPRESS_DB_HOST: mysql:3306
      WORDPRESS_DB_USER: wordpress_user
      WORDPRESS_DB_PASSWORD: [SECRET_USER_PASSWORD]
      WORDPRESS_DB_NAME: wordpress_db
    volumes:
```

```
    - wp_data:/var/www/html
  networks:
    - frontend
    - backend
  depends_on:
    - mysql

volumes:
  db_data:
  wp_data:

networks:
  frontend:
    driver: bridge
  backend:
    driver: bridge
```

2.4 Persistance des Données (Volumes)

Par défaut, les conteneurs Docker sont éphémères : toute modification de fichier est perdue à la suppression du conteneur. Pour assurer la persistance des données, deux volumes Docker nommés ont été mis en œuvre :

- db_data : Lié au dossier /var/lib/mysql du conteneur MySQL pour conserver les tables de la base de données.
- wp_data : Lié au dossier /var/www/html du conteneur WordPress pour préserver les fichiers du site web (thèmes, extensions, médias uploads).

---

3. Procédures de Vérification et d'Administration

3.1 Validation du fonctionnement des conteneurs

La commande docker ps permet de s'assurer que seuls les conteneurs légitimes de l'application s'exécutent, tout en contrôlant l'exposition des ports :

```
# docker ps
- Container wordpress_app (Up - Ports: 0.0.0.0:80->80/tcp) -> Accessible publiquement
- Container mysql_db      (Up - Ports: 3306/tcp)         -> Strictement interne
```

3.2 Validation du cloisonnement des réseaux Docker

La commande docker network ls confirme la création et l'isolation des réseaux logiques :

```
# docker network ls
```

- rp-docker_frontend (driver: bridge, scope: local)
- rp-docker_backend (driver: bridge, scope: local)

3.3 Validation de l'adressage IP de l'hôte Debian

La commande ip a permet de valider la présence de l'interface virtuelle liée au VLAN 20 du lycée ainsi que la création des ponts réseaux Docker (interfaces br-<id>) associés aux réseaux frontend et backend.