ARCHITECTURE CLIENT-SERVEUR

Projet : SERVEUR WEB (Public)

Documentation d’architecture

Date de début : 29 avril 2025
Date de fin : 20 juin 2025

GONZALES Arthur
PONS William

Présentation générale :

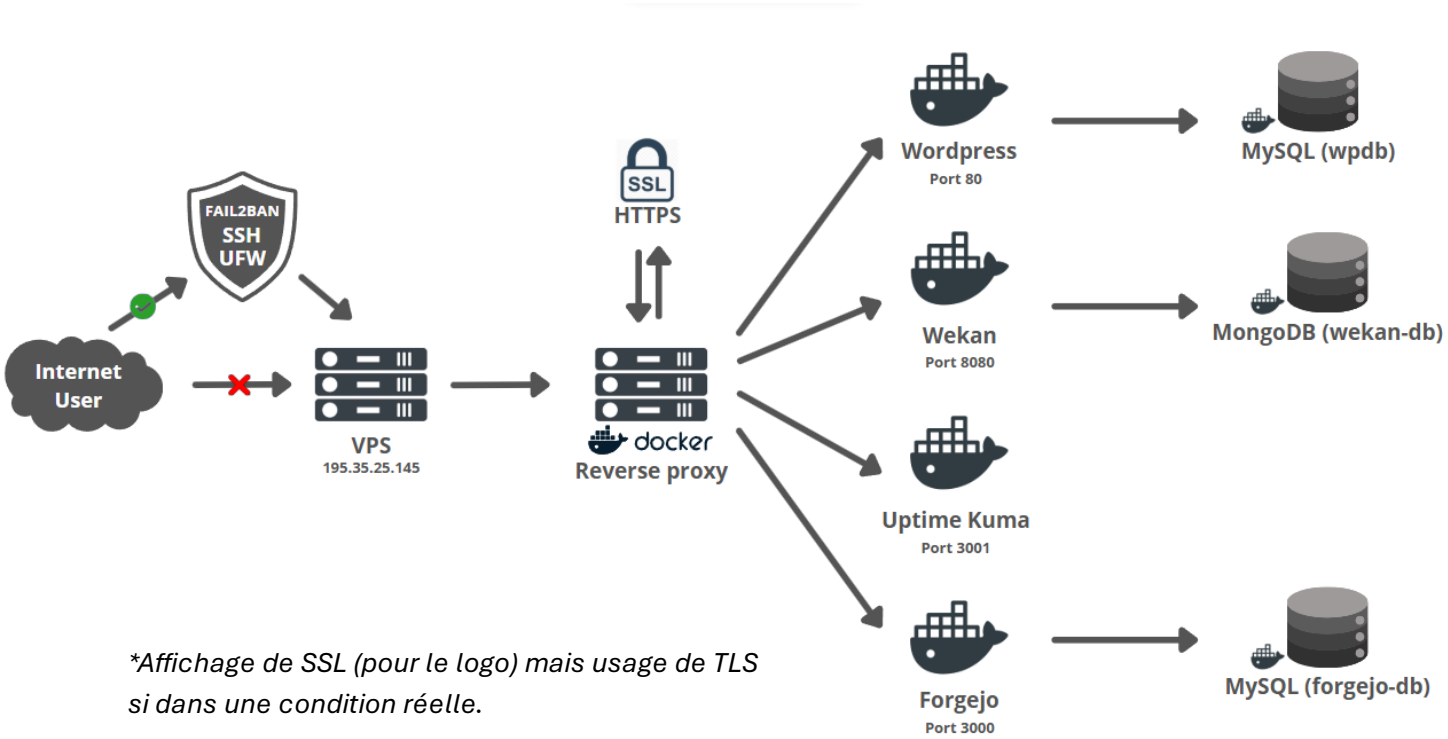
L'objectif de ce projet est de déployer un serveur web avancé, composé de plusieurs services développés dans des langages différents, tous accessibles via HTTPS avec certificats auto-signés. L'infrastructure est conçue pour séparer les rôles critiques (serveur applicatif vs reverse proxy) et pour rester facilement extensible directement en utilisant des conteneurs. En nous basant sur les consignes du document fourni, nous avons volontairement adapté la mise en situation afin de simuler un environnement exposé à des risques réels.

Contexte serveur :

- **Type :** VPS (machine virtuelle dédié)
- **Fournisseur :** Hostinger (KVM 2 vCores, 8 Go RAM)
- **Adresse IP publique :** 195.35.25.145
- **Système d'exploitation :** Debian 12 minimal (Les paquets sont supprimés)

Le VPS constitue l'unique point d'entrée du réseau, accessible directement via les ports 80 et 443 de l'adresse IP publique. Toutes les applications web sont isolées dans des conteneurs Docker, orchestrés à l'aide de Docker Compose, avec un reverse proxy également intégré dans l'environnement Docker en frontal.

Schéma logique :



**Affichage de SSL (pour le logo) mais usage de TLS si dans une condition réelle.*

Domaines locaux simulés (Windows / Linux) :

Répertoires à modifier pour appliquer le DNS local :
Windows : C:\Windows\System32\drivers\etc\hosts (fichier système)
Linux : \etc\hosts

Service	Domaine	Langage principal	Port exposé
Wordpress	b1vps.com	PHP	443 (HTTPS)
Wekan	wekan.b1vps.com	Node.js / JavaScript	443 (HTTPS)
Uptime Kuma	uptime.b1vps.com	Node.js / TypeScript	443 (HTTPS)
Forgejo	forgejo.b1vps.com	Golang	443 (HTTPS)

```
# Host personnalisé
195.35.25.145 uptime.b1vps.com
195.35.25.145 b1vps.com
195.35.25.145 wekan.b1vps.com
195.35.25.145 forgejo.b1vps.com
```

Services utilisés :

Reverse Proxy (NGINX) + Docker :

- Intercepte toutes les requêtes entrantes de l'utilisateur
- Redirige http (port 80) vers HTTPS (port 443) [passage via TLS]
- Gère les certificats TLS auto-signés, usage de Let's Encrypt pour un DNS Public
- Transfère chaque requête vers le bon conteneur via proxy_pass [conteneur:port]

Conteneurs de services (Docker) :

- WordPress : CMS en PHP, basé sur MySQL
- Wekan : gestion de tâches Kanban, basé sur MongoDB
- Uptime Kuma : supervision des services
- Forgejo : Git forge (équivalent à Gitea), basé sur Go + MySQL
- *Veuillez consulter le schéma logique pour obtenir les ports utilisés*

Organisation des fichiers :

```
webinfra/
├── docker-compose.yml
├── reverse-proxy/
│   ├── nginx.conf
│   └── certs/
│       ├── selfsigned.crt
│       └── selfsigned.key
```

L'ensemble de l'infrastructure repose sur un unique fichier **docker-compose.yml**, situé à la racine du projet dans le répertoire webinfra/. Ce fichier centralise la définition des services déployés : reverse proxy, WordPress, Wekan, Forgejo, etc.
Le dossier reverse-proxy/ contient le fichier **nginx.conf**, configuration principale du proxy inverse, ainsi qu'un sous-dossier certs/ regroupant les certificats TLS auto-signés (selfsigned.crt et selfsigned.key).

Réseaux Docker :

Docker permet de créer des réseaux virtuels isolés pour organiser la communication entre les conteneurs. Dans cette infrastructure, deux réseaux ont été définis dans le fichier **docker-compose.yml** : **frontend** et **backend**. Leur but est d'assurer une séparation logique et sécuritaire entre les composants exposés au public et ceux qui doivent rester en accès restreint.

Réseau frontend (R-P) : Ce réseau est destiné à recevoir les connexions entrantes depuis l'extérieur (HTTP/HTTPS). Actuellement, il n'est pas utilisé mais est présent à titre d'exemple dans la configuration du conteneur reverse-proxy. Il a vocation à écouter sur les ports 80 et 443, et permettrait de séparer logiquement les services exposés publiquement, comme une API REST, afin de mieux structurer l'architecture réseau et renforcer sa lisibilité.

Réseau backend (R-P) : Ce réseau est utilisé pour la communication interne entre les différents conteneurs déployés sur le VPS, tels que WordPress, Wekan, Uptime Kuma, Forgejo, ainsi que leurs bases de données respectives. Tous ces services sont reliés au reverse proxy via ce réseau interne. Cette architecture permet d'éviter toute exposition accidentelle des services à l'extérieur, tout en assurant une interconnexion fiable entre les applications et leurs bases de données.

Configuration du Docker (Extrait docker-compose.yml) :

```yaml
services:
  reverse-proxy:
    image: nginx:latest
    container_name: reverse-proxy
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./reverse-proxy/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./reverse-proxy/certs:/etc/nginx/certs:ro
    depends_on:
      - wordpress
      - wekan
      - uptime
    networks:
      - frontend
      - backend

  wordpress:
    image: wordpress:latest
    container_name: wordpress
    restart: always
    environment:
      WORDPRESS_DB_HOST: ${WORDPRESS_DB_HOST}
      WORDPRESS_DB_USER: ${WORDPRESS_DB_USER}
      WORDPRESS_DB_PASSWORD: ${WORDPRESS_DB_PASSWORD}
      WORDPRESS_DB_NAME: ${WORDPRESS_DB_NAME}
    networks:
      - backend
    volumes:
      - wordpress-html:/var/www/html

  db:
    image: mysql:5.7
    container_name: wpdb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${WORDPRESS_DB_NAME}
      MYSQL_USER: ${WORDPRESS_DB_USER}
      MYSQL_PASSWORD: ${WORDPRESS_DB_PASSWORD}
    volumes:
      - wpdata:/var/lib/mysql
    networks:
      - backend
```

Pour démarrer et stopper les conteneurs docker, il suffit de taper les commandes :

Start : docker compose up -d
Stop : docker compose down

*Les informations d'environnement des conteneurs Docker sont stockées dans un fichier **.env**, afin de renforcer la sécurité et d'éviter d'exposer des données sensibles*

Configuration du Reverse Proxy (NGINX) (Extrait nginx.conf) :

```
events {}

http {
    ssl_certificate /etc/nginx/certs/selfsigned.crt;
    ssl_certificate_key /etc/nginx/certs/selfsigned.key;

    server {
    listen 80;
    server_name b1vps.com uptime.b1vps.com wekan.b1vps.com;

    return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl;
        server_name b1vps.com;

        location / {
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_pass http://wordpress:80;
        }
    }
```

Voici la configuration du reverse proxy assurant la redirection en HTTPS vers le service WordPress via TLS, avec un port pointant directement vers le conteneur WordPress. Les clés TLS auto-signées utilisées pour établir la connexion sécurisée sont également spécifiées, confirmant l'usage du

Les explications détaillées concernant la configuration de chaque fichier ainsi que la mise en place des conteneurs Docker seront présentées dans la ***documentation d'exploitation****.*
C'est un choix de ma part de dissocier la partie « blabla » technique/théorique à la pratique, cela peut-être une erreur dans ces cas-là je m'en excuse.

Bonnes pratiques appliquées :

Séparation des rôles – Isolation du reverse proxy

Dans une logique de défense en profondeur, le rôle de reverse proxy a été isolé dans un conteneur Docker dédié, séparé de tous les autres services (tels que WordPress, Wekan, Forgejo, etc.). Cette approche respecte le principe "un service = un conteneur", ce qui permet :

- Une **meilleure lisibilité** de l'infrastructure.
- Une **limitation de la surface d'attaque** : si un service est compromis, les autres restent protégés.
- Une **gestion fine des accès réseau** entre conteneurs via les réseaux Docker (frontend (non utilisé) /backend).
- Un **redémarrage ou une mise à jour indépendante** du proxy sans impacter les autres applications.

Cela permet également d'**appliquer des règles spécifiques de sécurité et de journalisation** uniquement sur le reverse proxy, qui est le point d'entrée unique depuis l'extérieur.

Sécurité système - Renforcement de la machine
Plusieurs mesures ont été mises en œuvre pour durcir le système hôte et limiter les risques d'intrusion :

1. Pare-feu UFW (Uncomplicated Firewall)

Le pare-feu UFW a été activé et configuré pour autoriser uniquement les **ports strictement nécessaires** au fonctionnement des services :

> 80/tcp pour HTTP

> 443/tcp pour HTTPS

> 22/tcp pour SSH
(accès administrateur uniquement)

```
@srv864652:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
22/tcp                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Tous les autres ports entrants sont **bloqués par défaut**, assurant ainsi une **politique de sécurité par défaut restrictive** (deny by default).

2. Désactivation de l'accès SSH root

Pour éviter toute tentative d'attaque par force brute ciblant l'utilisateur root, **l'accès SSH direct à root a été désactivé** (*PermitRootLogin no* et *PasswordAuthentification no* dans **sshd_config**).

Seuls les **comptes utilisateurs explicitement autorisés** peuvent se connecter via SSH, avec ou sans clé SSH selon les cas. Cette méthode réduit fortement les tentatives automatisées d'intrusion sur le port SSH.

3. Installation de Fail2Ban

Fail2Ban a été installé et configuré pour surveiller les tentatives de connexion SSH. En cas de multiples échecs d'authentification sur une courte période, **l'adresse IP source est automatiquement bannie pendant une durée définie**.

Cela permet de **ralentir voire bloquer les attaques par force brute**, tout en enregistrant les tentatives malveillantes dans des journaux d'audit.

```
[sshd]
enabled = true
port = ssh
backend = auto
logpath = /var/log/auth.log
maxretry = 10
findtime = 10m
bantime = 30m
```

```
@srv864652:~$ apt install fail2ban
@srv864652:~$ sudo nano /etc/fail2ban/jail.d/sshd.conf
```

```
@srv864652:~$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd
```

4. Évolutivité et durcissement futur

Bien que les premières mesures de sécurité aient été mises en place, l'architecture reste **évolutive** et pourra être renforcée à tout moment.
Par exemple, pour améliorer encore la sécurité du VPS et de ses services, on peut prévoir de **restreindre davantage certains points sensibles**, notamment :

- Le contrôle plus fin des **accès aux fichiers de configuration du reverse proxy (NGINX)**.
- La mise en place de **règles d'accès IP sur certaines localisations sensibles** (accès admin, backend).
- L'ajout de **headers HTTP de sécurité** (HSTS, X-Frame-Options, X-XSS-Protection, etc.).
- Le passage à des **certificats signés par une autorité reconnue** (Let's Encrypt) en cas de mise en ligne publique.
- L'intégration d'une **authentification forte** pour l'accès aux interfaces critiques (panel admin, cockpit, etc.).

Cette modularité permet à l'infrastructure de s'adapter progressivement à un **niveau de sécurité plus élevé**, selon les besoins futurs du projet.

Veuillez trouver en page suivante les captures d'écran illustrant la configuration des conteneurs Docker essentiels au bon fonctionnement de l'infrastructure VPS.
*Les **accès aux différents services en ligne** seront quant à eux **détaillés dans la documentation d'exploitation**, afin de garantir une séparation claire entre la partie technique et l'usage utilisateur.*

Captures d’écrans :

Conteneurs docker en ligne (docker ps -a) :

```
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
71d1d24d740b   nginx:latest                           "/docker-entrypoint.…"   20 hours ago     Up 20 hours
b07137315327   wekanteam/wekan                        "bash -c 'ulimit -s …"   20 hours ago     Up 20 hours
8e9e654ea078   codeberg.org/forgejo/forgejo:11.0.1    "/usr/bin/entrypoint…"   20 hours ago     Up 20 hours
969839fd2384   mysql:8                                "docker-entrypoint.s…"   20 hours ago     Up 20 hours
d203b95b7d71   mysql:5.7                              "docker-entrypoint.s…"   20 hours ago     Up 20 hours
0cae9d01dc8e   louislam/uptime-kuma:latest            "/usr/bin/dumb-init …"   20 hours ago     Up 20 hours (healthy)
c465525631e2   wordpress:latest                       "docker-entrypoint.s…"   20 hours ago     Exited (0) 5 hours ago
e6f7951beff3   mongo:5.0                              "docker-entrypoint.s…"   20 hours ago     Up 20 hours

PORTS                                                                        NAMES
0.0.0.0:80->80/tcp, [::]:80->80/tcp, 0.0.0.0:443->443/tcp, [::]:443->443/tcp   reverse-proxy
8080/tcp                                                                     wekan
22/tcp, 3000/tcp                                                             forgejo
3306/tcp, 33060/tcp                                                          forgejo-db
3306/tcp, 33060/tcp                                                          wpdb
3001/tcp                                                                     uptime
                                                                             wordpress
27017/tcp                                                                    wekan-db
```

Contenu du docker-compose.yml (/opt/webinfra/…) :

```yaml
services:
  reverse-proxy:
    image: nginx:latest
    container_name: reverse-proxy
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./reverse-proxy/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./reverse-proxy/certs:/etc/nginx/certs:ro
    depends_on:
      - wordpress
      - wekan
      - uptime
      - forgejo
    networks:
      - frontend
      - backend

  wordpress:
    image: wordpress:latest
    container_name: wordpress
    restart: always
    environment:
      WORDPRESS_DB_HOST: ${WORDPRESS_DB_HOST}
      WORDPRESS_DB_USER: ${WORDPRESS_DB_USER}
      WORDPRESS_DB_PASSWORD: ${WORDPRESS_DB_PASSWORD}
      WORDPRESS_DB_NAME: ${WORDPRESS_DB_NAME}
    networks:
      - backend
    volumes:
      - wordpress-html:/var/www/html

  db:
    image: mysql:5.7
    container_name: wpdb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${WORDPRESS_DB_NAME}
      MYSQL_USER: ${WORDPRESS_DB_USER}
      MYSQL_PASSWORD: ${WORDPRESS_DB_PASSWORD}
    volumes:
      - wpdata:/var/lib/mysql
    networks:
      - backend

  wekan-db:
    image: mongo:5.0
    container_name: wekan-db
    restart: always
    volumes:
      - wekan-dbdata:/data/db
    networks:
      - backend

  wekan:
    image: wekanteam/wekan
    container_name: wekan
    restart: always
    environment:
      MONGO_URL: ${WEKAN_MONGO_URL}
      ROOT_URL: ${WEKAN_ROOT_URL}
      PORT: ${WEKAN_PORT}
    depends_on:
      - wekan-db
    networks:
      - backend

  uptime:
    image: louislam/uptime-kuma:latest
    container_name: uptime
    restart: always
    environment:
      ROOT_URL: ${UPTIME_ROOT_URL}
    expose:
      - "3001"
    networks:
      - backend
    volumes:
      - uptime-data:/app/data
      - /var/run/docker.sock:/var/run/docker.sock #écoute les dockers à l'exterieur

  forgejo:
    image: codeberg.org/forgejo/forgejo:11.0.1
    container_name: forgejo
    restart: always
    depends_on:
      - forgejo-db
    environment:
      - USER_UID=${FORGEJO_UID}
      - USER_GID=${FORGEJO_GID}
      - FORGEJO__database__DB_TYPE=${FORGEJO_DB_TYPE}
      - FORGEJO__database__HOST=${FORGEJO_DB_HOST}
      - FORGEJO__database__NAME=${FORGEJO_DB_NAME}
      - FORGEJO__database__USER=${FORGEJO_DB_USER}
      - FORGEJO__database__PASSWD=${FORGEJO_DB_PASS}
      - FORGEJO__server__DOMAIN=${FORGEJO_DOMAIN}
      - FORGEJO__server__ROOT_URL=${FORGEJO_ROOT_URL}
      - FORGEJO__server__PROTOCOL=${FORGEJO_PROTOCOL}
    volumes:
      - forgejo-data:/data
    expose:
      - "3000"
    networks:
      - backend
```

Contenu du docker-compose.yml part 2 :

```yaml
  forgejo:
    image: codeberg.org/forgejo/forgejo:11.0.1
    container_name: forgejo
    restart: always
    depends_on:
      - forgejo-db
    environment:
      - USER_UID=${FORGEJO_UID}
      - USER_GID=${FORGEJO_GID}
      - FORGEJO__database__DB_TYPE=${FORGEJO_DB_TYPE}
      - FORGEJO__database__HOST=${FORGEJO_DB_HOST}
      - FORGEJO__database__NAME=${FORGEJO_DB_NAME}
      - FORGEJO__database__USER=${FORGEJO_DB_USER}
      - FORGEJO__database__PASSWD=${FORGEJO_DB_PASS}
      - FORGEJO_server_DOMAIN=${FORGEJO_DOMAIN}
      - FORGEJO_server_ROOT_URL=${FORGEJO_ROOT_URL}
      - FORGEJO_server_PROTOCOL=${FORGEJO_PROTOCOL}
    volumes:
      - forgejo-data:/data
    expose:
      - "3000"
    networks:
      - backend

  ▷ Run Service
  forgejo-db:
    image: mysql:8
    container_name: forgejo-db
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=${FORGEJO_MYSQL_ROOT_PASSWORD}
      - MYSQL_USER=${FORGEJO_DB_USER}
      - MYSQL_PASSWORD=${FORGEJO_DB_PASS}
      - MYSQL_DATABASE=${FORGEJO_DB_NAME}
    volumes:
      - forgejo-mysqldata:/var/lib/mysql
    networks:
      - backend

volumes:
  wpdata:
  wekan-dbdata:
  uptime-data:
  wordpress-html:
  forgejo-data:
  forgejo-mysqldata:

networks:
  frontend:
  backend:
```

Contenu du nginx.conf (../reverse-proxy/...) :

```
events {}

http {
    ssl_certificate /etc/nginx/certs/selfsigned.crt;
    ssl_certificate_key /etc/nginx/certs/selfsigned.key;

    server {
    listen 80;
    server_name b1vps.com uptime.b1vps.com wekan.b1vps.com forgejo.b1vps.com;

    return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl;
        server_name b1vps.com;

        location / {
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_pass http://wordpress:80;
        }
    }


    server {
        listen 443 ssl;
        server_name wekan.b1vps.com;

        location / {
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_pass http://wekan:8080;
        }
    }

    server {
        listen 443 ssl;
        server_name uptime.b1vps.com;

        location / {
            proxy_pass http://uptime:3001/;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }

    server {
        listen 443 ssl;
        server_name forgejo.b1vps.com;

        location / {
            proxy_pass http://forgejo:3000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }

}
```