VELIKO

Déploiement du Projet BTS-SIO-G12-2025-VELIKO-Web

Contexte et Objectifs

Le projet **BTS-SIO-G12-2025-VELIKO-Web** est une application Symfony intégrant des technologies front-end (TailwindCSS, Stimulus) et back-end (MariaDB, Mock API). Le déploiement de cette application repose sur une architecture distribuée sur **trois serveurs** distincts.

Ce compte rendu couvre les étapes détaillées du déploiement, les problèmes rencontrés, les solutions apportées et inclut la configuration HTTPS avec redirection.

Architecture Détaillée

1. **Serveur 1 : Application Web**
   - Fonction : Héberger l'application Symfony.
   - Logiciels : Apache2, PHP, Composer, Symfony CLI.
   - Problèmes : Permissions, configuration des variables d’environnement, dépendances manquantes.
2. **Serveur 2 : Services Associés**
   - Fonction : Héberger des services Docker tels que Mailhog (simulateur SMTP) et Mock API.
   - Problèmes : Ports réseau bloqués, absence de variables d’environnement.
3. **Serveur 3 : Base de Données**
   - Fonction : Héberger MariaDB avec accès distant sécurisé.
   - Problèmes : Connexions distantes refusées, permissions utilisateur insuffisantes.

Étapes de Déploiement : Serveur 1 (Application Web)

1. Préparation et Installation

1. **Mise à jour du système :**

```
sudo apt update && sudo apt upgrade -y
```

2. **Installation des dépendances nécessaires :**

```
sudo apt install apache2 php libapache2-mod-php composer unzip curl git
-y
```

2. Déploiement de l'Application Symfony

1. **Clonage du dépôt Git :**

```
git clone git@github.com:ort-montreuil/BTS-SIO-G12-2025-VELIKO-Web.git
/var/www/BTS-SIO-G12-2025-VELIKO-Web
```

2. **Installation des dépendances PHP :**

```
cd /var/www/BTS-SIO-G12-2025-VELIKO-Web
composer install --no-dev
php bin/console cache:clear --env=prod
```

3. **Configuration des permissions :**

```
sudo chown -R www-data:www-data /var/www/BTS-SIO-G12-2025-VELIKO-Web
sudo chmod -R 775 /var/www/BTS-SIO-G12-2025-VELIKO-Web/var
```

---

3. Configuration Apache

1. **Création du fichier VirtualHost :**

```
sudo nano /etc/apache2/sites-available/veliko.conf
```

Contenu du fichier :

```
<VirtualHost *:80>
    ServerName veliko.lan
    DocumentRoot /var/www/BTS-SIO-G12-2025-VELIKO-Web/public
```

```
    <Directory /var/www/BTS-SIO-G12-2025-VELIKO-Web/public>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/veliko-error.log
    CustomLog ${APACHE_LOG_DIR}/veliko-access.log combined
</VirtualHost>
```

2. **Activation du site et des modules nécessaires :**

```
sudo a2ensite veliko.conf
sudo a2enmod rewrite
sudo systemctl reload apache2
```

3. **Tests et résolution des problèmes courants :**

Problème	Solution
Erreur 500	Vérifiez les permissions et nettoyez le cache :
	`sudo chown -R www-data:www-data /var/www/BTS-SIO-G12-2025-VELIKO-Web`
	`php bin/console cache:clear --env=prod`
CSS/JS non chargés	Compiler les assets avec : `symfony console tailwind:build`
Fichiers `.env` mal configurés	Remplacer les valeurs fixes par `$_ENV` et ajuster le fichier `.env`.

Étapes de Déploiement : Serveur 2 (Services Associés)

1. Installation de Docker

1. **Installation de Docker et Docker Compose :**

```
sudo apt install docker.io -y
sudo apt install docker-compose -y
```

2. **Installation de Portainer pour la gestion :**

```
sudo docker run -d -p 9000:9000 --name=portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock portainer/portainer-ce
```

---

2. Déploiement des Services Docker

1. **Création du fichier `docker-compose.yml` :**

```
version: "3.8"
services:
  mailhog:
    image: mailhog/mailhog
    ports:
      - "8025:8025"
      - "1025:1025"
  mock_api:
    image: bfoujols/mock-veliko-api:latest
    ports:
      - "9042:80"
```

2. **Lancement des services :**

```
docker-compose up -d
```

3. **Résolution des problèmes courants :**

Problème	Solution
Ports réseau bloqués	Ouvrir les ports avec `sudo ufw allow 8025 && sudo ufw allow 9042`.
Variables manquantes	Ajouter les variables nécessaires dans `.env` et les inclure au conteneur.

Étapes de Déploiement : Serveur 3 (Base de Données)

1. Installation de MariaDB

1. **Installation :**

```
sudo apt install mariadb-server -y
```

2. **Configuration pour connexions distantes :**
   - Modifier `/etc/mysql/mariadb.conf.d/50-server.cnf` :

     ```
     bind-address = 0.0.0.0
     ```

   - Redémarrer MariaDB :

     ```
     sudo systemctl restart mariadb
     ```

3. **Création de la base de données et de l'utilisateur :**

```
CREATE DATABASE app_db;
CREATE USER 'symfony_user'@'%' IDENTIFIED BY 'Respons11';
GRANT ALL PRIVILEGES ON app_db.* TO 'symfony_user'@'%';
FLUSH PRIVILEGES;
```

Ajout de HTTPS

1. Installation de Certbot

1. **Installation :**

```
sudo apt install certbot python3-certbot-apache -y
```

2. **Obtention des certificats SSL :**

```
sudo certbot --apache -d veliko.lan
```

2. Configuration des redirections

1. **Modification des fichiers Apache :**
   - `/etc/apache2/sites-available/veliko.conf` :

     ```
     <VirtualHost *:80>
         ServerName veliko.lan
     ```

```
    Redirect permanent / https://veliko.lan/
</VirtualHost>
```

- `/etc/apache2/sites-available/veliko-le-ssl.conf` :

```
<VirtualHost *:443>
    ServerName veliko.lan
    DocumentRoot /var/www/BTS-SIO-G12-2025-VELIKO-Web/public

    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/veliko.lan/fullchain.pem
    SSLCertificateKeyFile
/etc/letsencrypt/live/veliko.lan/privkey.pem

    <Directory /var/www/BTS-SIO-G12-2025-VELIKO-Web/public>
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

2. **Activation et redémarrage :**

```
sudo a2ensite veliko-le-ssl.conf
sudo a2enmod ssl
sudo systemctl reload apache2
```

3. **Renouvellement automatique :**
   - Ajout dans cron :

```
sudo crontab -e
```

   - Ligne à ajouter :

```
0 3 * * * certbot renew --quiet
```

Résumé des Problèmes Majeurs et Solutions

Problème	Cause	Solution
Erreur 500	Permissions ou cache Symfony	Réparer permissions et nettoyer le cache.

Problème	Cause	Solution
Connexion distante refusée	`bind-address` mal configuré	Modifier à `0.0.0.0` et vérifier privilèges utilisateur.
Ports réseau bloqués	Pare-feu	Autoriser avec `sudo ufw allow <port>`.
HTTPS non configuré	Absence de certificats SSL	Installer Certbot et configurer les redirections.

Étapes Complémentaires pour le Déploiement Optimal

Automatisation des Déploiements

Pour automatiser la mise à jour du projet lors d'un nouveau push sur GitHub, nous avons mis en place un système de webhooks avec **smee.io** et un script de déploiement.

1. Mise en Place des Webhooks

1. **Inscription et Configuration sur smee.io**
   - Rendez-vous sur https://smee.io pour générer une URL unique, par exemple : `https://smee.io/unique-id`.
2. **Ajout d'un Webhook GitHub**
   - Dans le dépôt GitHub, allez dans **Settings > Webhooks**.
   - Ajoutez l'URL générée par smee.io.
   - Définissez les paramètres :
     - **Payload URL** : `https://smee.io/unique-id`.
     - **Content type** : `application/json`.
     - Cochez **Just the push event**.
   - Validez.

2. Script de Déploiement

1. **Création du Script :**

```
sudo nano /var/www/deploy.sh
```

Contenu du script :

```
#!/bin/bash
cd /var/www/BTS-SIO-G12-2025-VELIKO-Web
```

```
git pull origin main
composer install --no-interaction --no-dev --optimize-autoloader
php bin/console cache:clear --env=prod
php bin/console cache:warmup --env=prod
systemctl reload apache2
```

- Rendez le script exécutable :

```
sudo chmod +x /var/www/deploy.sh
```

2. **Test du Script :**
   - Exécutez manuellement :

```
/var/www/deploy.sh
```

   - Vérifiez qu'aucune erreur ne survient.

3. Exécution Automatique avec smee.io et Supervisor

1. **Installation de Supervisor :**

```
sudo apt install supervisor -y
```

2. **Configuration de Supervisor pour smee.io :**
   - Créez un fichier de configuration :

```
sudo nano /etc/supervisor/conf.d/smee.conf
```

   - Contenu :

```
[program:smee]
command=/usr/bin/smee -u https://smee.io/unique-id -p 3000 | bash
/var/www/deploy.sh
autostart=true
autorestart=true
stdout_logfile=/var/log/smee.log
stderr_logfile=/var/log/smee-error.log
```

   - Rechargez Supervisor :

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start smee
```

Optimisations et Bonnes Pratiques

1. Sécurisation Avancée

- **Pare-feu :**
  - Activer et configurer `ufw` :

```
sudo ufw enable
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 22
sudo ufw allow 8025
sudo ufw allow 9042
```

- **SSH :**
  - Désactiver les connexions SSH par mot de passe :

```
sudo nano /etc/ssh/sshd_config
```

Modifiez les lignes suivantes :

```
PasswordAuthentication no
PermitRootLogin no
```

  - Redémarrez le service :

```
sudo systemctl restart sshd
```

2. Surveillance et Logs

- **Surveillance des ressources :**
  - Installez `htop` et `nmon` :

```
sudo apt install htop nmon -y
```

- **Surveillance des logs :**
  - Configurez `logrotate` pour nettoyer régulièrement les fichiers logs :

```
sudo nano /etc/logrotate.d/apache2
```

Exemple de configuration :

```
/var/log/apache2/*.log {
    weekly
    missingok
    rotate 12
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    postrotate
        /etc/init.d/apache2 reload > /dev/null
    endscript
}
```

3. Tests et Validation

1. **Validation de la Connexion HTTPS :**
   - Assurez-vous que les redirections fonctionnent correctement :
     - Essayez `http://veliko.lan` → redirige vers `https://veliko.lan`.
2. **Tests Fonctionnels :**
   - Validez que l'application fonctionne en production, notamment :
     - Connexion utilisateur.
     - Accès aux APIs.
     - Chargement des CSS/JS.
3. **Validation des Performances :**
   - Exécutez un test de charge avec `Apache Benchmark` (installé via `ab`) :

```
ab -n 1000 -c 10 https://veliko.lan/
```

Résumé Final

Configuration Réalisée :

1. **Serveur 1 : Application Web**
   - Symfony déployé avec Apache, composer et PHP.
   - Configuration HTTPS avec Certbot.
   - Gestion des permissions et des dépendances.
2. **Serveur 2 : Services Docker**
   - Mailhog et Mock API déployés via Docker Compose.
   - Pare-feu configuré pour les ports.
3. **Serveur 3 : Base de Données**
   - MariaDB configuré avec accès distant sécurisé.
   - Permissions utilisateur correctement appliquées.
4. **Automatisation**
   - Webhooks GitHub intégrés via smee.io.
   - Script de déploiement automatisé avec Supervisor.

Problèmes Résolus :

Problème	Solution
Erreur 500	Réparation des permissions et gestion du cache Symfony.
Connexion distante MariaDB	Configuration de `bind-address` et des privilèges.
Ports réseau bloqués	Configuration du pare-feu avec `ufw`.
Absence de HTTPS	Certificat SSL avec Certbot et redirections configurées.
Automatisation du déploiement	Intégration de webhooks avec smee.io et scripts Bash.
