DOCUMENTATION DE DÉPLOIEMENT

Innov'Events

ECF — Concepteur Développeur d'Applications (Studi)
Février 2026

1. Objectif et périmètre

Cette documentation décrit comment j'ai configuré et déployé Innov'Events sur un VPS Hostinger (Ubuntu 24.04 LTS). L'objectif est d'avoir une procédure reproductible : repartir d'un VPS vierge et retrouver le même résultat.

Le code est maintenu sur GitHub, mais le déploiement en production a été réalisé manuellement via FileZilla (SFTP). Je n'ai pas finalisé le déploiement synchronisé (CI/CD) : une proposition de pipeline est décrite en fin de document.

2. Architecture de production

En production, l'architecture est la suivante :

Domaines (DNS) :

- innovevents-app.fr (frontend + proxy API via /api/)
- api.innovevents-app.fr (API si exposée directement)
- mobile.innovevents-app.fr (optionnel)

Composant	Rôle	Implémentation	Remarques
Nginx	Serveur web + reverse-proxy HTTPS	Service système (host)	Route le front et proxy /api + /uploads vers le backend
Frontend	Interface Angular (SPA)	Fichiers statiques sur le host	Servi par Nginx (root : /var/www/innovevents-front)
Backend	API PHP	Conteneur Docker	Exposé en local (ex : 127.0.0.1:8080) ; Nginx reverse-proxy
MySQL	Base relationnelle	Conteneur Docker	Réseau interne Docker (pas d'exposition publique)
MongoDB	Base NoSQL	Conteneur Docker	Réseau interne Docker (pas d'exposition publique)
phpMyAdmin	Admin MySQL	Conteneur Docker	Optionnel ; à restreindre si exposé

Commande utile pour vérifier les ports réellement exposés :

```
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Ports}}"
```

3. Préparation du VPS (Ubuntu 24.04 LTS)

3.1 Mise à jour système

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

3.2 Installation de Nginx

```
sudo apt install -y nginx
sudo systemctl enable --now nginx
```

3.3 Installation de Docker et Docker Compose

J'utilise Docker pour isoler le backend et les bases.

```
sudo apt install -y docker.io docker-compose-plugin
sudo systemctl enable --now docker
docker --version
docker compose version
```

3.4 Arborescence projet

Le projet est placé dans /opt/innovevents (backend + frontend + docker-compose.yml).

```
sudo mkdir -p /opt/innovevents
sudo chown -R $USER:$USER /opt/innovevents
ls -lah /opt/innovevents
```

4. Déploiement du code (FileZilla / SFTP)

Le déploiement en production est réalisé manuellement : j'envoie les dossiers sur le VPS via FileZilla en SFTP.

4.1 Connexion SFTP

Paramètres typiques :

- Protocole : SFTP
- Hôte : IP du VPS Hostinger
- Port : 22
- Authentification : identifiants SSH (ou clé SSH si configurée)

4.2 Upload du projet

J'envoie (au minimum) ces éléments vers /opt/innovevents :

- backend/ (API PHP)
- frontend/ (Angular : src + dist si déjà build)
- docker-compose.yml
- dump SQL (ex : innovevents.sql) si initialisation base

Bon réflexe : éviter d'envoyer node_modules. En production, je privilégie dist/ pour le front.

5. Mise en route Docker (API PHP + MySQL + MongoDB + phpMyAdmin)

Depuis le VPS, je démarre les services Docker avec docker compose.

5.1 Démarrage des conteneurs

```
cd /opt/innovevents
sudo docker compose up -d
sudo docker ps
```

5.2 Rebuild uniquement du backend (recommandé)

Quand je modifie uniquement l'API, je rebuild seulement le container backend. Ça limite les risques sur les volumes MySQL/MongoDB.

```
cd /opt/innovevents
sudo docker compose up -d --build backend
```

5.3 Upload d'images : permissions (point critique)

Pour que l'upload fonctionne, le dossier backend/uploads doit être accessible en écriture par l'utilisateur du serveur web (www-data).

```
sudo chown -R www-data:www-data /opt/innovevents/backend/uploads
sudo find /opt/innovevents/backend/uploads -type d -exec chmod 755 {} \;
sudo find /opt/innovevents/backend/uploads -type f -exec chmod 644 {} \;
```

```
sudo docker exec -it ecf_backend ls -ld /var/www/html/uploads /var/www/html/uploads/prospects
/var/www/html/uploads/events
```

6. Configuration Nginx (reverse-proxy + SPA + uploads)

Nginx sert le frontend Angular (fichiers statiques) et reverse-proxy les requêtes API et uploads vers le backend (Docker).

6.1 Dossier du frontend

Le frontend est servi par Nginx depuis : /var/www/innovevents-front.

```
sudo mkdir -p /var/www/innovevents-front
sudo chown -R www-data:www-data /var/www/innovevents-front
ls -lah /var/www/innovevents-front
```

6.2 Exemple de configuration Nginx (innovevents-front)

Le point clé est : /api/ proxy vers le backend, /uploads/ proxy vers le backend, et fallback SPA sur /index.html.

```
# /etc/nginx/sites-enabled/innovevents-front
server {
    listen 80;
    server_name innovevents-app.fr www.innovevents-app.fr;
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    server_name innovevents-app.fr www.innovevents-app.fr;

    root /var/www/innovevents-front;
    index index.html;

    ssl_certificate /etc/letsencrypt/live/innovevents-app.fr/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/innovevents-app.fr/privkey.pem;

    location /api/ {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        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;
    }

    location ^~ /uploads/ {
        proxy_pass http://127.0.0.1:8080/uploads/;
        proxy_http_version 1.1;
        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;
    }
```

```
    location / {
        try_files $uri $uri/ /index.html;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff2?|ttf|eot)$ {
        expires 7d;
        add_header Cache-Control "public, max-age=604800";
        try_files $uri =404;
    }
}
```

6.3 Test et rechargement Nginx

```
sudo nginx -t
sudo systemctl reload nginx
```

7. HTTPS (Let's Encrypt)

Le site est accessible en HTTPS via Let's Encrypt (certbot).

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d innovevents-app.fr -d www.innovevents-app.fr
sudo certbot --nginx -d api.innovevents-app.fr
# optionnel
sudo certbot --nginx -d mobile.innovevents-app.fr
```

```
sudo systemctl status certbot.timer
```

8. Données : import SQL et vérifications

Pour initialiser la base relationnelle, j'utilise un dump SQL exporté depuis l'environnement local. L'import peut se faire via phpMyAdmin ou via la ligne de commande.

8.1 Import via ligne de commande (reproductible)

```
cd /opt/innovevents
sudo docker exec -i ecf_db mysql -u root -proot innovevents < ./innovevents.sql
sudo docker exec -it ecf_db mysql -u root -proot -e "SHOW TABLES;" innovevents
```

8.2 Import via phpMyAdmin

phpMyAdmin > base innovevents > Importer > sélectionner le fichier SQL > Exécuter.

8.3 Vérifications rapides

```
curl -s https://innovevents-app.fr/api/prospects/read.php | head -n 60
curl -I https://innovevents-app.fr/uploads/events/nom_image.jpg
```

9. Procédure de mise à jour (sans casse)

Je fais les mises à jour par petites étapes, en ne touchant qu'à ce qui change. Le principe : upload des fichiers concernés, puis rebuild ciblé.

9.1 Mise à jour du backend (API PHP)

1. Envoyer les fichiers modifiés via FileZilla dans /opt/innovevents/backend
2. Rebuild uniquement du backend

```
cd /opt/innovevents
sudo docker compose up -d --build backend
```

3. Tester l'endpoint concerné (curl ou interface)

9.2 Mise à jour du frontend (Angular)

Option simple : build en local, puis upload du dist vers /var/www/innovevents-front.

4. Build en local
5. Upload du dist vers /var/www/innovevents-front
6. Rechargement navigateur (Ctrl+F5)

9.3 Nginx

Je ne modifie Nginx que si nécessaire (routes / domaines).

```
sudo nginx -t
sudo systemctl reload nginx
```

10. Vérifications de recette (avant / après mise en prod)

Recette courte à faire après déploiement ou mise à jour :

- Connexion / déconnexion
- Création prospect via demande de devis
- Upload image prospect (si utilisé)
- Liste prospects (filtre / recherche)
- Création / modification d'un événement + upload image
- Création d'une tâche et affichage
- Avis client (si dispo)
- Vérifier qu'aucune page ne renvoie 404 et que /api/ répond

Commandes utiles (diagnostic rapide) :

```
sudo docker ps
sudo docker logs --tail=200 ecf_backend
sudo nginx -t
sudo systemctl status nginx
```

Je reste volontairement léger sur les logs dans ce document (demande du projet), mais ces commandes suffisent pour un diagnostic rapide.

11. Notes CI/CD (prévu / proposition GitHub Actions)

L'objectif initial était un déploiement synchronisé depuis GitHub (push sur main -> contrôles -> déploiement). Je ne l'ai pas finalisé : la production est restée en déploiement manuel via FileZilla.

Ci-dessous, une proposition simple de pipeline GitHub Actions. Elle nécessite un dépôt cloné sur le VPS et des secrets GitHub (clé SSH, host, user).

11.1 Exemple de workflow (.github/workflows/ci-cd.yml)

```
name: ci-cd
on:
  push:
    branches: [ "main" ]
```

```
jobs:
  build_and_deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Frontend install
        working-directory: frontend
        run: npm ci

      - name: Frontend build
        working-directory: frontend
        run: npm run build

      - name: Deploy (SSH)
        uses: appleboy/ssh-action@v1.0.3
        with:
          host: ${{ secrets.VPS_HOST }}
          username: ${{ secrets.VPS_USER }}
          key: ${{ secrets.VPS_SSH_KEY }}
          script: |
            cd /opt/innovevents
            ./deploy.sh
```

11.2 Exemple de script deploy.sh (sur le VPS)

```
#!/usr/bin/env bash
set -euo pipefail

echo "[1/3] Mise à jour du code"
git pull --ff-only

echo "[2/3] Rebuild backend"
docker compose up -d --build backend

echo "[3/3] Terminé"
```