EnvironnementDevIRIS

Documentation Technique Complète

Environnement de développement Docker isolé
pour étudiants et développeurs — IRIS / MediaSchool

Version 1.0 • Avril 2026

1. Présentation du projet

1.1 Qu'est-ce que EnvironnementDevIRIS ?

EnvironnementDevIRIS est un environnement de développement web complet, livré sous forme de stack Docker Compose. Il permet à chaque étudiant ou développeur de disposer, en une seule commande, d'une application web fonctionnelle accessible via HTTPS sur un domaine dédié.

Le projet est conçu pour être partagé sur un serveur commun (type serveur pédagogique IRIS) où plusieurs utilisateurs cohabitent. Chaque utilisateur obtient automatiquement ses propres URLs, totalement isolées des autres.

1.2 À quoi sert ce projet ?

Ce projet sert à plusieurs choses :

- Fournir un environnement de développement web clé en main pour les étudiants de MediaSchool.
- Permettre à plusieurs apprenants de travailler sur le même serveur sans se gêner.
- Apprendre l'architecture web moderne : frontend, backend, base de données, reverse proxy.
- Tester et développer des applications PHP avec une base de données MariaDB.
- Accéder à phpMyAdmin pour gérer la base de données via une interface graphique.

1.3 Ce que vous obtenez

Service	Rôle	URL d'accès
Frontend (Nginx)	Sert les fichiers HTML/CSS/JS et transfère les requêtes PHP au backend	https://env-<user>.iris.a3n.fr
Backend (PHP-FPM)	Exécute le code PHP, accède à la base de données	Interne (pas d'URL publique)
Base de données (MariaDB)	Stocke les données de l'application	Interne (accessible via phpMyAdmin)
phpMyAdmin	Interface graphique pour gérer la base de données	https://pma-env-<user>.iris.a3n.fr
Traefik (externe)	Reverse proxy HTTPS, gère les certificats SSL automatiquement	Déjà en place sur le serveur

Exemple pour un utilisateur Linux nommé tiago :

- Frontend : https://env-tiago.iris.a3n.fr
- phpMyAdmin : https://pma-env-tiago.iris.a3n.fr

2. Architecture technique

2.1 Vue d'ensemble

Le projet est composé de 4 conteneurs Docker orchestrés par Docker Compose, plus un reverse proxy Traefik déjà installé sur le serveur.

Schéma simplifié du flux de données

```
Navigateur de l'utilisateur
    |  HTTPS (port 443)
    v
 [ Traefik ]  <-- Reverse proxy / SSL termination
    |
    v
 [ Frontend : Nginx ]  <-- Fichiers statiques (HTML, CSS, JS)
    |  Requêtes PHP (.php)
    v
 [ Backend : PHP-FPM ]  <-- Exécution du code PHP
    |  Requêtes SQL
    v
 [ Base de données : MariaDB ]
```

2.2 Détail des composants

Frontend — Nginx

Le frontend est basé sur Nginx Alpine (image légère). Il joue deux rôles :

- Serveur de fichiers statiques : il sert directement les fichiers HTML, CSS, JavaScript et les polices (FontAwesome inclus).
- Proxy FastCGI : quand une URL se termine par .php, Nginx transmet la requête au conteneur backend via le protocole FastCGI sur le port 9000.

La page d'accueil (index.html) est le portail MediaSchool : un dashboard élégant avec fond étoilé (particles.js), qui affiche en temps réel les ressources du serveur (CPU, RAM, disque) en appelant l'API PHP du backend.

Backend — PHP 8.2 FPM

Le backend utilise PHP 8.2 en mode FPM (FastCGI Process Manager) sur une image Alpine. Il inclut :

- L'extension PDO et PDO_MySQL pour se connecter à MariaDB.
- Composer (gestionnaire de paquets PHP) pour ajouter des dépendances.
- Le client MySQL pour des tests en ligne de commande.

Deux fichiers PHP sont fournis par défaut :

- index.php — Endpoint de test : répond "Pong !" pour vérifier que la liaison frontend/backend fonctionne.
- host-info.php — API JSON qui lit /proc/cpuinfo et /proc/meminfo pour retourner les infos CPU, RAM et disque du serveur.

Base de données — MariaDB 11.5

MariaDB est le système de gestion de base de données relationnelle. Il est configuré avec :

- Base de données : mediaschooldb
- Utilisateur applicatif : mediaschooluser / mediaschoolpass
- Mot de passe root : rootpass

> **Important**
> Ces identifiants sont des valeurs par défaut pour un environnement de développement. Ne pas utiliser en production !

phpMyAdmin

phpMyAdmin est une interface web qui permet de gérer la base de données sans connaître SQL. Il est accessible via son propre sous-domaine HTTPS et est connecté automatiquement à MariaDB.

2.3 Réseau interne

Les conteneurs communiquent via deux réseaux Docker :

- app-net : réseau interne privé partagé entre frontend, backend, MariaDB et phpMyAdmin. Aucun accès externe direct.
- admin_proxy : réseau externe partagé avec Traefik. Seuls le frontend et phpMyAdmin y sont connectés pour recevoir le trafic public.

3. Structure des fichiers

Voici l'organisation complète du projet :

```
EnvironnementDevIRIS/
├── docker-compose.yml      # Définit tous les services et leur configuration
├── start.sh                # Script de démarrage (génère .env + lance Docker)
├── stop.sh                 # Script d'arrêt propre
├── .env                    # Variables d'environnement (généré
automatiquement)
├── readme.md               # Documentation originale
│
├── backend/
│   ├── dockerfile          # Image PHP 8.2 FPM Alpine + extensions
│   └── src/
│       ├── index.php       # Endpoint de test ("Pong !")
│       └── host-info.php   # API JSON des ressources système
│
└── frontend/
    ├── dockerfile          # Image Nginx Alpine
    ├── nginx.conf          # Configuration du serveur web et proxy FastCGI
    └── src/
        ├── index.html      # Portail MediaSchool (dashboard principal)
        ├── css/            # Feuilles de style FontAwesome
        └── webfonts/       # Polices d'icônes FontAwesome (.woff2)
```

4. Prérequis

4.1 Côté serveur

Le serveur Linux doit avoir les éléments suivants installés et opérationnels :

Prérequis	Commande de vérification	Remarque
Docker Engine	docker --version	Version récente recommandée
Docker Compose Plugin	docker compose version	Pas le docker-compose legacy
Traefik (reverse proxy)	docker ps | grep traefik	Doit déjà tourner sur le serveur
Réseau admin_proxy	docker network ls | grep admin_proxy	Créer si absent (voir ci-dessous)

Si le réseau admin_proxy n'existe pas encore, le créer avec :

```
docker network create admin_proxy
```

4.2 Côté machine locale

- Un terminal avec accès SSH au serveur.
- La commande scp pour transférer les fichiers (ou Git si le serveur y a accès).
- Aucune installation Docker nécessaire en local.

5. Déploiement et utilisation

5.1 Transférer le projet sur le serveur

Depuis votre machine locale, copiez le projet sur le serveur avec scp :

```
# Exemple : copier dans le dossier home de l'utilisateur sur le serveur
scp -r ./EnvironnementDevIRIS utilisateur@serveur.iris.a3n.fr:~/
```

Vous pouvez aussi cloner depuis Git si le dépôt est accessible depuis le serveur :

```
git clone <url-du-dépôt> ~/EnvironnementDevIRIS
```

5.2 Démarrer l'environnement

Connectez-vous au serveur en SSH, puis naviguez dans le dossier du projet :

```
ssh utilisateur@serveur.iris.a3n.fr
cd ~/EnvironnementDevIRIS
bash start.sh
```

Le script start.sh fait automatiquement ces étapes :

1. Lit votre nom d'utilisateur Linux ($USER).
2. Génère automatiquement le fichier .env avec USERNAME et COMPOSE_PROJECT_NAME.
3. Affiche vos URLs personnalisées dans le terminal.
4. Lance docker compose up -d --build pour construire et démarrer les conteneurs.

Exemple de sortie dans le terminal :

```
------------------------------------------------
  Environnement Dev IRIS : tiago
------------------------------------------------
  Frontend    : https://env-tiago.iris.a3n.fr:4433
  phpMyAdmin  : https://pma-env-tiago.iris.a3n.fr:4433
------------------------------------------------
[+] Building...
[+] Running 4/4
  Container tiago-db-1           Started
  Container tiago-backend-1      Started
  Container tiago-phpmyadmin-1   Started
  Container tiago-frontend-1     Started
```

5.3 Accéder à l'application

Une fois démarré, ouvrez votre navigateur et rendez-vous sur vos URLs :

- Application web : https://env-<votre-user>.iris.a3n.fr
- Gestion base de données : https://pma-env-<votre-user>.iris.a3n.fr

Connexion phpMyAdmin
Serveur : db
Utilisateur : mediaschooluser
Mot de passe : mediaschoolpass

5.4 Arrêter l'environnement

```
bash stop.sh
```

Cette commande arrête et supprime proprement tous les conteneurs. Les données MariaDB sont perdues (pas de volume persistant configuré par défaut).

6. Commandes utiles

6.1 Surveillance et debug

Action	Commande
Voir l'état des conteneurs	docker compose ps
Suivre les logs en temps réel	docker compose logs -f
Logs d'un seul service	docker compose logs -f frontend
Entrer dans un conteneur	docker compose exec backend sh
Redémarrer un service	docker compose restart backend
Reconstruire les images	docker compose up -d --build

6.2 Vérification du déploiement

Pour vérifier que tout fonctionne, testez ces URL depuis votre navigateur :

- URL principale : affiche le portail MediaSchool avec les stats système.
- URL/index.php : doit afficher "Pong ! requete bien recue".
- URL/host-info.php : doit retourner un JSON avec CPU, RAM et disque.

7. Développement et personnalisation

7.1 Modifier le frontend

Les fichiers frontend se trouvent dans frontend/src/. Modifiez index.html pour changer la page d'accueil. Tout fichier ajouté dans ce dossier sera automatiquement servi par Nginx.

> **Attention**
> Après toute modification des fichiers source, relancez : bash start.sh (ou docker compose up -d --build) pour reconstruire les images.

7.2 Modifier le backend

Les fichiers PHP se trouvent dans backend/src/. Ajoutez vos fichiers .php dans ce dossier. Ils seront accessibles depuis le frontend via les URLs se terminant par .php.

Exemple : créer backend/src/mon-api.php, accessible depuis :

```
https://env-<user>.iris.a3n.fr/mon-api.php
```

7.3 Utiliser la base de données en PHP

Exemple de connexion PDO à MariaDB depuis un fichier PHP :

```php
<?php
$dsn = 'mysql:host=db;dbname=mediaschooldb;charset=utf8';
$user = 'mediaschooluser';
$pass = 'mediaschoolpass';

try {
    $pdo = new PDO($dsn, $user, $pass);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo 'Connexion réussie !';
} catch (PDOException $e) {
    echo 'Erreur : ' . $e->getMessage();
}
?>
```

Le nom d'hôte de la base de données est db (nom du service Docker Compose), pas localhost.

8. Résolution de problèmes fréquents

Problème	Cause probable	Solution
L'URL affiche une erreur 404	Traefik ne connaît pas encore le conteneur	Attendre 10-15 secondes après le démarrage
Erreur de certificat SSL	Le domaine n'est pas encore résolu	Vérifier que les DNS pointent vers le serveur
Le conteneur ne démarre pas	Erreur dans un fichier de config	Consulter les logs : docker compose logs
La BDD ne répond pas	MariaDB met du temps à initialiser	Attendre et réessayer, ou docker compose restart db
Réseau admin_proxy introuvable	Le réseau n'a pas été créé	docker network create admin_proxy
Les modifications PHP ne s'affichent pas	L'image n'a pas été reconstruite	Relancer avec --build : bash start.sh

9. Résumé rapide

Voici les 4 commandes essentielles pour utiliser ce projet au quotidien :

```
# 1. Aller dans le dossier projet
cd ~/EnvironnementDevIRIS

# 2. Démarrer tout l'environnement
bash start.sh

# 3. Surveiller les logs
docker compose logs -f

# 4. Arrêter l'environnement
bash stop.sh
```

> **URLs générées (exemple pour l'utilisateur 'tiago')**
> Frontend (votre application) : https://env-tiago.iris.a3n.fr
> phpMyAdmin (base de données) : https://pma-env-tiago.iris.a3n.fr
>
> Remplacez 'tiago' par votre propre nom d'utilisateur Linux.