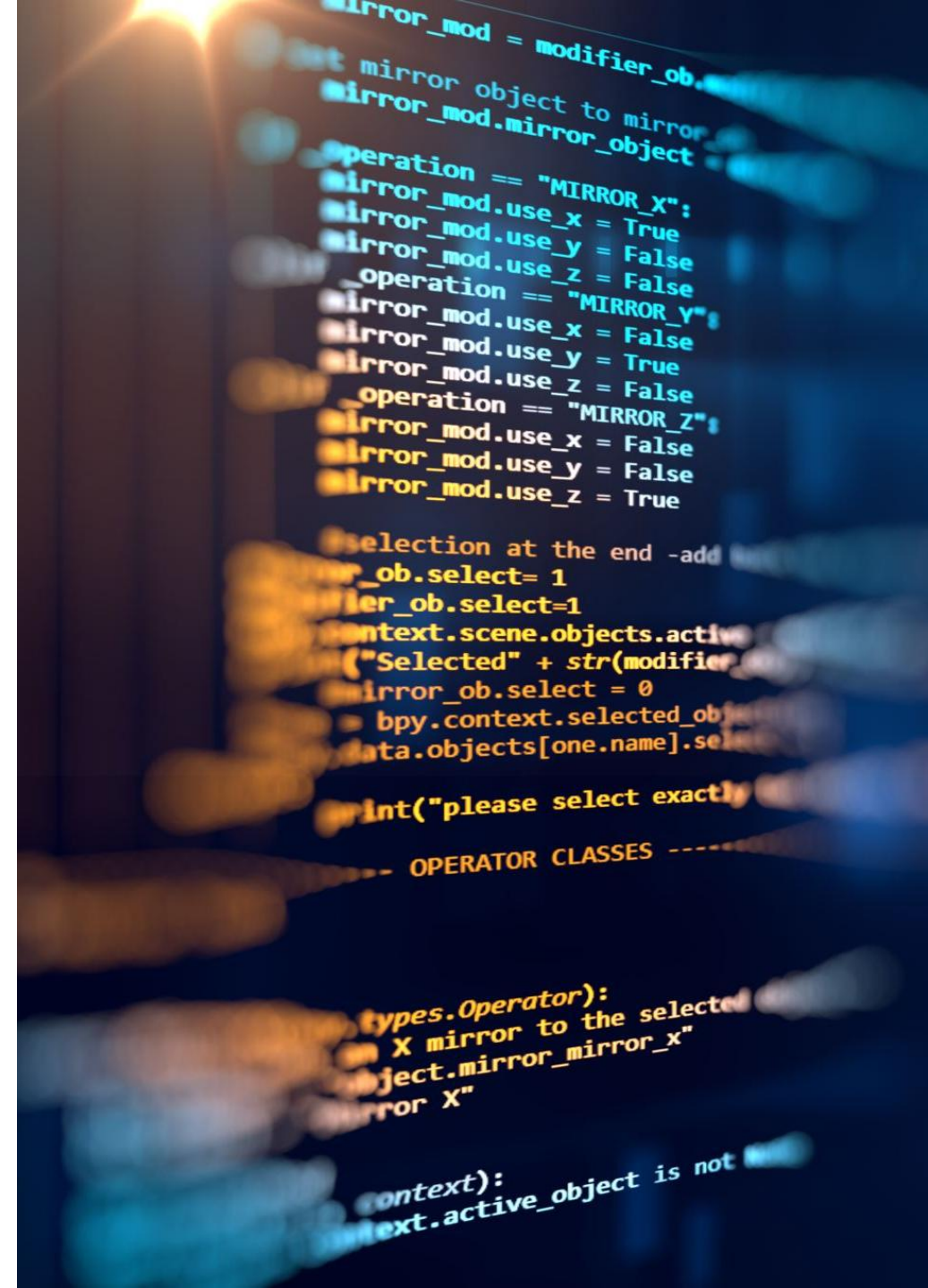


Documenter le  
déploiement d'une  
application dynamique  
web ou web mobile

# Procédure de déploiement

- Format texte, markdown, PDF ou dans un outil type Wiki/Confluence.
  - Étapes précises (préparation, transfert, configuration, tests post-déploiement)
  - Environnements concernés (développement, test, recette, production)
  - Rôles et responsabilités des intervenants
  - Solutions de repli en cas d'échec (rollback)
- Exemple d'étapes
  - Mise à jour du code depuis Git
  - Installation des dépendances (Composer, npm, etc.)
  - Migration de base de données
  - Compilation des assets (Webpack, Vite)
  - Tests automatisés
  - Redémarrage ou déploiement du service
  - Contrôle qualité post-déploiement







# Scripts de déploiement

- Outils usuels :
  - Bash (deploy.sh)
  - Docker (Dockerfile, docker-compose.yml)
  - CI/CD (.gitlab-ci.yml, GitHub Actions, Jenkinsfile)
- Exigences :
  - Scripts modulaires, versionnés, commentés
- Prévoir :
  - Installation des packages
  - Lancement des services
  - Exécution de tests
  - Nettoyage et gestion des logs
  - Utiliser des variables d'environnement sécurisées
  - Adapter aux outils du projet (ex. Symfony CLI, Laravel Forge, PM2, etc.)

A close-up, vertical view of a server rack. Several horizontal server units are visible, each featuring a glowing green indicator light. The lighting is dim, with the primary light source being the green LEDs, creating a blueish-green ambient glow. The focus is sharp on the unit in the foreground, showing its metallic texture and the circular light housing.

# Dépendances et compatibilités

- À prendre en compte :
  - Versions du serveur web (Apache/Nginx), PHP, Node.js, SGBD (MySQL/PostgreSQL/MongoDB)
  - Extensions ou modules nécessaires
  - Systèmes d'exploitation cibles
  - Services externes (API tierces, SSO, outils analytics...)





# Sécurité du déploiement

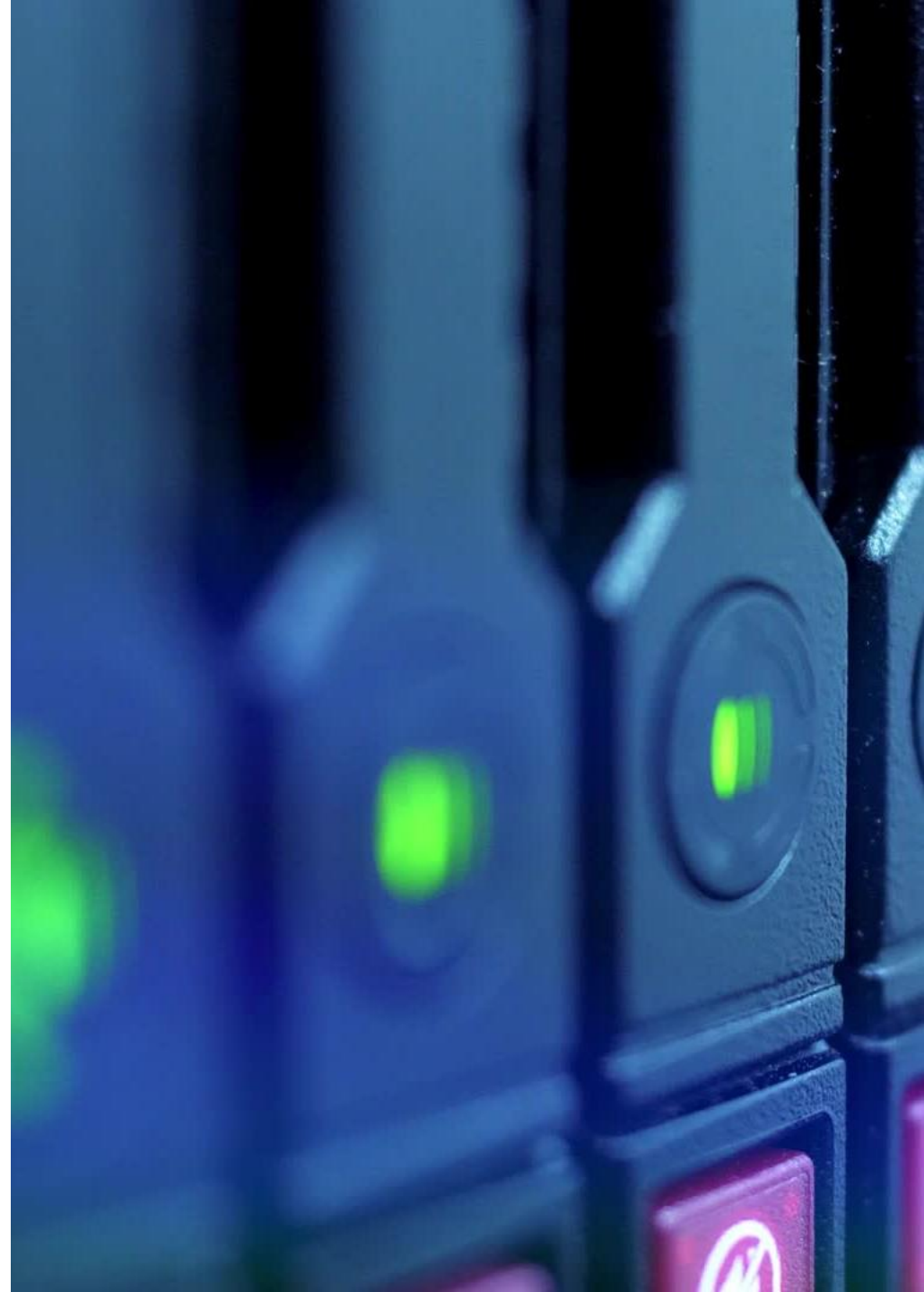
- Points de vigilance :
  - Transmission sécurisée des données : SSH/SCP, HTTPS
  - Rotation des clés ou tokens d'API
  - Configuration des droits des fichiers et des accès
  - Sécurisation du .env et autres fichiers sensibles
  - Références : ANSSI, OWASP, CERT-FR

# Intégration DevOps / CI/CD

- Définition :
  - CI = Intégration Continue : tests à chaque commit
  - CD = Déploiement Continu : mise en prod automatisée
- Bonnes pratiques :
  - Pipelines automatisés avec GitLab CI, GitHub Actions, Jenkins, CircleCI...
  - Tests unitaires, tests d'intégration, tests de sécurité
  - Déploiement sur container (Docker, Kubernetes)
  - Documentation synchronisée avec les releases

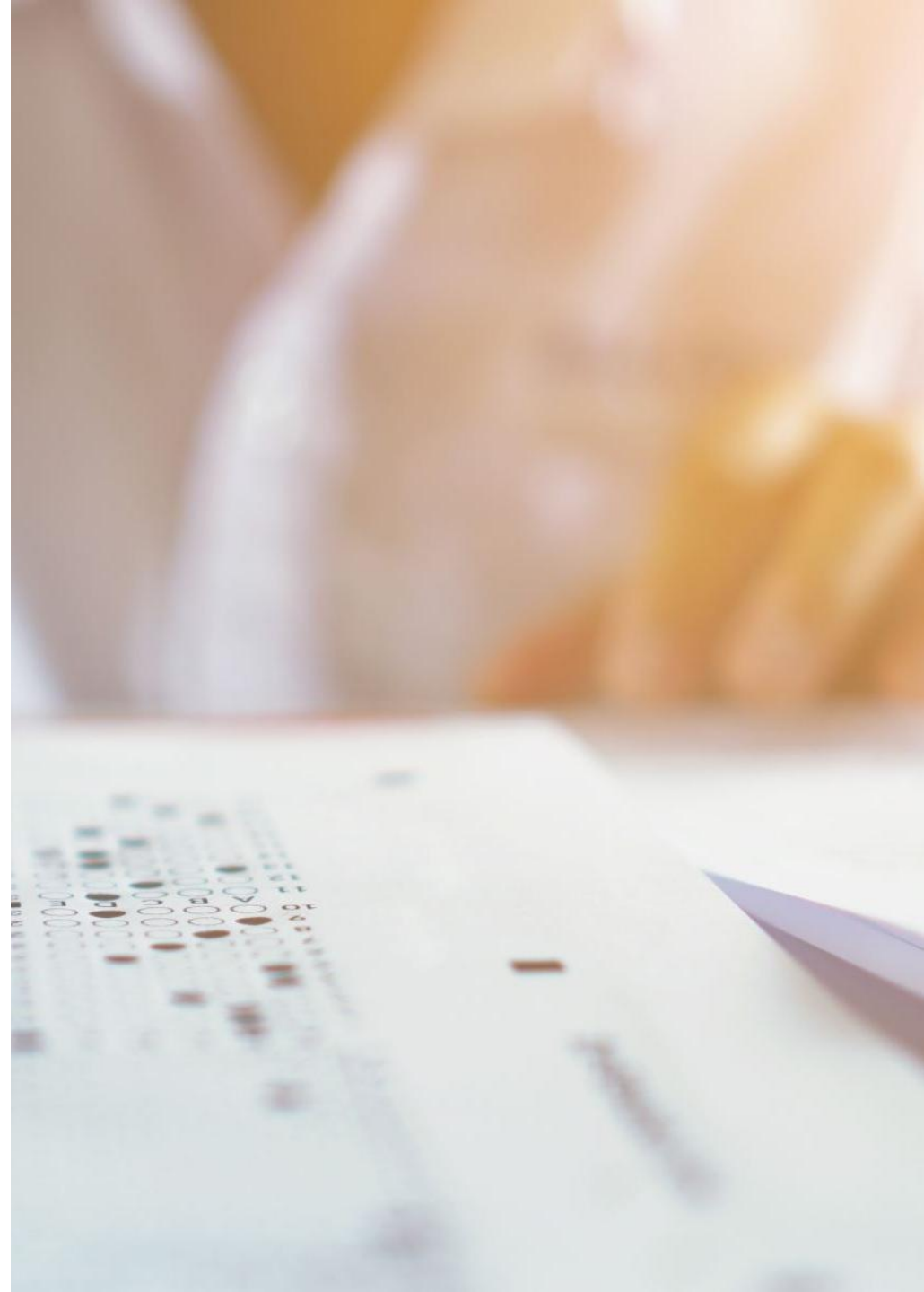
# Veille technologique

- Suivre les évolutions :
  - Outils de déploiement (Ansible, Capistrano, Deployer)
  - Hébergement (Cloud, VPS, Serveur dédié)
  - Sécurité des pipelines CI/CD
- Ressources :
  - CERT-FR
  - OWASP Top 10
  - GitLab/CI
  - GitHub Actions

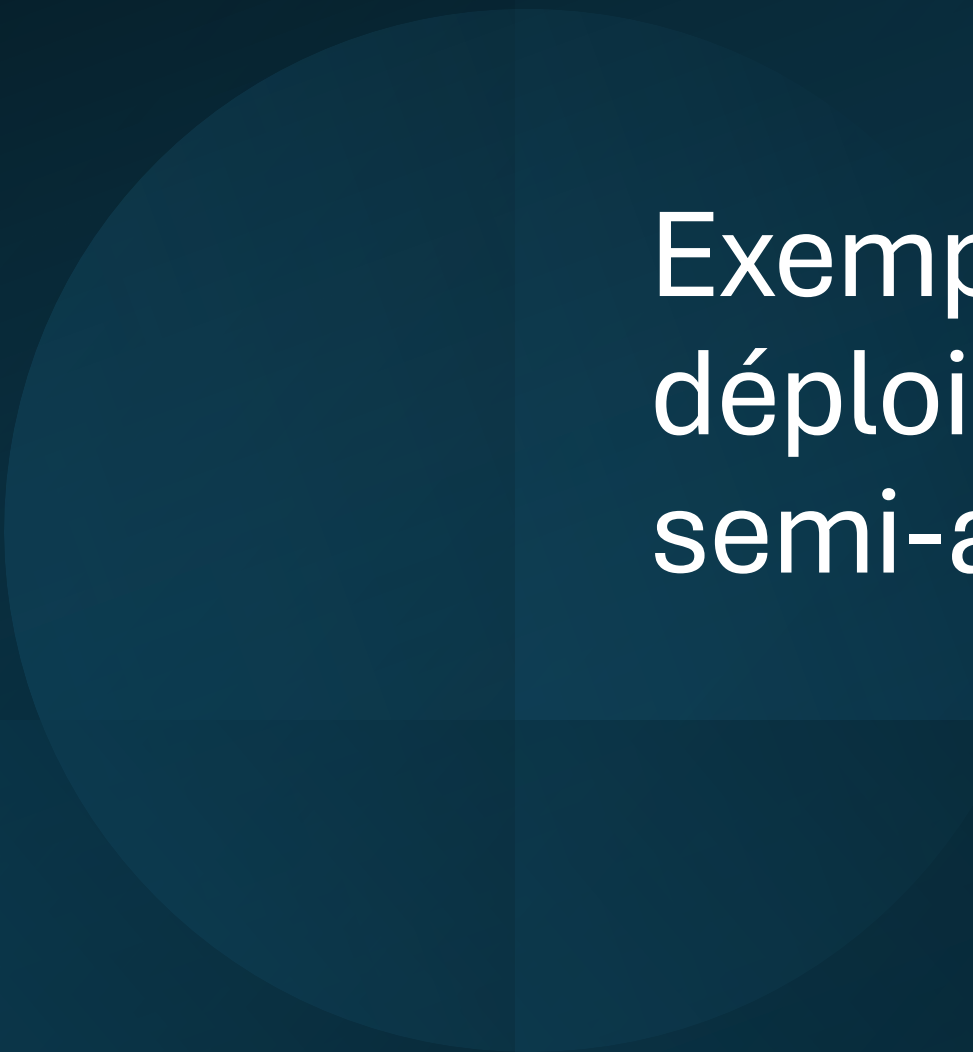


# Erreurs fréquentes

- Documentation obsolète ou non maintenue
- Scripts non testés, non versionnés
- Aucune procédure de rollback
- Trop de dépendance à un seul environnement
- Manque d'intégration dans la démarche DevOps







# Exemple de procédure de déploiement manuelle et semi-automatisée



# Objectif

- Déployer une application web PHP/MySQL en production chez un hébergeur mutualisé (ex : OVH, LWS, Ionos), sans CI/CD.

# Structure du projet

```
/mon-site/  
├── public/  
│   ├── index.php  
│   └── assets/  
├── config/  
│   └── config.php  
├── includes/  
├── uploads/  
├── .htaccess  
├── README.md  
└── db/  
    └── dump.sql
```





# Étapes du déploiement



# Étape 1 : Préparer les fichiers locaux

- Avant de transférer :
  - Nettoyer les fichiers inutiles (fichiers de tests, README, dump de dev)
  - Vérifier la configuration (ex. config.php : host = 'localhost' → 'sql123.serveur.com')
  - Ajouter un fichier .htaccess pour sécuriser certains dossiers (deny from all dans uploads, etc.)

## Étape 2 : Transfert des fichiers avec FileZilla

### Prérequis :

- Avoir les identifiants FTP de l'hébergeur
- Avoir installé FileZilla

### Procédure :

- Ouvrir FileZilla et se connecter à l'hébergeur :
- Hôte : ftp.votresite.com
- Identifiant / mot de passe : fourni par l'hébergeur
- Port : 21 ou 22 (si SFTP)
- Naviguer vers le dossier racine
- Transférer le contenu du projet dans ce dossier
- Créer manuellement les répertoires nécessaires : uploads/, includes/, etc.
- Mettre à jour les permissions si nécessaire (uploads/ : 755 ou 775)



## Étape 3 : Import de la base via phpMyAdmin

### Prérequis :

- Avoir accès à l'URL de phpMyAdmin (ex. : <https://sql.votresite.com/phpmyadmin>)

### Procédure :

- Se connecter avec les identifiants de la base
- Créer une nouvelle base de données (si nécessaire)
- Importer le fichier SQL (db/dump.sql) via l'onglet Importer

### Vérifier que :

- Les tables ont bien été créées
- Les données sont présentes
- Les utilisateurs ont bien leurs droits

# Script de déploiement simplifié (Bash)

```
#!/bin/bash
# deploy.sh - Script simplifié pour préparer un déploiement manuel
echo "== Déploiement du site web =="
# Vérification de la présence des fichiers essentiels
echo "[1/5] Vérification des fichiers..."
if [ ! -f ./public/index.php ]; then
    echo "Erreur : index.php manquant."
    exit 1
fi
# Nettoyage des fichiers temporaires
echo "[2/5] Nettoyage des fichiers inutiles..."
rm -rf ./tests ./README.md
# Compression des fichiers à transférer
echo "[3/5] Création de l'archive à transférer..."
tar -czf site.tar.gz ./public ./includes ./uploads ./config ./htaccess
# Message de fin
echo "[4/5] Archive prête à être transférée avec FileZilla !"
echo "[5/5] Pensez à importer votre base via phpMyAdmin"
exit 0
```

# Sécurité à vérifier manuellement

Élément	Action
uploads/	Vérifier que les scripts ne sont pas exécutables
config.php	S'assurer que les identifiants sont corrects mais non exposés
.htaccess	Bloquer l'accès à certains dossiers sensibles
Accès FTP	Utiliser <b>SFTP</b> de préférence



# Exemple de documentation à livrer

# Procédure de déploiement du projet "Site Boutique"

## ## 1. Transfert des fichiers

- Utiliser FileZilla pour envoyer les dossiers suivants :
  - `public/` → dans `www/`
  - `uploads/`, `includes/`, etc.

## ## 2. Configuration

- Modifier `config/config.php` :
  - DB\_HOST = "sql.votresite.com"
  - DB\_USER = "site\_user"
  - DB\_PASS = "password"

## ## 3. Import de la base

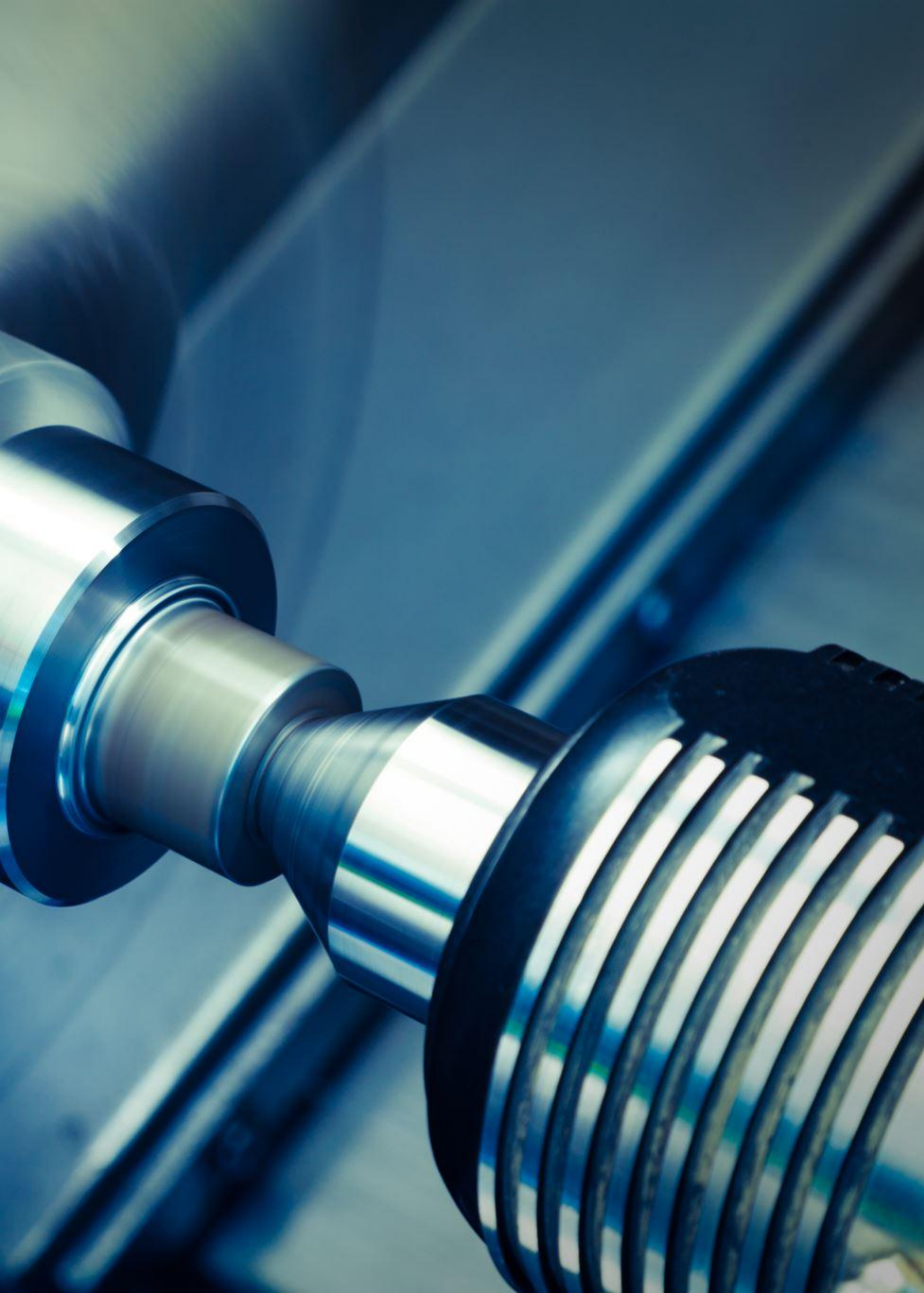
- Se connecter à phpMyAdmin : <https://sql.votresite.com/phpmyadmin>
- Créer la base `boutique\_db`
- Importer `db/dump.sql`

## ## 4. Vérifications

- Page d'accueil en ligne
- Connexion à la base OK
- Upload de fichiers OK
- Aucune erreur PHP

## ## 5. Sécurité

- Vérifier les permissions des fichiers
- Contrôler les accès au dossier `uploads/`



# En complément

- Outils alternatifs à découvrir :
  - WinSCP (pour Windows)
  - rsync (en SSH)
  - Deployer (outil PHP)
- Évolution possible :
  - Migrer cette procédure vers un pipeline CI/CD
  - Utiliser GitHub Actions ou GitLab CI pour automatiser les tests et le transfert FTP