Hébergement et Déploiement

Objectifs de cette section

Dans cette section, nous allons explorer **les différentes stratégies d'hébergement et de déploiement** pour nos applications en **PHP et Node.js**.
À la fin de cette section, vous serez capable de :

- Comprendre **les options d'hébergement** disponibles (serveur dédié, VPS, cloud, conteneurs).
- Déployer une application en **mode manuel** et en **mode automatisé**.
- Sécuriser l'environnement et mettre en place un **monitoring efficace**.
- Gérer les **sauvegardes et la scalabilité**.

---

1. Les différentes options d'hébergement

Il existe plusieurs solutions pour héberger une application backend. Le choix dépend de **la complexité du projet, du budget et des contraintes techniques**.

Serveur dédié

- Offre **un contrôle total** et de **hautes performances**.
- Requiert **une gestion et une maintenance manuelles**.
- Adapté aux **applications critiques nécessitant des ressources dédiées**.

VPS (Virtual Private Server)

- Offre un **bon compromis entre coût et performance**.
- Mutualise **les ressources d'un serveur physique** en plusieurs environnements isolés.
- Requiert une **gestion technique**, mais est plus accessible qu'un serveur dédié.

Hébergement cloud (AWS, GCP, Azure)

- Offre **une haute disponibilité** et une **scalabilité automatique**.
- Permet **de ne payer que les ressources utilisées**.
- Peut entraîner **des coûts variables** en fonction de la charge.

Conteneurs (Docker, Kubernetes)

- Permettent **de déployer des applications dans des environnements isolés**.
- Offrent **une portabilité facile** entre différents serveurs.
- Peuvent être **orchestrés avec Kubernetes pour une gestion avancée**.

---

2. Déploiement manuel : la méthode classique

Un **déploiement manuel** consiste à **installer, configurer et exécuter une application directement sur un serveur**.

Étapes du déploiement manuel :

Préparation du serveur

- Installation des services requis (serveur web, base de données, pare-feu).
- Configuration des utilisateurs et des permissions.

Transfert des fichiers

- Envoi du code source sur le serveur via **FTP, SCP ou Git**.

Configuration de l'environnement

- Définition des variables d'environnement.
- Configuration des fichiers `.env` et des permissions nécessaires.

Configuration du serveur web

- Mise en place d'un **serveur web (Apache, Nginx)**.
- Configuration des **hôtes virtuels et des certificats SSL**.

Démarrage et test de l'application

- Lancement de l'application et validation de son bon fonctionnement.

---

3. Déploiement automatisé avec Docker

Le déploiement manuel présente des **risques d'erreur et une gestion difficile des mises à jour**. **Docker** permet d'isoler l'application et de simplifier son exécution.

Avantages de Docker

- Assure **une portabilité totale** entre les environnements.
- Évite les **problèmes de compatibilité** entre les serveurs.
- Permet de **standardiser l'installation** grâce à une image préconfigurée.

Étapes du déploiement avec Docker :

Création d'une image Docker

- Définition des dépendances et configurations nécessaires.

Orchestration des services

- Gestion des **conteneurs (backend, base de données, reverse proxy, cache)**.

Déploiement et exécution

- Démarrage de l'ensemble des services sur le serveur cible.

---

4. Déploiement en production avec CI/CD

Un **workflow CI/CD (Continuous Integration / Continuous Deployment)** permet d'**automatiser les tests, le build et le déploiement**.

Outils couramment utilisés :

- **GitHub Actions / GitLab CI/CD** → Automatiser le build et le test.
- **Ansible** → Configurer et déployer l'application sur un serveur.
- **Kubernetes** → Orchestrer les conteneurs pour une **scalabilité optimale**.

Étapes d'un pipeline CI/CD :

1. **Validation du code source** → Vérification des modifications.
2. **Exécution des tests unitaires et d'intégration**.
3. **Build et création d'une image Docker**.
4. **Déploiement automatique sur le serveur**.

---

5. Gestion des sauvegardes

Une stratégie de sauvegarde efficace permet de **préserver les données et de garantir la reprise après incident**.

Bonnes pratiques :

- **Planifier des sauvegardes régulières** des bases de données et du code source.
- **Stocker les sauvegardes sur un serveur distant ou dans le cloud**.
- **Automatiser les sauvegardes** pour éviter les erreurs humaines.
- **Tester régulièrement la restauration des sauvegardes** pour s'assurer de leur fiabilité.

---

6. Sécurisation et monitoring du serveur

Un serveur mal sécurisé est une cible facile pour les cyberattaques. **Il est essentiel d'appliquer des mesures de protection dès le départ**.

Sécurisation du serveur

Bonnes pratiques :

- Désactiver **l'accès root en SSH** et restreindre les connexions.
- Mettre en place **un pare-feu** et limiter les services exposés.

- Configurer **Fail2Ban** pour bloquer les tentatives de connexion suspectes.
- Utiliser **des certificats SSL** pour sécuriser les communications.

Surveillance et monitoring

Outils recommandés :

- **Prometheus + Grafana** → Surveillance des performances du serveur et des requêtes API.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** → Gestion centralisée des logs.
- **Fail2Ban** → Protection contre les attaques par force brute.

Exemple d'arborescence d'un déploiement automatisé

```
/gestion-taches-deploiement
├── /php                          # Version PHP de l'application
│   ├── /src                      # Code source de l'application
│   ├── /config                    # Configuration de l'application
│   ├── /database                  # Migrations et Seeders
│   ├── /public                    # Point d'entrée HTTP
│   │   ├── index.php              # Routeur principal
│   ├── /scripts                   # Scripts d'automatisation
│   │   ├── deploy.sh              # Script de déploiement
│   │   ├── backup.sh              # Script de sauvegarde
│   │   ├── setup_server.sh        # Script d'installation sur le serveur
│   ├── /docker                    # Configuration Docker
│   │   ├── Dockerfile             # Image Docker de l'application PHP
│   │   ├── docker-compose.yml     # Orchestration des services Docker
│   ├── .env                       # Variables d'environnement
│   ├── composer.json              # Dépendances PHP
│   └── README.md                  # Documentation
│
├── /nodejs                        # Version Node.js de l'application
│   ├── /src                       # Code source de l'application
│   ├── /config                    # Configuration de l'application
│   ├── /database                  # Migrations et Seeders
│   ├── /public                    # Point d'entrée HTTP
│   │   ├── index.ts               # Routeur principal
│   ├── /scripts                   # Scripts d'automatisation
│   │   ├── deploy.sh              # Script de déploiement
│   │   ├── backup.sh              # Script de sauvegarde
│   │   ├── setup_server.sh        # Script d'installation sur le serveur
│   ├── /docker                    # Configuration Docker
│   │   ├── Dockerfile             # Image Docker de l'application Node.js
│   │   ├── docker-compose.yml     # Orchestration des services Docker
│   ├── .env                       # Variables d'environnement
│   ├── package.json               # Dépendances Node.js
│   └── README.md                  # Documentation
│
├── /infra                         # Configuration de l'infrastructure
│   ├── nginx/                     # Configuration du serveur Nginx
│   │   ├── nginx.conf             # Configuration globale
│   │   ├── sites-available/       # Fichiers de configuration des hôtes virtuels
│   ├── ansible/                   # Automatisation avec Ansible
│   │   ├── playbook.yml           # Playbook d'installation et configuration
│   ├── kubernetes/                # Déploiement Kubernetes
│   │   ├── deployment.yaml        # Configuration des pods
│   │   ├── service.yaml           # Configuration des services
│   ├── monitoring/                # Outils de surveillance
│   │   ├── prometheus.yml         # Configuration de Prometheus
│   │   ├── grafana-dashboard.json # Tableau de bord Grafana
│   ├── backup/                    # Scripts et configurations de sauvegarde
│   │   ├── cron-backup.sh         # Planification des sauvegardes
│   │   ├── s3-backup.sh           # Sauvegarde sur Amazon S3
│   ├── security/                  # Configuration de la sécurité
│   │   ├── firewall-rules.sh      # Règles de pare-feu (UFW/Iptables)
│   │   ├── fail2ban.conf          # Configuration de Fail2ban
│   │   ├── ssl-certificates/      # Certificats SSL (Let's Encrypt)
│   └── README.md                  # Documentation Infrastructure
```