Manuel de déploiement - Penpal AI

Ce manuel détaille les procédures de déploiement de la plateforme Penpal AI sur l'infrastructure Hetzner Cloud avec Coolify. Il couvre le déploiement des 7 services (frontend + 6 microservices backend) avec la configuration complète des environnements.

Vue d'ensemble du déploiement

Architecture cible

- **Infrastructure** : Hetzner Cloud (serveurs dédiés)
- **Orchestrateur** : Coolify (PaaS self-hosted)
- **Conteneurisation** : Docker multi-stage builds
- **Registre** : GitHub Container Registry (GHCR)
- **Domaines** :
  - **Production** : `prod.<service>.penpal-ai.maksou.dev`
  - **Staging** : `staging.<service>.penpal-ai.maksou.dev`

Services à déployer

1. **Frontend** (Next.js) - Interface utilisateur
2. **Auth Service** (NestJS) - Authentification et autorisation
3. **DB Service** (NestJS) - Gestion des données centralisée
4. **AI Service** (NestJS) - Intelligence artificielle conversationnelle
5. **Payment Service** (NestJS) - Paiements et abonnements Stripe
6. **Monitoring Service** (NestJS) - Métriques et observabilité
7. **Notify Service** (NestJS) - Notifications et emails

Prérequis

Infrastructure

Infrastructure Hetzner Cloud (existante)

- **Instance** : Serveur Hetzner Cloud configuré
- **OS** : Ubuntu 22.04 LTS
- **Network** : IPv4 publique avec firewall
- **Coolify** : PaaS self-hosted installé et opérationnel

Domaine configuré

- **Domaine principal** : `penpal-ai.maksou.dev`
- **DNS** : Configuration wildcard `*.penpal-ai.maksou.dev`
- **SSL** : Let's Encrypt automatique via Coolify
- **GitHub Registry** : GHCR configuré pour pull images

Secrets et clés API

GitHub

- **GITHUB_TOKEN** : Accès GHCR pour pull images
- **COOLIFY_API_TOKEN** : Token Coolify pour webhooks CI/CD

Services externes

- **OpenAI API Key** : `sk-...` pour AI Service
- **Anthropic API Key** : `sk-ant-...` pour AI Service (optionnel)
- **Stripe Keys** : `sk_live_...` et `pk_live_...` pour Payment Service
- **SendGrid API Key** : `SG.xxx` pour Notify Service

OAuth Providers

- **Google** : Client ID + Secret

Configuration des environnements

Variables globales

```
# Base URLs pour chaque environnement
# Production
FRONTEND_URL=https://app.penpal-ai.maksou.dev
AUTH_SERVICE_URL=https://prod.auth-service.penpal-ai.maksou.dev
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev
AI_SERVICE_URL=https://prod.ai-service.penpal-ai.maksou.dev
PAYMENT_SERVICE_URL=https://prod.payment-service.penpal-ai.maksou.dev
MONITORING_SERVICE_URL=https://prod.monitoring-service.penpal-
ai.maksou.dev
NOTIFY_SERVICE_URL=https://prod.notify-service.penpal-ai.maksou.dev

# Staging
FRONTEND_URL=https://staging.app.penpal-ai.maksou.dev
AUTH_SERVICE_URL=https://staging.auth-service.penpal-ai.maksou.dev
# ... (même pattern pour staging)
```

Configuration par service

1. Frontend (Next.js)

Variables d'environnement :

```
# Build-time variables (définies dans Coolify)
NEXT_PUBLIC_AUTH_SERVICE_URL=https://prod.auth-service.penpal-
ai.maksou.dev/api/v1
NEXT_PUBLIC_AI_SERVICE_URL=https://prod.ai-service.penpal-
ai.maksou.dev/api/v1
NEXT_PUBLIC_PAYMENT_API_URL=https://prod.payment-service.penpal-
```

```
ai.maksou.dev/api/v1
NEXT_PUBLIC_MONITORING_API_URL=https://prod.monitoring-service.penpal-
ai.maksou.dev
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_xxxxx
NEXT_TELEMETRY_DISABLED=1
NODE_ENV=production
PORT=3000
```

Dockerfile :

```
# Stage 0: Base (libc for Next.js SWC on Alpine)
FROM node:20-alpine AS base
# Install libc6-compat for Alpine compatibility
RUN apk add --no-cache libc6-compat

# Stage 1: Dependencies
FROM base AS deps
WORKDIR /app

# Copy package files
COPY package.json package-lock.json* ./

# Install all dependencies (including devDependencies needed for build)
RUN npm ci

# Stage 2: Builder
FROM base AS builder
WORKDIR /app

# Copy node_modules from deps stage
COPY --from=deps /app/node_modules ./node_modules

# Copy all source files (excluding node_modules, .git, .next)
COPY . .

# Set build-time environment variables
ARG NEXT_PUBLIC_AUTH_SERVICE_URL
ARG NEXT_PUBLIC_AI_SERVICE_URL
```

2. Auth Service (NestJS)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3002
API_PREFIX=/api
API_VERSION=v1
```

```
# Database
MONGODB_URI=mongodb://username:password@mongodb-prod:27017/penpal-auth
REDIS_HOST=redis-prod
REDIS_PORT=6379
REDIS_PASSWORD=redis_password_prod

# JWT Configuration
JWT_SECRET=jwt_secret_super_secure_prod
JWT_EXPIRATION=1h
JWT_REFRESH_SECRET=jwt_refresh_secret_super_secure_prod
JWT_REFRESH_EXPIRATION=7d

# OAuth Configuration
GOOGLE_CLIENT_ID=xxx.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=GOCSPX-xxx
GOOGLE_REDIRECT_URI=https://prod.auth-service.penpal-
ai.maksou.dev/api/v1/auth/google/callback

# Service Communication
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev/api/v1
NOTIFY_SERVICE_URL=https://prod.notify-service.penpal-ai.maksou.dev/api/v1
SERVICE_AUTH_TOKEN=service_token_super_secure_prod

# Security
BCRYPT_ROUNDS=12
RATE_LIMIT_TTL=3600
RATE_LIMIT_LIMIT=100
```

3. DB Service (NestJS)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3001
API_PREFIX=/api
API_VERSION=v1

# Database MongoDB
MONGODB_URI=mongodb://penpal_user:penpal_password_prod@mongodb-
prod:27017/penpal-ai
MONGODB_USER=penpal_user
MONGODB_PASSWORD=penpal_password_prod

# Cache Redis
REDIS_HOST=redis-prod
REDIS_PORT=6379
REDIS_PASSWORD=redis_password_prod
REDIS_TTL=3600

# API Documentation
```

```
SWAGGER_ENABLED=false  # Désactivé en production
API_DOCS_PATH=/docs

# Service Authentication
SERVICE_AUTH_TOKEN=service_token_super_secure_prod
API_KEY=api_key_super_secure_prod

# Monitoring
LOG_LEVEL=info
ENABLE_METRICS=true
```

4. AI Service (Asimov)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3003
API_PREFIX=/api
API_VERSION=v1

# AI Providers
OPENAI_API_KEY=sk-xxxxx
OPENAI_ORG_ID=org-xxxxx
ANTHROPIC_API_KEY=sk-ant-xxxxx

# AI Configuration
DEFAULT_AI_PROVIDER=openai
MAX_TOKENS=2000
TEMPERATURE=0.7
AI_CACHE_TTL=1800  # 30 minutes

# Database Communication
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev/api/v1
SERVICE_AUTH_TOKEN=service_token_super_secure_prod

# Cache Redis
REDIS_HOST=redis-prod
REDIS_PORT=6379
REDIS_PASSWORD=redis_password_prod

# WebSocket (futur)
WEBSOCKET_ENABLED=false
WEBSOCKET_PORT=3103

# Rate Limiting
AI_RATE_LIMIT=60  # Requêtes par minute par utilisateur
```

5. Payment Service (NestJS)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3004
API_PREFIX=/api
API_VERSION=v1

# Stripe Configuration
STRIPE_SECRET_KEY=sk_live_xxxxx
STRIPE_WEBHOOK_SECRET=whsec_xxxxx
STRIPE_PUBLISHABLE_KEY=pk_live_xxxxx

# Database Communication
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev/api/v1
SERVICE_AUTH_TOKEN=service_token_super_secure_prod

# Pricing Configuration
MONTHLY_PRICE_ID=price_xxxxx
YEARLY_PRICE_ID=price_xxxxx
TRIAL_PERIOD_DAYS=30

# Frontend URLs for redirects
FRONTEND_SUCCESS_URL=https://app.penpal-ai.maksou.dev/checkout/success
FRONTEND_CANCEL_URL=https://app.penpal-ai.maksou.dev/pricing

# Notification Service
NOTIFY_SERVICE_URL=https://prod.notify-service.penpal-ai.maksou.dev/api/v1
```

6. Monitoring Service (NestJS)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3005

# Prometheus Configuration
PROMETHEUS_PORT=9090
PROMETHEUS_ENDPOINT=/metrics

# Grafana Configuration
GRAFANA_URL=https://prod.grafana.penpal-ai.maksou.dev
GRAFANA_API_KEY=eyxxxxx
GRAFANA_ORG_ID=1

# Service URLs to Monitor
AUTH_SERVICE_URL=https://prod.auth-service.penpal-ai.maksou.dev
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev
AI_SERVICE_URL=https://prod.ai-service.penpal-ai.maksou.dev
```

```
PAYMENT_SERVICE_URL=https://prod.payment-service.penpal-ai.maksou.dev
NOTIFY_SERVICE_URL=https://prod.notify-service.penpal-ai.maksou.dev

# Health Check Configuration
HEALTH_CHECK_INTERVAL=30000  # 30 secondes
HEALTH_CHECK_TIMEOUT=5000    # 5 secondes
```

7. Notify Service (NestJS)

Variables d'environnement :

```
# Application
NODE_ENV=production
PORT=3007
API_PREFIX=/api
API_VERSION=v1

# SendGrid Configuration
SENDGRID_API_KEY=SG.xxxxx
FROM_EMAIL=noreply@penpal-ai.maksou.dev
FROM_NAME=Penpal AI

# Email Templates
TEMPLATE_DIR=./src/templates
WELCOME_TEMPLATE=welcome
RESET_PASSWORD_TEMPLATE=reset-password
SUBSCRIPTION_TEMPLATE=subscription

# Database Communication
DB_SERVICE_URL=https://prod.db-service.penpal-ai.maksou.dev/api/v1
SERVICE_AUTH_TOKEN=service_token_super_secure_prod

# Frontend URLs for email links
FRONTEND_URL=https://app.penpal-ai.maksou.dev
RESET_PASSWORD_URL=https://app.penpal-ai.maksou.dev/auth/reset-password
```

Procédures de déploiement

Infrastructure existante (déjà configurée)

> **Note** : Cette section décrit l'infrastructure déjà mise en place. Pour référence historique et maintenance.

1. Serveur Hetzner Cloud configuré

Configuration actuelle :

- **Instance** : Hetzner Cloud (type et specs à documenter)
- **OS** : Ubuntu 22.04 LTS

- **IP publique** : Configurée et routée
- **Firewall** : Ports 22, 80, 443, 8080 ouverts
- **DNS** : Domaine `penpal-ai.maksou.dev` pointant vers l'IP

2. Coolify déjà installé et configuré

Configuration existante :

- **Installation** : Coolify v4.x installé via script officiel
- **Accès** : Interface admin accessible via HTTPS
- **Docker Registry** : GitHub Container Registry (GHCR) configuré
- **SSL** : Let's Encrypt automatique pour tous les domaines
- **Webhooks** : Configurés pour CI/CD automatique

3. DNS et domaines configurés

Configuration domaine `penpal-ai.maksou.dev` :

```
# Structure DNS existante
penpal-ai.maksou.dev                    # Domaine principal
*.penpal-ai.maksou.dev                  # Wildcard pour services

# Production
app.penpal-ai.maksou.dev               # Frontend production
prod.auth-service.penpal-ai.maksou.dev # Auth service production
prod.db-service.penpal-ai.maksou.dev   # DB service production
prod.ai-service.penpal-ai.maksou.dev   # AI service production
prod.payment-service.penpal-ai.maksou.dev
prod.monitoring-service.penpal-ai.maksou.dev
prod.notify-service.penpal-ai.maksou.dev

# Staging
staging.app.penpal-ai.maksou.dev       # Frontend staging
staging.auth-service.penpal-ai.maksou.dev
staging.db-service.penpal-ai.maksou.dev
staging.ai-service.penpal-ai.maksou.dev
staging.payment-service.penpal-ai.maksou.dev
staging.monitoring-service.penpal-ai.maksou.dev
staging.notify-service.penpal-ai.maksou.dev
```

4. Bases de données configurées

Services existants dans Coolify :

```
# MongoDB Production
# - Container: mongodb-prod
# - Version: mongo:6.0
# - Base de données: penpal-ai
# - Utilisateur applicatif: penpal_user
```

```
# - Accès interne: mongodb-prod:27017

# Redis Production
# - Container: redis-prod
# - Version: redis:7.0-alpine
# - Configuration: persistence activée
# - Accès interne: redis-prod:6379

# MongoDB Staging
# - Container: mongodb-staging
# - Version: mongo:6.0
# - Base de données: penpal-ai-staging
# - Isolation complète de production

# Redis Staging
# - Container: redis-staging
# - Version: redis:7.0-alpine
# - Isolation complète de production
```

Déploiement des services

Ordre de déploiement :

1. **DB Service** (premier - hub central de données)
2. **Auth Service** (dépend de DB Service)
3. **AI Service** (dépend de DB Service)
4. **Payment Service** (dépend de DB Service)
5. **Notify Service** (dépend de DB Service)
6. **Monitoring Service** (monitore tous les autres)
7. **Frontend** (dernier - consomme tous les services)

Configuration par service dans Coolify :

```
# Exemple pour DB Service
name: penpal-db-service-prod
repository: ghcr.io/your-username/penpal-ai-db-service
tag: latest
domain: prod.db-service.penpal-ai.maksou.dev
port: 3001
environment: production
build_target: production

# Variables d'environnement (via interface Coolify)
# NODE_ENV=production
# PORT=3001
# MONGODB_URI=mongodb://penpal_user:password@mongodb-prod:27017/penpal-ai
# etc...

# Health check
health_check: /api/v1/health
```

Déploiement continu (CI/CD)

Workflow GitHub Actions

Chaque service possède un workflow automatisé :

1. **Tests** : Lint, unitaires, E2E
2. **Build** : Image Docker multi-arch
3. **Push** : GHCR avec tags (latest, sha, version)
4. **Deploy** : Webhook Coolify pour redéploiement

```
# Exemple webhook trigger
- name: Trigger Coolify deploy (production)
  if: github.ref_name == 'main'
  run: |
    curl -fsSL -X POST \
      -H "Authorization: Bearer ${{ secrets.COOLIFY_API_TOKEN }}" \
      "${{ secrets.COOLIFY_PROD_WEBHOOK_URL }}"
```

Configuration des webhooks

```
# Dans Coolify pour chaque service
# 1. Aller dans Settings > Webhooks
# 2. Copier l'URL de webhook
# 3. Ajouter dans GitHub Secrets :
#    - COOLIFY_PROD_WEBHOOK_URL_DB_SERVICE
#    - COOLIFY_STAGING_WEBHOOK_URL_DB_SERVICE
#    - etc.
```

Mise à jour des services

Déploiement staging

```
# 1. Push sur branch develop
git checkout develop
git add .
git commit -m "feat: nouvelle fonctionnalité"
git push origin develop

# 2. CI/CD automatique :
# - Tests automatiques
# - Build image Docker
# - Push vers GHCR avec tag develop-latest
# - Webhook Coolify staging
# - Déploiement automatique sur staging.*.penpal-ai.maksou.dev
```

```
# 3. Tests en staging
curl -f https://staging.db-service.penpal-ai.maksou.dev/api/v1/health
```

Déploiement production

```
# 1. Merge develop vers main
git checkout main
git merge develop
git push origin main

# 2. CI/CD automatique :
# - Tests complets
# - Build image production
# - Push vers GHCR avec tag latest + sha
# - Webhook Coolify production
# - Déploiement zero-downtime sur prod.*.penpal-ai.maksou.dev

# 3. Vérification production
curl -f https://prod.db-service.penpal-ai.maksou.dev/api/v1/health
```

Release avec tags

```
# 1. Créer une release tagged
git tag v1.2.3
git push --tags

# 2. CI/CD automatique :
# - Build avec tag de version (v1.2.3)
# - Push vers GHCR avec tous les tags
# - Pas de déploiement automatique (sécurité)

# 3. Déploiement manuel en production
# Via interface Coolify ou webhook manuel
```

Monitoring et observabilité

Métriques Prometheus

```
# Services exposant des métriques
https://prod.db-service.penpal-ai.maksou.dev/metrics
https://prod.auth-service.penpal-ai.maksou.dev/metrics
https://prod.ai-service.penpal-ai.maksou.dev/metrics
https://prod.payment-service.penpal-ai.maksou.dev/metrics
https://prod.notify-service.penpal-ai.maksou.dev/metrics
```

Dashboards Grafana

```
# Accès Grafana
https://prod.monitoring-service.penpal-ai.maksou.dev/grafana

# Dashboards principaux :
# - System Overview : CPU, RAM, disk, network
# - Application Metrics : Requêtes, erreurs, latence
# - Business Metrics : Utilisateurs actifs, conversations, paiements
# - AI Metrics : Coûts API, tokens consommés, temps de réponse
```

Logs centralisés

```
# Accès aux logs via Coolify
# 1. Interface Coolify > Service > Logs
# 2. Filtrage par niveau (info, warn, error)
# 3. Recherche par mots-clés
# 4. Export pour analyse

# Logs applicatifs structurés (JSON)
{
  "timestamp": "2024-01-15T10:30:00Z",
  "level": "info",
  "service": "auth-service",
  "message": "User login successful",
  "userId": "user123",
  "ip": "192.168.1.100"
}
```

Alerting

```
# Configuration alerts Grafana
# 1. CPU > 80% pendant 5 minutes
# 2. Memory > 90% pendant 3 minutes
# 3. Disk space < 10%
# 4. Service down (health check fail)
# 5. Error rate > 1%
# 6. AI API costs > budget quotidien

# Notifications :
# - Email : admin@penpal-ai.maksou.dev
# - Slack : #alerts channel (optionnel)
# - SMS : Numéros critiques (optionnel)
```

Sécurité en production

SSL/TLS

```
# Certificats automatiques via Coolify
# - Let's Encrypt pour tous les domaines
# - Renouvellement automatique
# - Redirection HTTP vers HTTPS
# - HSTS headers activés
```

Secrets Management

```
# Variables sensibles dans Coolify
# - Chiffrement au repos
# - Accès restreint par rôles
# - Rotation périodique
# - Audit des accès

# Bonnes pratiques :
# 1. Jamais de secrets en hard-code
# 2. Variables d'environnement uniquement
# 3. Différentes clés par environnement
# 4. Rotation tous les 90 jours
```

Network Security

```
# Firewall Hetzner
# - Ports 22, 80, 443, 8080 uniquement
# - Whitelist IP admin pour SSH
# - DDoS protection activée

# Container Network
# - Services isolés par réseau Docker
# - Communication inter-services sécurisée
# - Pas d'exposition ports inutiles
```

Backup et recovery

```
# Stratégie de sauvegarde
# 1. MongoDB : Backup quotidien automatique
# 2. Redis : Snapshot toutes les 6h
# 3. Volumes Docker : Backup hebdomadaire
# 4. Configuration Coolify : Export mensuel

# Rétention :
# - Backups quotidiens : 30 jours
# - Backups hebdomadaires : 12 semaines
```

```
# - Backups mensuels : 12 mois

# Tests de restore : Mensuel en staging
```

Troubleshooting

Problèmes courants

Service ne démarre pas

```
# 1. Vérifier les logs
# Via Coolify > Service > Logs

# 2. Vérifier les variables d'environnement
# Via Coolify > Service > Environment

# 3. Vérifier la connectivité base de données
docker exec -it service-container sh
curl -f http://mongodb-prod:27017

# 4. Vérifier les resources
docker stats
free -h
df -h
```

Erreur de connectivité inter-services

```
# 1. Vérifier le réseau Docker
docker network ls
docker network inspect coolify

# 2. Test de connectivité
docker exec -it db-service-container sh
curl -f http://auth-service:3002/api/v1/health

# 3. Vérifier les tokens de service
echo $SERVICE_AUTH_TOKEN
# Doit être identique sur tous les services
```

Performance dégradée

```
# 1. Métriques système
htop
iotop
nethogs
```

```
# 2. Métriques application
curl https://prod.db-service.penpal-ai.maksou.dev/metrics

# 3. Analyse base de données
# Connexion MongoDB
mongo mongodb://user:pass@mongodb-prod:27017/penpal-ai
db.conversations.find().limit(5).explain("executionStats")

# 4. Cache Redis
redis-cli -h redis-prod -a password
INFO memory
```

Problème de déploiement

```
# 1. Vérifier le webhook
curl -X POST -H "Authorization: Bearer $TOKEN" $WEBHOOK_URL

# 2. Vérifier l'image Docker
docker pull ghcr.io/user/service:latest
docker run --rm ghcr.io/user/service:latest --version

# 3. Logs du déploiement
# Via Coolify > Service > Deployments > View logs

# 4. Rollback si nécessaire
# Via Coolify > Service > Deployments > Previous version
```

Procédures de rollback

Rollback rapide (via Coolify)

```
# 1. Interface Coolify
# Service > Deployments > Deploy Previous

# 2. Ou via API
curl -X POST \
  -H "Authorization: Bearer $COOLIFY_API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"deployment_id": "previous_deployment_id"}' \
  https://coolify.yourdomain.com/api/v1/deploy
```

Rollback Git + redéploiement

```
# 1. Revert Git
git checkout main
git revert <commit-hash>
```

```
git push origin main

# 2. Attendre CI/CD automatique
# Ou déclencher manuellement le webhook

# 3. Vérification
curl -f https://prod.service.penpal-ai.maksou.dev/api/v1/health
```

Rollback base de données (si nécessaire)

```
# 1. Arrêter les services
# Via Coolify > Stop all services

# 2. Restore backup MongoDB
mongorestore --host mongodb-prod --db penpal-ai /backups/latest/

# 3. Redémarrer les services
# Via Coolify > Start all services

# 4. Vérification intégrité
# Tests complets en staging d'abord
```

Maintenance

Mises à jour système

```
# Maintenance mensuelle programmée
# 1. Notification utilisateurs 48h avant
# 2. Fenêtre de maintenance : Dimanche 02:00-04:00 UTC

# Procédure :
# 1. Backup complet
# 2. Mise à jour OS
apt update && apt upgrade -y

# 3. Mise à jour Docker
curl -fsSL https://get.docker.com | sh

# 4. Mise à jour Coolify
# Via interface admin

# 5. Redémarrage serveur si nécessaire
reboot

# 6. Vérification tous services
# 7. Tests fonctionnels complets
```

Scaling horizontal

```
# Ajout de serveurs (si croissance)
# 1. Nouveau serveur Hetzner
# 2. Installation Coolify worker
# 3. Configuration load balancer
# 4. Migration services haute charge

# Services prioritaires pour scaling :
# - AI Service (coûteux en CPU/GPU)
# - DB Service (I/O intensif)
# - Frontend (trafic utilisateur)
```

Optimisation continue

```
# Métriques à surveiller
# 1. Response time < 200ms (95e percentile)
# 2. Error rate < 0.1%
# 3. Uptime > 99.9%
# 4. CPU usage < 70%
# 5. Memory usage < 80%
# 6. Disk usage < 80%

# Actions d'optimisation :
# - Cache Redis pour requêtes fréquentes
# - Index MongoDB pour performances
# - CDN pour assets statiques
# - Compression gzip/brotli
# - Optimisation images Docker
```

Niveaux d'escalation

1. **L1** : Problèmes mineurs, redémarrages simples
2. **L2** : Problèmes services, rollbacks, debugging
3. **L3** : Problèmes infrastructure, corruption données
4. **L4** : Incidents majeurs, perte service > 1h

SLA

- **Disponibilité** : 99.9% (8.76h downtime/an maximum)
- **Support L1** : 15min response time
- **Support L2** : 1h response time
- **Recovery** : 4h maximum pour incidents critiques

Conclusion

Ce manuel couvre l'ensemble des procédures de déploiement pour la plateforme Penpal AI. L'architecture Hetzner + Coolify + Docker offre un excellent équilibre entre simplicité opérationnelle et robustesse enterprise.

Points clés :

- **Zero-downtime** deployments via Coolify
- **Multi-environment** avec promotion staging → production
- **Sécurité** SSL automatique + secrets management
- **Scalabilité** horizontale prête pour croissance