Guide de Déploiement - Formation Manager Itinova

Version: 7a737c05
Date: 24 novembre 2025
Auteur: Manus AI

Vue d’ ensemble

Ce document fournit les instructions complètes pour déployer l’ application **Formation Manager Itinova** sur un serveur de production. L’ application est construite avec React 19, Express 4, tRPC 11 et utilise une base de données MySQL/TiDB.

Prérequis système

Avant de commencer le déploiement, assurez-vous que votre serveur dispose des éléments suivants :

Configuration matérielle minimale

Composant	Spécification minimale	Recommandé
CPU	2 cœurs	4 cœurs
RAM	2 GB	4 GB
Stockage	20 GB	50 GB SSD
Bande passante	100 Mbps	1 Gbps

Logiciels requis

L'application nécessite les logiciels suivants installés sur le serveur :

Node.js version 22.13.0 ou supérieure : Le runtime JavaScript est essentiel pour exécuter l'application côté serveur. La version 22.13.0 garantit la compatibilité avec toutes les dépendances du projet.

pnpm : Le gestionnaire de paquets pnpm est utilisé pour installer les dépendances de manière efficace. Il offre de meilleures performances et une gestion optimisée de l'espace disque par rapport à npm.

MySQL 8.0 ou TiDB : La base de données relationnelle stocke toutes les données de l'application (formations, séquences, apprenants, inscriptions). TiDB est compatible MySQL et offre une scalabilité horizontale pour les déploiements à grande échelle.

Nginx ou Apache : Un serveur web reverse proxy est recommandé pour gérer le SSL/TLS, la compression et le cache statique. Nginx est préféré pour ses performances supérieures.

Systemd : Le gestionnaire de services Linux permet de gérer l'application comme un service système, assurant le démarrage automatique et la supervision.

Git : Le système de contrôle de version est nécessaire pour cloner le dépôt et effectuer les mises à jour.

Architecture de déploiement

L'application suit une architecture client-serveur moderne avec les composants suivants :

Composants principaux

Frontend React : L'interface utilisateur est construite avec React 19 et Tailwind CSS 4. Le build de production génère des fichiers statiques optimisés (HTML, CSS, JavaScript) qui sont servis par le serveur Express.

Backend Express + tRPC : Le serveur Node.js gère les requêtes API via tRPC, offrant une communication type-safe entre le client et le serveur. Toutes les routes API sont

préfixées par `/api/`.

Base de données MySQL/TiDB : La couche de persistance utilise Drizzle ORM pour interagir avec la base de données. Le schéma est défini dans `drizzle/schema.ts` et les migrations sont gérées automatiquement.

Authentification OAuth : Le système d'authentification utilise Manus OAuth pour la gestion des utilisateurs. Les sessions sont stockées dans des cookies HTTP-only sécurisés.

Flux de requêtes

Les requêtes utilisateur suivent ce chemin : **Navigateur → Nginx (SSL/TLS) → Express (port 3000) → tRPC → Base de données**. Les fichiers statiques sont servis directement par Express depuis le répertoire `dist/public/`.

Étape 1 : Préparation du serveur

Installation de Node.js

La première étape consiste à installer Node.js version 22.13.0 sur votre serveur Ubuntu. Utilisez les commandes suivantes pour installer Node.js via le gestionnaire de versions nvm :

```
# Installer nvm (Node Version Manager)
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/install.sh |
bash

# Recharger le profil shell
source ~/.bashrc

# Installer Node.js 22.13.0
nvm install 22.13.0

# Définir comme version par défaut
nvm use 22.13.0
nvm alias default 22.13.0

# Vérifier l'installation
node --version  # Doit afficher v22.13.0
```

Installation de pnpm

Une fois Node.js installé, installez pnpm globalement :

```
npm install -g pnpm

# Vérifier l'installation
pnpm --version
```

Configuration de la base de données

Créez une base de données MySQL dédiée pour l’ application. Connectez-vous à MySQL et exécutez les commandes suivantes :

```
CREATE DATABASE formation_manager CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
CREATE USER 'formation_user'@'localhost' IDENTIFIED BY
'MOT_DE_PASSE_SECURISE';
GRANT ALL PRIVILEGES ON formation_manager.* TO 'formation_user'@'localhost';
FLUSH PRIVILEGES;
```

Remplacez `MOT_DE_PASSE_SECURISE` par un mot de passe fort généré aléatoirement. Notez les informations de connexion pour la configuration ultérieure.

Création de l'utilisateur système

Pour des raisons de sécurité, créez un utilisateur système dédié pour exécuter l'application :

```
sudo useradd -r -s /bin/bash -d /opt/formation-manager formation-manager
sudo mkdir -p /opt/formation-manager
sudo chown formation-manager:formation-manager /opt/formation-manager
```

Étape 2 : Clonage et configuration

Clonage du dépôt

Connectez-vous en tant qu'utilisateur `formation-manager` et clonez le dépôt Git :

```
sudo su - formation-manager
cd /opt/formation-manager
git clone <URL_DU_DEPOT> app
cd app
git checkout 7a737c05
```

Remplacez `<URL_DU_DEPOT>` par l'URL réelle de votre dépôt Git.

Configuration des variables d'environnement

Créez un fichier `.env` à la racine du projet avec les variables suivantes :

```
# Base de données
DATABASE_URL=mysql://formation_user:MOT_DE_PASSE_SECURISE@localhost:3306/forma

# Authentification
JWT_SECRET=<GENERER_UNE_CLE_ALEATOIRE_64_CARACTERES>
OAUTH_SERVER_URL=https://api.manus.im
VITE_OAUTH_PORTAL_URL=https://auth.manus.im

# Identifiants propriétaire
OWNER_OPEN_ID=<VOTRE_OPEN_ID_MANUS>
OWNER_NAME=<VOTRE_NOM>

# Configuration application
VITE_APP_ID=<VOTRE_APP_ID_MANUS>
VITE_APP_TITLE="Gestion des Formations Manager Itinova"
VITE_APP_LOGO=/logo.png

# APIs Manus (fournis automatiquement par la plateforme)
BUILT_IN_FORGE_API_URL=<URL_API_MANUS>
BUILT_IN_FORGE_API_KEY=<CLE_API_MANUS>
VITE_FRONTEND_FORGE_API_KEY=<CLE_API_FRONTEND_MANUS>
VITE_FRONTEND_FORGE_API_URL=<URL_API_FRONTEND_MANUS>

# Analytics (optionnel)
VITE_ANALYTICS_ENDPOINT=<URL_ANALYTICS>
VITE_ANALYTICS_WEBSITE_ID=<ID_SITE>

# Production
NODE_ENV=production
PORT=3000
```

Important : Générez une clé JWT sécurisée avec la commande suivante :

```
openssl rand -hex 32
```

Installation des dépendances

Installez toutes les dépendances du projet :

```
pnpm install
```

Cette commande télécharge et installe toutes les bibliothèques nécessaires définies dans `package.json`. Le processus peut prendre plusieurs minutes selon votre connexion internet.

Étape 3 : Migration de la base de données

Application du schéma

Appliquez le schéma de base de données en exécutant :

```
pnpm db:push
```

Cette commande utilise Drizzle Kit pour créer automatiquement toutes les tables nécessaires dans la base de données. Les tables suivantes seront créées :

Table	Description
`users`	Utilisateurs et administrateurs
`formations`	Formations disponibles
`sequences`	Séquences de formation
`apprenants`	Apprenants inscrits
`inscriptions`	Inscriptions aux séquences
`emailTemplates`	Templates d’ emails personnalisés
`alertes`	Alertes et notifications

Vérification du schéma

Connectez-vous à MySQL et vérifiez que toutes les tables ont été créées correctement :

```
mysql -u formation_user -p formation_manager
```

```
SHOW TABLES;
DESCRIBE users;
DESCRIBE formations;
```

Étape 4 : Build de production

Compilation de l'application

Compilez l'application pour la production :

```
pnpm run build
```

Cette commande effectue les opérations suivantes :

1. **Compilation TypeScript** : Transpile le code TypeScript du serveur en JavaScript
2. **Build Vite** : Compile et optimise le frontend React (minification, tree-shaking, code splitting)
3. **Génération des assets** : Crée les fichiers statiques dans `dist/public/`

Le build de production génère un répertoire `dist/` contenant :

- `dist/index.js` : Le serveur Express compilé
- `dist/public/` : Les fichiers statiques du frontend (HTML, CSS, JS, images)

Vérification du build

Vérifiez que le build s'est terminé sans erreur et que les fichiers ont été générés :

```
ls -lh dist/
ls -lh dist/public/
```

Étape 5 : Configuration du service systemd

Création du fichier service

Créez un fichier de service systemd pour gérer l’ application :

```
sudo nano /etc/systemd/system/formation-manager.service
```

Ajoutez le contenu suivant :

```
[Unit]
Description=Formation Manager Itinova
After=network.target mysql.service
Wants=mysql.service

[Service]
Type=simple
User=formation-manager
Group=formation-manager
WorkingDirectory=/opt/formation-manager/app
Environment="NODE_ENV=production"
Environment="PORT=3000"
EnvironmentFile=/opt/formation-manager/app/.env
ExecStart=/home/formation-manager/.nvm/versions/node/v22.13.0/bin/node
dist/index.js
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=formation-manager

# Limites de sécurité
LimitNOFILE=65536
PrivateTmp=true
NoNewPrivileges=true

[Install]
WantedBy=multi-user.target
```

Activation du service

Rechargez systemd, activez et démarrez le service :

```
sudo systemctl daemon-reload
sudo systemctl enable formation-manager
sudo systemctl start formation-manager
```

Vérification du statut

Vérifiez que le service fonctionne correctement :

```
sudo systemctl status formation-manager
```

Vous devriez voir `active (running)` en vert. Consultez les logs en cas d’ erreur :

```
sudo journalctl -u formation-manager -f
```

Étape 6 : Configuration Nginx (reverse proxy)

Installation de Nginx

Si Nginx n’ est pas déjà installé :

```
sudo apt update
sudo apt install nginx
```

Configuration du site

Créez un fichier de configuration pour votre site :

```
sudo nano /etc/nginx/sites-available/formation-manager
```

Ajoutez la configuration suivante :

```
upstream formation_backend {
    server 127.0.0.1:3000;
    keepalive 64;
}

server {
    listen 80;
    server_name votre-domaine.com www.votre-domaine.com;

    # Redirection HTTP vers HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name votre-domaine.com www.votre-domaine.com;

    # Certificats SSL (à configurer avec Let's Encrypt)
    ssl_certificate /etc/letsencrypt/live/votre-domaine.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/votre-domaine.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    # Logs
    access_log /var/log/nginx/formation-manager-access.log;
    error_log /var/log/nginx/formation-manager-error.log;

    # Taille maximale des uploads
    client_max_body_size 10M;

    # Compression
    gzip on;
    gzip_vary on;
    gzip_types text/plain text/css application/json application/javascript
text/xml application/xml application/xml+rss text/javascript;

    # Proxy vers Node.js
    location / {
        proxy_pass http://formation_backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        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;
        proxy_cache_bypass $http_upgrade;
        proxy_read_timeout 300s;
        proxy_connect_timeout 75s;
    }

    # Cache des assets statiques
    location ~* \.(jpg|jpeg|png|gif|ico|css|js|svg|woff|woff2|ttf|eot)$ {
        proxy_pass http://formation_backend;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
```

Remplacez `votre-domaine.com` par votre nom de domaine réel.

Activation du site

Activez la configuration et redémarrez Nginx :

```
sudo ln -s /etc/nginx/sites-available/formation-manager /etc/nginx/sites-enabled/
sudo nginx -t  # Tester la configuration
sudo systemctl restart nginx
```

Configuration SSL avec Let’s Encrypt

Installez Certbot et obtenez un certificat SSL gratuit :

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d votre-domaine.com -d www.votre-domaine.com
```

Suivez les instructions à l’écran. Certbot configurera automatiquement Nginx pour utiliser HTTPS.

Étape 7 : Configuration du pare-feu

Configuration UFW

Si vous utilisez UFW (Uncomplicated Firewall), autorisez les ports nécessaires :

```
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw enable
sudo ufw status
```

Sécurisation SSH (recommandé)

Désactivez l’ authentification par mot de passe SSH et utilisez uniquement les clés :

```
sudo nano /etc/ssh/sshd_config
```

Modifiez les lignes suivantes :

```
PasswordAuthentication no
PermitRootLogin no
```

Redémarrez SSH :

```
sudo systemctl restart sshd
```

Étape 8 : Vérification du déploiement

Tests fonctionnels

Accédez à votre application via le navigateur à l'adresse `https://votre-domaine.com` et vérifiez les éléments suivants :

Test	Description	Résultat attendu
Page d'accueil	Chargement de la page d'accueil	Affichage correct sans erreur
Connexion	Authentification via Manus OAuth	Redirection vers le tableau de bord
Tableau de bord	Affichage des statistiques	Graphiques et cartes visibles
Formations	Liste des formations	Tableau avec données
Séquences	Liste des séquences	Tableau avec code couleur
Apprenants	Liste des apprenants	Tableau avec badges de statut
Templates d'emails	Modification d'un template	Éditeur TipTap fonctionnel

Vérification des logs

Surveillez les logs pour détecter d'éventuelles erreurs :

```
# Logs de l'application
sudo journalctl -u formation-manager -f

# Logs Nginx
sudo tail -f /var/log/nginx/formation-manager-error.log

# Logs système
sudo tail -f /var/log/syslog
```

Tests de performance

Utilisez des outils pour tester les performances de votre application :

```
# Test de charge avec Apache Bench
ab -n 1000 -c 10 https://votre-domaine.com/

# Analyse des temps de réponse
curl -w "@curl-format.txt" -o /dev/null -s https://votre-domaine.com/
```

Maintenance et mises à jour

Sauvegarde de la base de données

Créez un script de sauvegarde automatique :

```
sudo nano /opt/formation-manager/backup-db.sh
```

Contenu du script :

```
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="/opt/formation-manager/backups"
mkdir -p $BACKUP_DIR

mysqldump -u formation_user -p'MOT_DE_PASSE' formation_manager | gzip >
$BACKUP_DIR/formation_manager_$DATE.sql.gz

# Garder seulement les 30 dernières sauvegardes
find $BACKUP_DIR -name "formation_manager_*.sql.gz" -mtime +30 -delete
```

Rendez le script exécutable et ajoutez-le au cron :

```
chmod +x /opt/formation-manager/backup-db.sh
sudo crontab -e
```

Ajoutez la ligne suivante pour une sauvegarde quotidienne à 2h du matin :

```
0 2 * * * /opt/formation-manager/backup-db.sh
```

Mise à jour de l'application

Pour mettre à jour l'application vers une nouvelle version :

```
cd /opt/formation-manager/app
sudo systemctl stop formation-manager
git pull origin main
pnpm install
pnpm db:push  # Appliquer les migrations
pnpm run build
sudo systemctl start formation-manager
```

Surveillance et monitoring

Installez des outils de monitoring pour surveiller les performances :

PM2 : Alternative à systemd avec monitoring intégré **Prometheus + Grafana** : Monitoring avancé et dashboards **Uptime Kuma** : Surveillance de disponibilité **New Relic / DataDog** : Solutions SaaS complètes

---

Dépannage

L'application ne démarre pas

Vérifiez les logs pour identifier l'erreur :

```
sudo journalctl -u formation-manager -n 100
```

Causes courantes :

- **Erreur de connexion à la base de données** : Vérifiez `DATABASE_URL` dans `.env`
- **Port déjà utilisé** : Changez le port dans `.env` ou arrêtez le processus conflictuel
- **Permissions insuffisantes** : Vérifiez les droits sur `/opt/formation-manager/app`

Erreur 502 Bad Gateway

Cette erreur indique que Nginx ne peut pas se connecter au backend Node.js. Vérifications :

```
# Vérifier que le service tourne
sudo systemctl status formation-manager

# Vérifier que le port 3000 écoute
sudo netstat -tulpn | grep 3000

# Tester la connexion locale
curl http://localhost:3000
```

Problèmes de performance

Si l'application est lente :

1. **Optimiser les requêtes SQL** : Ajoutez des index sur les colonnes fréquemment interrogées
2. **Activer le cache Redis** : Mettez en cache les résultats des requêtes coûteuses
3. **Augmenter les ressources** : Ajoutez plus de RAM ou de CPU
4. **Activer la compression Nginx** : Déjà configurée dans l'exemple ci-dessus

Base de données corrompue

En cas de corruption de la base de données, restaurez depuis une sauvegarde :

```
gunzip < /opt/formation-
manager/backups/formation_manager_YYYYMMDD_HHMMSS.sql.gz | mysql -u
formation_user -p formation_manager
```

Sécurité

Bonnes pratiques

Suivez ces recommandations pour sécuriser votre déploiement :

Mettez à jour régulièrement : Appliquez les mises à jour de sécurité du système d'exploitation et des dépendances Node.js.

Utilisez HTTPS uniquement : Forcez la redirection HTTP vers HTTPS et activez HSTS (HTTP Strict Transport Security).

Limitez les accès SSH : Utilisez des clés SSH au lieu de mots de passe et limitez l'accès par IP si possible.

Configurez un WAF : Utilisez un Web Application Firewall comme ModSecurity ou Cloudflare pour bloquer les attaques courantes.

Surveillez les logs : Configurez des alertes pour détecter les tentatives d'intrusion ou les comportements anormaux.

Sauvegardez régulièrement : Automatisez les sauvegardes de la base de données et testez régulièrement la restauration.

Checklist de sécurité

Avant de mettre en production, vérifiez les points suivants :

- [ ] Certificat SSL valide configuré
- [ ] Pare-feu activé et configuré
- [ ] Authentification SSH par clé uniquement
- [ ] Variables d'environnement sécurisées (pas de valeurs par défaut)

- [ ] Sauvegardes automatiques configurées
- [ ] Monitoring et alertes en place
- [ ] Logs rotatifs configurés
- [ ] Permissions fichiers correctes (pas de 777)
- [ ] Base de données accessible uniquement en local
- [ ] Rate limiting configuré sur Nginx

Support et contact

Pour toute question ou problème concernant le déploiement :

- **Documentation technique** : Consultez le README.md du projet
- **Support Manus** : https://help.manus.im
- **Contact développeur** : o.pareige@itinova.org

Fin du guide de déploiement