# Guide Complet de Déploiement de l'Application de Gestion d'Immobilisations

Ce document fournit des instructions détaillées pour déployer l'application de gestion d'immobilisations sur différents environnements : un PC local, un Raspberry Pi (Raspbian) et un serveur OVH. Chaque section est autonome et couvre les prérequis, l'installation, la configuration et le lancement de l'application.

---

## 1. Guide de Déploiement Local (PC)

## 2. Guide de Déploiement sur Raspberry Pi (Raspbian)

# Guide de Déploiement sur Raspberry Pi (Raspbian)

Ce guide vous expliquera comment déployer l'application de gestion d'immobilisations sur un Raspberry Pi fonctionnant sous Raspbian (maintenant appelé Raspberry Pi OS). Le processus est similaire au déploiement sur un PC Linux, mais avec quelques spécificités liées à l'architecture ARM et à l'environnement du Raspberry Pi.

## 1. Prérequis

Avant de commencer, assurez-vous que votre Raspberry Pi est configuré et que les logiciels suivants sont installés :

### 1.1. Système d'Exploitation

Assurez-vous que votre Raspberry Pi exécute une version récente de Raspberry Pi OS (anciennement Raspbian). Vous pouvez le télécharger et l'installer en suivant les instructions officielles [3].

## 1.2. Mises à Jour du Système

Il est crucial de mettre à jour votre système avant d'installer de nouveaux paquets. Ouvrez un terminal sur votre Raspberry Pi et exécutez les commandes suivantes :

Bash

```
sudo apt update
sudo apt upgrade -y
```

## 1.3. Node.js et npm

Pour le frontend React, vous aurez besoin de Node.js et npm. L'installation de Node.js sur Raspberry Pi peut être légèrement différente de celle sur un PC de bureau. Il est recommandé d'utiliser `nvm` (Node Version Manager) pour gérer les versions de Node.js, mais pour une installation simple, vous pouvez utiliser `apt` ou télécharger les binaires précompilés.

**Option A: Installation via `apt` (recommandé pour la simplicité)**

Bash

```
sudo apt install nodejs npm -y
```

**Option B: Installation via `nvm` (pour une meilleure gestion des versions)**

Bash

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh |
bash
# Redémarrez votre terminal ou sourcez le fichier .bashrc/.profile
source ~/.bashrc # ou ~/.profile
nvm install node
nvm use node
```

Vérifiez l'installation :

Bash

```
node -v
npm -v
```

### 1.4. Python et pip

Python est généralement préinstallé sur Raspbian. Assurez-vous d'avoir Python 3 et pip :

Bash

```
python3 --version
pip3 --version
```

Si pip3 n'est pas installé :

Bash

```
sudo apt install python3-pip -y
```

### 1.5. Git (Optionnel mais Recommandé)

Bash

```
sudo apt install git -y
```

## 2. Téléchargement du Code Source

Comme pour le déploiement local, vous pouvez cloner le dépôt Git ou télécharger l'archive ZIP. Il est recommandé de cloner le dépôt directement sur votre Raspberry Pi.

Bash

```
git clone <URL_DU_DEPOT>
cd <NOM_DU_DOSSIER_APPLICATION>
```

## 3. Configuration du Backend (Flask)

Le backend Flask fonctionnera de la même manière que sur un PC local, utilisant SQLite.

### 3.1. Accéder au Dossier du Backend

Bash

```bash
cd gestion-immobilisations-backend
```

### 3.2. Création et Activation de l'Environnement Virtuel

Bash

```bash
python3 -m venv venv
source venv/bin/activate
```

### 3.3. Installation des Dépendances du Backend

Bash

```bash
pip install -r requirements.txt
# Ou manuellement si requirements.txt n'est pas fourni:
# pip install Flask Flask-SQLAlchemy Flask-CORS
```

### 3.4. Lancement du Backend

Bash

```bash
python3 src/main.py
```

Laissez ce terminal ouvert. Pour un déploiement en production, vous utiliserez un service comme `systemd` pour le faire fonctionner en arrière-plan (voir section 5).

## 4. Configuration du Frontend (React)

Le frontend est déjà compilé et inclus dans le dossier `static` du backend. Vous n'avez donc pas besoin de le recompiler sur le Raspberry Pi pour une utilisation normale. Cependant, si

vous souhaitez le modifier ou le recompiler :

### 4.1. Accéder au Dossier du Frontend

```bash
cd gestion-immobilisations
```

### 4.2. Installation des Dépendances du Frontend

```bash
npm install
```

### 4.3. Compilation du Frontend

```bash
npm run build
```

Après la compilation, copiez les fichiers générés dans le dossier `static` de votre backend :

```bash
cp -r dist/* ../gestion-immobilisations-backend/src/static/
```

## 5. Accès à l'Application

Une fois le backend lancé sur votre Raspberry Pi, vous pouvez y accéder depuis un autre appareil sur le même réseau en utilisant l'adresse IP de votre Raspberry Pi et le port 5000.

Pour trouver l'adresse IP de votre Raspberry Pi :

```bash
hostname -I
```

Si l'adresse IP de votre Raspberry Pi est `192.168.1.100`, l'application sera accessible à :

Plain Text

```
http://192.168.1.100:5000/
```

# 6. Déploiement en Production (Service Systemd)

Pour que l'application s'exécute en arrière-plan et démarre automatiquement au démarrage du Raspberry Pi, vous pouvez créer un service `systemd`.

## 6.1. Créer un Fichier de Service

Créez un fichier de service, par exemple `/etc/systemd/system/immobilisations.service` :

Bash

```
sudo nano /etc/systemd/system/immobilisations.service
```

Collez le contenu suivant (adaptez les chemins si nécessaire) :

Plain Text

```
[Unit]
Description=Application de Gestion d'Immobilisations
After=network.target

[Service]
User=pi
WorkingDirectory=/home/pi/gestion-immobilisations-backend
ExecStart=/home/pi/gestion-immobilisations-backend/venv/bin/python
/home/pi/gestion-immobilisations-backend/src/main.py
Restart=always

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

**Note** : Remplacez `/home/pi/gestion-immobilisations-backend` par le chemin réel de votre dossier backend et `User=pi` par votre nom d'utilisateur si ce n'est pas `pi`.

### 6.2. Activer et Démarrer le Service

```bash
sudo systemctl daemon-reload
sudo systemctl enable immobilisations.service
sudo systemctl start immobilisations.service
```

Pour vérifier le statut du service :

```bash
sudo systemctl status immobilisations.service
```

## 7. Dépannage Courant

- **Permissions** : Assurez-vous que l'utilisateur sous lequel le service s'exécute a les permissions nécessaires pour accéder aux fichiers de l'application et à la base de données SQLite.
- **Pare-feu** : Si vous avez un pare-feu activé sur votre Raspberry Pi (par exemple `ufw`), assurez-vous que le port 5000 est ouvert :

---

## Références

[3] Raspberry Pi. (n.d.). *Télécharger Raspberry Pi OS*. Consulté le 27 juin 2025, à l'adresse https://www.raspberrypi.com/software/

## 3. Guide de Déploiement sur Serveur OVH

# Guide de Déploiement sur Serveur OVH

Ce guide vous expliquera comment déployer l'application de gestion d'immobilisations sur un serveur dédié ou un VPS (Virtual Private Server) chez OVH. Le déploiement sur un serveur

distant implique des étapes supplémentaires par rapport à un déploiement local, notamment la configuration d'un serveur web (Nginx ou Apache) et d'un serveur d'applications (Gunicorn ou uWSGI) pour gérer le backend Flask en production.

# 1. Prérequis

Avant de commencer, assurez-vous d'avoir :

## 1.1. Un Serveur OVH

Vous devez disposer d'un serveur dédié ou d'un VPS chez OVH avec un accès SSH. Assurez-vous que le système d'exploitation est une distribution Linux récente (par exemple, Ubuntu Server, Debian).

## 1.2. Accès SSH

Vous aurez besoin d'un client SSH (PuTTY pour Windows, terminal pour macOS/Linux) pour vous connecter à votre serveur. Assurez-vous d'avoir les identifiants de connexion (adresse IP du serveur, nom d'utilisateur, mot de passe ou clé SSH).

## 1.3. Noms de Domaine (Optionnel mais Recommandé)

Si vous souhaitez que votre application soit accessible via un nom de domaine (par exemple, `mon-appli.com`), assurez-vous d'avoir configuré les enregistrements DNS (A record) pour pointer votre domaine vers l'adresse IP de votre serveur OVH.

## 1.4. Mises à Jour du Système

Connectez-vous à votre serveur via SSH et mettez à jour le système :

Bash

```
sudo apt update
sudo apt upgrade -y
```

## 1.5. Node.js et npm

Installez Node.js et npm sur votre serveur. Il est recommandé d'utiliser `nvm` pour une meilleure gestion des versions, mais `apt` est suffisant pour une installation simple :

**Option A: Installation via `apt` (recommandé pour la simplicité)**

Bash

```
sudo apt install nodejs npm -y
```

**Option B: Installation via `nvm` (pour une meilleure gestion des versions)**

Bash

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh |
bash
source ~/.bashrc # ou ~/.profile
nvm install node
nvm use node
```

Vérifiez l'installation :

Bash

```
node -v
npm -v
```

## 1.6. Python et pip

Installez Python 3 et pip :

Bash

```
sudo apt install python3 python3-pip -y
```

## 1.7. Git

Bash

```
sudo apt install git -y
```

## 2. Téléchargement du Code Source

Clonez le dépôt Git de l'application sur votre serveur. Choisissez un répertoire approprié, par exemple `/var/www/` ou `/opt/`.

Bash

```
cd /var/www/
sudo git clone <URL_DU_DEPOT> gestion-immobilisations-app
cd gestion-immobilisations-app
```

**Note** : Vous devrez peut-être ajuster les permissions du dossier si vous utilisez `sudo` pour le clonage. Il est souvent préférable de cloner en tant qu'utilisateur non-root, puis d'ajuster les permissions.

## 3. Configuration du Backend (Flask avec Gunicorn)

Pour la production, nous utiliserons Gunicorn comme serveur d'applications WSGI pour exécuter Flask, et Nginx comme proxy inverse pour servir les requêtes web.

### 3.1. Accéder au Dossier du Backend

Bash

```
cd gestion-immobilisations-backend
```

### 3.2. Création et Activation de l'Environnement Virtuel

Bash

```
python3 -m venv venv
source venv/bin/activate
```

### 3.3. Installation des Dépendances du Backend

Installez les dépendances Python, y compris Gunicorn :

Bash

```
pip install -r requirements.txt gunicorn
# Ou manuellement:
# pip install Flask Flask-SQLAlchemy Flask-CORS gunicorn
```

### 3.4. Test de Gunicorn

Testez que Gunicorn peut lancer votre application Flask :

Bash

```
gunicorn --bind 0.0.0.0:5000 src.main:app
```

Laissez-le tourner quelques instants, puis arrêtez avec `Ctrl+C`. Si vous voyez des erreurs, corrigez-les avant de continuer.

## 4. Configuration du Frontend (React)

Le frontend est déjà compilé et inclus dans le dossier `static` du backend. Vous n'avez pas besoin de le recompiler sur le serveur pour une utilisation normale. Si vous avez mis à jour le frontend localement, assurez-vous de copier les fichiers `dist` mis à jour dans le dossier `src/static` de votre backend avant de transférer le code sur le serveur.

## 5. Configuration de Nginx comme Proxy Inverse

Nginx servira les fichiers statiques du frontend et transmettra les requêtes API au backend Gunicorn.

### 5.1. Installer Nginx

Bash

```
sudo apt install nginx -y
```

### 5.2. Configurer Nginx

Créez un nouveau fichier de configuration Nginx pour votre application, par exemple `/etc/nginx/sites-available/immobilisations` :

Bash

```
sudo nano /etc/nginx/sites-available/immobilisations
```

Collez le contenu suivant (adaptez `server_name` et les chemins) :

Plain Text

```
server {
    listen 80;
    server_name your_domain.com www.your_domain.com; # Remplacez par votre
nom de domaine ou l'IP du serveur

    location /static/ {
        alias /var/www/gestion-immobilisations-app/gestion-immobilisations-
backend/src/static/;
        try_files $uri $uri/ =404;
    }

    location / {
        proxy_pass http://127.0.0.1:5000;
        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;
    }
}
```

## 5.3. Activer la Configuration Nginx

Créez un lien symbolique vers `sites-enabled` et testez la configuration :

Bash

```
sudo ln -s /etc/nginx/sites-available/immobilisations /etc/nginx/sites-
enabled/
sudo nginx -t
```

Si le test est réussi, redémarrez Nginx :

Bash

```
sudo systemctl restart nginx
```

# 6. Création d'un Service Systemd pour Gunicorn

Pour que Gunicorn s'exécute en arrière-plan et démarre automatiquement, créez un service `systemd`.

## 6.1. Créer un Fichier de Service

Créez un fichier de service, par exemple `/etc/systemd/system/immobilisations-backend.service` :

Bash

```
sudo nano /etc/systemd/system/immobilisations-backend.service
```

Collez le contenu suivant (adaptez les chemins et l'utilisateur) :

Plain Text

```
[Unit]
Description=Gunicorn instance for Immobilisations App
After=network.target

[Service]
User=www-data # Ou l'utilisateur sous lequel vous voulez exécuter
l'application
WorkingDirectory=/var/www/gestion-immobilisations-app/gestion-
immobilisations-backend
ExecStart=/var/www/gestion-immobilisations-app/gestion-immobilisations-
backend/venv/bin/gunicorn --workers 3 --bind unix:/tmp/immobilisations.sock
src.main:app
Restart=always

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

**Note** : Nous utilisons un socket Unix ( `unix:/tmp/immobilisations.sock` ) pour la communication entre Nginx et Gunicorn, ce qui est plus performant et sécurisé que les ports TCP locaux

pour les communications inter-processus sur le même serveur.

### 6.2. Activer et Démarrer le Service

Bash

```
sudo systemctl daemon-reload
sudo systemctl enable immobilisations-backend.service
sudo systemctl start immobilisations-backend.service
```

Pour vérifier le statut du service :

Bash

```
sudo systemctl status immobilisations-backend.service
```

## 7. Configuration du Pare-feu (UFW)

Si vous utilisez `ufw` (Uncomplicated Firewall), assurez-vous que les ports 80 (HTTP) et 443 (HTTPS, si vous configurez SSL) sont ouverts :

Bash

```
sudo ufw allow 'Nginx HTTP'
# sudo ufw allow 'Nginx HTTPS' # Si vous configurez SSL
sudo ufw enable
```

## 8. Accès à l'Application

Votre application devrait maintenant être accessible via votre nom de domaine (si configuré) ou l'adresse IP de votre serveur OVH.

Plain Text

```
http://your_domain.com/
# ou
http://YOUR_SERVER_IP/
```

## 9. Configuration SSL (HTTPS) (Recommandé)

Pour sécuriser votre application, il est fortement recommandé de configurer HTTPS avec un certificat SSL. Let's Encrypt offre des certificats gratuits et faciles à installer avec Certbot.

### 9.1. Installer Certbot

Bash

```
sudo apt install certbot python3-certbot-nginx -y
```

### 9.2. Obtenir et Installer le Certificat

Bash

```
sudo certbot --nginx -d your_domain.com -d www.your_domain.com
```

Suivez les instructions. Certbot configurera automatiquement Nginx pour utiliser HTTPS et mettra en place le renouvellement automatique du certificat.

## 10. Dépannage Courant

- **Erreurs Nginx** : Vérifiez les logs Nginx ( `sudo tail -f /var/log/nginx/error.log` ).
- **Erreurs Gunicorn/Flask** : Vérifiez les logs du service systemd ( `sudo journalctl -u immobilisations-backend.service -f` ).
- **Permissions** : Assurez-vous que l'utilisateur sous lequel Gunicorn s'exécute ( `www-data` ou autre) a les droits de lecture/écriture sur les fichiers de l'application et la base de données SQLite.
- **Pare-feu** : Vérifiez que les ports nécessaires sont ouverts.

---

## Références

[4] Nginx. (n.d.). *Welcome to Nginx*. Consulté le 27 juin 2025, à l'adresse https://nginx.org/
[5] Gunicorn. (n.d.). *Gunicorn Documentation*. Consulté le 27 juin 2025, à l'adresse https://gunicorn.org/
[6] Certbot. (n.d.). *Certbot Documentation*. Consulté le 27 juin 2025, à l'adresse https://certbot.eff.org/