Manuel d'installation

NIS CUBE

Abdelwaheb SEBA
Mohammed Ryad DERMOUCHE
Chatodit LOKONGA KONGOLO
Sedik SI MEHAND

Groupe L3L1
2024 - 2025

Informations sur le document

Les informations d'identification du document	
Référence du document :	NIS-MI-V1.00
Version du document :	1.00
Date du document :	04/05/2025
Auteur(s) :	Groupe L3L1

Les éléments de vérification du document	
Validé par :	Nicolas DENIS
Validé le :	04/05/2025
Soumis le :	04/05/2025
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	Réservé aux membres du groupe L3L1
	L'encadrant monsieur Nicolas DENIS
	Le responsable de l'UE monsieur David JANISZEK
	Les membres du jury lors de la soutenance du projet

Table des matières

1 Introduction

1.1 Contexte

Avec l'évolution constante des cybermenaces et l'entrée en vigueur de la directive **NIS2** (Network and Information Security), les entreprises et administrations doivent renforcer leur niveau de sécurité et assurer leur conformité aux nouvelles exigences européennes. Afin de répondre à ce besoin, le projet **NIS CUBE** propose une solution innovante sous la forme d'une **interface web** permettant aux entreprises d'estimer leur niveau de mise en conformité et, par conséquent, d'améliorer leur sécurité.

Le principe du projet repose sur un processus d'**évaluation interactive** : les utilisateurs répondent à un ensemble de questions visant à mesurer la conformité des entités qu'ils représentent par rapport aux exigences de la directive NIS2. En fonction des réponses fournies, des **scores** sont attribués, et les résultats sont affichés sous forme de **représentations graphiques** afin d'identifier clairement les aspects à améliorer.

1.2 Historique

Historiquement, la première directive **NIS** (2016) avait pour objectif d'établir un niveau élevé commun de sécurité des réseaux et des systèmes d'information à travers l'Union européenne. Face à l'augmentation des menaces informatiques et à la nécessité d'impliquer un plus grand nombre d'acteurs, la directive **NIS2** est venue renforcer et étendre les obligations de sécurité dans plusieurs secteurs (énergie, transports, santé, infrastructures numériques, etc.).

Ces évolutions ont généré de nouveaux besoins : les organisations doivent désormais rendre compte de leur conformité plus précisément et faire preuve de davantage de réactivité en matière de cybersécurité. C'est dans ce contexte que s'inscrit **NIS CUBE**, afin de fournir aux acteurs concernés un outil simple et adapté pour piloter leur mise en conformité.

1.3 Objectifs du document

Ce document a pour objectif de présenter l'ensemble des étapes nécessaires pour déployer et mettre en production le projet **NIS CUBE** sur un serveur distant (VPS). Il constitue un manuel technique détaillé, destiné à toute personne souhaitant installer, configurer et sécuriser l'environnement complet de l'application, du backend à la base de données, en passant par le frontend et le nom de domaine.

Le guide couvre notamment :

- la préparation et la sécurisation du serveur VPS (utilisateurs, pare-feu, accès SSH, ports) ;
- l'installation et la configuration de PostgreSQL, ainsi que l'importation des données ;
- le déploiement du backend `Express.js` et sa gestion avec `PM2` ;
- la configuration et la compilation du frontend développé avec `Vue.js` et `Vite` ;
- la mise en place d'un serveur `Nginx` en tant que reverse proxy ;
- l'association du nom de domaine `niscube.fr` au serveur ;
- la génération d'un certificat SSL avec `Certbot` pour activer le protocole HTTPS.

Ce manuel constitue donc une documentation complète pour assurer une installation reproductible, fiable et sécurisée de l'application **NIS CUBE**.

2 Préparation et sécurisation du serveur VPS

Cette section décrit les étapes nécessaires pour préparer un serveur VPS vierge à accueillir le projet **NIS CUBE** dans un environnement sécurisé et maintenable.

2.1 Connexion initiale au VPS

Après la commande du serveur chez l'hébergeur (par exemple IONOS), un accès SSH est fourni. Il est recommandé de se connecter avec l'utilisateur `root` uniquement pour les premières étapes.

```
ssh root@<IP_DU_SERVEUR>
```

Listing 1 – Connexion SSH initiale

2.2 Création du super-utilisateur

Pour éviter l'usage prolongé du compte `root`, un utilisateur privilégié est créé avec les droits `sudo`. Dans le cadre du projet, cet utilisateur est nommé `nicolasdenis`.

```
adduser nicolasdenis
usermod -aG sudo nicolasdenis
```

Listing 2 – Ajout de l'utilisateur nicolasdenis

Une fois ce compte créé, toutes les opérations d'administration doivent être réalisées via cet utilisateur.

2.3 Désactivation de l'accès root en SSH

Pour renforcer la sécurité, on désactive la connexion SSH directe en root :

```
sudo nano /etc/ssh/sshd_config
# Modifier ou ajouter :
PermitRootLogin no
```

Listing 3 – Désactivation de root dans sshd_config

Puis on redémarre le service SSH :

```
sudo systemctl restart ssh
```

2.4 Configuration du pare-feu UFW

On utilise `ufw` pour limiter les connexions entrantes aux services essentiels :

```
sudo apt install ufw -y
sudo ufw allow 22 # SSH
sudo ufw allow 80 # HTTP
sudo ufw allow 443 # HTTPS
sudo ufw enable
```

Listing 4 – Activation de UFW

2.5 Installation de logiciels essentiels

On installe les outils nécessaires pour le déploiement du projet :

```
sudo apt update && sudo apt upgrade -y
sudo apt install curl git unzip net-tools build-essential -y
```

Listing 5 – Mise à jour et outils

2.6 Configuration du nom de domaine (chez l'hébergeur)

Dans le panneau de configuration DNS de l'hébergeur (IONOS), on crée un enregistrement `A` pointant vers l'adresse IP du VPS :

- Domaine : `niscube.fr`
- Type : `A`
- Valeur : `<IP publique du VPS>`

Une fois la propagation DNS terminée, le nom de domaine pourra être utilisé pour le serveur web.

3 Installation et configuration de PostgreSQL

Le projet **NIS CUBE** repose sur une base de données PostgreSQL pour le stockage des informations relatives aux utilisateurs, aux évaluations et aux résultats. Cette section décrit l'installation de PostgreSQL, sa sécurisation, ainsi que l'importation des données depuis un fichier `.sql` existant.

3.1 Installation de PostgreSQL

On installe PostgreSQL à l'aide du gestionnaire de paquets :

```
sudo apt update
sudo apt install postgresql postgresql-contrib -y
```

Listing 6 – Installation de PostgreSQL

Une fois l'installation terminée, le service PostgreSQL est automatiquement lancé.

3.2 Connexion à PostgreSQL

On se connecte au terminal PostgreSQL via l'utilisateur système `postgres` :

```
sudo -u postgres psql
```

Listing 7 – Connexion à PostgreSQL

3.3 Création de la base de données

Dans l'invite PostgreSQL, on crée la base de données utilisée par le projet :

```
CREATE DATABASE niscube;
\q
```

Listing 8 – Création de la base de données niscube

3.4 Changement du mot de passe de l'utilisateur postgres

Toujours via le terminal, on attribue un mot de passe fort à l'utilisateur `postgres` (requis pour les connexions à distance) :

```
sudo -u postgres psql
```

Listing 9 – Changer le mot de passe

Puis dans l'invite PostgreSQL :

```
ALTER USER postgres WITH PASSWORD 'motdepassefort';
\q
```

3.5 Sécurité : laisser PostgreSQL accessible uniquement en local

Pour des raisons de sécurité, la base de données PostgreSQL ne doit pas être exposée publiquement. Aucune connexion distante (depuis l'extérieur du serveur) n'est autorisée : seul le backend Express, exécuté sur le même VPS, pourra accéder à la base en local.

Il n'est donc pas nécessaire de modifier les fichiers `postgresql.conf` ni `pg_hba.conf`, puisque PostgreSQL écoute déjà sur `localhost` uniquement.

- L'adresse de connexion dans le backend est : `localhost:5432`
- Le mot de passe `postgres` défini précédemment est requis

Le port 5432 peut également être fermé dans le pare-feu (UFW), sauf si une administration distante temporaire est nécessaire via `pgAdmin`.

3.6 Importation des données via un fichier .sql

Un fichier de sauvegarde nommé `dump.sql` est fourni avec le projet. Il contient la structure complète de la base ainsi que des données d'exemple permettant de cloner l'environnement de développement.

Pour restaurer cette base dans PostgreSQL, on utilise la commande suivante :

```
psql -U postgres -d niscube -h localhost -f dump.sql
```

Listing 10 – Import du dump SQL

Cette commande suppose que :

- le fichier `dump.sql` est situé dans le répertoire courant sur le serveur ;
- le mot de passe de l'utilisateur `postgres` a été correctement défini ;
- la base `niscube` existe déjà (voir section précédente).

Cette procédure permet de répliquer rapidement l'environnement de travail du projet **NIS CUBE** sans devoir recréer manuellement les tables ni les données.

4 Déploiement du backend Express.js et gestion avec PM2

Cette section décrit comment déployer la partie **backend** du projet **NIS CUBE**, basée sur `Node.js` et `Express.js`, et comment l'exécuter en arrière-plan grâce à `PM2`, un gestionnaire de processus adapté à la production.

4.1 Installation de Node.js et npm

On commence par installer Node.js (version LTS recommandée) et son gestionnaire de paquets `npm` :

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install nodejs -y
```

Listing 11 – Installation de Node.js et npm

4.2 Récupération du code source depuis le dépôt SVN

Le code source du projet **NIS CUBE** (frontend et backend) est versionné dans un dépôt SVN. Il doit être récupéré sur le serveur VPS avant de poursuivre le déploiement.

```
sudo apt install subversion -y
svn checkout <URL_DU_DEPOT_SVN> /var/www/niscube
```

Listing 12 – Installation de Subversion et clonage du dépôt

Le dépôt doit contenir la structure suivante, avec les deux sous-dossiers `backend` et `frontend` :

```
/var/www/niscube/
|---backend/
      |---.env
      |---server.js
      |---package.json
      '---src/
            |---config/
            |---controllers/
            |---middleware/
            |---models/
            |---routes/
            |---services/
            '---utils/
```

Listing 13 – Structure attendue du projet

Vérifiez que les permissions appartiennent à l'utilisateur système responsable du déploiement (par exemple `abdou`) :

```
sudo chown -R nicolasdenis:nicolasdenis /var/www/niscube
```

Listing 14 – Ajustement des droits d'accès

4.3 Installation des dépendances du backend

Une fois dans le répertoire du backend, on installe les dépendances définies dans le fichier `package.json` :

```
cd /var/www/niscube/backend
npm install
```

Listing 15 – Installation des dépendances npm

4.4 Création du fichier `.env`

Le backend utilise des variables d'environnement stockées dans un fichier `.env`. En voici un exemple à adapter :

```
DB_USER=postgres
DB_HOST=localhost
DB_NAME=niscube
DB_PASSWORD=<mot_de_passe>
DB_PORT=5432

JWT_SECRET=<clef_secrete>

SMTP_HOST=smtp.exemple.com
SMTP_PORT=465
SMTP_SECURE=true
SMTP_USER=apikey
SMTP_PASSWORD=<clé_API>
SMTP_FROM="Équipe NIS CUBE <info@niscube.fr>"
```

Listing 16 – Exemple de fichier .env

4.5 Test local du serveur Express.js

Avant de passer en production, on vérifie que l'API démarre bien :

```
node server.js
```

Listing 17 – Test du backend en local

Une fois l'API démarrée, on peut tester un endpoint depuis le serveur :

```
curl http://localhost:3000/api/axes
```

Listing 18 – Test avec curl

4.6 Installation de PM2

PM2 est installé globalement pour gérer les processus Node.js en tant que services persistants :

```
sudo npm install -g pm2
```

Listing 19 – Installation de PM2

4.7 Démarrage du backend avec PM2

On lance le backend avec PM2 depuis le bon répertoire :

```
pm2 start server.js --name niscube --cwd /var/www/niscube/backend
```

Listing 20 – Démarrage du backend

4.8 Persistance du service au redémarrage

Pour garantir que le backend redémarre automatiquement après un redémarrage du VPS, on configure le démarrage automatique de PM2 :

```
pm2 save
pm2 startup
\# Puis exécuter la commande suggérée par PM2
```

Listing 21 – Activation du redémarrage automatique

4.9 Suivi et supervision

Voici quelques commandes utiles pour superviser le backend :

- `pm2 list` : liste les applications gérées.
- `pm2 logs niscube` : affiche les logs en temps réel.
- `pm2 restart niscube` : redémarre l'application.
- `pm2 stop niscube` : arrête temporairement l'application.
- `pm2 delete niscube` : supprime l'application de PM2.

5 Configuration et compilation du frontend (Vue.js + Vite)

Cette section explique comment configurer l'environnement frontend du projet **NIS CUBE**, développé avec `Vue.js 3` et `Vite`, puis comment le compiler en version de production.

5.1 Récupération du code frontend via SVN

Le code source du frontend est versionné dans un dépôt `SVN`, conjointement avec le backend. Il est cloné dans le répertoire `/var/www/niscube` à l'aide de la commande suivante (voir section précédente) :

```
svn checkout <URL_DU_DEPOT_SVN> /var/www/niscube
```

Listing 22 – Clonage du dépôt SVN

Après clonage, le dossier `frontend/` se trouve dans `/var/www/niscube/frontend`.

5.2 Installation des dépendances du frontend

Depuis le répertoire `frontend`, on installe les dépendances définies dans le fichier `package.json` :

```
cd /var/www/niscube/frontend
npm install
```

Listing 23 – Installation des dépendances frontend

5.3 Fichier `.env.production`

Le frontend utilise un fichier `.env.production` pour définir l'URL du backend (API). Ce fichier doit se trouver à la racine du dossier `frontend` :

```
VITE_API_BASE_URL=https://niscube.fr/api
```

Listing 24 – Exemple de .env.production

Attention : assurez-vous que l'URL corresponde bien au nom de domaine final, configuré dans Nginx.

5.4 Compilation en mode production

Une fois les dépendances installées et l'environnement configuré, on compile le projet en version optimisée pour la production :

```
npm run build
```

Listing 25 – Compilation du frontend

Cette commande génère un dossier `dist/` contenant les fichiers statiques optimisés (HTML, JS, CSS, images, etc.).

5.5 Structure typique du dossier frontend

Le répertoire `frontend` suit l'organisation suivante :

```
frontend/
|---.env.production
|---index.html
|---vite.config.js
|---package.json
|---public/
'---src/
      |---assets/
      |---components/
      |---data/
      |---store/
      |---style/
      |---types/
      '---views/
```

Listing 26 – Structure du frontend

6 Mise en place du serveur Nginx en tant que reverse proxy

Dans cette section, nous configurons **Nginx** pour qu'il agisse comme *reverse proxy* devant l'API backend (port 3000) et serve le frontend (fichiers statiques du dossier `dist/` généré avec Vite).

6.1 Installation de Nginx

On commence par installer Nginx via le gestionnaire de paquets :

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

Listing 27 – Installation de Nginx

6.2 Structure des répertoires

Les fichiers du projet sont supposés être organisés ainsi :

```
/var/www/niscube/
|---backend/ # Serveur Express lancé via PM2
'---frontend/
'---dist/ # Fichiers statiques après la compilation Vite
```

Listing 28 – Répertoire cible

6.3 Configuration du serveur Nginx

On crée un fichier de configuration pour le domaine `niscube.fr` :

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

Listing 29 – Création du fichier de configuration Nginx

Voici un exemple de configuration à adapter :

```
server {
      listen 80;
      server_name niscube.fr www.niscube.fr;

      root /var/www/niscube/frontend/dist;
      index index.html;

      location / {
            try_files $uri $uri/ /index.html;
      }

      location /api/ {
            proxy_pass http://localhost:3000/;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
      }
}
```

Listing 30 – Exemple de configuration pour Nginx

Remarques importantes :

- Le frontend est servi directement depuis le dossier `dist`.
- Les requêtes commençant par `/api/` sont redirigées vers le backend Express.

6.4 Activation de la configuration et redémarrage

On active le site et redémarre Nginx :

```
sudo ln -s /etc/nginx/sites-available/niscube /etc/nginx/sites-enabled/
sudo nginx -t # Vérification de la syntaxe
sudo systemctl restart nginx
```

Listing 31 – Activation de la configuration

6.5 Accès au site

Le projet est désormais accessible via le nom de domaine configuré, par exemple :

- `https://niscube.fr` — frontend (interface utilisateur)
- `https://niscube.fr/api/axes` — endpoint d'API backend

7 Association du nom de domaine `niscube.fr` au serveur

Pour que le projet soit accessible via une adresse claire comme `niscube.fr`, il est nécessaire d'associer ce nom de domaine à l'adresse IP publique du serveur.

7.1 Configuration DNS chez l'hébergeur

L'association se fait depuis l'espace client du fournisseur de domaine (par exemple IONOS). Il faut créer un enregistrement de type `A` dans la zone DNS du domaine :

- **Nom** : `@` (ou vide, selon l'interface)
- **Type** : `A`
- **Valeur** : `<IP_publique_du_serveur>`
- **TTL** : 1h ou automatique

7.2 Redirection avec ou sans www

Il est recommandé d'ajouter un second enregistrement pour la version `www.niscube.fr` :

- **Nom** : `www`
- **Type** : `A`
- **Valeur** : `<IP_publique_du_serveur>`

7.3 Propagation DNS

Une fois les modifications effectuées, il faut attendre la propagation DNS. Celle-ci prend généralement quelques minutes, mais peut parfois durer jusqu'à 24 heures.

Pour vérifier que le domaine pointe bien vers le serveur, on peut utiliser :

```
dig niscube.fr +short
ping niscube.fr
```

Listing 32 – Vérification DNS

Si l'adresse IP renvoyée correspond bien à celle du VPS, le nom de domaine est correctement associé.

7.4 Intégration dans Nginx

Le fichier de configuration Nginx doit faire référence à ce domaine, comme montré précédemment dans la directive :

```
server_name niscube.fr www.niscube.fr;
```

7.5 Test final

Une fois le DNS propagé et Nginx configuré, accéder à `http://niscube.fr` depuis un navigateur permet de vérifier que le site est bien accessible.

8 Activation du HTTPS avec un certificat SSL Let's Encrypt

Pour sécuriser les échanges entre les utilisateurs et le serveur, le site **niscube.fr** doit être accessible via le protocole `HTTPS`. Cette section décrit l'installation et la configuration de `Certbot`, un outil permettant d'obtenir automatiquement un certificat SSL gratuit via **Let's Encrypt**.

8.1 Installation de Certbot et du plugin Nginx

On installe Certbot ainsi que le plugin Nginx pour gérer automatiquement les certificats :

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

Listing 33 – Installation de Certbot

8.2 Obtention automatique du certificat

Certbot est capable de détecter la configuration Nginx existante et d'y ajouter automatiquement les directives HTTPS :

```
sudo certbot --nginx -d niscube.fr -d www.niscube.fr
```

Listing 34 – Obtention du certificat SSL

Lors de l'exécution, Certbot :

- vérifie que le nom de domaine pointe vers le serveur,
- génère un certificat valide pour `niscube.fr` et `www.niscube.fr`,
- modifie automatiquement la configuration Nginx pour rediriger le trafic HTTP vers HTTPS.

8.3 Vérification du renouvellement automatique

Les certificats Let's Encrypt expirent tous les 90 jours. Pour s'assurer qu'ils soient automatiquement renouvelés, on vérifie la tâche cron installée :

```
sudo certbot renew --dry-run
```

Listing 35 – Simulation du renouvellement

Si aucune erreur n'apparaît, le renouvellement automatique est bien en place.

8.4 Test final

Une fois le certificat installé, on accède au site via `https://niscube.fr`. Le cadenas dans la barre d'adresse du navigateur confirme que la connexion est chiffrée.

9 Conclusion

Ce manuel a présenté de manière détaillée les étapes nécessaires à l'installation, la configuration et la sécurisation complète de la solution **NIS CUBE** sur un serveur distant.

À travers ce guide, nous avons couvert :

- la préparation d'un environnement VPS sécurisé et minimal,
- l'installation et la configuration de PostgreSQL comme base de données centrale,
- le déploiement du backend Node.js avec Express et sa gestion via PM2,
- la configuration et la compilation du frontend développé avec Vue.js et Vite,
- l'intégration de Nginx comme reverse proxy avec prise en charge du HTTPS grâce à Certbot,
- l'association du nom de domaine `niscube.fr` avec l'infrastructure.

L'ensemble de ces étapes permet de mettre en production une application web sécurisée, maintenable et évolutive, en respectant les bonnes pratiques de déploiement modernes. Grâce à cette documentation, tout administrateur peut désormais reproduire l'environnement ou intervenir efficacement pour assurer sa maintenance.

Nous recommandons vivement de compléter ce déploiement par une surveillance régulière des journaux, des sauvegardes automatiques de la base de données, et l'application des mises à jour de sécurité du système et des dépendances.