Guide pour le déploiement du site ZALATOSHOP en ligne

-premier auteur: Hajar Benkhadra-

suivant c'est un guide et documentation pour déployer un site web en ligne et les outils et précautions nécessaire pour la réussite de ce déploiement, voyez lire tout le guide d'abord on cas de doute voyez chercher sur internet, suivre les étapes dans l'ordre, ne pas sauter les étapes ou faire une étape avant de finir la celle qui précède. BONNE CHANCE!!

dans ce cas de déploiement en travail sur un site web (ZALATOSHOP) qui est réaliser en fonction de React pour le FrontEnd, Express pour le BackEnd, Oracle sql developer DB pour la base de donnée, mais en totalité c'est presque la même logique de déploiement à suivre pour les autres cas.

I.Un Hébergeur:

C'est quoi un hébergeur? Un hébergeur c'est une entreprise qui met à disposition un espace de stockage sur un serveur connecté en permanence à Internet, permettant ainsi à un site web d'être accessible aux internautes,En théorie, vous pouvez avoir votre propre serveur chez vous pour y héberger votre site web, mais la plupart de gens optent généralement pour un fournisseur d'hébergement en raison de la complexité de maintenance d'un tel serveur. Dans l'hébergement en compte six principaux types d'hébergement web :

- **Hébergement web mutualisé**: permet à plusieurs propriétaires de sites web de partager un serveur pour héberger un site et ainsi minimiser les coûts opérationnels
- **Hébergement web dédié**: offre un serveur entier à un seul utilisateur pour héberger un site
- **Hébergement web sur VPS** (serveur privé virtuel): divise un serveur en compartiments virtuels qui fonctionnent de manière indépendante
- **Hébergement web en colocation**: fournit à une entreprise son propre serveur dédié pour prendre en charge son site internet et le stocke dans un centre de donnée parmi les serveurs d'autres entreprises
- **Hébergement web cloud**: fournit à votre site web ses propres ressources dédiées sur un réseau de plusieurs serveurs, plutôt que sur un seul
- **Hébergement web multi cloud** consiste à héberger simultanément des sites web sur plusieurs réseaux
- **serveurs domestiques**:soit le fait d'héberger un site web à partir de votre propre serveur, dans vos locaux, peuvent être l'option idéale si vous faites partie des personnes qui souhaitent contrôler totalement la façon dont le site web est gérer

Comment choisir un type d'hébergement et l'hébergeur?

Pour bien choisir un hébergeur pour un site web, il faut comprendre et déterminer les facteurs pour ce choix notamment ; les besoin et la fonction.Dans ce cas on a un site e-commerce, une boutique e-commerce qui vient de débuter, le trafic des clients sera estimé entre 50-500 visiteurs par jour, avec un budget modeste, l'objectif est la stabilité et la simplicité.Vous devrez également tenir compte de vos besoins en matière de fiabilité, pas seulement pendant les périodes de pics de trafic, mais si les serveurs tombent en panne, par exemple. Votre offre d'hébergement comprendra-t-elle des options de secours ? Ou encore, si vous acceptez les paiements en ligne, comment allez-vous vous assurer que votre solution d'hébergement protège les données sensibles de vos clients ?
L'idéal ici c'est soit un hébergement mutualisé ou cloud hosting ou bien un serveur domestique ce qui est le cas.
Voici quelques paramètres à prendre en compte également au moment d'héberger un site :

- Le **temps de disponibilité**: qui désigne la durée pendant laquelle un service d'hébergement web est disponible pour les utilisateurs sur une période donnée, ou la durée durant laquelle les sites web stockés sur un serveur ou un réseau sont opérationnels.
- La **bande passante**, qui représente la quantité de données que votre site web peut transférer aux visiteurs dans le temps. Vous pouvez calculer les besoins en bande passante de votre site web en fonction de votre volume de trafic prévu et de la taille moyenne de vos pages.
- La **sécurité web**, car les sites web peuvent être attaqués via leurs serveurs et subir en particulier des attaques DOS (Deny of Service ou déni de service) et de phishing. Celles-ci constituent un risque pour votre site qui peut devenir vulnérable aux violations et aux vols de données, et ainsi mettre votre entreprise et vos clients en danger.

Proposition des hébergeurs :

Hébergement Mutualisé:

Hébergeur	Prix	Avantages	Inconvénients
Ovh Perso	3,99€/mois	Serveurs en France 1-click PrestaShop	Support lent limité en CPU
Hostinger	2,99€/mois	Interface intuitive SSl gratuit	Stockage SSD limité
LWS E-commerce	5€/mois	Optimisé pour PrestaShop Backup Quotidien	Peu scalable

Hébergement Cloud :

Hébergeur	Prix	Modèle	Points Forts
AWS Lightsail	5€/mois	VPS Cloud	Intégration AWS Scalable facilement
Kinsta	35€/mois	Managed Cloud	Optimisé WooCommerce CDN inclus
Scaleway	10€/mois	VPS	Data centers en Europe Bon rapport perf/prix

Différence: à part la définition pour chacun des deux en haut, les offres mutualisées souvent bloquent les requêtes Oracle DB intensives, le cloud par contre est plus cher mais évite les crashes en période de pic, le conseil qu'on trouve souvent c'est de commencer pas du mutualisé, puis migrer vers le cloud quand votre trafic dépasse 500 visiteurs par jour.

La configuration minimales et le matériels requise pour un serveur domestique

D'abord un serveur domestique est un ordinateur ou un logiciel qui fournit des services à des appareils connectés à un réseau domestique.Il permet de stocker, partager et gérer des fichiers, de diffuser du contenu multimédia, de gérer la domotique, d'héberger des sites web, et bien plus encore.Les avantages d'un serveur domestique incluent une gestion centralisé des données, une meilleur sécurité, une économie d'espace de stockage, et une flexibilité accrue pour les utilisateurs.Pour configurer un serveur domestique il est possible d'utiliser un ordinateur ancien ou un mini PC, qui peut être transformé en serveur grâce à des logiciels comme Docker, Plex, ou Pihole.
Pour créer un serveur domestique, vous aurez besoin de plusieurs éléments essentiels qui dépendent des besoins spécifiques que vous souhaitez couvrir, tels que le stockage de fichiers, la diffusion multimédia, l'hébergement de sites web ou la domotique.

Matériels requis:

- Mini-PC ou ordinateur
- Serveur Physique
  =>CPU: Intel Core i5/i7 (4+ coeurs) ou Xeon, AMD Ryzen 5/7
  => RAM: Minimum 8 Go, recommandé 16 Go+
  =>Stockage/Disque: SSD NVMe avec minimum 500 Go, HDD avec 1 To en RAID1 pour le sauvegarde
  =>Alimentation/Cooling: Alimentation 80+ Bronze/Gold pour la stabilité et Ventilation silencieuse comme Noctua, Be Quiet!
- Connectivité réseau: prise ethernet gigabit,wifi 5 doub bande….
  =>Box Internet: débit symétrique recommandé au moi 100 Mbps en upload si site public

=>Routeur: un bon routeur comme TP-Link Omada si on gère plusieurs connexions

=>Adresse IP: IP PUBLIQUE FIXE demander à l'ISP ou DNS DYNAMIQUE

=>Port Forwarding:Ouvrir les ports 80 et 443 vers le serveur

- Stockage:le stockage essentiel.Il est recommandé d'utiliser un Mini-PC avec au moins deux disques pour construire votre serveur domestique.Si le pc ne supporte qu'un seul disque interne, un disque externe peut être utilisé pour sauvegarder vos fichiers importants.
- Système d'exploitation: vous pouvez choisir entre différents systèmes d'exploitation, tels que Windows, Linux ou des systèmes NAS comme FreeNas, ces systèmes offrent des fonctionnalités variées pour gérer le serveur.

  =>Ubuntu: le plus simple, bonne documentation

  =>Debian:stable et léger

  =>Proxmox:Si on est dans la virtualisation (VM ou Conteneurs)

  =>Windows Server: si besoin d'ASP.NET /Active directory

- Logiciels de gestion : des logiciels comme BackupPC ou "UrBackup" peuvent être utilisés pour automatiser les sauvegardes de vos fichiers.Des logiciels comme Samba peuvent être installés pour permettre le partage de fichiers sur le réseau.

  =>Web Server:Nginx ou Apache : ils sont performant et plus simple

  =>Base de donnée:MariaDB/MySQL ou PostgreSQL

  =>Langages: selon le site

  =>Sécurité:Fail2Ban pour les attaques par force brute, Certbot pour des certificats HTTPS gratuits

- Sécurité:il est important de mettre en place des mesures de sécurité, telles que des mots de passe robustes et des mises à jour logicielles régulières afin de protéger votre serveur domestique contre les menaces potentielles.
- Expansion et évolutivité: choisir un matériel qui offre des possibilités d'extension, comme des emplacements PCIe pour ajouter des cartes réseau ou des contrôleurs de stockage supplémentaires.

II. Installation d'un OS serveur:

- regarde les documentation d'installation.

I.III Configurer Nginx + Base Donnée :

NGINX est un serveur Web, un serveur proxy, un proxy email et un répartiteur de charge; il est basé sur une architecture événementielle, ce qui permet de gérer efficacement un grand nombre de connexions simultanées tout en économisant les ressources.Pour configurer NGINX vous pouvez commencer par installer le logiciel.

INSTALLATION NGINX:

- L'installation est simple et se fait via le gestionnaire de paquet

```
sudo apt-get install nginx
```

- Après sur votre serveur vous devriez déjà avoir une page qui va s'afficher, en générale cette page se retrouve dans le dossier /usr/share/nginx/html

CONFIGURATION NGINX:

- la configuration de nginx se trouve dans le dossier /etc/nginx
  =>nginx.conf contient la configuration générale
  =>conf.d/.conf contient la configuration des différents "servers"
- Les virtuals hosts: ces hôtes virtuels permettent à Nginx de gérer plusieurs arborescences Web en simultanées.Dans le dossier conf.d vous remarquez que vous n'avez pour le moment qu'une configuration qui est la configuration par défaut.
- Maintenant on va faire pointer notre site Zalatoshop vers notre serveur, nous allons devoir créer un nouveau virtual host :

```
server {

    listen 80;
    server_name Zalatoshop.com;

    # Les urls commençant par / (toutes les urls)
    location / {
        root /home/dev/www;
        index index.html index.htm;
        try_files $uri $uri/ $uri.html =404;
    }

    # Les urls contennant /. (dotfiles)
    location ~ /\. {
        deny all;
        access_log off;
        log_not_found off;
    }

    # On va placer les logs dans un dossier accessible
    error_log /home/dev/logs/error.log;
    access_log /home/dev/logs/acess.log;

    # Les pages d'erreurs
    error_page 404 500 501 /error.html;

}

server {
    # On redirige les www. vers la version sans www
    listen 80;
    server_name www.monsupersite.fr;
    return 301 http://monsupersite.fr$request_uri;
}
```

- une fois ces modifications sont effectuées il ne faut pas oublier de recharger la configuration en redémarrant nginx

```
# Pour tester la configuration
sudo nginx -t
# Pour recharger la configuration
sudo service nginx restart
```

Ici la configuration et l'installation de NGINX est prête et finie.

Maintenant on passe à la configuration et installation de la base donnée et la connecter à NGINX

- Installer MariaDB/MySQL:

```
# Installer MariaDB (fork open-source de MySQL)
sudo apt update
sudo apt install mariadb-server

# Sécuriser l'installation
sudo mysql_secure_installation
```

au cours de l'installation voyez répondez aux questions comme suivant:
=>Définir un mot de passe root pour MySQL
=>Supprimer les utilisateurs anonymes? OUI
=>Désactiver la connexion root à distance? OUI
=>Supprimer La base de test? OUI

- Maintenant créer la base donnée et un utilisateur:

```
# Se connecter à MySQL
sudo mysql -u root -p

# Créer une base de données
CREATE DATABASE nom_de_ma_base;

# Créer un utilisateur et lui donner les droits
CREATE USER 'utilisateur'@'localhost' IDENTIFIED BY 'mot_de_passe_securise';
GRANT ALL PRIVILEGES ON nom_de_ma_base.* TO 'utilisateur'@'localhost';
FLUSH PRIVILEGES;

# Quitter
EXIT;
```

- Pour configurer NGINX pour se connecter à la Base de donnée dans notre qui est fait par Node.js/Express est oracle sql developer et react pour le frontend on aura besoin que oracle database soit installé localement et oracle instant client et Node.js avec le module oracledb et Nginx comme reverse proxy.
=>installer oracle instant client sur le serveur :

```
# Téléchargez le package depuis Oracle (ex: version 21)
wget https://download.oracle.com/otn_software/linux/instantclient/instantclient-basic-linuxx64.zip
unzip instantclient-basic-linuxx64.zip -d /opt/oracle
sudo mv /opt/oracle/instantclient_21_* /opt/oracle/instantclient

# Configurer les variables d'environnement
echo 'export LD_LIBRARY_PATH=/opt/oracle/instantclient:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc
```

- Configuration NGINX comme Reverse Proxy :
=>Modifier /etc/nginx/sites-available/le-nom-du-site pour servir le frontend React et rediger les requetes /api vers le backend Node.js

```
server {
    listen 80;
    server_name mondomaine.com www.mondomaine.com;

    # Frontend React (build)
    location / {
        root /var/www/mon-site/frontend/build;
        try_files $uri /index.html;
    }

    # Backend Node.js (API)
    location /api/ {
        proxy_pass http://localhost:3000;  # Port de votre serveur Express
        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;
    }
}
```

et redémarrez Nginx par

```
sudo systemctl restart nginx
```

- Sécurisation de la Connexion Oracle
  => ne stockez pas les identifiants en clair utiliser (dotenv)
  =>Restreignez les droits SQL; créer un utilisateur Oracle dédié avec uniquement les permissions nécessaires.

```
CREATE USER app_user IDENTIFIED BY "mot_de_passe_complexe";
GRANT CONNECT, RESOURCE TO app_user;
GRANT SELECT, INSERT ON ma_table TO app_user;
```

- Optimisations pour production
  =>pool de connexions Oracle :

```
// Dans db.js
const pool = await oracledb.createPool({
  user: process.env.ORACLE_USER,
  password: process.env.ORACLE_PASSWORD,
  connectString: process.env.ORACLE_CONN_STRING,
  poolMin: 4,
  poolMax: 10,
});
```

=> HTTPS: utiliser Certbot pour obtenir un certificat SSL gratuit

```
sudo certbot --ngix -d mondomaine.com
```

III.Nom de Domaine:

- Pour choisir un nom de domaine il est préférable que ça soit court et facile à retenir, évitez les caractères spéciaux ou chiffres, vérifiez la disponibilité sur WHOIS ou via des registres.Puis il faut acheter un nom de domaine l'option la plus souvent c'est les registrars comme Namecheap(10-15 eur/an), OVH(5-20 eur/an), Google domains(12-20 eur/an).
- Configurer le DNS pour le serveur si votre si vou avez une Ip publique fixe ce qui est le cas la plupart de temps

=>connectez-vous au panneau de gestion de votre registrar et modifiez les DNS comme suivant :

Type A: pointez vers l'IP publique du serveur
@ → 123.123.123.123 (votre IP)
www → 123.123.123.123

Type CNAME: si sous-domaine
blog → monsite.com

- Puis on associe le domaine à Nginx :
  il faut modifiez la configuration Nginx (`/etc/nginx/sites-available/monsite`)

```
server {
    listen 80;
    server_name monsite.com www.monsite.com;  # ← Votre domaine ici
    root /var/www/html;
    index index.html;
}
```

  et puis redémarrer Nginx
- Activer HTTPS :
  =>Installez Certbot:

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

  =>Générer le certificat :

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

  =>Renouvellement automatique :

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

  - La dernière étape c'est de vérifier la propagation DNS

ici il faut attendre 1-24h pour que le DNS se propage puis vérifiez avec :

```
ping monsite.com
nslookup monsite.com
```

VI.Transfert des fichiers

- Pour le frontend en React
  =>d'abord il faut générer un dossier /build

```
cd frontend
npm run build
```

- Pour le backend en node.js/express

=>il faut vérifier que les dépendances sont à jour et excluez les dossiers inutiles avec .gitignore

- Transfert vers le server : il y a deux méthodes pour le transfert soit :
  =>transfert par SCP (SSH)
  # Transférer le frontend (React)
  scp -r ./frontend/build user@votre-server-ip:/var/www/mon-site/frontend

  # Transférer le backend (Node.js)
  scp -r ./backend user@votre-server-ip:/var/www/mon-site/backend
  =>transfert par GIT :
  sudo apt install git
  cd /var/www/mon-site
  git clone https://github.com/votre-repo.git .
  et installez les dépendances suivant :
  cd /var/www/mon-site/backend
  npm install
- Configuration des permissions et des dossiers
  => si le dossier racine n'existe pas il faut le créer :
  sudo mkdir -p /var/www/mon-site/{frontend,backend}
  sudo chown -R $USER:$USER /var/www/mon-site
  =>Permissions : sudo chmod -R 755 /var/www/mon-site
- Configurer Nginx pour le FrontEnd et Proxy pour le Backend
  =>il faut éditer le fichier Nginx `/etc/nginx/sites-available/mon-site` comme suivant :

```
server {
    listen 80;
    server_name mondomaine.com www.mondomaine.com;

    # Frontend React
    location / {
        root /var/www/mon-site/frontend/build;
        try_files $uri /index.html;
    }

    # Backend Node.js (API)
    location /api/ {
        proxy_pass http://localhost:3000;  # Port de votre serveur Express
        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;
```

  =>Puis tester et recharger Nginx :
  sudo nginx -t
  sudo systemctl restart nginx

- Démarrer le BackEnd
  =>Installer PM2 : sudo npm install -g pm2
  =>démarrer le serveur :
  cd /var/www/mon-site/backend
  pm2 start server.js --name "mon-api"
  pm2 save
  pm2 startup
  =>Vous pouvez vérifier si tout fonctionne depuis les logs :
  sudo tail -f /var/log/nginx/error.log
  pm2 logs mon-api

V.Migration de la base donnée du local vers serveur

- il faut exporter les données de oracle database depuis l'environnement local avec expdp :

```
# Sur votre machine locale (où la DB Oracle est installée)
expdp USER/password@LOCAL_DB_SCHEMA \
  DIRECTORY=DATA_PUMP_DIR \
  DUMPFILE=export_local.dmp \
  LOGFILE=export_local.log \
  SCHEMAS=VOTRE_SCHEMA
```

- Puis transférer le fichier .dmp vers le serveur à travers SCP :
  scp export_local.dmp user@votre-server-ip:/chemin/vers/dossier_destination
- Ensuite en importe les données sur le serveur,d'abord il faut assurer qu'il y a oracle instant client et oracle DB installé (voir les étapes d'avant), puis créez un schéma cible comme suivant :
  CREATE USER VOTRE_SCHEMA IDENTIFIED BY "mot_de_passe";
  GRANT CONNECT, RESOURCE TO VOTRE_SCHEMA;

- Import avec impdp :

```
# Sur le serveur
impdp USER/password@SERVER_DB_SCHEMA \
  DIRECTORY=DATA_PUMP_DIR \
  DUMPFILE=export_local.dmp \
  LOGFILE=import_server.log \
  SCHEMAS=VOTRE_SCHEMA \
  REMAP_SCHEMA=VOTRE_SCHEMA_LOCAL:VOTRE_SCHEMA_SERVEUR
```

- Configuration de Node.js pour la DB sur le serveur:voyez mettre à jour le backend avec les nouvelles informations de connexion ;

```
const dbConfig = {
  user: 'VOTRE_USER_SERVEUR',
  password: 'VOTRE_MDP_SERVEUR',
  connectString: 'localhost:1521/SERVER_DB_SCHEMA', // Service name/SID du serveur
};
```

VI.Configuration du site

- Configuration du FrontEnd : optimisation du Build
  cd frontend
  npm run build
  => vérifier si les variables d'environnement sont pointent vers l'URL de production et utiliser des variables d'environnement par .env.production
  REACT_APP_API_URL=https://mondomaine.com/api
  =>Pour la sécurité voyez désactiver les sources maps :
  "build": "GENERATE_SOURCEMAP=false react-scripts build"
- configuration du BackEnd
  => si n'est pas déjà fait, créez un fichier .env :
  # Oracle DB
  ORACLE_USER=user_prod
  ORACLE_PASSWORD=mdp_complexe
  ORACLE_CONN_STRING=localhost:1521/ORCLPDB1

  # Sécurité
  JWT_SECRET=clé_secrète_complexe
  NODE_ENV=production
  et protégez le fichier avec : chmod 600 .env
  =>Sécurité Express :
  // Dans server.js
  const helmet = require('helmet');
  const rateLimit = require('express-rate-limit');

  app.use(helmet()); // Protège contre les vulnérabilités HTTP
  app.use(rateLimit({ windowMs: 15 * 60 * 1000, max: 100 })); // Anti-DDoS
- Configuration d'Oracle
  =>Pool de connexions pour éviter les surcharges :

```
const pool = await oracledb.createPool({
  user: process.env.ORACLE_USER,
  password: process.env.ORACLE_PASSWORD,
  connectString: process.env.ORACLE_CONN_STRING,
  poolMin: 5,
  poolMax: 20,
});
```

  =>Sécurité :
  REVOKE ALL ON VOTRE_SCHEMA.* FROM 'user_prod';
  GRANT SELECT, INSERT, UPDATE ON VOTRE_TABLE TO 'user_prod';
- Configuration de Nginx :
  =>fichier de configuration optimisé
  /etc/nginx/sites-available/mondomaine.com

```
server {
    listen 443 ssl;
    server_name mondomaine.com www.mondomaine.com;

    # SSL (Let's Encrypt)
```

```
    ssl_certificate
/etc/letsencrypt/live/mondomaine.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/mondomaine.com/privkey.pem;

    # Frontend React
    root /var/www/mon-site/frontend/build;
    index index.html;
    try_files $uri /index.html;

    # Backend Node.js
    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;
    }

    # Sécurité
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-Content-Type-Options "nosniff";
}
```

=>Rediriger HTTP→HTTPS

```
server {
  listen 80;
  server_name mondomaine.com www.mondomaine.com;
  return 301 https://$host$request_uri;
}
```

=>Activer Gzip pour compression :

```
gzip on;
gzip_types text/css application/javascript;
```

- Déploiement avec PM2

  =>Lancer Node.js en arrière-plan ;

  ```
  cd /var/www/mon-site/backend
  npm install --production  # Installe uniquement les dépendances nécessaires
  pm2 start server.js --name "api-prod" --time
  pm2 save
  pm2 startup  # Active le démarrage automatique au reboot
  ```

- Monitoring et Maintenance :

  =>Surveillance : pour PM2

  ```
  pm2 monit
  pm2 logs
  ```

Pour Nginx sudo tail -f /var/log/nginx/error.log

- Sauvegardes Automatisées :
  =>Base de données :
  0 3 * * * /usr/bin/expdp user/password@db SCHEMAS=votre_schema DUMPFILE=/backups/export_$(date +\%Y\%m\%d).dmp
  =>Fichiers :
  0 4 * * * tar -czvf /backups/frontend_$(date +\%Y\%m\%d).tar.gz /var/www/mon-site/frontend/build

VII.Certificat SSL

pour cette étape assurer qu'il y a bien un nom de domaine configuré et pointant vers votre serveur et que NGINX est bien installé et configuré, les ports ouverts dans le pare feu

- Installer cerbot (de base c'est déjà installé sinon fait le d'abord)
- Génération du Certificat :
  sudo certbot --nginx -d mondomaine.com -d www.mondomaine.com
- Le certificat généré et stocké dans `/etc/letsencrypt/live/mondomaine.com/`
- Certbot peut modifie votre fichier Nginx
  `/etc/nginx/sites-available/mondomaine.com`
- Tester et rechargez Nginx
  sudo nginx -t # Vérifie la configuration
  sudo systemctl restart nginx
- Les certificats de Let's Encrypt expirent après 90 jours du coup il faut automatiser le renouvellement : sudo certbot renew --dry-run ça va créer un cron job est automatiquement créé pour renouveler les certificats .

Ici on arrive à la fin du guide à ce point là vous aurez bien déployé votre site en ligne avec tout ce qu'il faut.