Documentation de Déploiement

VPS, Serveur Web Go, Nginx et Certbot

2025

Table des matières

Contents

1 Introduction

Dans cette documentation, nous allons détailler la procédure pour déployer une application web développée en Go sur un Serveur Privé Virtuel (**VPS**) en utilisant **Nginx** comme reverse proxy et **Certbot** pour la gestion des certificats **SSL/TLS** (HTTPS).

1.1 Prérequis

- Une application **Go-web** prête à être compilée (exemple de code).
- Un **nom de domaine** enregistré.
- Un **VPS**, OS : **Debian**.

2 Configuration du VPS

Pour cette documentation, les placeholders suivants seront utilisés :

- Adresse IP du VPS : **xx.xx.xx.xx**
- Nom de domaine : **mondomaine.com** (ou **hello-world.mondomaine.com** pour le sous-domaine)
- Nom de l'application : **hello-world**
- Nom de l'utilisateur par défaut : **user**

Rappelez-vous de remplacer ces valeurs par les vôtres.

2.1 Connexion et Mise à jour Initiale

Après l'achat du VPS, vous devriez disposer de l'IP publique, du nom d'utilisateur initial (souvent **root** ou un utilisateur standard) et du mot de passe.

> **†Note :** *Si vous utilisez l'utilisateur **root**, il est fortement recommandé de créer un utilisateur non-root après la première connexion pour des raisons de sécurité.*

Connectez-vous via SSH :

```
ssh user@xx.xx.xx.xx
```

Listing 1: Connexion initiale au VPS

Mettez à jour les paquets du système :

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

Listing 2: Mise à jour du système

2.2 Configuration des Clés SSH pour la Sécurité

L'utilisation de clés SSH est la méthode de connexion la plus sécurisée.

2.2.1 Génération de la Clé (Local)

Sur votre machine locale (macOS/Linux terminal ou PowerShell/Git Bash sur Windows), générez une paire de clés :

```
ssh-keygen -t ed25519 -C "email@exemple.com"
```

Listing 3: Génération de la paire de clés Ed25519

Cela crée la clé **privée** (`~/.ssh/id_ed25519`) et la clé **publique (`~/.ssh/id_ed25519.pub`). **Ne partagez jamais la clé privée !****

2.2.2 Copie de la Clé Publique (Recommandée)

Utilisez l'outil standard pour copier la clé sur le VPS :

```
ssh-copy-id user@xx.xx.xx.xx
```

Listing 4: Copie de la clé publique

2.2.3 Copie Manuelle de la Clé Publique (Alternative)

Si `ssh-copy-id` n'est pas disponible, suivez ces étapes après vous être connecté au VPS :

- Affichez le contenu de votre clé publique en local : `cat ~/.ssh/id_ed25519.pub`.
- Créez le dossier **.ssh** s'il n'existe pas et donnez-lui les bonnes permissions :

```
mkdir -p ~/.ssh
chmod 700 ~/.ssh
```

- Copiez le contenu de votre clé publique dans le fichier **authorized_keys** :

```
echo "VOTRE\_CLE\_PUBLIQUE" >> ~/.ssh/authorized_keys
```

- Appliquez les permissions restrictives nécessaires :

```
chmod 600 ~/.ssh/authorized_keys
```

†Note : *Après avoir testé la connexion par clé SSH, pensez à désactiver la connexion par mot de passe dans la configuration SSH du VPS pour renforcer la sécurité.*

3 Déploiement de l'Application Go

3.1 Compilation (Build) du Projet Go (Local)

Dans le dossier source de votre projet Go (en local), la compilation se fait ainsi.

> **†Note :** *Si vous ciblez un autre OS/Architecture, utilisez les variables d'environnement* `GOOS` *et* `GOARCH` *(ex:* `GOOS=linux GOARCH=amd64 go build ...`*).*

L'architecture du projet (exemple) :

```
|-- assets/
| |-- css/
| |-- js/
|-- config/
|-- controllers/
|-- views/
|-- go.mod
\-- main.go
```

Listing 5: Structure du projet Go (avant build)

La compilation crée l'exécutable :

```
go build -o hello-world
```

Listing 6: Compilation de l'exécutable

Après la compilation, supprimez les fichers go et le dossier de déploiement (sans les fichiers source Go) ressemble à ceci (le fichier **hello-world** est un fichier éxécutable) :

```
|-- assets/
| |-- css/
| |-- js/
|-- views/
\hello-world
```

Listing 7: Structure du dossier de déploiement (local)

3.2 Transfert des Fichiers vers le VPS

3.2.1 Préparation sur le VPS

Connectez-vous au VPS et créez la structure de dossiers pour votre application dans le répertoire de l'utilisateur (`/home/user/`).

```
mkdir -p app/hello-world
cd app/hello-world
```

Listing 8: Création des répertoires sur le VPS

3.2.2 Transfert via SCP (Local)

Depuis votre machine locale, utilisez **scp** (Secure Copy Protocol) pour transférer l'intégralité du contenu du dossier de déploiement :

```
scp -r ./hello-world/* user@xx.xx.xx.xx:/home/user/app/hello-world/
```

Listing 9: Transfert récursif des fichiers

3.3 Création du Service Systemd

Pour que l'application soit lancée automatiquement au démarrage et gérée comme un service, nous utilisons **Systemd**.

Créez le fichier de service sur le VPS :

```
sudo nano /etc/systemd/system/hello-world.service
```

Listing 10: Création du fichier de service

Remplissez-le avec le contenu suivant :

```
[Unit]
Description=Service Hello World Go Web
After=network.target

[Service]
ExecStart=/home/user/app/hello-world/hello-world
WorkingDirectory=/home/\ref{user-ref}/app/hello-world/
Restart=always
User=user
Environment=PORT=8000

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

Listing 11: Contenu du fichier **hello-world.service**

Détails du service :

- **ExecStart**: Path jusqu'au fichier éxécutable
- **WorkingDirectory**: Path du sosier de l'application
- **User**: l'utilisateur qui va tourner le service (pas root)
- **Environment**: Le port de l'application web

3.3.1 Activation et Démarrage du Service

1. Rechargez la configuration de Systemd :

```
sudo systemctl daemon-reload
```

2. Activez le service pour qu'il démarre au boot :

```
sudo systemctl enable hello-world.service
```

3. Démarrez le service immédiatement :

```
sudo systemctl start hello-world.service
```

3.3.2 Vérification du Service

Vérifiez l'état du service et les logs :

```
sudo systemctl status hello-world.service
sudo journalctl -u hello-world.service -f
```

Listing 12: Vérification de l'état du service

Testez la communication locale sur le port **8000** (le port défini dans le fichier `.service`) :

```
curl http://localhost:8000
```

Listing 13: Test de l'application via cURL (localement)

Si vous recevez une réponse HTTP, l'application Go fonctionne correctement en arrière-plan.

4 Configuration Nginx et SSL

4.1 Installation de Nginx

Si Nginx n'est pas déjà installé :

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

Listing 14: Installation de Nginx

4.2 Configuration Reverse Proxy Nginx

Nous allons configurer Nginx pour écouter sur le port **80** (HTTP) et transférer les requêtes vers notre application Go qui tourne sur le port interne **8000**.

Créez le fichier de configuration :

```
sudo nano /etc/nginx/sites-available/hello-world.mondomaine.com.conf
```

Listing 15: Création du fichier de configuration Nginx

Ajoutez la configuration suivante. Elle est essentielle pour le bon fonctionnement d'un **reverse proxy** :

```
server {
  listen 80;
  server_name hello-world.mondomaine.com;

  location / {
    proxy_pass http://localhost:8000;
    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;
  }
}
```

Listing 16: Configuration Nginx initiale (**HTTP**)

Détails de la configuration **Nginx**

- **server** : Débute la définition d'un nouveau bloc de serveur virtuel (*virtual host*). Tout ce qui est défini dans ce bloc s'applique uniquement au domaine spécifié.
- **listen 80;** : Définit le **port d'écoute**. Le port `80` est le port standard et par défaut pour le trafic HTTP non chiffré.
- **server_name hello-world.mondomaine.com;** : Le nom de domaine ou sous-domaine que ce bloc de configuration est censé servir. Nginx utilise cet identifiant pour sélectionner le bon bloc de configuration en fonction de l'en-tête `Host` de la requête client.
- **location /** : Démarre un bloc qui définit comment traiter les requêtes dont l'**URI** (chemin d'accès) correspond au motif. Le simple slash (`/`) signifie que ce bloc gère toutes les requêtes pour ce serveur, car tous les chemins commencent par `/`.
  - **proxy_pass http://localhost:8000;** : C'est le cœur du **Reverse Proxy**. Cette directive transfère la requête reçue par Nginx à l'application Go qui tourne en interne sur l'interface locale (`localhost`) et le port `8000`.
  - **proxy_http_version 1.1;** : Définit la version du protocole HTTP à utiliser pour la communication entre Nginx et le serveur backend (Go). La version 1.1 est courante et recommandée.

- **proxy_set_header Upgrade $http_upgrade;** : Transmet l'en-tête `Upgrade` du client au serveur *backend*. Nécessaire pour supporter des protocoles comme les **WebSockets**.
- **proxy_set_header Connection 'upgrade';** : Travaille de pair avec la ligne précédente. Indique au backend de maintenir la connexion ouverte si un protocole de niveau supérieur (exemple : WebSocket) est demandé.
- **proxy_set_header Host $host;** : Transmet l'**en-tête Host** original de la requête client au serveur Go.
- **proxy_set_header X-Real-IP $remote_addr;** : Transmet l'**adresse IP réelle** du client à l'application Go. Sans cela, le serveur Go ne verrait que l'IP locale de Nginx (`127.0.0.1`).
- **proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;** : Ajoute l'IP du client à la liste des adresses par lesquelles la requête est passée. Utile pour le traçage et l'analyse des logs.
- **proxy_set_header X-Forwarded-Proto $scheme;** : Indique au serveur Go quel protocole (`http` ou `https`) le client a utilisé pour se connecter à Nginx, ce qui est vital pour les redirections HTTPS.
- **proxy_cache_bypass $http_upgrade;** : Contourne les mécanismes de cache de Nginx pour les requêtes qui tentent de mettre à jour le protocole.

4.2.1 Activation de la Configuration

Créez le lien symbolique (symlink) pour activer la configuration :

```
sudo ln -s /etc/nginx/sites-available/hello-world.mondomaine.com.conf /etc/nginx/
    sites-enabled/
```

Listing 17: Activation de la configuration

†Note : *N'hésitez pas à supprimez la configuration 'default' si elle est en conflit (sudo rm /etc/nginx/sites-enabled/default)*

Testez la syntaxe Nginx :

```
sudo nginx -t
```

Listing 18: Test de la configuration Nginx

Et si il n'y a pas d'erreurs, vous pouvez redémarrer le service :

```
sudo systemctl restart nginx.service
```

Listing 19: Redémarrage de Nginx

4.3 Configuration SSL avec Certbot (Let's Encrypt)

4.3.1 Installation de Certbot et du Plugin Nginx

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

Listing 20: Installation de Certbot

4.3.2 Obtention du Certificat

L'outil **certbot** va automatiquement lire votre configuration Nginx, obtenir le certificat et modifier le fichier de configuration pour ajouter la redirection **HTTP vers HTTPS**.

```
sudo certbot --nginx -d hello-world.mondomaine.com
```

Listing 21: Exécution de Certbot

Suivez les instructions : entrez votre email et acceptez les conditions. Choisissez l'option de redirection **2** (**Redirect**) pour forcer le HTTPS.

> **†Note :** *Certbot installe également un 'timer/cronjob' pour renouveler automatiquement le certificat avant son expiration.*

5 Gestion du Sous-domaine (DNS)

Cette étape doit être effectuée auprès de votre **registrar** (OVH, Hostinger, Infomaniak, etc.).

Vous devez créer un enregistrement **A** (pour IPv4) ou **AAAA** (pour IPv6) pour le sous-domaine que vous avez utilisé dans votre configuration Nginx.

- **Type** : **A** (si votre VPS a une IPv4).
- **Name** (ou **Host**) : **hello-world** (le nom de votre sous-domaine).
- **Points to** (ou **Value**) : **xx.xx.xx.xx** (l'adresse IP publique de votre VPS).
- **TTL** (Time To Live) : La valeur par défaut est souvent acceptable (ex: 3600 secondes). Il s'agit du temps que les résolveurs DNS gardent l'information en cache.

  †Note : *La propagation DNS (le temps que prend la mise à jour pour être effective partout dans le monde) peut prendre de quelques minutes à 48 heures, bien que ce soit généralement très rapide aujourd'hui.*

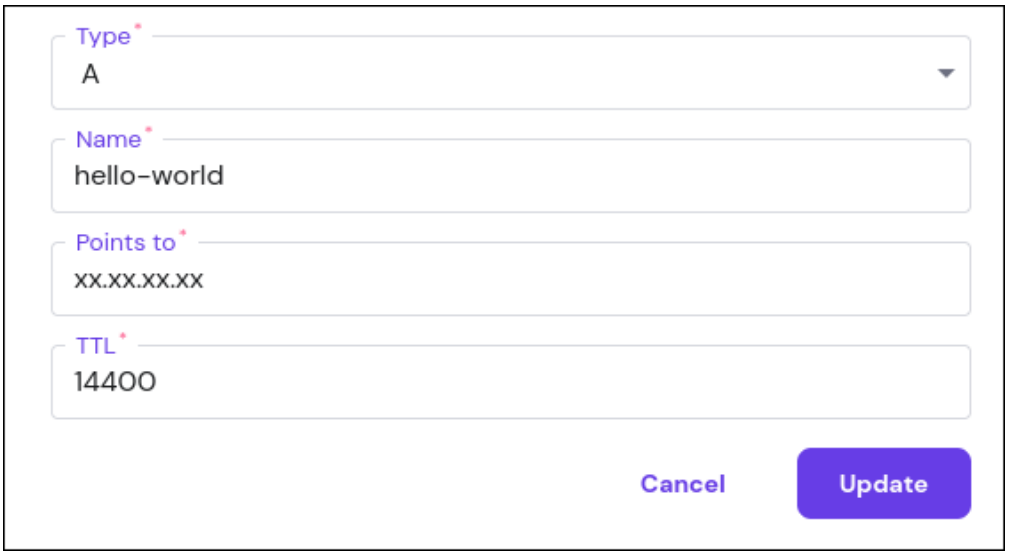

Figure 1: Exemple de configuration d'un enregistrement DNS de type A pour le sous-domaine **hello-world**.

Une fois la propagation effectuée, votre application Go sera accessible via

`https://hello-world.mondomaine.com`