

# Serveur web go et certificat auto-signé

Un guide pratique pour les **débutants** souhaitant créer un **serveur web en Go** et **générer des certificats auto-signés**. Parfait pour apprendre les bases du développement web sécurisé, étape par étape.

- [Sommaire](#)
- [Mise à jour du système et installation des outils nécessaires](#)
- [Création d'un certificat SSL auto-signé](#)
- [Développement d'un serveur HTTPS en Go et mise en place du certificat.](#)
- [Permissions pour écouter sur le port 443](#)

# Sommaire

Nous allons procéder étape par étape pour configurer un serveur web sécurisé avec Go sur une nouvelle installation Debian. Voici le plan que nous allons suivre :

- Mise à jour et installation des paquets de base : Mettre à jour Debian et installer les outils nécessaires.
- Installation de Go : Installer le langage Go pour développer et exécuter votre programme.
- Création d'un certificat SSL auto-signé : Générer un certificat SSL auto-signé pour sécuriser le site.
- Développement d'un serveur HTTPS en Go : Créer un programme Go simple qui utilise ce certificat.
- Configurer les permissions et Firewall : S'assurer que tout est correctement configuré pour servir en HTTPS.

# Mise à jour du système et installation des outils nécessaires

Commençons par mettre à jour le système et installer les outils nécessaires (comme `git`, `curl`, et `openssl`).

Ouvrez un terminal et exécutez ces commandes :

```
# Mise à jour des paquets
sudo apt update && sudo apt upgrade -y

# Installer les outils de base
sudo apt install -y git curl vim wget openssl
```

Exécutez les commandes suivantes pour télécharger et installer Go :

```
# Télécharger la version Go 1.20.4 (ou la dernière version)
wget https://golang.org/dl/go1.20.4.linux-amd64.tar.gz

# Extraire et installer Go
sudo tar -C /usr/local -xzf go1.20.4.linux-amd64.tar.gz

# Configurer les variables d'environnement pour Go
echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.profile
source ~/.profile

# Vérifiez l'installation de Go
go version
```

# Création d'un certificat SSL auto-signé

Nous allons créer un certificat SSL auto-signé avec OpenSSL. Cela permet de sécuriser votre serveur en HTTPS.

Exécutez les commandes suivantes pour générer un certificat et une clé privée :

```
# Créer le répertoire du projet
mkdir ~/go-https-server && cd ~/go-https-server

# Générer la clé privée et le certificat auto-signé (valide pour 365 jours)
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
```

Lors de l'exécution de cette commande, OpenSSL vous demandera quelques informations.

`server.key` : la clé privée

`server.crt` : le certificat

# Développement d'un serveur HTTPS en Go et mise en place du certificat.

Création de l'environnement et mise en place du projet.

```
# Initialiser un nouveau module Go
go mod init go-https-server

# Créer le fichier principal main.go
vim main.go
```

N'oubliez pas que vous devez appliquer ces commandes dans **~/go-https-serve**

```
package main

import (
    "crypto/tls"
    "log"
    "net/http"
)

func main() {
    // Gestionnaire simple pour la route principale
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Bienvenue sur mon serveur sécurisé en HTTPS !"))
    })

    // 📄 Charger le certificat et la clé
    cert, err := tls.LoadX509KeyPair("server.crt", "server.key")
    if err != nil {
        log.Fatal(err)
    }
}
```

```

}

// Configurer TLS
tlsConfig := &tls.Config{
    Certificates: []tls.Certificate{cert},
}

// Configuration du serveur HTTP/HTTPS
server := &http.Server{
    Addr:    ":443",    // Écoute sur le port HTTPS 443
    TLSConfig: tlsConfig,    // Config TLS
    Handler: nil,        // Utilise le gestionnaire par défaut
}

log.Println("Serveur HTTPS démarré sur https://localhost:443")

// Démarrer le serveur HTTPS
err = server.ListenAndServeTLS("", "") // Les certificats sont déjà dans tlsConfig
if err != nil {
    log.Fatal(err)
}
}

```

Avant d'exécuter le serveur, assurez-vous que les fichiers de certificat sont dans le même répertoire que votre fichier Go ( `server.crt` et `server.key` ).

Compilez et exécutez votre programme Go :

```

# Exécuter le serveur Go
go run main.go

```

# Permissions pour écouter sur le port 443

**Vous verrez probablement une alerte de sécurité, car le certificat est auto-signé.**

Sous Linux, pour écouter sur le port 443 (qui est un port privilégié), il faut avoir les permissions root. Vous pouvez soit exécuter votre serveur Go en tant que super-utilisateur, soit utiliser un outil comme `setcap` pour permettre à l'exécutable d'accéder aux ports privilégiés sans root.

Pour accorder ces permissions, exécutez :

```
# Compiler votre programme
go build -o https-server main.go

# Accorder les permissions nécessaires à l'exécutable
sudo setcap 'cap_net_bind_service=+ep' ./https-server

# Exécuter le serveur sans sudo
./https-server
```

Configurer le pare-feu :

Si vous utilisez `ufw` comme pare-feu, assurez-vous d'autoriser le trafic HTTPS (port 443) :

```
# Autoriser le trafic HTTPS
sudo ufw allow 443

# Vérifier le statut du pare-feu
sudo ufw status
```

Félicitation, Ouvrez un navigateur et allez à l'adresse suivante : **`https://[IP_DU_SERVEUR]`**