**INSTALLATION ET CONFIG ASTERISK SUR DEBIAN 12**

1. **ETAPE**

**On installe debian 12 en graphique pour plus de confort**

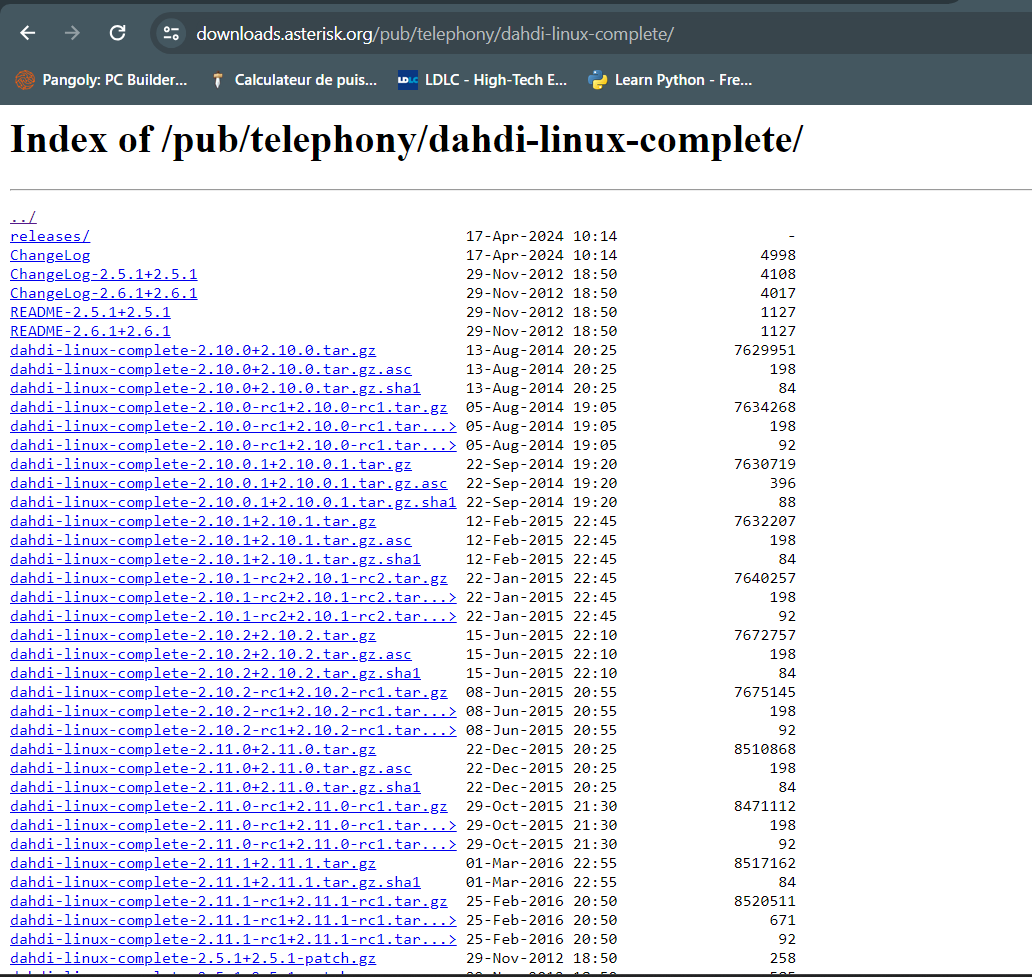
**On va sur le terminal :**

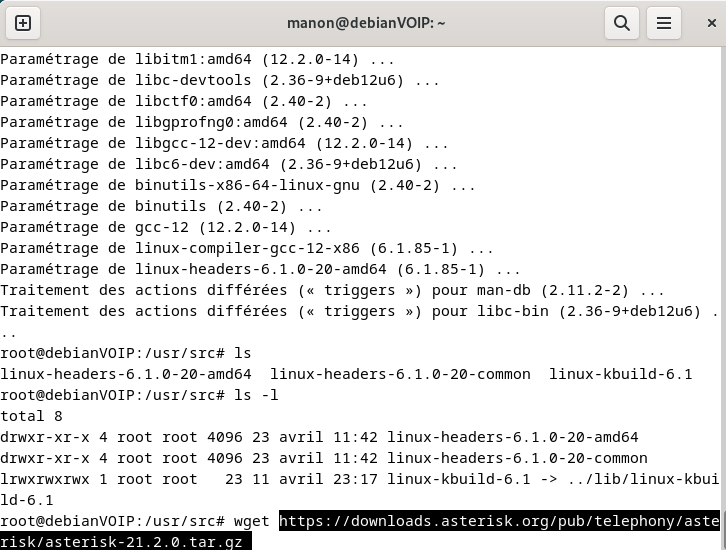
**On fait un apt update && apt upgrade**

**Puis on télécharge les paquets asterisk en récupérant le lien sur le site asterisk**

**Pour avoir les options ifconfig etc il suffit de télécharger le paquet suivant**

**apt install net-tools**

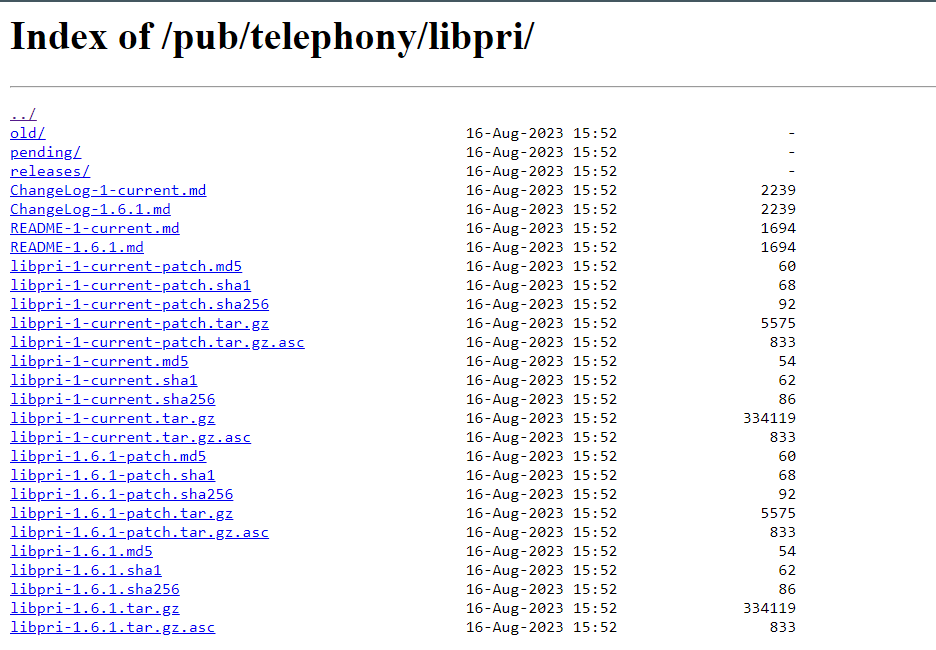
****

****

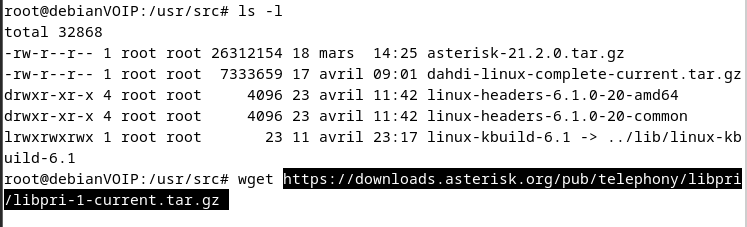
**Une fois le paquet asterisk intallé, on fait de meme pour le paquet dahdi-linux-complete On prend le fichier current sur le site et on tape la commande suivante**

****

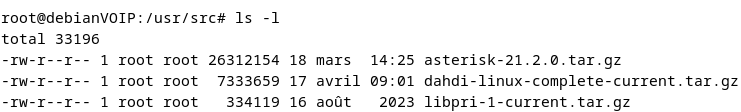
**Apres on recupere le paquet libpri en current.tar.gz egalement**

****

**Puis on refait wget avec le lien recupéré**

****

Voici les 3 fichiers telechargé

****

Maintenant on les dezippé, onfait cette commande pour nos 3 fichiers**, dahdi, libpri et asterisk**

****

Installé les outils necessaire pour compiler nos fichier pour se faire :

**« sudo apt install build-essential** » et les commande make seront disponible

La commande make est un utilitaire largement utilisé dans le développement de logiciels, en particulier dans le contexte de la compilation de programmes à partir de leur code source. Son principal objectif est d'automatiser le processus de compilation en utilisant des fichiers appelés "makefiles" qui spécifient comment les différents fichiers source doivent être compilés et liés pour créer le programme final.

Voici quelques-unes des utilisations principales de make :

1. Compilation de logiciels: make est utilisé pour compiler des programmes à partir de leur code source. Il peut prendre en charge la compilation de projets complexes avec de nombreux fichiers source et dépendances.
2. Gestion de projets: En utilisant des makefiles, make peut gérer les dépendances entre les fichiers source et effectuer uniquement les tâches nécessaires pour reconstruire les parties modifiées du projet. Cela permet d'économiser du temps en évitant de recompiler les parties du projet qui n'ont pas été modifiées.
3. Portabilité: Les makefiles peuvent être écrits pour fonctionner sur différents systèmes d'exploitation et architectures, ce qui facilite la portabilité des projets logiciels.
4. Automatisation: make peut être utilisé pour automatiser diverses tâches liées au développement logiciel, telles que la génération de documentation, l'exécution de tests, la gestion des versions, etc.

En résumé, make est un outil puissant et polyvalent utilisé par les développeurs pour gérer efficacement le processus de compilation et d'automatisation dans le développement de logiciels.

****

**On fait make install**

**Pareil pour notre fichier libpri**

**Apres on va sur le fichier asterisk**

****

**Puis on execute le fichier install prereq install**

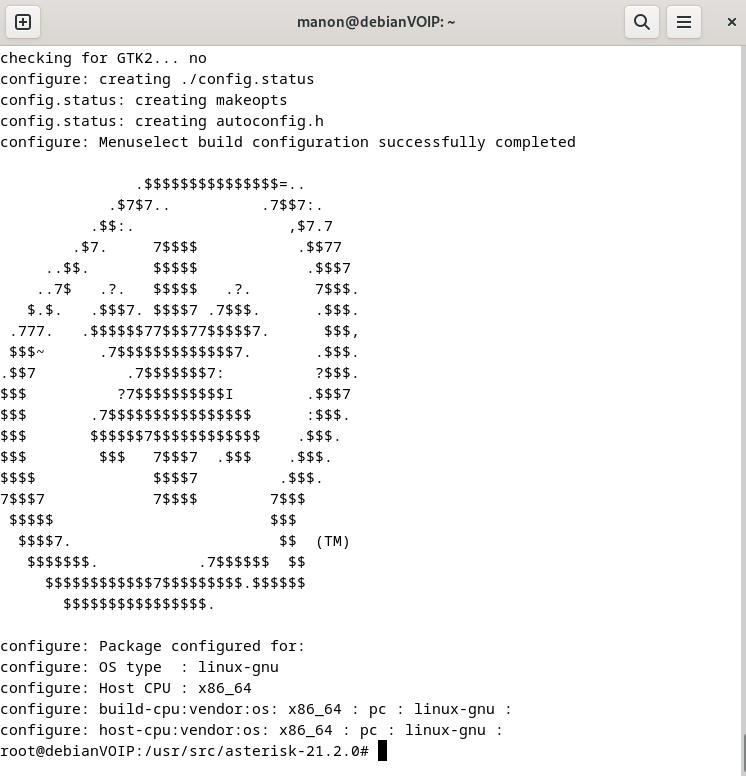
****

**Une fois l’installations terminé**

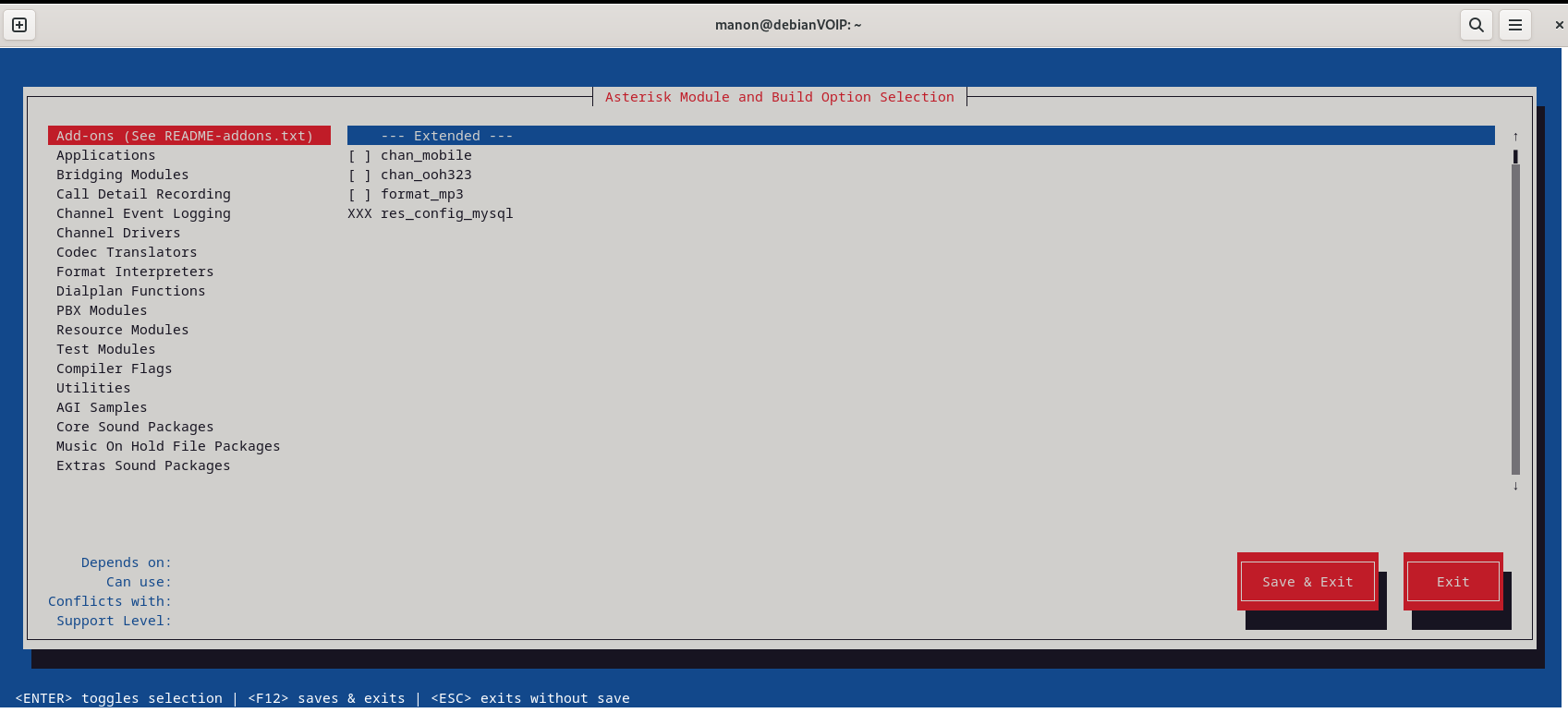
**On retourne au debut du fichiers asterisk et on fait la commande suivante**

****

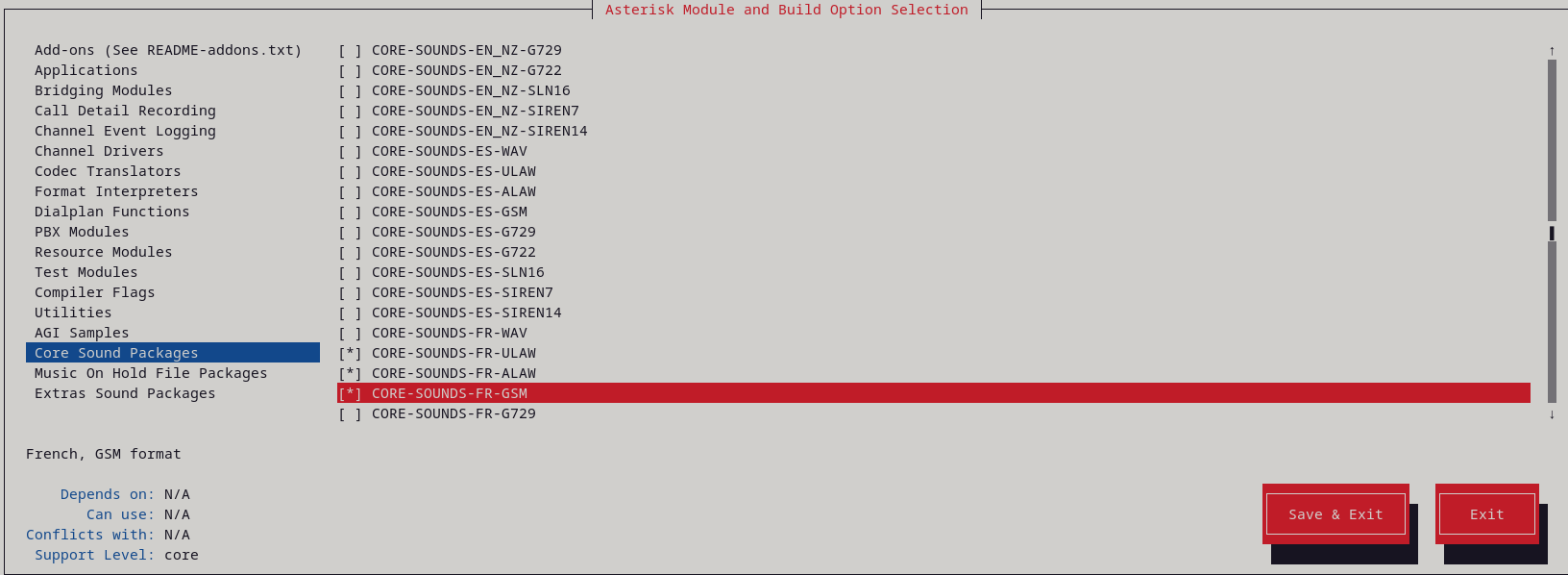
**Si le logo apparait c’est que asterisk est bien configuré**

****

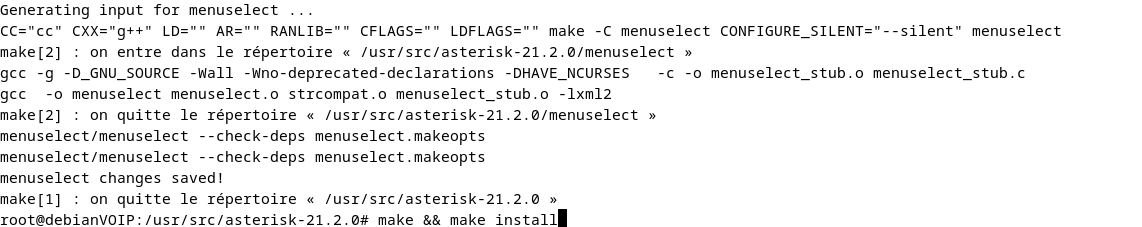
**On tape la commande make menuselect pour ouvrir la fenetre**

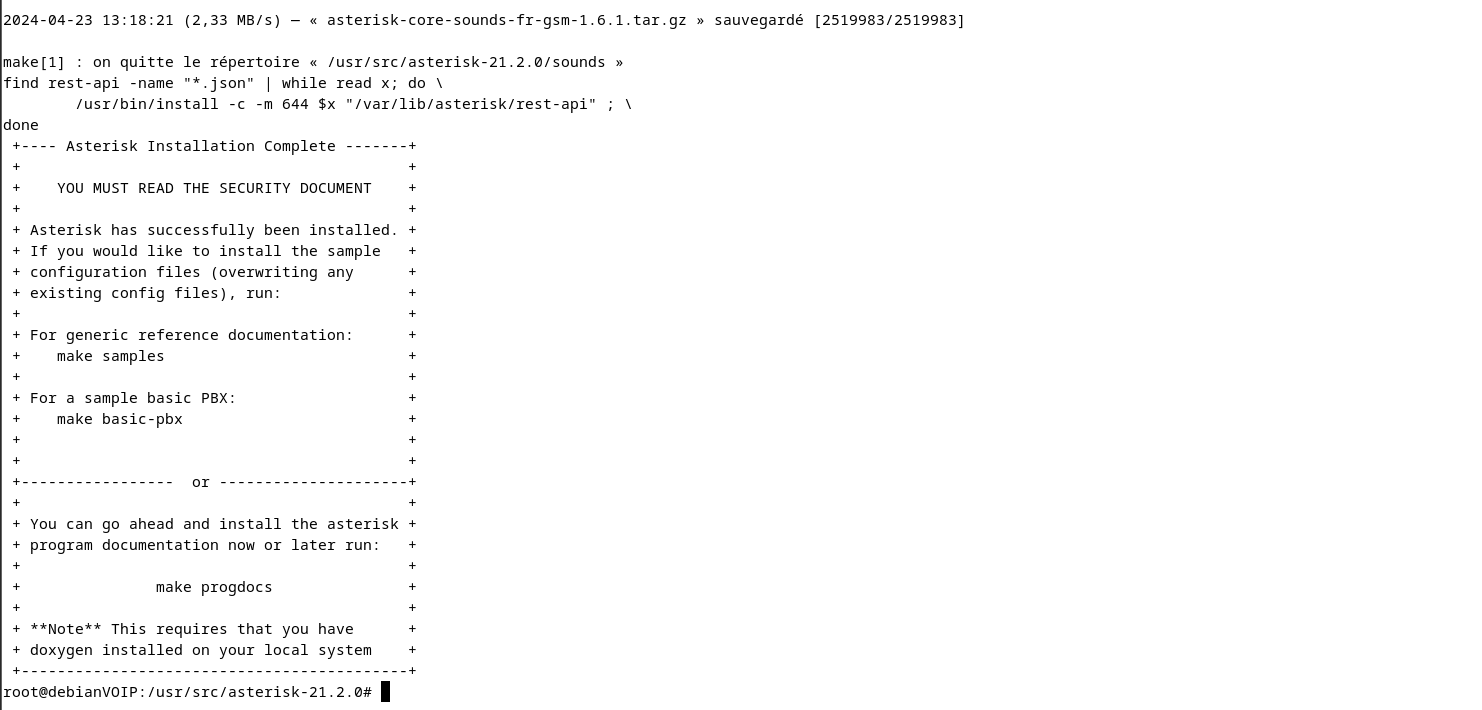
****

**On choisi le codec dans la langue que l’on veut, nous francais**

****

**On sauvegarde et quitte la fenetre puis on fait les commande suivante**

****

****

Faire la commande make samples 

La commande `**make samples**` dans Asterisk est essentielle pour compiler et installer les exemples de configuration et de scripts inclus avec le logiciel. Cela permet aux utilisateurs de comprendre et de configurer Asterisk plus facilement en leur fournissant des modèles pratiques et des guides de configuration pour divers scénarios téléphoniques.

Puis 

La commande `**make config`** dans Asterisk génère les fichiers de configuration réels à partir des modèles fournis dans le répertoire de configuration de base, facilitant ainsi la personnalisation et le déploiement rapide d'un système de téléphonie.

On termine par cette commande



La commande `make install-logrotate` dans Asterisk installe un fichier de configuration spécifique pour logrotate, facilitant la gestion efficace des journaux d'Asterisk. Voici les points clés à retenir :

1. \*\*Gestion des Journaux\*\* : Elle permet de gérer automatiquement les journaux d'Asterisk en définissant les règles de rotation, d'archivage et de compression des fichiers journaux.

2. \*\*Installation Automatique\*\* : Elle installe le fichier de configuration logrotate spécifique à Asterisk, définissant les paramètres de gestion des journaux pour assurer un fonctionnement efficace du système.

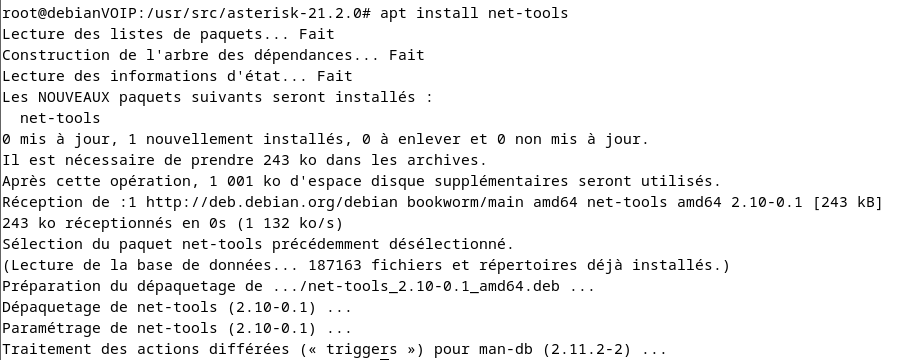
3. \*\*Prévention de la Croissance Incontrôlée des Journaux\*\* : Elle aide à éviter que les fichiers journaux ne deviennent trop volumineux en les rotant périodiquement, ce qui permet de conserver un historique des événements tout en limitant l'espace disque utilisé.

En somme, `make install-logrotate` simplifie la gestion des journaux d'Asterisk en automatisant le processus de rotation et d'archivage, assurant ainsi une surveillance efficace du système de téléphonie.

On installe le paquet net-tools qui est un ensemble d'outils en ligne de commande permettant de gérer divers aspects du réseau sur les systèmes Linux. L'installation de la bibliothèque "net-tools" via la commande **sudo apt install net-tools** vous donne accès à plusieurs outils réseau utiles, notamment :

1. **ifconfig** : Affiche et configure les interfaces réseau sur votre système, y compris les adresses IP, les masques de sous-réseau et les informations sur le trafic réseau.
2. **netstat** : Affiche les connexions réseau actives, les tables de routage, les statistiques d'interface et d'autres informations sur les connexions réseau en cours.
3. **arp** : Affiche et modifie la table ARP (Address Resolution Protocol) du système, qui associe les adresses IP aux adresses MAC des périphériques réseau.
4. **route** : Affiche et manipule la table de routage du système, qui détermine la manière dont les paquets réseau sont acheminés à travers le réseau.
5. **iptunnel** : Permet de créer et de gérer des tunnels réseau, notamment des tunnels IP (IPv4 et IPv6) et des tunnels Ethernet.

Ces outils sont essentiels pour diagnostiquer, configurer et gérer les paramètres réseau sur les systèmes Linux. Ils sont largement utilisés par les administrateurs système pour surveiller et dépanner les réseaux.



La commande `netstat -anu` est utilisée pour afficher les connexions réseau actives ainsi que les statistiques de réseau sur un système Linux. Voici une explication de chaque option utilisée dans cette commande :

- `netstat`: C'est l'utilitaire en ligne de commande utilisé pour afficher les informations sur les connexions réseau.

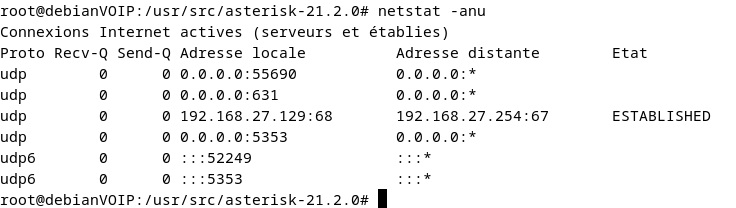
- `-a` : Cette option signifie "afficher toutes les connexions". Elle indique à `netstat` d'afficher les connexions réseau en écoute et les connexions établies.

- `-n` : Cette option signifie "afficher les adresses IP numériques". Elle empêche `netstat` de résoudre les adresses IP en noms d'hôte, ce qui accélère l'affichage des informations.

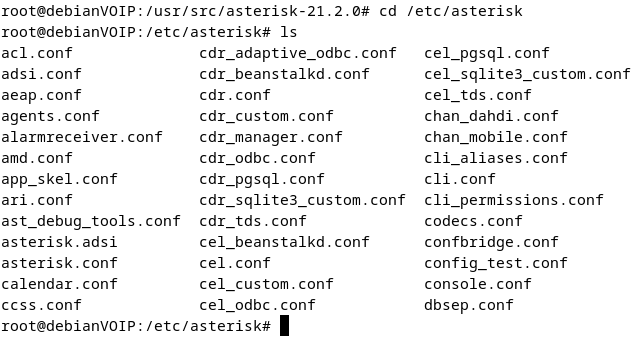
- `-u` : Cette option signifie "afficher les connexions UDP". Elle indique à `netstat` d'afficher uniquement les connexions réseau de type UDP.

En combinant ces options, la commande `netstat -anu` affiche toutes les connexions réseau actives, y compris les connexions UDP, en affichant les adresses IP numériques plutôt que les noms d'hôte.

Cela peut être utile pour surveiller les connexions réseau en temps réel, diagnostiquer les problèmes de réseau et vérifier quelles applications ou services utilisent les connexions UDP sur votre système.



Configuration de base d’asterisk



On va dans le fichier pjsip.conf pour déclaré nos telephones

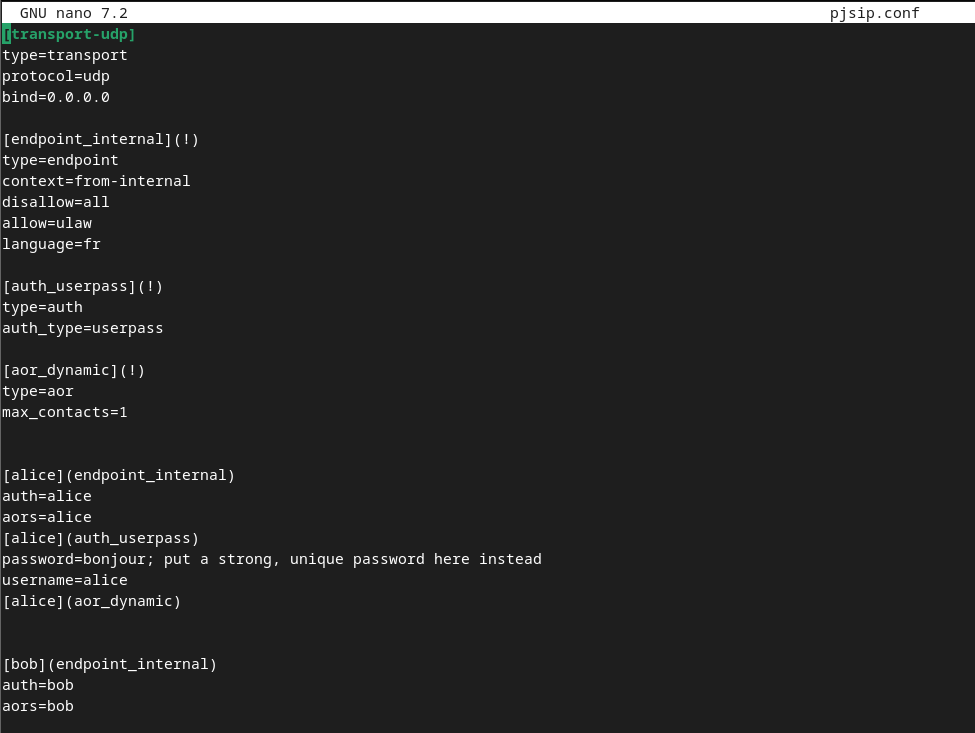
On renomme nos fichiers que l’on va modifier cela crée une sauvegarde au cas ou

Les 2 fichiers concerné sont pjsip.conf et extensions.conf



Voici le fichier pjsip.conf

1. **[transport-udp]**
   * C'est une section qui définit un type de transport pour le protocole UDP. Dans ce cas, il s'agit d'un transport UDP.
2. **type=transport**
   * Cela indique le type de cette section, qui est un transport.
3. **protocol=udp**
   * Spécifie le protocole utilisé par ce transport, qui est UDP dans ce cas.
4. **bind=0.0.0.0**
   * Cette ligne spécifie l'adresse IP à laquelle le transport est lié. **0.0.0.0** signifie que le serveur écoute sur toutes les interfaces disponibles.
5. **[endpoint\_internal](!)**
   * Une autre section qui définit un endpoint pour les utilisateurs internes.
6. **type=endpoint**
   * Indique que cette section définit un endpoint.
7. **context=from-internal**
   * Spécifie le contexte dans lequel les appels entrants de cet endpoint seront traités.
8. **disallow=all**
   * Désactive tous les codecs pour cet endpoint.
9. **allow=ulaw**
   * Autorise uniquement le codec μ-law (ulaw) pour cet endpoint.
10. **language=fr**
    * Définit la langue pour cet endpoint à français.
11. **[auth\_userpass](!)**
    * Une section qui définit un type d'authentification basée sur un nom d'utilisateur et un mot de passe.
12. **type=auth**
    * Indique que cette section définit un type d'authentification.
13. **auth\_type=userpass**
    * Spécifie que le type d'authentification est basé sur un nom d'utilisateur et un mot de passe.
14. **[aor\_dynamic](!)**
    * Une section qui définit un type de ressource d'adresse pour les utilisateurs dynamiques.
15. **type=aor**
    * Indique que cette section définit un type de ressource d'adresse.
16. **max\_contacts=1**
    * Spécifie le nombre maximal de contacts pour cette ressource d'adresse. Dans ce cas, il est fixé à 1.
17. **[alice](endpoint\_internal)**
    * Une section qui définit un endpoint nommé "alice" avec des paramètres hérités de la section "endpoint\_internal".
18. **auth=alice**
    * Spécifie l'authentification utilisée par cet endpoint, qui est "alice" (probablement une référence à une section d'authentification).
19. **aors=alice**
    * Spécifie les ressources d'adresses utilisées par cet endpoint, qui sont également "alice" (probablement une référence à une section de ressource d'adresse).
20. **[alice](auth\_userpass)**
    * Une section qui définit les informations d'authentification pour l'utilisateur "alice".
21. **password=bonjour; put a strong, unique password here instead**
    * Définit le mot de passe de l'utilisateur "alice". La recommandation est de remplacer "bonjour" par un mot de passe fort et unique.
22. **username=alice**
    * Spécifie le nom d'utilisateur pour l'utilisateur "alice".
23. **[alice](aor\_dynamic)**
    * Une section qui définit les informations de ressource d'adresse pour l'utilisateur "alice"



Voici les modifications du fichiers extensions

Ce fichier extensions.conf configure le système téléphonique basé sur Asterisk. Voici un résumé de ce que chaque ligne fait :

1. Le contexte `[from-internal]` définit un groupe d'extensions pour les appels internes au système.

2. Les lignes commençant par `exten=>` définissent des règles pour le routage des appels :

- `exten=>300,1,Dial(PJSIP/alice,10)` : Appeler l'extension 300 et acheminer l'appel vers l'endpoint SIP nommé "alice" pendant 10 secondes.

- `exten=>301,1,Dial(PJSIP/bob,10)` : Appeler l'extension 301 et acheminer l'appel vers l'endpoint SIP nommé "bob" pendant 10 secondes.

- `exten=>309,1,VoiceMailMain()` : Appeler l'extension 309 et accéder au système de messagerie vocale principal.

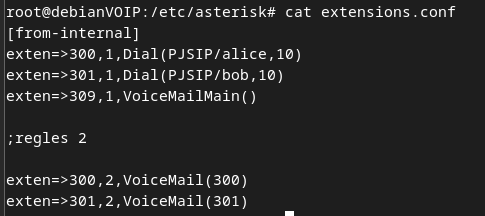
3. La ligne `;regles 2` est un commentaire indiquant une section de règles supplémentaires.

4. Les lignes suivantes spécifient des actions alternatives pour les extensions en cas d'occupation ou de non-réponse :

- `exten=>300,2,VoiceMail(300)` : Si l'appel à l'extension 300 n'est pas répondé ou est occupé, transférer vers la boîte vocale de l'extension 300.

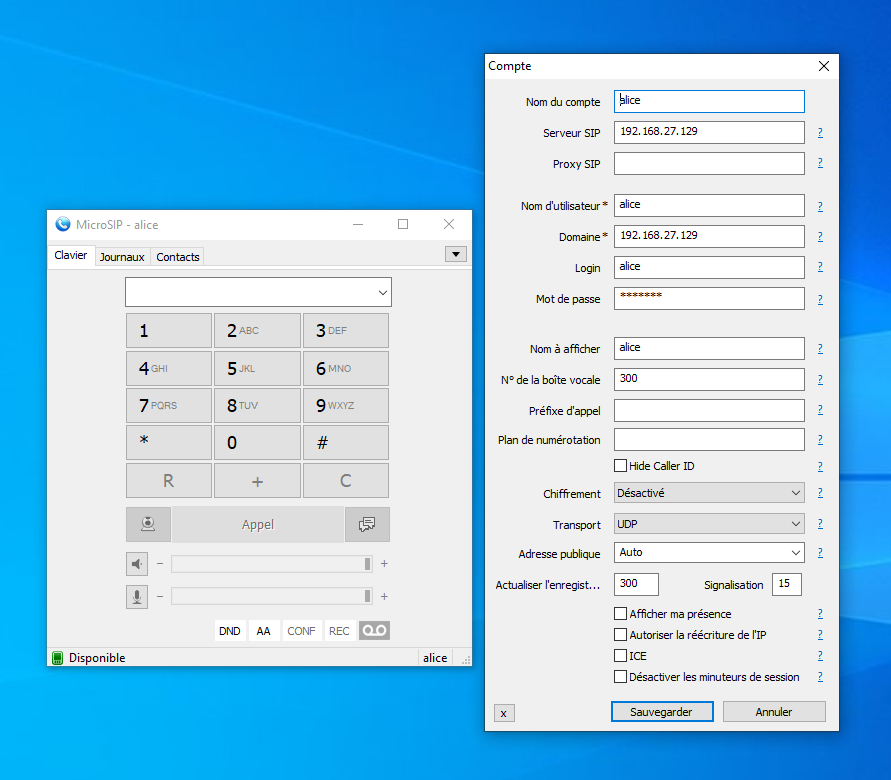
- `exten=>301,2,VoiceMail(301)` : Si l'appel à l'extension 301 n'est pas répondé ou est occupé, transférer vers la boîte vocale de l'extension 301.

En résumé, ce fichier définit comment les appels vers certaines extensions sont acheminés vers des endpoints spécifiques, et comment gérer les situations où l'appel n'est pas répondu ou est occupé en redirigeant vers la messagerie vocale.



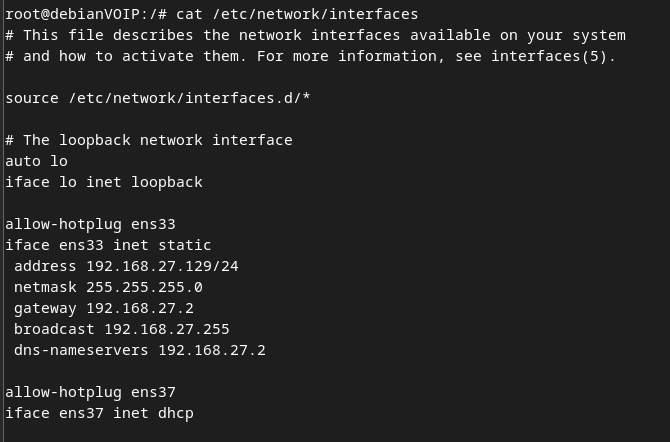
Installer un softphone comme **microsip** et **mizudroid** pour le téléphone.

Voici la configuration quand on ajoute un compte sur microsip :



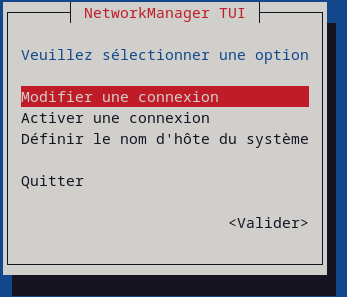
Pour créer un pont entre votre machine virtuelle et votre appareil mobile, ajoutez une deuxième carte réseau à votre machine virtuelle et configurez-la en mode "Bridge" dans les paramètres de virtualisation. Redémarrez ensuite votre machine virtuelle et configurez la nouvelle carte réseau pour utiliser le même réseau que votre appareil mobile en modifiant le fichier d'interfaces réseau sur la machine virtuelle. Une fois cela fait, votre appareil mobile et votre machine virtuelle seront connectés au même réseau local, leur permettant de communiquer directement entre eux.

Pour configurer la seconde carte réseau , il suffit de se rendre dans le fichier network/interfaces pour déclarer notre second réseau

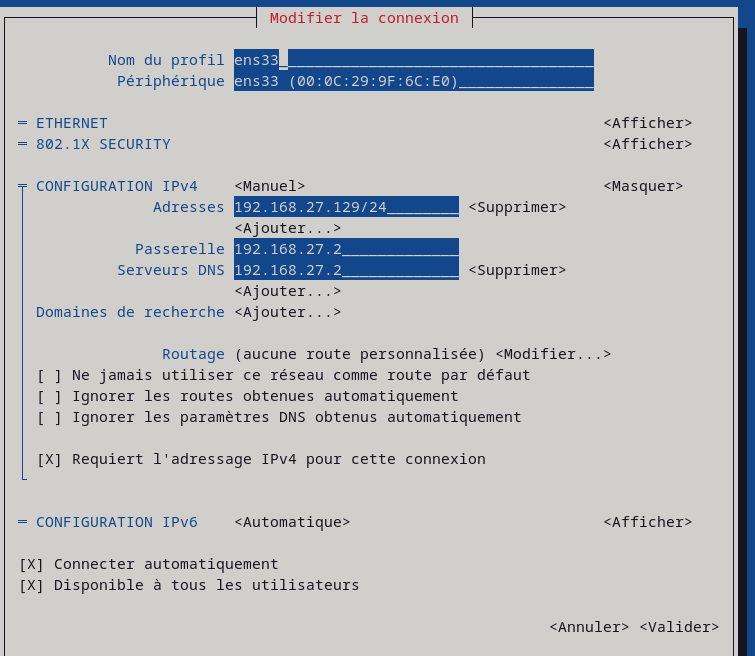


Pour accéder au **network manager :**

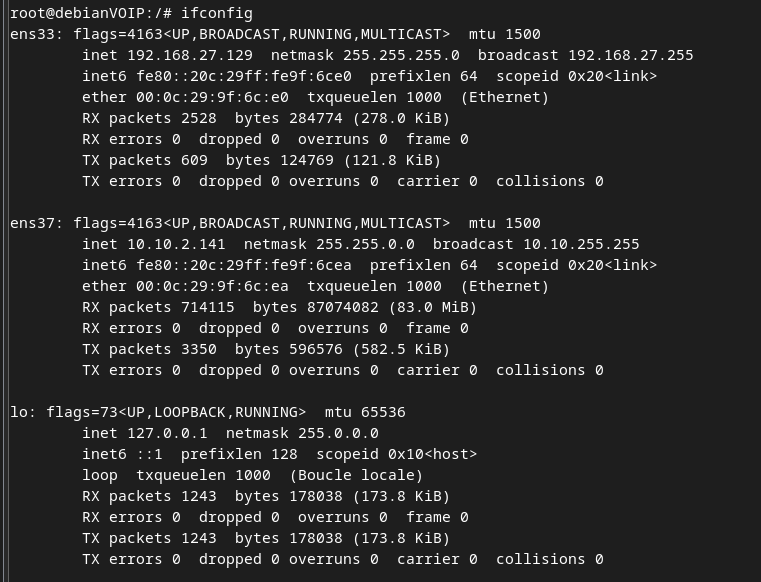


Puis on va attribué une adresse static a notre ens33 qui correspond a notre nat



On redémarre la machine et **ifconfig** pour vérifier ou **ip a**



SECURISATION TLS/SRTP

 Il est possible de se protéger très facilement en adoptant une démarche pragmatique, en réalisant une simple configuration d'Asterisk et en adoptant les outils fail2ban et iptables.

Definitions :

***fail2ban*:**

C’est un outil d'analyse de journaux (log) dont l'objectif premier est de détecter des tentatives d'intrusions ou de connexions infructueuses sur un service et de bannir les adresses IP à l'origine de ces tentatives d'intrusion.

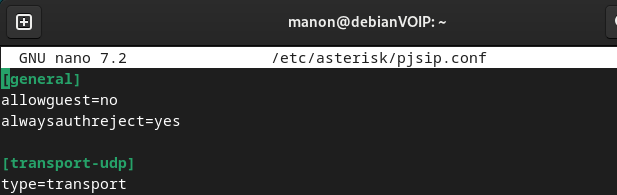
1. **Protection contre les attaques par force brute** : Fail2ban surveille les journaux système pour détecter les tentatives répétées de connexion échouées, typiques des attaques par force brute.
2. **Blocage automatique des adresses IP** : Lorsqu'il détecte une activité suspecte, Fail2ban peut automatiquement bloquer les adresses IP des attaquants en ajoutant des règles au pare-feu.
3. **Configuration flexible** : Les administrateurs peuvent ajuster les paramètres tels que le seuil de tentatives de connexion et la durée de blocage des adresses IP selon leurs besoins.
4. **Notifications d'activité** : Fail2ban peut envoyer des notifications par e-mail ou d'autres moyens pour informer les administrateurs des actions prises, comme le blocage d'une adresse IP.

Pour sécuriser Asterisk, il est important d'adopter une approche logique et pragmatique. Pour cela, nous respecterons deux règles très simples :

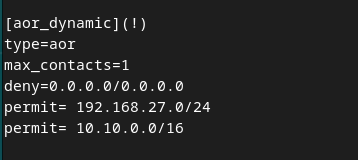
1. **n'autoriser que ce qui est nécessaire**. C'est-à-dire que l'on n'activera pas les services que l'on n'utilise pas d'une part, et que l'on appliquera des filtrages par adresse IP lorsque cela est possible d'autre part.
2. **ne pas utiliser un identifiant et/ou un mot de passe simple.** Jamais. Il ne faut pas utiliser d'identifiant trivial comme : "100", "abc", "demo", "Pierre", "test", "temporaire", etc. Idem pour les mots de passe.

On commence par filtrer l’authentification SIP par adresse IP

Pour se faire on va dans le fichier pjsip.conf, on va préciser l’adresse IP ou les adresses IP autorisées



Avec cette configuration, les paramètres généraux spécifiés dans la section **[general]** seront appliqués avant les autres sections de configuration. Cela garantit que les paramètres tels que **allowguest=no** et **alwaysauthreject=yes** sont pris en compte dans tout le fichier de configuration.



**Configurer tls sur asterisk**

Tout d'abord, créons un emplacement pour nos clés.

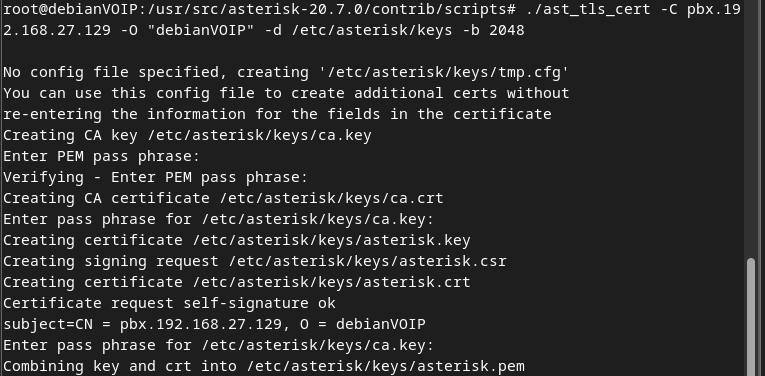
mkdir /etc/asterisk/keys Ensuite, utilisez le script "ast\_tls\_cert" dans le répertoire source "contrib/scripts" d'Asterisk pour créer une autorité de certification auto-signée et un certificat Asterisk.

./ast\_tls\_cert -C pbx.mycompany.com -O "Ma Super Entreprise" -d /etc/asterisk/keys -b 2048 L'option "-C" est utilisée pour définir notre hôte - nom DNS ou notre adresse IP.

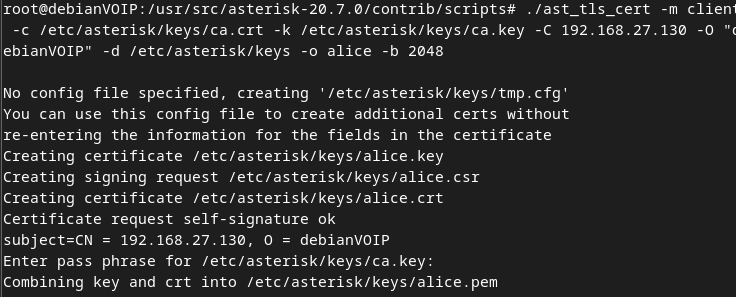
L'option "-O" définit le nom de notre organisation. L'option "-d" est le répertoire de sortie des clés. L'option "-b" spécifie la taille du fichier de clé privée, par défaut est 1024 sauf sur la branche principale.

Vous serez invité à saisir une phrase secrète pour /etc/asterisk/keys/ca.key, entrez quelque chose que vous vous souviendrez pour plus tard. Cela créera le fichier /etc/asterisk/keys/ca.crt. Vous serez invité à saisir à nouveau la phrase secrète, puis le fichier /etc/asterisk/keys/asterisk.key sera créé. Le fichier /etc/asterisk/keys/asterisk.crt sera automatiquement généré. Vous serez invité à saisir la phrase secrète une troisième fois, et le fichier /etc/asterisk/keys/asterisk.pem sera créé, une combinaison des fichiers asterisk.key et asterisk.crt.

Phrase secrete : happybilly



Maintenant nous allons générer un certificat mais pour notre client pour nous alice



L'option "-m client" indique au script que nous voulons un certificat client, et non un certificat serveur.

L'option "-c /etc/asterisk/keys/ca.crt" spécifie l'autorité de certification (nous-mêmes) que nous utilisons.

L'option "-k /etc/asterisk/keys/ca.key" fournit la clé pour l'autorité de certification définie ci-dessus.

L'option "-C", puisque nous définissons un client cette fois, est utilisée pour définir le nom d'hôte ou l'adresse IP de notre téléphone SIP.

L'option "-O" définit le nom de notre organisation.

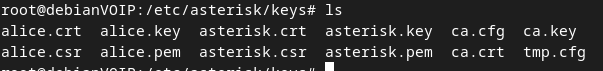
L'option "-d" est le répertoire de sortie des clés.

L'option "-o" est le nom de la clé que nous produisons.

L'option "-b" spécifie la taille du fichier de clé privée, par défaut elle est de 1024 bits sauf sur la branche principale.

Vous serez invité à saisir la phrase secrète précédente pour déverrouiller /etc/asterisk/keys/ca.key.

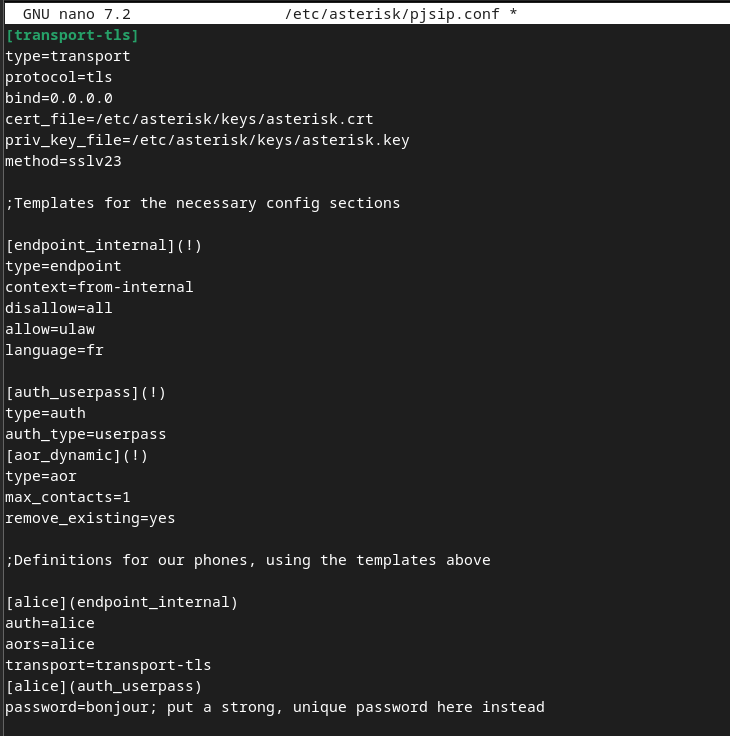
Vérifier que les certificats ont bien été éditer :

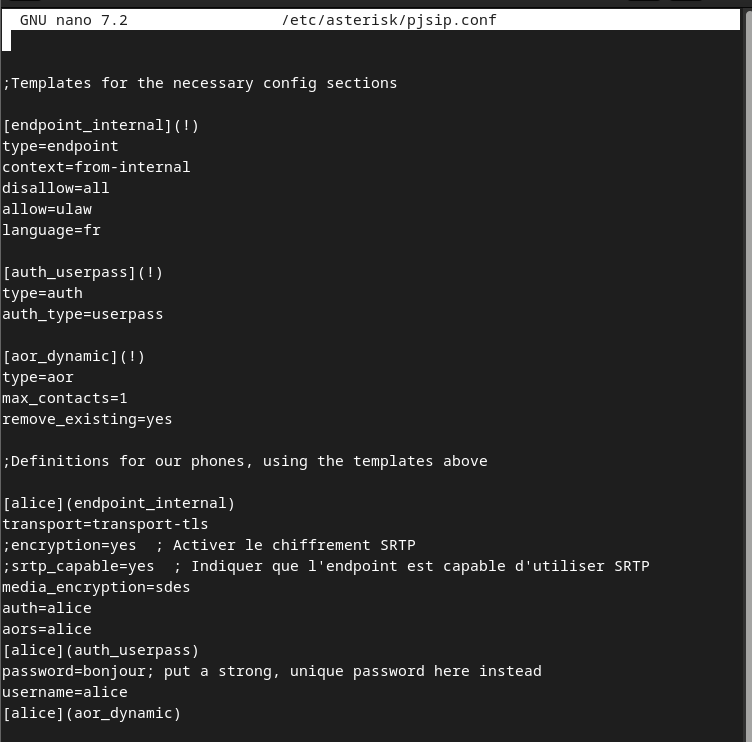


Une fois les certifcat créé transferer les fichiers alice.pem et ca.crt sur la vm client ou se trouve le microsip d’alice

Pour se faire par usb ou par mail

On met les fichiers sur la vm client puis on les place dans le fichier microsip

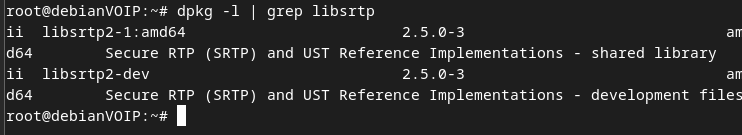




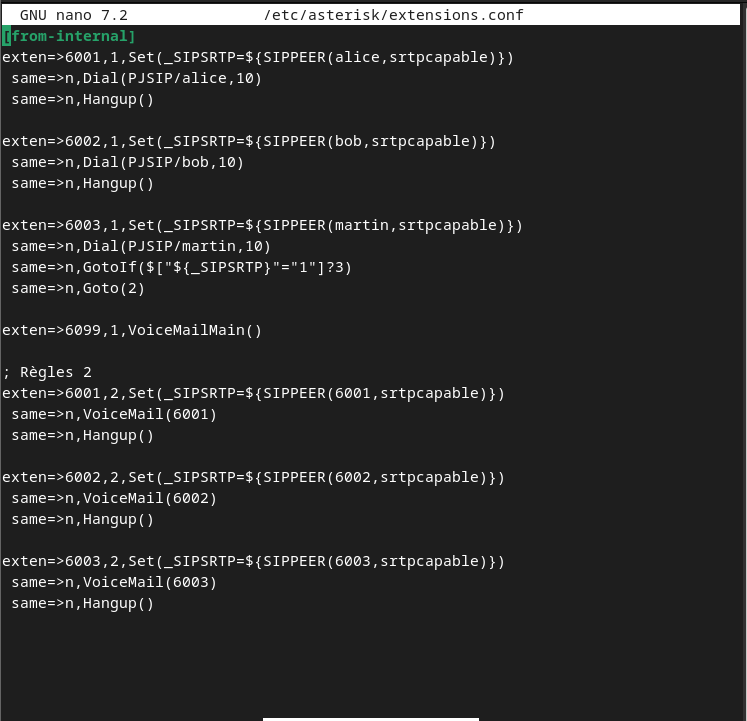
Dans le fichier pjsip.conf, les lignes **encryption=yes** et **srtp\_capable=yes** sont effectivement obsolètes car elles sont implicitement définies lorsque vous utilisez **media\_encryption=sdes**

En spécifiant **media\_encryption=sdes** dans la configuration de l'endpoint dans le fichier pjsip.conf, vous indiquez à Asterisk d'activer le chiffrement SRTP pour les appels utilisant cet endpoint et de sécuriser les médias échangés lors de ces appels en utilisant la méthode de chiffrement SDES. Ainsi, tous les médias, tels que la voix, seront cryptés pour garantir la confidentialité des communications. Par conséquent, les lignes **encryption=yes** et **srtp\_capable=yes** deviennent superflues et peuvent être retirées car elles sont implicitement définies par **media\_encryption=sdes**

Voici la commande pour voir si un paquet est bien installé



Config du fichiers extensions pour srtp



Bien sûr, voici une explication de chaque ligne de votre extension :

```ini

exten=>6001,1,Set(\_SIPSRTP=${SIPPEER(alice,srtpcapable)})

```

- `exten=>6001,1`: Cette ligne indique que cette extension s'applique lorsque le numéro 6001 est composé et qu'elle est prioritaire 1 dans le contexte actuel.

- `Set(\_SIPSRTP=${SIPPEER(alice,srtpcapable)})`: Cette ligne utilise la fonction Set pour définir une variable appelée `\_SIPSRTP`. La valeur de cette variable est déterminée en appelant la fonction SIPPEER() avec deux arguments : le nom de l'endpoint ("alice" dans ce cas) et la capacité SRTP (srtpcapable). Cela permet de vérifier si l'endpoint "alice" est capable d'utiliser SRTP.

```ini

same=>n,Dial(PJSIP/alice,10)

```

- `same=>n`: Cette ligne indique que la priorité suivante est dans la même extension (6001) et qu'elle est la priorité "n" (numéro suivant).

- `Dial(PJSIP/alice,10)`: Cette ligne effectue un appel sortant vers l'endpoint PJSIP "alice" avec une durée de sonnerie de 10 secondes. Cela signifie qu'Asterisk va essayer de joindre l'endpoint "alice" pendant 10 secondes. Si l'appel est réussi, il passera à la prochaine priorité. Sinon, il exécutera les instructions de la prochaine priorité.

```ini

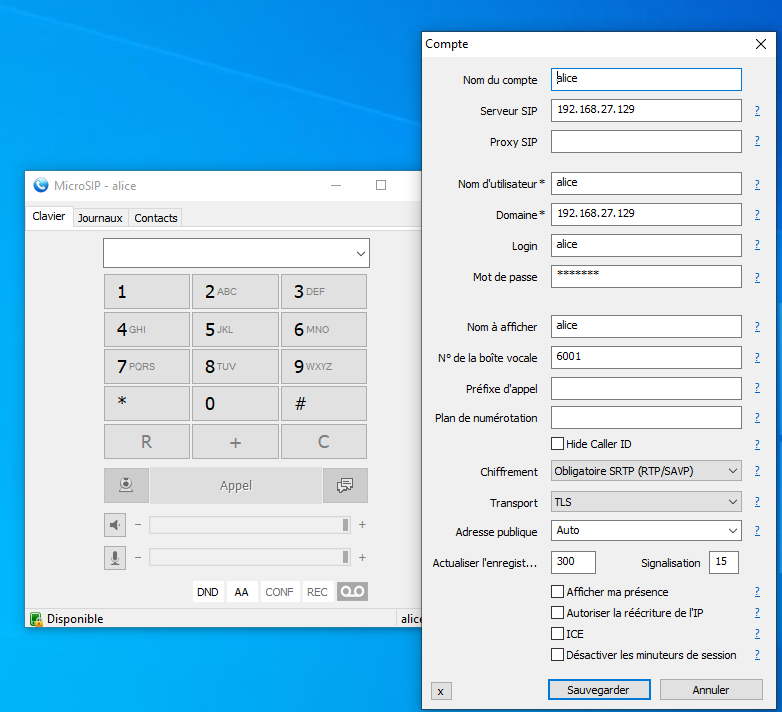
same=>n,Hangup()

```

- `same=>n`: Cette ligne indique que la priorité suivante est dans la même extension (6001) et qu'elle est la priorité "n" (numéro suivant).

- `Hangup()`: Cette ligne met fin à l'appel. Elle s'exécute après que l'appel soit terminé, que ce soit parce que l'appel a été connecté ou parce qu'il a échoué. Cela garantit que la ligne soit libérée après l'appel, quelle que soit son issue.

Ces lignes sont utilisées pour vérifier la capacité SRTP de l'endpoint "alice" et pour effectuer un appel sortant vers cet endpoint avec SRTP activé, si l'endpoint est capable de l'utiliser.



En activant le chiffrement SRTP obligatoire, vous renforcez la sécurité de vos communications vocales en empêchant les attaques d'écoute passive et en garantissant que les données vocales sont cryptées pendant le transport. Cela est particulièrement important lorsque vous effectuez des appels sur des réseaux non sécurisés ou sur Internet, où il peut y avoir des risques potentiels pour la confidentialité des communications.