Voix sur IP Déploiement d'une solution open source de VoIP

Partie 1

1- Installation d'Asterisk

Asterisk est un logiciel open source qui permet de transformer un simple ordinateur en autocommutateur téléphonique privé (PABX). Il permet de gérer des appels, conférences, transferts, messagerie vocale, etc.

Commandes d'installation (Debian/Ubuntu): sudo apt install asterisk
Vérification du bon fonctionnement : asterisk -rvvv

```
Asterisk 22.2.0, Copyright (C) 1999 - 2025, Sangoma Technologies Corporation and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 22.2.0 currently running on hp-HP-ProBook-440-G4 (pid = 16655)
```

2- Configuration d'Asterisk

- **Définition du plan de numérotation – extensions.conf**

Le fichier **extensions.conf** permet de spécifier le comportement d'Asterisk selon les numéros composés. C'est ici que l'on définit la logique de traitement des appels.

```
;====================== Appels directs
exten => 1001,1,Dial(PJSIP/twinkle1,,Tt)    ; twinkle1 extension
same => n,Hangup()

exten => 1002,1,Dial(PJSIP/twinkle2,,Tt)    ; twinkle2 extension
same => n,Hangup()

exten => 1003,1,Dial(PJSIP/zoiper1,,Tt)     ; Zoiper1 extension
same => n,MixMonitor(/var/lib/asterisk/sounds/recordings/${UNIQUEID}.wav)
same => n,Hangup()
```

Ces lignes de code font partie d'un **plan de numérotation dans Asterisk**. Elles définissent des **extensions (numéros internes)** et les actions associées :

- `exten => 1001,1,Dial(PJSIP/twinkle1,,Tt):`
  Lorsque l'on compose **1001**, Asterisk appelle le terminal SIP nommé **twinkle1**.
  Le paramètre **`-Tt -`** permet au **correspondant ou à l'appelant** de pouvoir transférer l'appel.
- `same => n,Hangup():`
  Une fois l'appel terminé ou s'il échoue, l'appel est **raccroché**.

Même principe pour :

- `1002` → appelle **twinkle2**
- `1003` → appelle **zoiper1**

- **Définition des utilisateurs SIP – pjsip.conf**

Le fichier **`pjsip.conf`** permet de configurer les clients SIP (Softphones, téléphones IP, etc.) qui se connectent au serveur Asterisk.
Chaque utilisateur est identifié par un nom (**username**) et un mot de passe (**secret**).

```
[twinkle1]
type=endpoint
context=internal
disallow=all
allow=ulaw
auth=twinkle1-auth
aors=twinkle1

[twinkle1-auth]
type=auth
auth_type=userpass
password=twinklepass1
username=twinkle1

[twinkle1]
type=aor
max_contacts=1
```

Cela permet de créer un poste SIP (twinkle1) avec un mot de passe sécurisé, relié au contexte d'appel interne, et autorisant un seul appareil à se connecter.

- **Établissement d'un Appel entre Extensions**

Dans cet extrait, on observe le déroulement d'un appel entre deux extensions via le protocole PJSIP dans Asterisk. L'extension 1002 tente d'établir un appel vers l'extension **twinkle2**, ce qui est confirmé par le message **"Called PJSIP/twinkle2"**, suivi de **"twinkle2 is ringing"**, indiquant que l'appel sonne bien chez le destinataire. Ensuite, un autre appel est lancé depuis l'extension 1001 vers **twinkle1**, avec un fonctionnement similaire. Ces lignes montrent comment le serveur Asterisk gère le processus d'appel, en signalant les étapes de **composition, appel et sonnerie**.

```
Executing [1002@internal:1] Dial("PJSIP/twinkle1-00000000", "PJSIP/twinkle2,,Tt") in new stack
Called PJSIP/twinkle2
PJSIP/twinkle2-00000001 is ringing
```

```
Executing [1001@internal:1] Dial("PJSIP/twinkle1-00000002", "PJSIP/twinkle1,,Tt") in new stack
Called PJSIP/twinkle1
PJSIP/twinkle1-00000003 is ringing
```

- **Mise en place d'un appel en conférence avec Asterisk (ConfBridge)**

```
[bridge-default]
type=bridge
max_members=10  ; optional, can remove if not needed

[user-default]
type=user
startmuted=no
dsp_drop_silence=yes
announce_user_count=yes
```

Dans `confbridge.conf`, la section **[bridge-default]** configure le pont de conférence, définissant le type comme **bridge** et limitant le nombre de participants à **10** avec `max_members=10`. La section **[user-default]** définit les paramètres des utilisateurs, incluant des options comme `startmuted=no` pour permettre aux participants de parler dès leur entrée, `dsp_drop_silence=yes` pour ignorer les silences, et `announce_user_count=yes` pour annoncer le nombre de participants dans la conférence.

```
exten => 800,1,Answer()
 same => n,ConfBridge(1234,bridge-default,user-default)
 same => n,Hangup()
```

Cette configuration permet à tout utilisateur appelant l'extension **800** de rejoindre une conférence. Lors de l'appel, l'extension répond avec `Answer()`, puis l'utilisateur est dirigé vers la conférence **1234** en utilisant les profils **bridge-default** (pour la configuration du pont de conférence) et **user-default** (pour les paramètres des utilisateurs). Enfin, l'appel est raccroché avec `Hangup()` après que l'utilisateur quitte la conférence.

- **Enregistrement des appels Zoiper avec MixMonitor dans Asterisk**

```
exten => 1003,1,Dial(PJSIP/zoiper1,,Tt)    ; Zoiper1 extension
same => n,MixMonitor(/var/lib/asterisk/sounds/recordings/${UNIQUEID}.wav)
same => n,Hangup()
```

En ajoutant la commande `MixMonitor` dans le plan de numérotation pour l'extension `1003`, nous permettons à Asterisk d'enregistrer les appels entrants et sortants vers cette extension. Lorsque l'extension `1003` est appelée, la commande `MixMonitor` démarre l'enregistrement d' appel et le sauvegarde sous un fichier **`.wav`** unique dans le répertoire spécifié, basé sur l'ID unique de l'appel. Cela permet de conserver un enregistrement de l'appel Zoiper à chaque fois qu'un appel est passé à cette extension.

- **Menu vocal interactif (IVR) dans Asterisk**

```
exten => s,1,Answer()
 same => n(loop),Background(vm-options)
 same => n,WaitExten()

exten => 1,1,Playback(you-entered)
 same => n,SayNumber(1)
 same => n,Goto(s,loop)

exten => 2,1,Playback(you-entered)
 same => n,SayNumber(2)
 same => n,Goto(s,loop)

exten => t,1,Playback(vm-goodbye)
 same => n,Hangup()

exten => i,1,Playback(invalid)
 same => n,Goto(s,loop)

exten => 6598,1,Goto(internal,s,1)
```

Ce code définit un menu vocal interactif (IVR) dans Asterisk, permettant aux appelants d'écouter un message audio et de faire un choix en appuyant sur une touche (ex. 1 ou 2). Selon le choix, un message de confirmation est joué et le menu démarrer. Il gère aussi les cas où l'appelant ne répond pas (timeout) ou entre une touche invalide. L'appel peut être dirigé vers ce menu en composant un numéro spécifique. Ce type de configuration est couramment utilisé pour orienter les appels automatiquement dans un standard téléphonique.

- **Musique en attente dans Asterisk**

```
exten => 1001,1,Dial(PJSIP/twinkle1,,Tt)
same => n,MusicOnHold(default)                ; Play Music on Hold from the "default" class
same => n,Hangup()                            ; Hang up the call when done
```

Le code suivant permet d'ajouter de la musique en attente lors d'un appel. Lorsque l'extension `1001` est appelée, la musique est jouée à partir de la classe "default" pendant que l'extension `twinkle1` sonne. Une fois que l'appel est terminé ou après avoir été décroché, la musique en attente cesse et l'appel est

raccroché. Ce processus améliore l'expérience de l'appelant en attendant que la ligne soit connectée.

Partie 2

Dans cette partie, nous allons implémenter un **Serveur Vocal Interactif (SVI)** avec **Asterisk**, permettant à un utilisateur d'obtenir son solde en appelant un numéro d'accès **(115)**. Nous allons connecter Asterisk à une base de données **MySQL** afin d'authentifier les utilisateurs et récupérer leurs soldes.

- Technologies et Matériels Utilisés
  - Système d'exploitation : Linux (Ubuntu/Debian)
  - Logiciel VoIP : Asterisk
  - Base de données : MySQL
  - Langage de programmation : Python
  - Protocoles utilisés : DTMF, SIP
  - Matériel : PC/Serveur local avec Zoiper pour les tests
- Configuration du Plan de Numérotation (extensions.conf)

```
;=============== menu ==============
exten => 115,1,Answer()
 same=> n,Goto(ivr-menu,start,1)

[ivr-menu]
exten => start,1,Answer()
    same => n(loop),Background(en/enter-phoneNumber)
    same => n,Read(phone,,10)

    same => n,Playback(en/enter-pin)
    same => n,Read(pin,,4)

    same => n,AGI(check_balance.py,${phone},${pin})
    same => n,NoOp(Balance is ${balance})

    same=> n,AGI(googletts.agi,"your balance is ${balance}",en)
    same => n,Hangup()
```

Configuration de la Base de Données MySQL

- ### Création de la base et des tables

```sql
CREATE DATABASE asterisk;

USE asterisk;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    phone VARCHAR(20) UNIQUE,
    pin VARCHAR(10),
    balance DECIMAL(10,2)
);

INSERT INTO users (phone, pin, balance) VALUES
('0600000001', '1234', 20.50),
('0600000002', '5678', 35.00);
```

```
MariaDB [(none)]> use asterisk;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [asterisk]> SELECT * FROM users;
+----+------------+------+---------+
| id | phone      | pin  | balance |
+----+------------+------+---------+
|  1 | 0600000001 | 1234 |   20.50 |
|  2 | 0600000002 | 5678 |   35.00 |
+----+------------+------+---------+
2 rows in set (0.001 sec)
```

Développement du Script Python

Script Python pour la Récupération du Solde (**check_balance.py**)

```python
#!/usr/bin/env python3

import sys
import mysql.connector

def get_balance(phone, pin):
    try:
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="password",
            database="asterisk"
        )
        cursor = conn.cursor()
        cursor.execute("SELECT balance FROM users WHERE phone=%s AND pin=%s",
(phone, pin))
        result = cursor.fetchone()
        return result[0] if result else None
    except mysql.connector.Error as err:
        print(f"Database error: {err}", file=sys.stderr)
        return None
    finally:
        if 'conn' in locals() and conn.is_connected():
            conn.close()

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: get_balance.py <phone> <pin>", file=sys.stderr)
        sys.exit(1)

    phone = sys.argv[1]
    pin = sys.argv[2]
    balance = get_balance(phone, pin)

    if balance is not None:
        # Output the balance in a format Asterisk can read
        print(f"SET VARIABLE balance {balance}")
    else:
        # Output 0 if no balance is found or an error occurs
        print("SET VARIABLE balance 0")
```

Nous avons commencé par **configurer Asterisk** en définissant un plan de numérotation permettant aux utilisateurs de saisir leur numéro et leur code PIN. Ensuite, nous avons créé une **base de données MySQL** pour stocker les informations des utilisateurs et leurs soldes. Nous avons développé un **script Python** permettant de récupérer et retourner le solde d'un utilisateur après authentification. Pour rendre l'expérience utilisateur plus naturelle, nous avons intégré la **synthèse vocale Google TTS** via le module asterisk-googletts, afin d'annoncer vocalement le solde à l'appelant.

Déduction de solde pour Asterisk

Le script deduct_balance.py est un script Python conçu pour être utilisé avec Asterisk dans le but de déduire automatiquement le coût d'un appel du solde d'un utilisateur en fonction de la durée de l'appel.

Fonctionnement du script :

1. **Paramètres d'entrée :**
   - `phone` : numéro de téléphone (identifiant de l'utilisateur dans la base de données).
   - `Durée` de l'appel en secondes.
2. **Calcul du coût :**
   - Le coût par seconde est fixé à `0.01€`.
   - Le coût total est calculé en multipliant la durée de l'appel par le tarif par seconde.
3. **Connexion à MySQL :**
   - Le script se connecte à une base de données MySQL nommée `asterisk`.
   - Les identifiants de connexion sont configurés en dur dans le script (`root/password`).
4. **Mise à jour du solde :**
   - Il vérifie si le numéro de téléphone existe dans la table `users`. Si oui, il met à jour le champ `balance` en soustrayant le coût de l'appel.
   - La valeur du solde ne peut pas être inférieure à zéro (protection contre les soldes négatifs).
   - Affiche un message `VERBOSE` avec les anciens et nouveaux soldes.
5. **Gestion des erreurs :**
   - Les erreurs éventuelles (connexion à la BDD, requêtes) sont capturées et affichées via `VERBOSE`.

```python
#!/usr/bin/env python3

import sys
import mysql.connector

def main():
    phone = sys.argv[1]
    duration = int(sys.argv[2])

    cost_per_second = 0.01
    total_cost = duration * cost_per_second

    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='password', # Replace with your MySQL root password
            database='asterisk' # Replace with your database name
        )
        cursor = conn.cursor()

        # Step 3: Update balance
        cursor.execute("SELECT balance FROM users WHERE phone = %s", (phone,))
        result = cursor.fetchone()

        if result:
            current_balance = float(result[0])
            new_balance = max(current_balance - total_cost, 0.00)
            cursor.execute("UPDATE users SET balance = %s WHERE phone = %s", (new_balance, phone))
            conn.commit()
            print(f"VERBOSE \"Balance updated: {current_balance} -> {new_balance}\" 1")
        else:
            print(f"VERBOSE \"Phone number {phone} not found in DB.\" 1")

    except Exception as e:
        print(f"VERBOSE \"Error: {str(e)}\" 1")
    finally:
        cursor.close()
        conn.close()

if __name__ == "__main__":
    main()
```

Intégration dans Asterisk (extensions.conf)

Lorsqu'un appel se termine, la priorité spéciale `h` dans le plan de numérotation d'Asterisk est déclenchée, ce qui permet d'exécuter automatiquement le script `deduct_balance.py`. Ce script reçoit deux arguments : `${PIN}`, qui correspond à l'identifiant de l'utilisateur (généralement le numéro de téléphone ou un identifiant unique), et `${CDR(billsec)}`, qui représente la durée facturable de l'appel en secondes. À partir de ces informations, le script calcule le coût de l'appel en appliquant un tarif prédéfini, puis met à jour le solde de l'utilisateur dans la base de données MySQL. Cette opération est consignée dans le fichier de

log `master.csv`, généré automatiquement par Asterisk, qui enregistre les détails de chaque appel **(date, durée, coût, etc.)**. Ce fichier joue un rôle crucial dans la traçabilité et l'audit des communications, permettant de vérifier que les déductions de solde correspondent bien aux appels passés. Ainsi, l'ensemble du processus assure une gestion automatisée, transparente et fiable de la facturation des utilisateurs.

```
exten => 1001,1,NoOp(Incoming call....)
 same => n,Set(PIN=1001)
 same => n,Dial(PJSIP/zoiper1,,Tt)
 same => n,Hangup()

exten => 1002,1,NoOp(Incoming call...)
 same => n,Set(PIN=1002)
 same => n,Dial(PJSIP/zoiper2,,Tt)
 same => n,Hangup()

exten => h,1,AGI(deduct_balance.py,${PIN},${CDR(billsec)})
 same => n,NoOp(Balance updated for ${PIN})
```

Ce système permet de **facturer automatiquement les utilisateurs selon la durée de leurs appels** et de **gérer les soldes en base de données de manière transparente** depuis Asterisk, assurant une meilleure gestion des comptes utilisateurs.

Questions complémentaires

1. Motiver la solution VoIP proposée et donner la différence entre VoIP et ToIP ?

- **VoIP (Voice over Internet Protocol)** permet de transmettre des appels vocaux via des réseaux de données (comme Internet), en utilisant des protocoles standards comme SIP (Session Initiation Protocol) ou H.323. L'avantage de VoIP est de réduire les coûts de communication en utilisant l'infrastructure Internet existante, tout en offrant des fonctionnalités avancées comme les appels vidéo, la messagerie instantanée, et la conférence.
- **ToIP (Telephony over IP)** fait référence à un système de téléphonie qui utilise des réseaux IP, mais dans un contexte plus large que VoIP. ToIP peut intégrer des services de téléphonie plus complexes, y compris des solutions

de communication unifiées et des intégrations avec d'autres systèmes d'entreprise.

Différence principale : VoIP se concentre principalement sur la transmission de la voix sur Internet, tandis que ToIP inclut également des éléments comme la gestion des appels, la messagerie et d'autres services téléphoniques professionnels.

2. Quels sont les bénéfices de cette solution pour une entreprise ?

- **Réduction des coûts** : VoIP permet de réduire les frais de téléphonie traditionnelle (appels longue distance, équipements coûteux).
- **Flexibilité** : Les employés peuvent utiliser des appareils mobiles ou des ordinateurs pour passer des appels, ce qui permet une plus grande mobilité.
- **Fonctionnalités avancées** : comme la messagerie vocale, la gestion d'appels en conférence, la redirection d'appels, et l'intégration avec d'autres outils d'entreprise (CRM, ERP).
- **Scalabilité** : Facilité d'ajouter de nouvelles lignes sans nécessiter de câblage complexe.
- **Amélioration de la productivité** : La possibilité d'intégrer des solutions de communication unifiée (voix, vidéo, chat) dans une seule plateforme.

3. Est-ce que le déploiement d'une solution VoIP de type Asterisk n'engendre pas des investissements supplémentaires ?

- **Oui, des investissements peuvent être nécessaires** pour :
  - L'achat de serveurs pour héberger Asterisk.
  - Les équipements spécifiques (comme les téléphones IP ou des adaptateurs pour les téléphones analogiques).
  - Des coûts de maintenance et de formation pour administrer et dépanner le système.
  - Des coûts pour le trafic Internet et la gestion de la bande passante (la qualité de service peut nécessiter des investissements supplémentaires). Toutefois, ces coûts sont généralement inférieurs à ceux des systèmes téléphoniques traditionnels.

4. Quels sont les différents messages de signalisation SIP échangés entre un client et le serveur lors de l'établissement d'une communication VoIP ?

Voici les principaux messages SIP échangés lors de l'établissement d'une communication :

- **INVITE** : Le client envoie une demande d'appel au serveur.

- **TRYING** : Le serveur répond pour indiquer qu'il a reçu la demande.
- **RINGING** : Le serveur indique que l'appel est en cours de sonner sur le destinataire.
- **OK (200)** : Le destinataire répond que l'appel est accepté.
- **ACK** : Le client confirme qu'il a reçu la réponse OK.
- **BYE** : Pour mettre fin à l'appel.
- **CANCEL** : Si l'appel est annulé avant d'être répondu.

5. Quels sont les différents messages de signalisation SIP échangés entre deux clients lors de l'établissement d'une communication VoIP ?

Les messages SIP échangés entre deux clients sont similaires :

- **INVITE** : Le client A envoie une demande d'appel au client B.
- **RINGING** : Le client B sonne l'appel.
- **OK (200)** : Le client B accepte l'appel.
- **ACK** : Le client a confirmé la réception du message OK.
- **BYE** : L'un des clients met fin à l'appel.

6. Quel est le protocole utilisé pour le transport des données VoIP ?

Le protocole utilisé pour le transport des données VoIP est généralement **RTP (Real-time Transport Protocol)**. RTP permet le transport des flux audio et vidéo en temps réel, tandis que SIP ou H.323 gère la signalisation des appels.

7. Quels sont les problèmes responsables de la dégradation de la qualité du son ?

Les problèmes pouvant affecter la qualité de la voix sur une solution VoIP sont :

- **Latence** : Délai de transmission des paquets de données.
- **Jitter** : Variation du délai entre les paquets, ce qui perturbe la qualité de la communication.
- **Perte de paquets** : Certains paquets ne sont pas transmis correctement.
- **Bande passante insuffisante** : Manque de bande passante pour supporter les appels de haute qualité.
- **Interférences réseau** : Congestion du réseau, ce qui peut entraîner des appels coupés ou de la distorsion.
- **Équipement défectueux** : Problèmes avec les microphones, écouteurs ou câbles réseau.

8. Proposer une méthode permettant l'utilisation du fax sous Asterisk.

- **T.38 Fax over IP** : Asterisk supporte le transport de fax via le protocole T.38, qui permet de transmettre des fax sur des réseaux IP. Il faut configurer Asterisk pour utiliser T.38, et s'assurer que les équipements de terminaux (comme les télécopieurs ou les adaptateurs VoIP) sont compatibles avec ce protocole.

9. Donner quelques limitations du serveur de VoIP Asterisk.

- **Scalabilité** : Asterisk peut devenir difficile à gérer à grande échelle sans une configuration complexe ou l'ajout de ressources supplémentaires.
- **Gestion de la qualité du service** : Asterisk ne gère pas nativement la qualité de service (QoS) réseau, ce qui nécessite une configuration manuelle des priorités de trafic pour éviter des problèmes de qualité audio.
- **Compatibilité avec certains équipements** : Certains périphériques ou systèmes propriétaires peuvent ne pas être entièrement compatibles avec Asterisk.
- **Sécurité** : Comme pour toute solution open source, la configuration sécurisée d'Asterisk nécessite une attention particulière pour éviter les attaques par déni de service (DoS) ou d'autres menaces liées aux télécommunications.

Cela couvre les principaux aspects de la VoIP, Asterisk, et les solutions associées. Si vous avez besoin de détails supplémentaires, n'hésitez pas à demander !

Conclusion

Ce projet implique plusieurs aspects de la gestion des appels interactifs avec Asterisk, notamment l'intégration de la messagerie vocale, la musique en attente, l'IVR pour la consultation de solde, et la conversion de chiffres en voix avec TTS. La gestion du solde, de la facturation et de la recharge nécessite aussi une base de données pour stocker les informations des abonnés et un système de recharge intégré. Vous pouvez enrichir cette solution en intégrant un système de paiement ou un autre mécanisme pour permettre aux utilisateurs de recharger leur solde en temps réel.