Rapport Project SysG5 : Time Setup

42933 Achetouan Mohammed - 45682 Kardillo Younes

7 mai 2020

Table des matières

Chapitre 1

Introduction

L'objectif de ce travail est de créer une application permettant de régler l'heure de différentes machines se trouvant dans un même réseau en utilisant le protocole «**NTP**».
Tout notre script a été fait en utilisant le langage «**Python**». Afin que notre application se lance dans des meilleures conditions, nous avons opté pour un environnement virtuel «**Python3.6.3**».

Dans notre application, on donne la possibilité à l'utilisateur de régler l'horloge des machines avec deux méthodes différentes :

1. Une machine va régler l'heure des différentes machines du serveur.
2. Chaque machine demandera à se faire régler son heure à l'aide d'un «**Cron**».

1.1 Concepts théoriques

1.1.1 Protocole *NTP* :

Le NTP est défini comme un protocole de synchronisation de plusieurs horloges d'un réseau à l'aide d'un ensemble de clients et de serveurs distribués. Il met à disposition des mécanismes de protocole fondamentaux et nécessaires à la synchronisation de l'heure de différents systèmes jusqu'à une précision de l'ordre de la nanoseconde. D'autre part, il contient des dispositions visant à spécifier la précision et les sources d'erreur possibles de l'horloge système locale ainsi que les propriétés de l'heure de référence. [1]

1.1.2 Programme *Cron* :

cron est un programme qui permet aux utilisateurs des systèmes Unix d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée à l'avance, ou selon un cycle défini à l'avance. [2]

1. https ://www.ionos.fr/digitalguide/serveur/know-how/network-time-protocol
2. https ://fr.wikipedia.org/wiki/Cron

Chapitre 2

Installation et configuration

2.1 Installation des différentes dépendances

Avant de lancer l'application, il vous faudra installer les différentes dépendances.
Pour ce faire, vous devrez exécuter le script **installationPython** dont voici la commande :

```
./installationPython
```

Une fois l'environnement installé, il faudra activer celui-ci en exécutant cette commande :

```
source ~/pythonsProjet-SysG5_TimeSetup/python3.6.3/bin/activate
```

2.2 Modification des différentes adresses IP

En fin, il vous faudra insérer les «**adresses IP**» des machines du serveur dans le fichier **ressources/src/ipAdresses** sans préciser celle de la machine qui va lancer le script comme ceci :

/Projet − SysG5_TimeSetup/ressources/src/ipAdresses

```
192.168.6.130
192.168.6.131
192.168.6.132
192.168.6.133
```

2.3 Déreglement de l'heure

Afin que les résultats de notre application soit visible, il est préférable d'ajuster l'heure des autres machines en utilisant cette commande :

```
sudo date -s "2 OCT 2006 10:00:00"
```

2.4 Lancement de l'application

Une fois tout installés et préparés, il ne vous reste plus qu'à exécuter la commande ci-dessous qui permettra de lancer l'application :

```
python ressources/scriptPython/TimeSetup.py
```

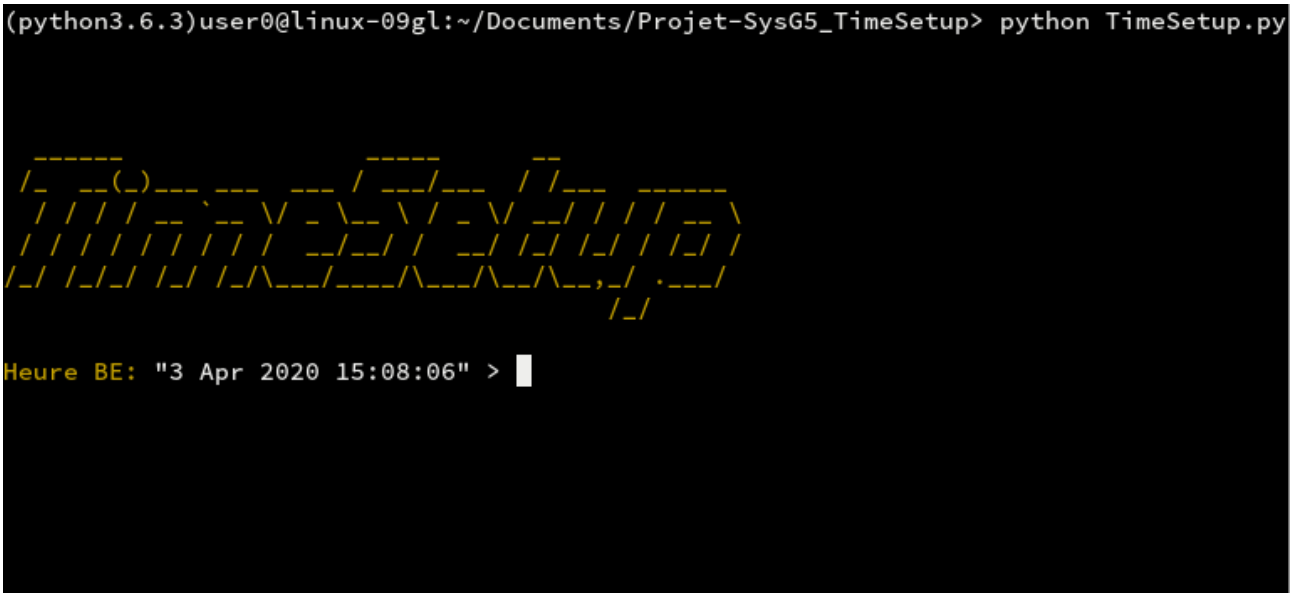
Chapitre 3

Vue de l'application

Après toutes les différentes installations et configurations.

3.1 Lancement de l'application

Voici la vue au démmarage de l'application :



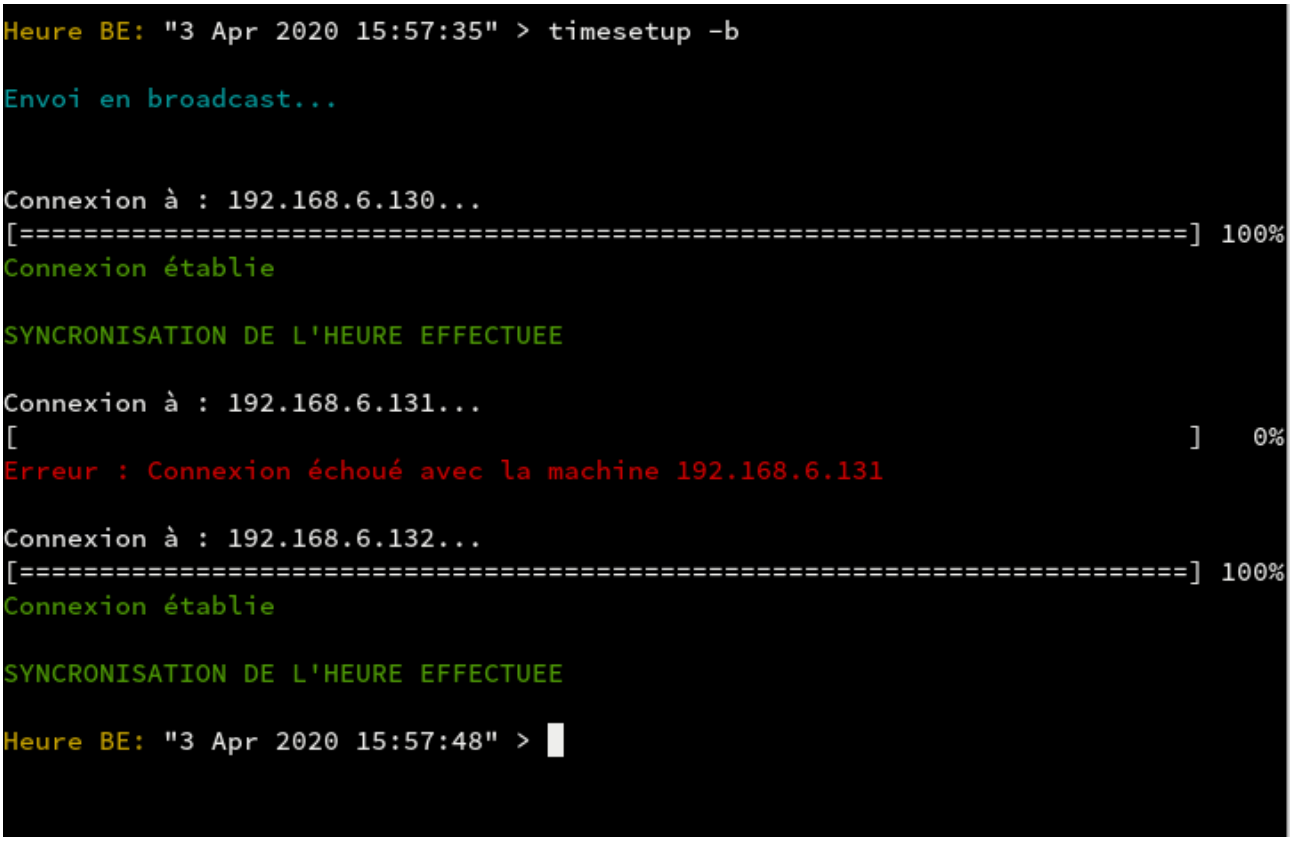
3.2 Aide

Affichage de l'aide :

```
Heure BE: "3 Apr 2020 15:20:45" > timesetup -h

NAME
         timesetup - the stupid hour manager

         clear - clear the terminal screen

SYNOPSIS
         timesetup [-b] [--broadcast] [-c] [--configure] [-h] [--help] [-e] [--exit]

         clear

OPTIONS
        -b, --broadcast
          Règle l'heure de toutes les machines du serveur, dont l'adresse IP se trouve dan
s le fichier "ipAdresses" à partir de votre machine.

        -c, --configure
          Ajoute un script qui règle l'heure sur chaque machines du serveur, dont l'adress
e IP se trouve dans le fichier "ipAdresses" à partir de votre machine afin qu'un Cron se la
nce chaque jour à 8:45.

        -h, --help
          Afficher l'aide.

        -e, --exit
          Quitter.

AUTHORS
        42933 Achetouan Mohammed.
        45682 Kardillo Younes.

Heure BE: "3 Apr 2020 15:20:45" >
```

3.3 Broadcast

Affichage après exécution de la méthode broadcast :



3.3.1 Explication :

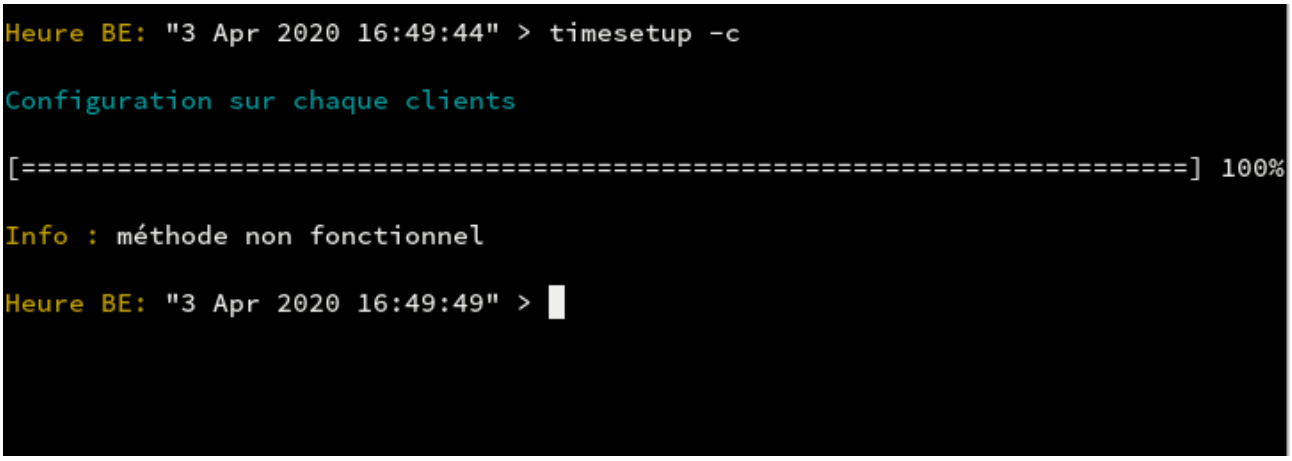
Vous pouvez observer sur l'image, que l'application travaille avec 3 machines.

1. 192.168.6.130 : *Allumée*
2. 192.168.6.131 : *Eteinte*
3. 192.168.6.132 : *Allumée*

La synchronisation a été faite sur les deux machines allumées tant dit que sur celle éteinte, l'application n'a pas pu établir une connexion. Un message d'erreur s'est donc affiché.

3.4 Configuration

Affichage après exécution de la méthode configuration :



3.4.1 Explication :

Par manque de temps, cette méthode ne fait rien de spéciale.

Chapitre 4

Fonctionnement de l'application

Dans cette section, nous nous intéresserons au fonctionnement de l'application, plus particulièrement aux deux méthodes permettant de régler l'horloge des machines.

4.1 Méthode 1 : Envoie par broadcast

Lorsque vous entrez la commande suivante :

```
timesetup -b
```

Ou bien

```
timesetup --broadcast
```

Vous réglerez l'heure de toutes les machines du serveur, dont l'adresse IP se trouve dans le fichier *ressources/src/ipAdresses* à partir de votre machine.

4.2 Méthode 2 : Configuration sur chaque client

Lorsque vous entrez la commande suivante :

```
timesetup -c
```

Ou bien

```
timesetup --configure
```

Vous ajoutez un script qui règle l'heure sur chaque machines du serveur, dont l'adresse IP se trouve dans le fichier *ressources/src/ipAdresses* afin qu'un «**Cron**» se lance chaque jour à 8 :45.

4.3 Aide

Pour afficher l'aide :

```
timesetup -h
```

Ou bien

```
timesetup --help
```

4.4 Vider l'affichage

Pour vider l'invite de commande :

```
clear
```

4.5 Quitter l'application

Pour quitter l'application, il vous suffit de taper la commande suivante :

```
timesetup -e
```

Ou bien

```
timesetup --exit
```

Chapitre 5

Explications du fichier TimeSetup.py

Dans cette section, nous expliquerons les variables et les fonctions du fichier *TimeSetup.py*.

5.1 Code source

```
import paramiko
import ntplib
import progressbar
import os

from scp import SCPClient
from time import sleep
from time import ctime
from termcolor import colored

#Variables globales
username = "user0"
password = "user0"
root_password = "Labo503"
adressIpFilepath = "ressources/src/ipAdresses"
scriptNtp = "ressources/src/scriptNtp"




#Cette fonction récupére l'heure local via NTP
def setupTime():
    try:
        c = ntplib.NTPClient()
        response = c.request('pool.ntp.org', version=3)
        ts = ctime(response.tx_time)
        liste = ts.split(' ')
        return "\""+liste[3]+' '+liste[1]+' '+liste[5]+' '+liste[4]+"\""
    except:
        print(colored("\nErreur :",'red')+" Impossible de récupérer l'heure avec le
            protocole NTP")
        exit()



#Récupération de l'heure local via NTP
heure = setupTime()


#Cette fonction crée un crone et le répend dans les différentes machines
def configuration():

    #Modification du script ntp en y ajoutant le bon root password
    f = open(scriptNtp,"w+")
```

```
    f.write("#!/bin/bash\n\necho \""+root_password+"\" | sudo -S service ntpd stop\necho
        \""+root_password+"\" | sudo -S ntpdate be.pool.ntp.org\necho \""+root_password
        +"\" | sudo -S service ntpd start")
    f.close()


    with open(adressIpFilepath) as fd:
        lines = fd.read().splitlines()
        for line in lines:
            try:
                #Connexion au clients
                print("\nConnexion à : "+line+"...")
                bar = progressbar.ProgressBar(maxval=10, \
                    widgets=[progressbar.Bar('=', '[', ']'), ' ', progressbar.Percentage
                        ()])
                bar.start()
                ssh_client = paramiko.SSHClient()
                ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                ssh_client.connect(hostname = line, username = username, password =
                    password)

                #Affichage de la bar de chargement
                for i in range(10):
                    bar.update(i+1)
                    sleep(0.1)
                bar.finish()

                #Envoi du fichier scriptNtp
                scp = SCPClient(ssh_client.get_transport())
                scp.put('ressources/src/scriptNtp')
                scp.close()

                #Ajout de la tache de syncronisation de l'heure au crontab des
                    différentes machines
                transport = ssh_client.get_transport()
                session = transport.open_session()
                session.set_combine_stderr(True)
                session.get_pty()

                #session.exec_command("echo \"40 8 * * * ./scriptNtp\" | crontab -")
                session.exec_command("echo \"* * * * * ./scriptNtp\" | crontab -")

                print(colored("SYNCRONISATION DE L'HEURE EFFECTUEE",'green'))
            except:
                print(colored("\nErreur :",'red')+" Connexion échoué avec la machine "+
                    line)


#Cette fonction se connecte en ssh au différentes machines et règle l'heure de ceux-ci.
def broadcast():
    with open(adressIpFilepath) as fd:
        lines = fd.read().splitlines()
        for line in lines:
            try:
                #Connexion au clients
                print("\nConnexion à : "+line+"...")
                bar = progressbar.ProgressBar(maxval=10, \
                    widgets=[progressbar.Bar('=', '[', ']'), ' ', progressbar.Percentage
                        ()])
                bar.start()
                ssh_client = paramiko.SSHClient()
                ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                ssh_client.connect(hostname = line, username = username, password =
                    password)
                for i in range(10):
                    bar.update(i+1)
                    sleep(0.1)
                bar.finish()

                print(colored("Connexion établie\n",'green'))
```

```
                transport = ssh_client.get_transport()
                session = transport.open_session()
                session.set_combine_stderr(True)
                session.get_pty()

                #Lancement des commandes
                heure = setupTime()
                session.exec_command("sudo date -s "+heure)
                stdin = session.makefile('wb', -1)
                stdout = session.makefile('rb', -1)

                #Ecrit le password root
                stdin.write(root_password +'\n')
                stdin.flush()
                print(colored("SYNCRONISATION DE L'HEURE EFFECTUEE",'green'))
                #Affichage de la console client
                #for lines in stdout.read().splitlines():
                    #print('%s' % lines)
            except:
                print(colored("\nErreur :",'red')+" Connexion échoué avec la machine "+
                    line)



#Cette fonction  affiche l'aide de l'application.
def displayHelp():
    print('\033[1m' + "\nNAME" + '\033[0m' + "\n\t timesetup - the stupid hour manager\n\
        n\t clear - clear the terminal screen ")
    print('\033[1m' + "\nSYNOPSIS" + '\033[0m' + "\n\t "+'\033[4m'+"timesetup"+'\033[0m
        '+" [-b] [--broadcast] [-c] [--configure] [-h] [--help] [-e] [--exit]\n\n\t
        "+'\033[4m'+"clear"+'\033[0m')
    print('\033[1m'+"\nOPTIONS"+'\033[0m')
    print("\t-b, --broadcast\n\t   Règle lheure de toutes les machines du serveur, dont l
        'adresse IP se trouve dans le fichier \"ipAdresses\" à partir de votre machine.\n
        ")
    print("\t-c, --configure\n\t   Ajoute un script qui règle l'heure sur chaque machines
         du serveur, dont l'adresse IP se trouve dans le fichier \"ipAdresses\" à partir
        de votre machine afin qu'un "+'\033[1m'+"Cron"+'\033[0m'+" se lance chaque jour à
         8:45.\n")
    print("\t-h, --help\n\t   Afficher l'aide.\n")
    print("\t-e, --exit\n\t   Quitter.")
    print('\033[1m' + "\nAUTHORS" + '\033[0m' + "\n\t42933 Achetouan Mohammed.\n\t45682
        Kardillo Younes.")









#Cette fonction affiche le nom du projet avec style.
def displayTitle():
    print("\n")
    print(colored("  ______                    _____       __ ",'yellow'))
    print(colored(" /_  __(_)___ ___  ___ / ___/___  / /___  ______ ",'yellow'))
    print(colored("  / / / / __ `__ \\/ _ \\__ \\ / _ \\/ __/ / / / __ \\ ",'yellow'))
    print(colored(" / / / / / / / / /  __/__/ /  __/ /_/ /_/ / /_/ /  ",'yellow'))
    print(colored("/_/ /_/_/ /_/ /_/\\___/____/\\___/\\__/\\__,_/ .___/ ",'yellow'))
    print(colored("                                         /_/  ",'yellow'))

#Cette fonction permet de vider le terminal.
def clear():
    os.system('clear')


#Cette fonction lance l'application et demande à l'utlisateur quelle méthode il souhaite
    utiliser.
```

```
def main():
    displayTitle()
    while True:
        heure = setupTime()
        value = input(colored("\nHeure BE: ",'yellow')+ heure + " > ")
        if (value == "timesetup -b" or value == "timesetup --broadcast" ):
            print(colored("\nEnvoi en broadcast...\n",'cyan'))
            broadcast()

        elif(value == "timesetup -c" or value == "timesetup --configure" ):
            print(colored("\nConfiguration sur chaque clients\n",'cyan'))
            configuration()

        elif(value == "timesetup -h" or value == "timesetup --help" ):
            clear()
            displayHelp()

        elif(value == "clear"):
            clear()

        elif (value == "timesetup -e" or value == "timesetup --exit" ):
            print(colored("\nDéconnexion...",'red'))
            break
        else:
            print(colored("\nErreur :",'red')+" '"+value+"': n'est pas une commande
                timesetup. Voir 'timesetup --help'")

    print("\n")



#Lancement du main
main()
exit()
```

5.2 Les variables globales

— *username* : représente le nom de l'utilisateur.
— *password* : représente le mot de passe de l'utilisateur.
— *root_Password* : représente le mot de passe du SuperUser.
— *adressIpFilepath* : représente le path du fichier contenant la liste d'adresse IP.
— *heure* : représente l'heure retourner par la méthode setupTime().

5.3 Les fonctions

setupTime() :

Cette fonction permet de récuperer l'heure local via le protocole NTP.

configuration() :

Pour le moment cette fonction ne fait rien de spéciale.

broadcast() :

Cette fonction récupere les différentes adresses IP du fichier **ressources/src/ipAdresses**. Ensuite elle essaye d'établir une connexion avec les machines grâce au protocole de communication SSH. Une fois la connexion établie, elle lance une multitude de commandes permettant de synchroniser l'heure sur les différentes machines.

Pour chaque machine, si la connexion a pu être établie et que la synchronisation a bien été éffectuée alors un message de confirmation est affiché, sinon un message d'erreur apparait expliquant que la connexion a échoué.

displayTitle() :

Cette fonction affiche le nom du projet avec style.

displayHelp() :

Cette fonction affiche l'aide de l'application.

clear() :

Cette fonction permet de vider le terminal.

main() :

Cette fonction lance l'application et demande à l'utlisateur quelle méthode il souhaite utiliser.

Chapitre 6

Conclusion

Ce projet s'est révélé très enrichissant. En effet, nou avons développé notre prise d'initiative, le respect des délais et le travail en équipe qui sont des aspects essentiels pour notre futur vie de développeur. De plus, il nous a permis d'appliquer nos connaissances en recherche et en programmation dans un domaine pratique.

Les principaux problèmes, que nous avons rencontrés, sont :

- l'adaptation au système d'exploitation *OpenSuse*.
- l'installation d'un environnement *Python*3.6.3 compatible à nos besoins
- le paramétrage du protocole de communication *SSH* entre les machines virtuelles
- le manque de matériel (manque de machines pour tester notre application)
- perte de temps majeur dû à la configuration des machines virtuelles.

Toutefois, nous avons fait le maximum pour mener à bien ce projet et le rendre agréable pour les utilisateurs.