Réalisation d'un utilitaire d'archivage

Mémoire technique

BIER Cyril – BOUASRIA Ayoub – EL GUERMAT BOUCHAMA Mohamed

Table des matières

I – Introduction 2

II – Architecture du système 2

A- Serveur web 2

B- Système local 2

C- Serveur de destination pour archivage 3

III – Fonctionnement de l'utilitaire 3

A- Fichier de configuration 3

B- Récupération de l'archive via HTTPS 3

C- Contrôle de modifications 4

D- Renommage 4

E- Envoi d'une archive via SFTP + Historisation 4

F- Envoi de mail via SMTPS 5

G- Programme principal 5

H- Scripts d'installation des prérequis **5**

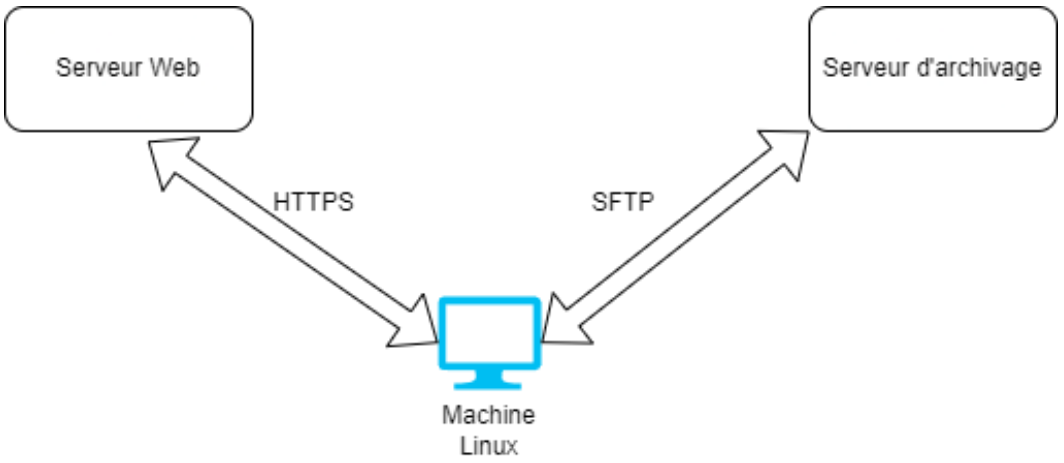
1) Dépendances 5

2) Création d'un serveur web 6

IV – Automatisation du lancement 6

I – Introduction

Ce document a pour objectif de présenter et justifier les différents choix techniques permettant de créer l'utilitaire relatif à ce projet. L'utilitaire créé permet de télécharger, de manière automatique, l'archive .zip d'un fichier .sql sur un serveur web, de la dézipper, de vérifier que le contenu du fichier est différent du contenu d'un fichier téléchargé précédemment et si celui-ci est différent d'archiver le nouveau fichier au format .tgz sur un autre serveur distant. Le lancement de cet utilitaire doit se faire de manière automatisée. Ce document présentera ainsi l'environnement dans lequel sera utilisé l'utilitaire et le fonctionnement général du programme.



II – Architecture du système

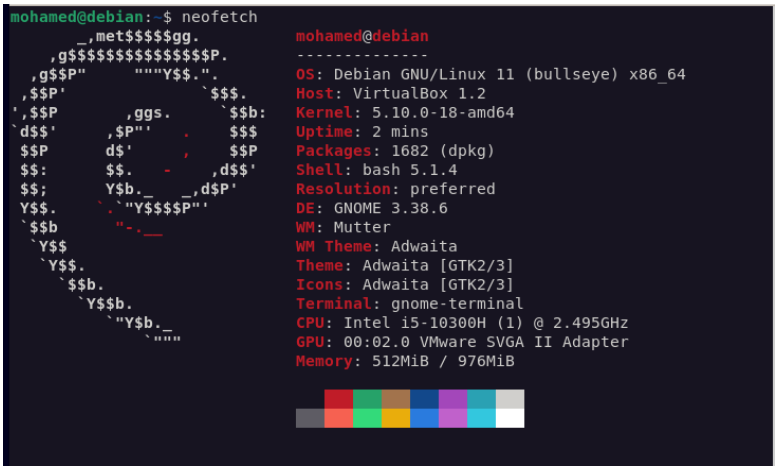
Cette partie présente les différents choix fait concernant l'architecture du système : le serveur source, la machine sur laquelle est exécutée le programme et le serveur utilisé pour l'archivage.

A- Serveur web

Le serveur source, contenant l'archive .zip, est un serveur web utilisant la technologie Apache2. Cette technologie est open-source, facile à mettre en place et stable. Son principal défaut réside dans sa capacité à gérer un nombre important de connexions simultanés (+ 10 000), dans le cadre de ce projet cela ne nous pose donc pas de problèmes. Pour ce projet il est demandé que le serveur soit un serveur accessible en https. Il a donc été nécessaire de mettre en place un chiffrement SSL. Pour réaliser ce chiffrement nous avons notamment généré un certificat d'authenticité localement. Ce certificat n'est donc pas reconnu par les navigateurs qui considèrent ainsi que le serveur n'est pas totalement sécurisé.

B- Système local

La machine sur laquelle est exécutée l'utilitaire est une machine utilisant la distribution Linux Debian 11.



Afin de fonctionner l'utilitaire a besoin que plusieurs packages tels que python3, apache, unzip et d'autres soient installés sur cette machine. Un script bash est fourni avec l'utilitaire afin d'installer toutes les dépendances nécessaires à son bon fonctionnement.

C- Serveur de destination pour archivage

Le serveur utilisé pour l'archivage sera quant à lui accessible en SSH via le protocole SFTP. Ce protocole permet d'accéder de manière sécurisée à une machine distante pour le transfert de fichier. Cela permet ainsi de récupérer des fichiers dessus, mais aussi d'en ajouter ou supprimer, ce qui correspond à notre besoin. Pour utiliser ce protocole nous avons utilisé l'outil libre openssh. Pour que l'utilitaire fonctionne, il est nécessaire de se connecter manuellement en SFTP au serveur d'archivage lors de la première utilisation (procédure décrite dans le fichier Guide_installation_utilisation.pdf).

III – Fonctionnement de l'utilitaire

Sur le plan logiciel, l'utilitaire est réalisé en langages de script Bash et Python. L'utilitaire est composé de plusieurs scripts implémentant chacun une fonctionnalité. Ces scripts sont ensuite tous utilisés dans un programme principal.

A- Fichier de configuration

La configuration des différentes options proposées par l'utilitaire est possible via un fichier de configuration qui pourra être configuré par un technicien. Les différents champs modifiables sont explicités dans le guide d'installation et d'utilisation.

Ce fichier de configuration est au format json qui est plutôt simple à lire et il est facile d'en extraire les informations via Python. Nous avons ainsi créé un script read_configuration.py qui permet d'extraire tous les champs du fichier de configuration pour les utiliser dans les scripts suivants à l'aide de la librairie json.

B- Récupération de l'archive via HTTPS

Un premier script implémente la récupération de l'archive .zip sur un serveur web en HTTPS. Ce script implémente également le dézippage de l'archive. Le script a été écrit en bash car il permet de télécharger facilement le fichier depuis le serveur web avec wget. Wget permet également de

contourner la vérification de notre certificat non reconnu par les navigateurs. Pour le dézippage, on utilise la commande unzip pour sa simplicité.

C- Contrôle de modifications

Un script Python, implémente la fonction permettant de contrôler si deux fichiers sont différents, on retrouve également dans ce script les fonctions permettant la compression et la décompression d'un fichier au format tar.

Ainsi, le fichier modification_zip.py implémente une fonction "modification" qui compare deux fichiers et indique s'ils sont identiques via un booléen. La comparaison s'effectue ligne à ligne, donc seul le contenu est comparé (une différence dans le nom des fichiers n'est donc pas prise en compte).

D- Renommage

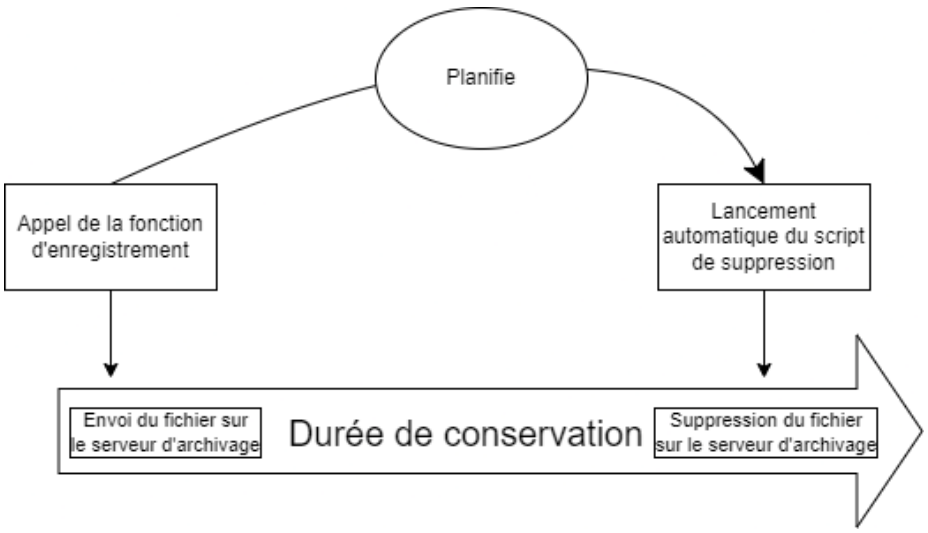
Un fichier rename.py implémente une fonction permettant de renommer automatiquement à la date du jour un fichier tout en conservant son extension. Ce script utilise pour cela la bibliothèque Python datetime pour récupérer la date et la bibliothèque OS pour renommer le fichier.

E- Envoi d'une archive via SFTP + Historisation

Le fichier link_stfp.py implémente les fonctions permettant de se connecter au serveur d'archivage en SFTP pour télécharger, envoyer ou supprimer un fichier. Ce script importe la bibliothèque pysftp permettant l'utilisation de SFTP.

Ces fonctions sont utilisées dans la fonction enregistrement du script historisation.py. Le but de cette fonction est d'enregistrer le fichier sur le serveur distant mais également de planifier sa suppression en fonction de la durée de conservation configurée par le technicien.

Pour cela, la fonction enregistrement appelle la fonction d'envoi de fichier via SFTP puis va planifier le lancement du script suppression_stp.py qui lui-même appelle la fonction de suppression de fichier par SFTP. On utilise pour cela des commandes Linux via la bibliothèque Python OS. Pour la planification on utilise la commande Linux at.



F- Envoi de mail via SMTPS

L'envoi du mail se fait via la bibliothèque smtplib sur python et le chiffrement du mail se fait à l'aide de la bibliothèque SSL. On peut personnaliser les destinataires, l'objet, le contenu du mail et ajouter une pièce jointe. Ces paramètres sont gérés depuis le fichier de configuration

G- Programme principal

Le fichier "main.py" est le script utilisant toutes les fonctions présentées plus haut afin d'enchaîner toutes les étapes et mettre en place le fonctionnement de l'utilitaire. Ce programme peut être divisé en 4 parties.

- Dans un premier temps, le programme main fait appel au script file_apache.sh qui s'occupe de télécharger l'archive .zip présente sur le serveur web et de la décompresser, ensuite la fonction get_file du fichier link_sftp.py est utilisée pour télécharger la dernière archive présente sur le serveur SFTP et la décompresser. Si les fichiers ont été téléchargés sans difficultés, la deuxième étape débute.

- Les deux fichiers sql sont ensuite comparés ligne à ligne à l'aide de la fonction modification du fichier modification_zip.

- Si le fichier provenant du serveur web est une version modifiée de celle stockée en SFTP alors il est renommé avec la date du jour puis est compressé sous format ".tgz". (Dans le cas où le fichier n'aurait pas été modifié, cela sera précisé dans le rapport de log et aucune action ne sera réalisée). De même si le serveur SFTP ne contient aucun fichier stocké (dans le cas où le serveur SFTP vient d'être mis en place, ou à cause d'une perte de la sauvegarde) le fichier du serveur Apache sera immédiatement stocké sur le serveur SFTP. L'enregistrement s'effectue à l'aide de la fonction enregistrement située dans le fichier "historisation.py".

- La dernière étape consiste à l'envoi du mail. Si l'option d'envoi de mail est choisie dans le fichier de configuration, alors la fonction mail_send du fichier "mail.py" est appelée et un mail de validation est envoyé. Si une erreur est survenue durant les différentes opérations un mail d'erreur est envoyé avec la possibilité d'ajouter le fichier log du jour.

H- Scripts d'installation des prérequis

1) Dépendances

Le script d'installation (bash) appelée "dependances.sh" qui gère les dépendances doit être exécuté par l'utilisateur lors de la mise en place du système d'archivage. Ce script permet également l'installation des différents serveurs : le serveur web Apache, le serveur OpenSSH/OpenSSL ainsi que le serveur SFTP.

Il permet également l'installation de python et de python-pip qui gère les librairies et les dépendances de python.

Le paquet wget installé par le script est utilisé pour le téléchargement des fichiers présent sur le serveur web et le paquet unzip permet d'effectuer une décompression des fichiers.

2) Création d'un serveur web

La création du serveur web se fait via apache2 et est géré avec les autres dépendances. Pour upload le fichier, il suffit de suivre la procédure indiquée dans le guide d'installation et d'exécuter le fichier publish_WebServer.sh.

IV – Automatisation du lancement

Le programme principal doit pouvoir s'exécuter de manière automatique et périodique. Nous utilisons pour cela Crontab via le script automatisation_crontab.sh qui écrit, directement dans le fichier cron, l'ordre d'exécution du fichier main.py à la période indiquée dans le fichier de configuration. En revanche ce script supprime également les anciennes instructions crontab, il faut donc l'exécuter en premier avant d'ajouter d'autres tâches crontab (cette procédure est détaillée dans le guide d'utilisation).