rsyncFTP

1.2

Athors : Julien Vermeil, Nicolas Sobczak, Vincent Reynaert

Contents

Chapter 1

Présentation

rsyncFTP est un programme permettant de créer un dossier miroir sur un serveur ftp distant.

Le principe de ce programme est de mettre à jour le dossier situé sur le serveur distant en fonction de l'état du dossier situé en local une machine. Pour synchoniser ce fichier, il faut surveiller le dossier local et réaliser les actions nécessaires à la mise à jour du dossier miroir.

Ce programme a été réalisé en python. La version utilisée est la version 3.5.2.

Ce projet est disponible en ligne sur github à l'adresse suivante : `https://github.com/nsobczak/rsyncFTP`

Chapter 2

Choix d'organisation

Nous avons voulu séparer le plus possible les fonctionnalités en créant des fichiers différents:

- logger.py
- parserRsyncFTP.py
- gestionFTP.py
- directorySupervisor.py
- main.py

A la fin de chaque fichier se trouvent des tests unitaires.

2.1 logger.py

Le package logger contient la gestion du logger. C'est ici qu'on s'occupe de créer le logger. Il est possible d'utiliser un fichier de configuration *.conf* pour le faire. Il est également possible de créer ce fichier dans un dossier spécifique, ou encore de laisser le programme gérer le logger lui-même.

2.2 parserRsyncFTP.py

Le package parserRsyncFTP contient la fonction de définition de gestion du parser. C'est ici que nous définissons les paramètres que nous passons en lignes de commandes. Nous avons également ajouté une description pour le programme et ses paramètres.

2.3 gestionFTP.py

Le package gestionFTP contient toutes les fonctions utiles pour gérer les actions basiques avec le serveur FTP. Elle est basée sur la librairie python ftplib.

2.4 directorySupervisor.py

Le package directorySupervisor correspond à la supervision des dossiers. Il s'agit des fonctions du tp1 que nous avons adaptées pour rsyncFTP. Nous n'en avons repris que le coeur de façon à garder une certaine souplesse par rapport à ce que nous avions déjà réalisé. C'est à dire que nous l'avons amélioré notamment en supprimant les variables globales qui était dangereuses.

Cependant, il y a toujours des cas où ce module ne fonctionne pas. Nous n'avons pas su mettre en évidence ces cas. Nous pensons que de réaliser trop d'actions dans une courte période peut générer des problèmes.

2.5 main.py

Le package main contient les fonctions d'initialisation du programme et les fonctions principales. On trouve ainsi les fonctions de supervision du dossier local et de mise à jour du dossier miroir.

La fonction *monMain()* est le coeur du programme. Nous avons chercher à la réduire à l'essentiel de façon à ce qu'elle reste lisible et compréhensible au premier coup d'oeil.

Les quelques étapes que nous réalisons sont les suivantes :

- Nous définissons notre parser.
- Nous initialisons le logger.
- Nous lançons ensuite notre boucle principale qui supervise le dossier et synchornise le répertoire situé sur le serveur FTP distant.

Chapter 3

Choix techniques

La version de python utilisée est la 3.5.2. Le programme a été testé sur avec les serveurs FTP :

- *Filezilla* (version ..) pour windows
- *Proftpd* pour linux, avec son interface graphique Gadmin ProFTPD (version 0.4.2)

Pour utiliser ce programme, il est nécessaire d'installer les bibliothèques suivantes :

- os
- os.path
- ftplip
- argparse
- logging
- logging.config
- time

3.1 Synchronisation

Pour synchroniser le dossier miroir, nous avons choisi de surveiller notre dossier local et alors de mettre à jour le dossier miroir en cas de modification.

Au lancement du programme, si le serveur FTP contient déjà un dossier avec le nom de celui dont on souhaite réaliser le miroir, nous le supprimons. Nous copions ensuite le dossier que l'ont veut superviser sur le serveur FTP.

3.2 Paramètres

Nous avons choisi de passer en ligne de commande les paramètres suivants:

Paramètre	Type	Variable
donnees pour le site FTP distant: hote, identifiant, mot de passe| ex : localhost nsobczak ISEN	obligatoire	ftp
chemin vers le dossier local (directory path)	obligatoire	dp
liste des fichiers à inclure puis celle à exclure, * pour tout selectionner, ',' pour séparer les extensions | ex : * .odt,.docx pour dire de tout inclure sauf les .odt et .docx	obligatoire	ie
chemin ou generer le fichier log	optionnel	"–lp", "–logPath"
chemin vers le fichier conf du log (gestion des handler)	optionnel	"-lc", "–logConf"
profondeur de la supervision du dossier, default = 2	optionnel	"-p", "–profondeur"
taille maximale des fichiers transferes en Mo, default = 500 Mo	optionnel	"-sf","–sizeFile"
frequence de supervision en s, default = 1 s	optionnel	"-f", "–frequence"
temps de supervision en s, -1 pour infini, default = 60 sec	optionnel	"-st", "–supervisionTime"

3.3 Gestion des extensions de fichiers à inclure/exclure

Les fichiers à inclure/exclure entrés en paramètres sont pris en compte lors l'ajout/modifaction/suppression de fichiers. Ils ne sont pas pris en compte lors de la copie de fichiers. Ils ne sont donc pas non plus pris en compte lors de la copie initiale du dossier.

Le problème vient du fait que lors de la copie d'un dossier nous utilisons la fonction de la bibliothèque gestionFTP dans laquelle nous n'avons pas pris en compte les listes d'extensions de fichiers à inclure/exclure.

3.4 Fichier log

Concernant le fichier de log, si aucun chemin pour enregistrer le fichier n'est précisé, nous utilisons le fichier *.conf* qui nous définit des handlers proprement et enregistre le fichier log dans le répertoire du projet. Si un chemin est précisé, nous n'utilisons pas le fichier *.conf*. Nous créons le logger dans le code du fichier logger.py. Le fichier rsyncFTP.log est alors enregistré dans le répertoire précisé par le chemin entré en ligne de commande. Il est recommandé d'utiliser le fichier *.conf* qui crée également un fichier de log de debug.

3.5 Librairie gestionFTP

Dans la librairie *gestionFTP*, nous avons choisi de définir des fonctions permettant de réaliser des actions basiques telles que :

- se connecter/deconnecter au seveur ftp
- lister les éléments présents sur le serveur ftp
- créer un fichier sur le seveur ftp
- transferer un fichier vers le seveur ftp

- effacer un fichier sur le seveur ftp
- créer un dossier sur le seveur ftp
- transferer un dossier vers le seveur ftp
- effacer un dossier sur le seveur ftp

Nous pouvons alors executer ces actions pour réaliser des fonctions plus complexes.

Nous avons rencontré une difficulté concernant la fonction qui supprime un dossier. Elle est un peu similaire à celle qui copie un dossier à ceci près qu'elle que l'on rencontre un probleme lorsque l'on veut supprimer un dossier ou un fichier car il n'existe plus en local. On ne peut donc pas vérifier si ce qu'on veut supprimer sur le serveur ftp est un fichier ou un dossier. En outre, pour supprimer un dossier, nous devons en supprimer tout son contenu. Il nous a donc fallu créer une fonction récursive pour vider les dossiers avant de les supprimer.

Avec les fonctions de cette librairie, il faut faire attention à la forme des chemins que l'on doit passer en paramètre. En effet, certaine fonctions prenne en paramètre des chemin absolus tandis que d'autre prenne des chemins relatifs. De surcroit, il est parfois nécessaire d'avoir le nom du fichier ou répertoire à la fin du chemin et parfois il ne faut pas le mettre.

Nous aurions du discuter des chemins au préalable pour définir une convention. Il est en effet très embêtant de devoir réfléchir au type de chemin à chaque utilisation de fonction.

3.6 Librairie directorySupervisor

Pour pouvoir superviser un dossier, nous utilisons l'organisation des répertoire en "arbre". Nous avons choisi de ne pas créer de classe arbre mais plutôt de matérialiser un arbre avec une liste.

Nous créons donc d'abord l'arbre du dossier à chaque fois qu'il s'écoule une période correspondant à la fréquence de supervision. Puis nous comparons cette arbre avec l'arbre précédemment créé. Si on observe une modification, on l'écrit dans le fichier de log et on va signaler le type de modification effectuée afin de pouvoir mettre à jour le dossier situé sur le serveur distant.

Les modifications sont retournées dans des listes. Il y en a une pour les ajouts, une pour les suppressions et une pour les modifications. Chaque liste contient des tuples, un pour chaque modification effectuée. Ainsi, si 2 ajouts de fichiers sont effectuée, la liste d'ajout contiendra 2 tuples alors que les 2 autres listes seront vides. Un tuple contient le chemin absolu vers l'élément modifié et le moment quand la modification a été effectuée.

Cette librairie ne fonctionne pas pour certains cas que nous avons décidé d'exclure. Ces cas sont les suivants:

- fichiers sans extension tels que les fichiers binaires
- fichiers et dossiers avec des caracteres spéciaux dans le nom tels que les accents

Chapter 4

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 5

Namespace Documentation

5.1 directorySupervisor Namespace Reference

Functions

- def createSurveyList (tree, startinglevel, depth)
- def comparateSurveyList (oldListe, newListe)
- def logTheMADLists (logger, M, A, D)
- def loop (logger, frequence, supervisionTime, arbrePrecedent, dp, startinglevel)
- def **monMain** ()

5.1.1 Detailed Description

```
############
# rsyncFTP #
########################
# directorySupervisor #
########################

@author: Julien Vermeil and Vincent Reynaert and Nicolas Sobczak
```

5.1.2 Function Documentation

5.1.2.1 def directorySupervisor.comparateSurveyList (*oldListe, newListe*)

```
Fonction qui compare 2 listes :
:param oldListe: old list
:type oldListe: list
:param newListe: new list
:type newListe: list
:return: tuple of list for the modified files, list for the added files, list for the deleted files
:rtype: tuple
```

5.1.2.2 def directorySupervisor.createSurveyList (*tree, startinglevel, depth*)

```
Function which create a list of tupples form by (fileName, dateOfLastModif) corresponding to the files in the
:param tree: tree
:type tree: list
:param startinglevel: starting level to compute the current depth
:type startinglevel: int
:param depth: depth of supervision
:type depth: int
:return listOfModifFiles: list for he deleted files
:rtype listOfModifFiles: list
```

5.1.2.3 def directorySupervisor.logTheMADLists (*logger, M, A, D*)

```
Function that log information contained in the 3 lists :
:param M: modified files
:type M: list
:param A: added files
:type A: list
:param D: deleted files
:type D: list
:param logger: logger
:type logger: log
```

5.1.2.4 def directorySupervisor.loop (*logger, frequence, supervisionTime, arbrePrecedent, dp, startinglevel*)

```
Fonction: si stop() => arret, sinon compareArbre()
:param logger: logger
:type logger: log
:param frequence: frequence de supervision
:type frequence: int
:param supervisionTime: temps de supervision, si -1 alors infini
:type supervisionTime: int
:param arbrePrecedent: tree arbre precedent
:type arbrePrecedent: list
:param dp: chemin du dossier
:type dp: str
:return: 1
:rtype: int
```

5.2 gestionFTP Namespace Reference

Functions

- def connectionAuServeurFTP (host, user, password)
- def deconnexionAuServeur (ftp)
- def affichageFTP (ftp)
- def etatConnexion (ftp)
- def listerFichiers (ftp)
- def listerDossiers (ftp)
- def listerElements (ftp)
- def envoyerUnFichier (fichier_chemin, fichier_nom, ftp)
- def supprimerFichier (ftp, fichier_chemin, fichier_nom)
- def creerDossier (ftp, dossier_chemin, dossier_nom)
- def supprimerDossier (ftp, dossier_chemin, dossier_nom)
- def copierContenuDossier (ftp, chemin_ftp, chemin_local, nom_dossier, profondeure_copie_autorisee)
- def **monMain** ()

5.2.1 Detailed Description

```
############
# rsyncFTP #
##############
# gestionFTP #
##############

@author: Julien Vermeil and Vincent Reynaert and Nicolas Sobczak
```

5.2.2 Function Documentation

5.2.2.1 def gestionFTP.affichageFTP (*ftp*)

```
Fonction qui affiche les infos (fichiers et dossiers) contenues dans le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.2 def gestionFTP.connectionAuServeurFTP (*host, user, password*)

```
Fonction qui initialise la connection au serveur FTP distant
:param host: host name
:type host: str
:param user: user id
:type user: str
:param password: password
:type password: str
:return: ftp
:rtype: class 'ftplib.FTP'
```

5.2.2.3 def gestionFTP.copierContenuDossier (*ftp, chemin_ftp, chemin_local, nom_dossier, profondeure_copie_autorisee*)

```
Fonction qui copie les fichiers d'un dossier specifie
:param ftp: serveur ftp
:type ftp:
:param chemin_ftp: chemin dans le dossier distant
:type chemin_ftp: str
:param chemin_local: chemin absolu (avec nom du dossier) du dossier
:type chemin_local: str
:param nom_dossier: nom du dossier
:type nom_dossier: str
:param profondeure_copie_autorisee: profondeur de copie autorisee
:type profondeure_copie_autorisee: int
```

5.2.2.4 def gestionFTP.creerDossier (*ftp, dossier_chemin, dossier_nom*)

```
Fonction qui cree un dossier dans le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
:param dossier_chemin: chemin relatif (sans le nom du dossier) vers le dossier local
:type dossier_chemin: str
:param dossier_nom: nom du fichier
:type dossier_nom: str
```

5.2.2.5 def gestionFTP.deconnexionAuServeur (*ftp*)

```
Fonction qui se deconnecte du serveur
:param ftp: nom de la variable dans laquelle la connexion a ete declaree
:type ftp: class 'ftplib.FTP'
```

5.2.2.6 def gestionFTP.envoyerUnFichier (*fichier_chemin, fichier_nom, ftp*)

```
Fonction qui envoie un fichier existant vers le serveur ftp
:param fichier_chemin: chemin absolu (avec le nom du fichier) vers le fichier local
:type fichier_chemin: str
:param fichier_nom: nom du fichier
:type fichier_nom: str
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.7 def gestionFTP.etatConnexion (*ftp*)

```
Fonction qui affiche l'etat de la connexion au serveur ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.8 def gestionFTP.listerDossiers (*ftp*)

```
Fonction qui liste les dossiers presents dans le repertoire observe sur le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.9 def gestionFTP.listerElements (*ftp*)

```
Fonction qui liste les fichiers et les dossiers presents dans le repertoire observe sur le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.10 def gestionFTP.listerFichiers (*ftp*)

```
Fonction qui liste les fichiers presents dans le repertoire observe sur le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
```

5.2.2.11 def gestionFTP.supprimerDossier (*ftp, dossier_chemin, dossier_nom*)

```
Fonction qui cree un dossier dans le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
:param dossier_chemin: chemin relatif vers le dossier local contenant le dossier a supprimer
:type dossier_chemin: str
:param dossier_nom: nom du dossier
:type dossier_nom: str
```

5.2.2.12 def gestionFTP.supprimerFichier (*ftp, fichier_chemin, fichier_nom*)

```
Fonction qui supprime un fichier dans le ftp
:param ftp: serveur ftp
:type ftp: class 'ftplib.FTP'
:param fichier_chemin: chemin relatif (sans le nom du fichier) vers le fichier local
:type fichier_chemin: str
:param fichier_nom: nom du fichier
:type fichier_nom: str
```

5.3 main Namespace Reference

Functions

- def init (args)
- def initialisationDossierFTP (args, logger, connectFTP)
- def donneCheminRelatif (args, chemin_element)
- def updateFTP_M (args, logger, connectFTP, M)
- def updateFTP_A (args, logger, connectFTP, A)
- def updateFTP_D (args, logger, connectFTP, D)
- def updateFTP (args, logger, connectFTP, M, A, D)
- def loop (args, logger, arbrePrecedent, startingLevel, connectFTP)
- def **monMain** ()

5.3.1 Detailed Description

```
############
# rsyncFTP #
############
# main     #
############

@author: Julien Vermeil and Vincent Reynaert and Nicolas Sobczak
```

5.3.2 Function Documentation

5.3.2.1 def main.donneCheminRelatif (*args, chemin_element*)

```
Fonction qui retourne le chemin relatif a partir des chemins absolus du dossier surveille et de l'element dont
:param args: parametres entres en lignes de commandes
:param chemin_element: chemin absolu vers l'element (fichier ou dossier) dont on veut le chemin relatif
:type args: dict
:type chemin_element: str
:return: chemin_relatif
:rtype: str
```

5.3.2.2 def main.init (*args*)

```
Initialise les variables, constantes utiles, et se connecte au serveur ftp
:param args: parametres entres en lignes de commandes
:type args: dict
:return: connectFTP, includes, excludes, arbrePrecedent, startinglevel
:rtype: tuple
```

5.3.2.3 def main.initialisationDossierFTP (*args, logger, connectFTP*)

```
Fonction qui initialise le dossier sur le serveur FTP : supprime puis copie le dossier local
:param args: parametres entres en lignes de commandes
:param logger: fichier de log
:param connectFTP: objet ftp
:type args: dict
:type logger: log
:type connectFTP: class 'ftplib.FTP'
```

5.3.2.4 def main.loop (*args, logger, arbrePrecedent, startingLevel, connectFTP*)

```
Fonction de boucle principale. Elle fonctionne en 2 actions:
- Surveiller
- Si modif, mettre a jour le serveur FTP
:param args: parametres entres en lignes de commandes
:param logger: fichier de log
:type args: dict
:type logger: log
:param connectFTP: objet ftp
:type connectFTP: class 'ftplib.FTP'


:return: 1
```

5.3.2.5 def main.updateFTP (*args, logger, connectFTP, M, A, D*)

```
Fonction qui gere la mise a jour du dossier distant
:param args: parametres entres en lignes de commandes
:param logger:
:type args: dict
:type logger: log
:param connectFTP: objet ftp
:type connectFTP: class 'ftplib.FTP'
```

5.3.2.6 def main.updateFTP_A (*args, logger, connectFTP, A*)

```
Fonction qui s'occupe de mettre a jour le dossier situe sur le serveur FTP en cas d'ajout
:param args: parametres entres en lignes de commandes
:param logger: fichier de log
:param connectFTP: objet ftp
:param A: liste des elements ajoutes
:type args: dict
:type logger: log
:type connectFTP: class 'ftplib.FTP'
:type A: list
```

5.3.2.7 def main.updateFTP_D (*args, logger, connectFTP, D*)

```
Fonction qui s'occupe de mettre a jour le dossier situe sur le serveur FTP en cas d'ajout
:param args: parametres entres en lignes de commandes
:param logger: fichier de log
:param connectFTP: objet ftp
:param D: liste des elements supprimes
:type args: dict
:type logger: log
:type connectFTP: class 'ftplib.FTP'
:type D: list
```

5.3.2.8 def main.updateFTP_M (*args, logger, connectFTP, M*)

```
Fonction qui s'occupe de mettre a jour le dossier situe sur le serveur FTP en cas d'ajout
:param args: parametres entres en lignes de commandes
:param logger: fichier de log
:param connectFTP: objet ftp
:param M: liste des elements modifies
:type args: dict
:type logger: log
:type connectFTP: class 'ftplib.FTP'
:type M: list
```

5.4 parserRsyncFTP Namespace Reference

Functions

- def initVariables ()
- def logArgs (args, logger)
- def **monMain** ()

Variables

- int **test** = 0

5.4.1 Detailed Description

```
#############
# rsyncFTP #
#############
# parser   #
#############

@author: Julien Vermeil and Vincent Reynaert and Nicolas Sobczak
```

5.4.2 Function Documentation

5.4.2.1 def parserRsyncFTP.initVariables ( )

```
Fonction qui initialise le logger et les variables en fonction de ce que
l'utilisateur a entre.
La fonction genere une info recapitulant la liste des parametres entres.
:return: ARGS
:rtype: dict
```

5.4.2.2 def parserRsyncFTP.logArgs (*args, logger*)

```
Procedure qui ecrit les parametres utilises dans le logger
:param args: parametres entres en lignes de commandes
:type args: dict
```

Index