Solutions to lab 04-06

1

See the python source code down below.

2

The backup_plan.conf file is not very eligible to dynamic systems because new servers will have new IP-adresses. Also a new pair of SSH- keys are required.

One possible solution is to have each new server sending their internal IP-address and their SSH-key over scp to the backup server so they can be stored in the configuration file.

Another, possible solution that supports dynamic thinking better, is to have one large Docker virtual machine that runs most of the servers as docker containers, especially those containers (virtual machines) that should be backuped. Then the config file on the backup server only need the IP-adress of the Docker virtual machine and no SSH- keys needed. The config file can be reorganized to store backup data based on docker instances, not virtual machines.

```
#!/usr/bin/python

#Importing libraries
import os
import shutil
import argparse

VERBOSE = 0
DEBUG = 0

ITERATIONS = 7
BACKUP_FOLDER = "/home/ubuntu//backups/"
CONFIG = "/etc/backup.conf"

SCP_USER = "ubuntu@"

########################################
# Parsing
parser = argparse.ArgumentParser(prog='pull_backup.py')
parser.add_argument('-v','--verbose',dest="verbose",help='Turn verbosity
on',default=False,action="store_true")
parser.add_argument('-d','--debug',dest="debug",help='Turn debug_messages
on',default=False,action="store_true")
parser.add_argument('-c','--config',dest="config", help='What config file to use',
metavar="FILE",default=CONFIG)
parser.add_argument('-i','--iterations',dest="iterations",type=int, help='How many
backup iterations to do', metavar="N",default=7)
parser.add_argument('-b','--backup-directory',dest="backup_folder", help="Where to
keep the backup files", metavar="FOLDER",default=BACKUP_FOLDER)
arguments = parser.parse_args()

VERBOSE = arguments.verbose
DEBUG = arguments.debug
ITERATIONS = arguments.iterations
BACKUP_FOLDER = arguments.backup_folder
CONFIG = str(arguments.config)
########################################

def verbose(text):
    if VERBOSE :
        print text

def debug(text):
    if DEBUG :
        print text

#Opening CONFIG file for reading:
# for each line store away the internal IP-adress and those folders that should be
backed up.
verbose("Opening config file: " + CONFIG);
with open(CONFIG) as config:
    for line in config:
        verbose("Read line: " + line)
        configlist = line.split(":")
        pathlist = configlist[1].split(",")
        verbose("host: " + configlist[0] )
        verbose("path: " + str(pathlist))
        host = configlist[0]

        # step 0: Check there is a backup folder
        host_backup_path = BACKUP_FOLDER
        if not os.path.isdir(host_backup_path):
            verbose("Creating backup folder " + host_backup_path)
            os.makedirs(host_backup_path)

        # Step 1: remove the oldest folder
```

```
        if os.path.isdir(host_backup_path + "." + str(ITERATIONS)):
            verbose("Deleting oldest version of backup directories")
            shutil.rmtree(host_backup_path + "." + str(ITERATIONS))

        # step 2: move the other folders up
        for i in range((ITERATIONS - 1),0,-1):
            debug("Checking if " + str(i) + "th folder exists")
            if os.path.isdir(host_backup_path + "." + str(i)):
                verbose("Moving " + host_backup_path + " from " + str(i) + " to "
+ str(i + 1))
                shutil.move(host_backup_path + "." + str(i),host_backup_path + "."
+ str(i + 1))


        # step 3: cp -al the current folder
        verbose("Copying main folder with hard links")
        verbose("host_backup_path = " + host_backup_path)
        os.system("cp -al " +  host_backup_path + " " + host_backup_path + ".1")

        # step 4: sync the current folder
        verbose("Synchronizing folders")
        for folder in pathlist:
            folder = folder.rstrip()
            verbose("-> " + folder)
            if not os.path.isdir(host_backup_path + folder):
                os.makedirs(host_backup_path + folder)

            verbose_rsync = "v" if VERBOSE else ""

            #Handling exception if a server is down.
            try:
                os.system("rsync -a" + verbose_rsync + " --delete " + SCP_USER +
host + ":" + folder + " " + host_backup_path + folder)
                break
            except OSerror as error:
                print("Something went wrong. Perhaps " + SCP_USER + host + " is
down?")
```