vnextcloud.py

Introduction

The purpose of this code is to create quicklinks automatically as they are added to nextcloud. It is assumed, that the "Flow" component has been added to nextcloud and flows have been added to determine whether a file should be added to quicklinks or not. This might be because a file is in a folder, owner by a certain user or the user is a member of a certain group.

The program must be called with a file owner (--owner) and a full path to the file (--file).

The program will create a public share if:

- The program is called with --create-public-share
- The program is called with --public-keyword followed by the keyword. In which case the ini file must contain the publickeyword key. The public keyword will cause the program to create a public share if the keyword occurs anywhere in the path (including the filename).

In Addition:

- If configured, the program will add the file and the url for the public share to a text file.
- If configured, the program will perform some logging
- If configured, the program will pass the file, owner and URL to a Pronto API for adding to quicklinks.

Python Version

Written and tested on Python 3.10.13. Package versions as per REQUIREMENTS.txt

Configuration

NextCloud User

A user should be configured in NextCloud with a name like "quicklinks" (or similar). It should be in a group with all other users. The folder structure should be shared with all other users so that users can add files to the share. Note that when a user adds a file to a folder that has been shared by another user (e.g. quicklinks) then the file owner is the owner of the shared folder (e.g. quicklinks)

NextCloud Flows

NextCloud should be configured to use "flows". There is an flow that can be downloaded called "Workflow external scripts". This should be downloaded and installed.

Create a flow that occurs when a file is created. It should have a filter that selects the public files (e.g. where user = "public"). The flow should call the vnextcloud python script using the parameters specified below:

```
/root/PycharmProjects/vnextcloud/vnextcloud.sh %o %f  --create-public-share
```

If required a log can be created using:

```
/root/PycharmProjects/vnextcloud/vnextcloud.sh %o %f  --create-public-share >> /tmp/flow.log 2>&1
```

vNextCloud Configuration

This application requires a configuration file. An example is shown here:

```
[SETUP]
url = https://nextcloud189.velocityglobal.co.nz
nextcloudroot = /var/www/html/nextcloud/data
applog = /var/www/html/nextcloud/vnextcloud.log
loglevel = DEBUG
datalog = /tmp/vnextcloud.csv
prontourl = https://raydemo.velocityglobal.co.nz:8440
prontoapiuser = someuser
prontoapipassword = somepassword
#publickeyword = public

[USERPASSWORDS]
public = mypublicpassword
quicklinks = myquicklinkspasswords
```

The following notes apply:

key	Description
url	This is the url of the nextcloud server. It should have proper certificate installed and working for this application to work properly. ** The Program will not run without this configuration item **
nextcloudroot	This is the root of the nextcloud data folder. In order to create the public share this part of file path will be removed from the front of the newly created file. ** The Program will not run without this configuration item **
applog	The name of a file into which this application will create log records.
loglevel	The logging Level. Must be one of CRITICAL, ERROR, WARNING, INFO, DEBUG NOTSET
datalog	This the name of file on the nextcloud server. If provided a a record will be written to this file with the date, time, The filename, a public URI and the file no
prontourl	These parameters define a PRONTO url upon which an api (vapiql.spl)
prontoapiwebresource	must be available. The URL should be the same url as that the
prontoapiuser	pronto web ui is running (including the same port number).
prontoapipassword	This program will call the api passing the appropriate parameters The user and password and the relevant user and passwords to call the api. The webresource is the name of the webresource containing the the vapiql program.
USERPASSWORDS	This section should contain a list of users and their passwords that are to be used in the quicklinks shares. i.e. It should only have users and passwords for those users who will have files added to pronto quicklinks. These are the "dummy" users (not real users). Calls to this program pass a file and an owner. It is the owner that must appear in this list of passwords. IT is assumed that a "quicklinks" user of some kind is defined and all files and folders owned by that user are available to all other users. Only files added to folders owned by these people will be processed.

A note regarding passwords.

The ini file contains passwords for both the nextcloud users and the pronto api user. It is possible that to store those passwords in clear text, though this is not recommended.

The system supports encrytped passwords that use the standard pronto encrypt and decrypt as defined in the Pronot 4gl programming manual. To use encrypted passwords the ini file entry must contain the encrypted password and the initialisation vector and these should be comma separated. For example

```
rayb = RFcP+D3OmvpeNXVieA==,*UGd:0M/jJpI=KpD
raysharee = Yh/J/xduRtaH9Zrpww==,hE82^)D8Nk_M+Qvw
public = e2Kdera521Ti8YmqviY=,//?wiB@Im;8ks(@1
```

The pronto program vqlmntauto contains an option under "parameters" that will generate these entries. As per the Pronto documentation, the encryted text is base64 encoded, therefore there is a guarantee that no comma will exist in the encoded text. Vqlmntauto has code in it that also ensure the initialisation vector can contain no commas.

Finally note that the secret key part of the encryption algorthim is hard coded into both the python code and the pronto code. Therefore it is important that access to both sets of code is restricted.

Calling Procedure

The python program has the following parameters

Parameter	Description
--file	The path to the file. Should be the full path from the root ** MANDATORY **
--owner	The owner of the file ** MANDATORY **
--configfile	The path to the vnextcloud.ini file. ** OPTIONAL **

Testing

Regression Testing.

When changing the code, there is a full set of regression tests defined in the testcases.py file. This should be run each time the code changes.

Setup

If pays to review the vnextcloud.ini file. It may be useful to change the logging level from INFO to DEBUG

Logging on to nextcloud

Testing on nextcloud189.velocityglobal.co.nz should be done by logging into Nextcloud as rayb or any user that is not public. Whichever user is used, the user must belong to the group "staff" and be able to see the quicklinks share folder eg "Quicklinks Documents" or "Public Documents".

Processing and checking

Then:

- Add a file to the Public Documents/Stock folder
- Wait... an email has been sent to ray.burns@velocityglobal.co.nz (see vnextcloud.sh)
- Check the application log
- Check the csv

- Check Pronto to see if quicklink has been added.
  - Should have url
  - User only nums have shareid and fileid
- Check /pro/data/sat/vapiql.log. This has a log of what the api processed

Under the Covers

onxtcld.py

onxtcld contains the definition of the NxtCld class. This is where the majority of the work is done.

Instances of the class should be instantiated with the url, user and password for nextcloud. The user and password should be the user and password for the owner of the file that is about to be processed. Additionally, the class can be instantiated with a filename.

The class must be instantiated with a URL to the next Cloud server, a user id to connect with and a password for that user. Optionally a path can be added to the instantiation or set via the setter.

Essentially the oject is instantiated with the url, username and password. Other variables that control the behaviour of the class should then be set.

It is MOST important to note that the determination of the shares is done by __process_shares. Each getter of the readonly variables (share_url, public_share_id and file_id) will invoke __process_shares if it has not already been executed. The two methods that do things with the shares (add_to_csv_file and add_pronto_quicklink) will also invoke the process_shares function if has not already been done. Therefore, all the object consumer has to do is to access a property or invoke a mehthod for the share processing to execute.

The Object then does three essential functions:

a. A share url is deterined. This could be a public share, or a private share. If so configured the object will create a public share and return the url or, if no public share exists and the no option has been taken to create one, then a private url is returned. A file id is always available, but the public_share_id is only available if a public share already exists or has been newly created. The boolean variable __shares_processed is set when the object has completed this task
b. An entry is made in a log file.
c. A pronto API is called to add the quicklink.

Usage:

```
# Instantiate the class with user
thiscloud = onxtcld.NxtCld(global_url, pgm_args.owner, ownerpassword(pgm_args.owner))
# Set critical properties
# public shares are created only if optionally_create_public_share is true or
# a keyword has been defined.  If a keyword has been defined then the public share
# is created if the keyword is found in the path or filename.
thiscloud.optionally_create_public_share = TRUE
# or
thiscloud.public_share_keyword = 'shared'
# Finally set the file path.  This invokes the processing.
thiscloud.file_path = Fullpathtofile

# Access the properties
# The determination of returned properties is in the property getter itself.  The object keeps track
# of whether the determination of the values has already been done.
print(thiscloud.first_public_share_url)
```

```
print(thiscloud.first_public_share_id)
# create publicshare:
```

```
thiscloud.create_public_share()
# access URL
print(thiscloud.url)
print(thiscloud.fileid)
print(thiscloud.shareid)
# add to logfile
thiscloud.add_to_log_file('/tmp/logfile.log')
# or
thiscloud.log_file = '/tmp/logfile.log'
thiscloud.add_to_log_file()
# Call pronto API
thiscloud.add_pronto_quicklink(url,user,password)
```

Pronto Hosted Services

The following notes apply to running the application within the Pronto Hosted Services environment.

There is a special user in nextcloud called quicklinks. Files and folders belonging to this user are the application and it runs from the folders beloning to that user.

Creating the Virtual Environment

This application runs in a Python Virtual Environment using Venv. The code is all stored within nextcloud itself.

There is a script to build the virtual environment called crtvenv.sh.

Before running, this script needs reviewing. There are four key variables that are defined at the top of the script and these will need to be reviewed before execution.

Creating the flow job

There is a different version of the script for running at PHS. It is called vnextcloudphs.sh. We cannot make the executing procedure "executable", therefore it is called by bash:

```
/bin/bash /var/www/html/nextcloud/data/quicklinks/files/test_310a/vnextcloudphs.sh %o %f  >> /tmp/testflow.log 2>&1
```

or, to create a share:

```
/bin/bash /var/www/html/nextcloud/data/quicklinks/files/test_310a/vnextcloudphs.sh %o %f --create-share >> /tmp/testflow.log 2>&1
```

Distributing the Code

There is a shell script called phsdistro.sh which builds /tmp/vnextcloud.tar This can be sent to PHS.