hcpupd Documentation

Release 0.2.5

Thorsten Simons

May 03, 2022

Contents:

hcpupd is a daemon that automatically uploads files to HCP.

It's using the Linux inotify kernel subsystem[1] to monitor a folder (aka *watchdir*), sending every file that is moved or written to it to HCP, immediately. Folders created in the *watchdir* will be watched as well.

Features:

- Two different upload modes:
  1. Transfer the folder structure created in *watchdir* to HCP as it is
     - human-readable
     - performance-wise not the best possible solution
     - not tolerant against filename duplicates (except if the Namespace has Versioning enabled)

  or

  2. Obfuscate the folder structure on HCP by creating an UUID per file, used as filename as well as to construct a path to it
     - best possible ingest performance
     - 64k folders created at max. (*to be precise: 256**2 + 256*)
     - an unlimited number of **hcpupd**'s can write into the same Namespace without the risk of filename conflicts
     - supports search for the original filename by adding an annotation to each file, which can be used by *HCP's Metadata Query Engine*, *Hitachi Content Intelligence* or any other indexer that is able to crawl a HCP Namespace
- Optionally:
  - on start-up, upload existing files already stored in the *watchdir*
  - auto-delete files from *watchdir* after successful upload
  - auto-delete folders after the last file has been uploaded
- Made to run as a Linux daemon (but can run in an interactive session as well)
- Extended logging available, incl. log rotation

Tip: Please note the *Limitations*.

---

[1] https://en.wikipedia.org/wiki/Inotify

CHAPTER 1

Installation

1.1 Binary Distribution

For most modern Linux derivates, you should be able to simply run the binary provided here[2].

Grab it there[3] and follow the instructions in chapter *Run hcpupd*.

1.2 Build your own Binary

In case the provided binary fails to run on your Linux, you need to build it on your own. Here's how to do that:

Warning: Make sure the `objcopy` utility is installed:

```
$ objcopy
```

If the command isn't found, you need to install the *GNU binutils* package. For Fedora 24, this is:

```
$ sudo dnf install binutils.x86_64
```

- Clone the repository from GitLab[4]:

```
$ git clone https://gitlab.com/simont3/hcpupd.git
```

- Change into the project folder and create a Python 3 virtual environment and activate it:

```
$ cd hcpupd/src
```

- Update pip and setuptools, then load all required dev-packages:

```
$ sudo pip3 install --upgrade pip setuptools
$ sudo pip3 install -r pip-requirements-dev.txt
```

[2] https://gitlab.com/simont3/hcpupd/blob/master/src/dist/hcpupd
[3] https://gitlab.com/simont3/hcpupd/blob/master/src/dist/hcpupd
[4] https://gitlab.com/simont3/hcpupd.git

- Run the build tool:

```
$ pyinstaller hcpupd.spec
```

  You should find the executable in the `dist` subfolder.

- Now follow the instructions in chapter *Run hcpupd*.

CHAPTER 2

Configuration

hcpupd's behavior is controlled by a configuration file, which is searched for in this order:

If specified, the file identified by the `-c` parameter, otherwise

1. Current working directory: `.hcpupd.conf`
2. User's home directory: `~/.hcpupd.conf`
3. System-wide: `/var/lib/misc/hcpupd.conf`

Tip: **hcpupd** will offer to create a template configuration file in case it can't find one in any of the given locations:

```
$ hcpupd
No configuration file found.
Do you want me to create a template file in the current directory (y/n)? y

A template file (.hcpup.conf) has been created in the current directory.
Please edit it to fit your needs...
```

2.1 The configuration file explained

The configuration file is an ini-style text file with several sections, each of them holding configuration values.

2.1.1 The [src] section

describes from where files are to be uploaded to HCP, and how.

```
[src]
watchdir = /watchdir
upload existing files = yes
delete after upload = yes
remove empty folders after upload = yes
```

- **watchdir** - is the folder that will be monitored; every file written into it will be uploaded to HCP. **The folder specified here must exist when hcpupd is started.**
- **upload existing files** - enable discovery of files that are already in `watchdir` when **hcpupd** is started.
- **delete after upload** - enabled auto-deletion of files as soon as they have been uploaded successfully.
- **remove empty folders after upload** - enable auto-deletion of folders as soon as the last file has been uploaded.

> **Warning:** Be aware that setting `remove empty folders after upload = yes` will cause **hcpupd** to immediately delete a folder when the last file it contained has been uploaded.
>
> **This may cause applications writing into the watchdir to fail, as they might still expect a folder to exist they created earlier.**

2.1.2 The [tgt] section

describes where to store the files found in [src], and how.

```
[tgt]
namespace = namespace.tenant.hcp.domain.com
path = hcpupd/application
user = username
password = her_password
ssl = yes
obfuscate = yes
local DNS resolver = no
upload threads = 2
```

- **namespace** - the HCP Namespace to write to
- **path** - the path within the Namespace
- **user** - a user with write access to the Namespace
- **password** - her password
- **ssl** - enable transfer encryption (HTTPS)
- **obfuscate** - enable obfuscation of the file names stored to HCP
- **local DNS resolver** - set to `no` to use the built-in resolver
- **upload threads** - the number of uploader threads to use

2.1.3 The [meta] section

describes how to build the custom metadata annotation stored with each files (if `obfuscate = yes`, only).

```
[meta]
annotation = hcpupd
tag_timestamp = yes
tag_note = files from my application
retention = 0
```

- **annotation** - the name of the annotation to write
- **tag_timestamp** - enable adding the file's creation time

- **tag_note** - a note that will be added
- **retention** - `0` (zero) - the only supported value at this time

2.1.4 The [log] section

defines the logfile to write and if extended debug logging shall be performed.

```
[log]
logfile = /var/log/hcpupd.log
log uploaded files = yes
debug = yes
```

- **log uploaded files** - this will enable logging of uploaded files even if *debug = no*

Tip: Make sure to create the folder into which the `logfile` shall be stored before you start **hcpupd** the first time!

CHAPTER 3

Run **hcpupd**

3.1 In the foreground

Easiest way to run **hcpupd** is having a private `~/.hcpupd.conf` file and simply starting it:

```
$ ./hcpupd
```

3.2 As a Daemon

Adding `-d` to the command will run **hcpupd** in daemon mode, releasing the terminal session for further use:

```
$ ./hcpupd -d
```

> **Warning:** Due to the foundation technology, this will fail silently (!) if started by user **root**!

3.3 Using systemd

In a unattended production environment, you'll want to run **hcpupd** as a daemon in the background, automatically started on system boot.

> **Warning:** For security reasons, you shouldn't run **hcpupd** with root privileges. You might consider to create a technical user for it...
>
> We assume that you have <*user*> and its <*group*> created, already...

- Move the **hcpupd** binary to `/usr/local/bin` and set proper permissions:

```
$ sudo cp hcpupd /usr/local/bin
$ sudo chmod 755 /usr/local/bin/hcpupd
```

- Create a *systemd* service file (`/etc/systemd/system/hcpupd.service`) with this content:

```
[Unit]
Description=HCP upload daemon
# tested with Fedora 24:
Requires=network.service
# tested with Ubuntu 17.04:
# Requires=networking.service

[Service]
Type=simple
# make sure you set the correct path for bash!
ExecStart=/usr/bin/bash -c /usr/local/bin/hcpupd
Restart=always

User=<user>
Group=<group>

[Install]
WantedBy=multi-user.target
```

  hcpupd requires networking up and running, so you might need to find the proper entry for *[Unit] Requires=*, depending on your Linux distribution.

- Run **hcpupd** once to create a template config file:

```
$ hcpupd
No configuration file found.
Do you want me to create a template file in the current directory (y/n)? y

A template file (.hcpup.conf) has been created in the current directory.
Please edit it to fit your needs...
```

- Edit the `.hcpupd.conf` template file to your specific needs.

> **Warning:** Make sure that you have created the folder to watch, as well as the folder to store the logfile with appropriate permissions (see Config section).

- Move the config file to it's place and set the permissions properly (**remember, there's a password in it - you don't want to expose it**):

```
$ sudo mv .hcpupd.conf /var/lib/misc/hcpupd.conf
$ sudo chown <user>:<group> /var/lib/misc/hcpupd.conf
$ sudo chmod 600 /var/lib/misc/hcpupd.conf
```

- If you stay with the default logging path set in the config file, create the folder:

```
$ sudo mkdir /var/log/hcpupd
$ sudo chown <user>:<group> /var/log/hcpupd
$ sudo chmod 700 /var/log/hcpupd
```

- Manually start **hcpupd** in foreground mode to give it a try:

```
$ sudo -u <user> /usr/local/bin/hcpupd
```

  If everything works fine (which means the config file is correct), stop **hcpupd** by pressing CTRL-C.

- Now start it as a daemon using *systemd*:

```
$ sudo systemctl start hcpupd.service
```

  Check if it's running by:

```
$ sudo systemctl status hcpupd.service
```

- If everything is fine, you can enable the daemon to be started on Linux boot:

```
$ sudo systemctl enable hcpupd.service
```

CHAPTER 4

Limitations

- Moving a folder structure into the *watchdir*...

  ... will lead to the files in the top-level folder to be uploaded, but everything else will not. Reason for this is that the inotify mechanism in charge is not getting the information for all the sub-folders when moving in a folder structure.

  Workaround: avoid moving in folder structures, copy them instead.

- Renaming a folder (or moving a folder within *watchdir*) is not supported.

CHAPTER 5

Release History

0.2.5 2017-06-11

- fixed a bug that caused installation through pip to fail
- changed documentation telling not to run **hcpupd** as root, plus some systemd-related info

0.2.4 2017-03-27

- in case of inotify queue overflow, a directory scan is triggered to make sure no new files get lost
- in case the queue runs empty, we now preventively trigger a directory scan, as well
- new config item 'log uploaded files' allows to log uploaded files in non-debug logging mode
- added a message on shutdown that tells how many files are left for later upload

0.2.3 2017-03-01

- re-factored configuration file handling
- now surviving connection loss to HCP (missed file recovery still requires *hcpupd* restart)

0.2.2 2017-02-15

- fixed a bug that caused moved_in folders not to be processed
- added some more debug output for watched folder handling

0.2.1 2017-02-12

- various fixes related to publishing throuh gitlab and readthedocs.org

0.2.0 2017-02-11

- First public release

CHAPTER 6

License / Trademarks

6.1 The MIT License (MIT)

6.2 Trademarks and Copyrights of used material

Hitachi Content Platform is a registered trademark of Hitachi Data Systems Corp., in the United States and other countries.

All other trademarks, service marks, and company names in this document or web site are properties of their respective owners.

The logo used in the documentation is based on the picture found at pixabay.com[5], where it is declared under CC0 Public Domain[6] license.

[5] https://pixabay.com/en/paper-planes-fly-send-aircraft-1513032/
[6] https://pixabay.com/service/terms/#usage