GitP4Transfer - Git to P4 Migration

Perforce Professional Services

2022-04-19

Table of Contents

Chapter 1. GitP4Transfer.py

1.1. Overview

This is a functional script to import a git repo with LFS commits for a single branch (e.g. master/main) into Helix Core.

It includes some ideas from `git-p4` and `git-filter-repo.py`.

1.2. Pre-requisites

- Install recent version of git (2.x)
- Install Python 3.8+ and modules p4python

1.2.1. Install Git/Git LFS

Easiest to install these from Wandisco to get recent versions (need 2.x and not 1.8 for example):

```
sudo yum install http://opensource.wandisco.com/centos/7/git/x86_64/wandisco-git-release-7-2.noarch.rpm
sudo yum install git git-lfs
git --version
```

1.2.2. Clone repo

This assumes a filesystem such as `/hxdepots` with plenty of free space.

```
cd /hxdepots
mkdir work
cd work
git clone <url of your git repo>
```

1.2.3. Fetch all LFS objects

1. First ensure that git LFS credentials are stored

```
git config --global credential.helper store
```

```
$ git branch
* master
```

© 2010-2021 Perforce Software, Inc.

```
$ git lfs fetch --all
fetch: 163739 object(s) found, done.
fetch: Fetching all references...
Username for 'https://git.example.com': fred.bloggs
Password for 'https://fred.bloggs@example.com':
Downloading LFS objects:   4% (6561/163738), 9.1 GB | 100 MB/s
```

2. After the above you can Ctrl+C to abort because credentials should be in place.

```
cat ~/.git-credentials
```

3. If you want to check, the re-run the command and you should not be prompted.

```
git lfs fetch --all
```

4. Finally you can spawn the full fetch of all LFS versions (which often takes hours depending on size of your repo):

```
nohup git lfs fetch --all > ../fetch.out &
```

```
perforce@ip-10-0-0-151 gitrepo]$ cat ../fetch.out
fetch: 163739 object(s) found, done.
fetch: Fetching all references...
```

5. Check for LFS files not found too - all files less than 140 bytes in size are possible candidates to be checked:

```
find .git/lfs/objects/ -type f -size -140c
```

6. LFS files which have not been replaced with their proper contents will be similar to this sort of format:

```
version https://git-lfs.github.com/spec/v1
oid sha256:8923f38904c1ae21cd3d3e6e93087c07fda86fe97ee01d8664bb95fc20cd1de
size 858449
```

   If such files are found, you will need to determine why they were not fetched and try to fix that to get proper LFS contents downloaded.

© 2010-2021 Perforce Software, Inc.

1.2.4. Install Python3.8

Unfortunately 3.6 is missing some required changes in the `subproc` library, so you may ßneed to build from source. Ubuntu is similar (but different dependencies to install first!)

```
yum install wget yum-utils make gcc openssl-devel bzip2-devel libffi-devel zlib-devel
VER="3.8.12"
wget https://www.python.org/ftp/python/$VER/Python-$VER.tgz
tar zxvf Python-$ver.tgz
cd Python-$ver
./configure
make install
```

1.2.5. Install GitP4Transfer.py

We install dependencies and then the script itself.

1. Run the following as `root`:

```
cat << EOF > /etc/yum.repos.d/perforce.repo
[Perforce]
name=Perforce
baseurl=http://package.perforce.com/yum/rhel/7/x86_64/
enabled=1
gpgcheck=1
EOF

rpm --import https://package.perforce.com/perforce.pubkey

yum install perforce-p4python3
```

2. As normal user, e.g. `perforce`:

```
pip3 install --user requests ruamel.yaml
```

3. Clone the gitp4transfer repo

```
git clone https://github.com/perforce/gitp4transfer.git
```

4. Ensure dependencies setup

```
cd gitp4transfer
python3 GitP4Transfer.py -h
```

5. Setup config file

© 2010-2021 Perforce Software, Inc.

```
python3 GitP4Transfer.py --sample-config > transfer_config.yaml
```

6. Create appropriate target depot, e.g. `//git_import/repoA/master` and ensure setup in config file.
7. Do a test of config:

   ```
   python3 GitP4Transfer.py -c transfer_config.yaml -n
   ```

   Validate log files for success.

8. Consider setting up `p4 typemap` as appropriate for your import (e.g. for Unreal Engine or Unity)
9. Do a first test of one commit (note this is often quite a big commit so may still take a while!)

   ```
   python3 GitP4Transfer.py -c transfer_config.yaml -m1
   ```

10. If the above works, kick off a full transfer and monitor log/output file:

    ```
    nohup python3 GitP4Transfer.py -c transfer_config.yaml > out &
    ```

1.2.6. Note about temp branch

The script works by replaying each commit. To do this it executes:

```
for each commitid in reverse order:
    git switch -C p4_exportBranch <commitid>
    parse the output of git diff-tree against previous commit
    run various p4 commands
```

As a result, expect the new branch `p4_exportBranch` to be created and continually updated. This is effectively a dummy branch.

When the script has finished you may need to: `git checkout master` or similar to reset to your current branch.

if the script fails, then the active branch is going to be the temp one - don't assume it is HEAD/master!

1.2.7. Things to do

- Adjust unknown_git user
- Date times for changes update
- Interleave in date/time order

© 2010-2021 Perforce Software, Inc.

- More informative commit messages

1.2.8. Branch diffs

Generated by:

```
git log --first-parent --oneline master > ../b_master.txt
```

© 2010-2021 Perforce Software, Inc.

Chapter 2. gitp4transfer - Go program

This uses git's fast-import file format.

For git LFS files, this might work via `git lfs migrate`??

Not yet functional - very much a work in progress!!!

© 2010-2021 Perforce Software, Inc.