CompuCell3D

Download

- Binaries
- Source Code
- Developer Zone

Help

- CC3D User Forum
- Manuals
- Tutorials
- F.A.Q.

Demos

- Simulation Movies
- Screenshots
- Model Repository

Publications

- Publications
- Theses
- Talks and Posters

Events

- Workshops

About

- People
- Contact Us
- Mailing List

Search Site

Search

Titles Text

Cleaning Git repository

1. Clone big repository to a separate folder

```
mkdir BIG_GIT
cd BIG_GIT
git clone <git-address-of-your-repository> .
```

2. Save backup copy of the repository (Optional) From the BIG_GIT type:

```
git bundle create <directory-where-you-want-to-store-the-
bundle>/BIG_GIT.bundle --all
```

e.g.

```
git bundle create ~/BIG_GIT.bundle --all
```

3. Start removing the obvious big directories (or files) that might contain e.g. large files or unneeded files

```
git filter-branch --tag-name-filter cat --index-filter 'git rm -r --
cached --ignore-unmatch dir1 dir2 dirn file1 file2 filen' --prune-empty
-f -- --all
```

e.g.

```
git filter-branch --tag-name-filter cat --index-filter 'git rm -r --
cached --ignore-unmatch 2.1.0 3.2.0 3.2.2  3.3.1  3.4.05  3.4.2 3.5.0
3.6.0  3.6.2  PIF_Generator twedit twedit++ Workshop08 3.1.18  3.2.1
3.3.0  3.4.0  3.4.1 3.4.4-unstable  3.5.1  3.6.1  src-parallel ' --
prune-empty -f -- --all
```

4. Run cleanup commands (in the order as below). You may get these commands as a script

```
rm -rf .git/refs/original/
git reflog expire --expire=now --all
git gc --prune=now
git gc --aggressive --prune=now
```

the last command might take a while to complete. You force git garbage collector to actually remove directories and files that you removed in step 3

WARNING: if your repository is very large the last command might fail with "Out Of Memory" error. If this is the case limit the amount of memory and number of simultaneous threads performing cleanup by executing these two commands

```
git config --global pack.threads 1
git config --global pack.windowMemory 256m
```

You can increase memory from 256 to bigger value or increase number of threads depending on your machine specification

5. If you think you are done you might actually still have large files hidden in your repository. To discover them you will need to run the following bash script Get the script here

```
#set -x

# Shows you the largest objects in your repo's pack file.
# Written for osx.
#
# @see http://stubbisms.wordpress.com/2009/07/10/git-script-to-show-largest-pack-objects-and-trim-your-waist-line/
# @author Antony Stubbs

# set the internal field spereator to line break, so that we can iterate easily over the verify-pack output
IFS=$'\n';

# list all objects including their size, sort by size, take top 10
objects=`git verify-pack -v .git/objects/pack/pack-*.idx | grep -v chain | sort -k3nr | head`

echo "All sizes are in kB's. The pack column is the size of the object, compressed, inside the pack file."

output="size,pack,SHA,location"
for y in $objects
do
        # extract the size in bytes
        size=$((`echo $y | cut -f 5 -d ' '`/1024))
        # extract the compressed size in bytes
        compressedSize=$((`echo $y | cut -f 6 -d ' '`/1024))
        # extract the SHA
        sha=`echo $y | cut -f 1 -d ' '`
        # find the objects location in the repository tree
        other=`git rev-list --all --objects | grep $sha`
        #lineBreak=`echo -e "\n"`
        output="${output}\n${size},${compressedSize},${other}"
done

echo -e $output | column -t -s ', '
```

It will output 10 biggest files (or actually objects rep[resenting git deltas for these files) in remaining still in your repository. Those files might be hard to find by browsing repository because they might have been removed from Git but the git deltas representing remove operation for this files is still in the git repository in case you want to reinstate the files at some point in the future

You run above script and then pass the file names you want to definitely remove to step 3. After runnign step 3 you run step 4 and then go back to step 5 to look for more files

6. Now you can push your repository to a newly created remote repository (you create new repository on github.com or other git hosting provider)

```
git remote add neworigin <your-new-remote-git-url>
```

e.g.

```
git remote add neworigin git@github.com:compucell3d/CompuCell3D.git
```

7.Push trimmed down Git repository to the neworigin (here I push all the branches separately)

```
git push -u neworigin branch1
git push -u neworigin branch2
git push -u neworigin branch3
```

e.g.

```
git push -u neworigin master
git push -u neworigin 3.6.2
git push -u neworigin 3.7.0
```

8. Auxiliary scripts:

- git_cleaner
- big_finder
- deep_fetch clones repository and pulls all the branches
- git_shrink sequence of commands described above in the form of script
- pull_all_branches - pulls all branches from remote git