Nutch 0.9 Crawl Script Tutorial

Nutch 0.9 Crawl Script Tutorial

This is a walk through of the nutch 0.9 crawl.sh script provided by Susam Pal. (Thanks for getting me started Susam) I am only a novice at the whole nutch thing so this article may not be 100% accurate, but I think this will be helpful to other people just getting started, and I am hopeful other people who know more about Nutch will correct my mistakes and add more useful information to this document. Thanks to everyone in advance! (by the way I made changes to Susam's script, so if I broke stuff or made stupid mistakes, please correct me. 😉

```
#!/bin/sh

# Runs the Nutch bot to crawl or re-crawl
# Usage: bin/runbot [safe]
#        If executed in 'safe' mode, it doesn't delete the temporary
#        directories generated during crawl. This might be helpful for
#        analysis and recovery in case a crawl fails.
#
# Author: Susam Pal
```

First we specify some variables.

Depth tells how many times to crawl the web page. It seems like about 6 will get us all the files, but to be really thorough 9 should be enough.

Threads sets how many threads to crawl with, though ultimately this is limited by the conf file's max threads per server setting because for intranet crawling (like we are doing) there is really only one server.

adddays is something I don't know... need to figure out how to use this to our advantage for only crawling updated pages.

topN is not used right now because we want to crawl the whole intranet. You can use this during testing to limit the maximum number of pages to crawl per depth. But it will make it so you don't get all the possible results.

```
depth=0
threads=50
adddays=5
#topN=100 # Comment this statement if you don't want to set topN value

NUTCH_HOME=/data/nutch
CATALINA_HOME=/var/lib/tomcat5.5
```

Nutch home and catalina home have to be configured to point to where you installed nutch and tomcat respectively.

```
# Parse arguments
if [ "$1" == "safe" ]
then
  safe=yes
fi

if [ -z "$NUTCH_HOME" ]
then
  NUTCH_HOME=.
  echo runbot: $0 could not find environment variable NUTCH_HOME
  echo runbot: NUTCH_HOME=$NUTCH_HOME has been set by the script
else
  echo runbot: $0 found environment variable NUTCH_HOME=$NUTCH_HOME
fi

if [ -z "$CATALINA_HOME" ]
then
  CATALINA_HOME=/opt/apache-tomcat-6.0.10
  echo runbot: $0 could not find environment variable NUTCH_HOME
  echo runbot: CATALINA_HOME=$CATALINA_HOME has been set by the script
else
  echo runbot: $0 found environment variable CATALINA_HOME=$CATALINA_HOME
fi

if [ -n "$topN" ]
then
  topN="--topN $rank"
else
  topN=""
fi
```

This last part just looks at the incoming variables and sets defaults, etc... Now on to the real work!!

Step 1 : Inject

First thing is to inject the crawldb with an initial set of urls to crawl. In our case we are injecting only a single url contained in the nutch/seed/urls file. But in the future this file will probably get filled with "out of date" pages in order to hasten their recrawl.

```
steps=10
echo "----- Inject (Step 1 of $steps) -----"
$NUTCH_HOME/bin/nutch inject crawl/crawldb seed
```

Step 2 : Crawl

Next we do a for loop for $depth number of times. This for loop performs a couple steps which make up a basic 'crawl' procedure.

First it generates a segment which (I think?) is filled with empty data for each url in the crawldb that has reached it's expiration (i.e. has not been fetched in a month). I am not really sure what this does yet...

Then it fetches pages for those urls and stores that data in the segment. During this fetch phase, it also fills the crawldb with any new urls it finds (as long as they are not excluded by the filters we configured). This is really the key to making this for loop work, because the next time it gets to the segment generation there will be more urls in the crawldb for it to crawl. Notice however that the crawldb never gets cleared in this script... so if I am not mistaken there is no need to re-inject the root url.

Then we parse the data in the segments. Although depending on your configuration in the xml files this can be done automatically, in our case we are parsing it manually cause I read it would be faster this way... Haven't really given it a good test yet.

After these steps are done we have a nice set of segments that are full of data to be indexed.

```
echo "----- Generate, Fetch, Parse, Update (Step 2 of $steps) -----"
for((i=0; i < $depth; i++))
do
  echo "--- Beginning crawl at depth `expr $i + 1` of $depth ---"
  $NUTCH_HOME/bin/nutch generate crawl/crawldb crawl/segments $topN -adddays $adddays
  if [ $? -ne 0 ]
  then
    echo "runbot: Stopping at depth $depth. No more URLs to fetch."
    break
  fi
  segment=`ls -d crawl/segments/* | tail -1`

  $NUTCH_HOME/bin/nutch fetch $segment -threads $threads
  if [ $? -ne 0 ]
  then
    echo "runbot: fetch $segment at depth $depth failed. Deleting it."
    rm -rf $segment
    continue
  fi

  echo "--- Parsing Segment $segment ---"
  $NUTCH_HOME/bin/nutch parse $segment

  $NUTCH_HOME/bin/nutch updatedb crawl/crawldb $segment
done
```

Step 3 : Stop Tomcat

Not sure if this is necessary, but I seemed to have problems with files being in use if I didn't.

```
echo "----- Stopping Tomcat (Step 3 of $steps) -----"
sudo /etc/init.d/tomcat5.5 stop
```

Step 4 : Merge Segments

This part is pretty straight forward. It takes the existing segments in crawl/segments and merges them into a single one. If we are recrawling then there should have been one (or more perhaps) segment files in crawl/segments/ before we started, but at the very least we should have one for each of the depths in step 2.

Here we backup the old segments, then we merge them into a temporary folder MERGEDsegments. When the merge is complete we delete the originals and replace them with the single merged segment folder. I made it so that if the Merge Segments fails for some reason we bail on the script also. This is because I often found myself deleting the backed up segments after merging failed. :,(shiku

```
echo "----- Merge Segments (Step 4 of $steps) -----"
$NUTCH_HOME/bin/nutch mergesegs crawl/MERGEDsegments crawl/segments/*

if [ $? -eq 0 ]
then
    if [ "$safe" != "yes" ]
    then
      rm -rf crawl/segments/*
    else
      mkdir crawl/FETCHEDsegments
      mv --verbose crawl/segments/* crawl/FETCHEDsegments
    fi

    mv --verbose crawl/MERGEDsegments/* crawl/segments
    rmdir crawl/MERGEDsegments
else
    exit
fi
```

Step 5 : Invert Links

I think this updates the crawldb scores so that people who point

```
echo "----- Invert Links (Step 5 of $steps) -----"
$NUTCH_HOME/bin/nutch invertlinks crawl/linkdb crawl/segments/*
```

Step 6,7,8 : Index

This is the important part! We index the segments into a temp folder NEWIndexes. Then we remove duplicates from the new index. Then we merge the new indexes with the old ones into a temp folder called MERGEDindexes. Then finally we replace the old index with the new one (backing up the old one to OLDindexes).

```
echo "----- Index (Step 6 of $steps) -----"
$NUTCH_HOME/bin/nutch index crawl/NEWindexes crawl/crawldb crawl/linkdb crawl/segments/*

echo "----- Dedup (Step 7 of $steps) -----"
$NUTCH_HOME/bin/nutch dedup crawl/NEWindexes

echo "----- Merge Indexes (Step 8 of $steps) -----"
$NUTCH_HOME/bin/nutch merge crawl/MERGEDindexes crawl/NEWindexes

# in nutch-site, hadoop.tmp.dir points to crawl/tmp
rm -rf crawl/tmp/*

# replace indexes with indexes_merged
mv --verbose crawl/index crawl/OLDindexes
mv --verbose crawl/MERGEDindexes crawl/index

# clean up old indexes directories
if [ "$safe" != "yes" ]
then
  rm -rf crawl/NEWindexes
  rm -rf crawl/OLDindexes
fi
```

Step 9, 10 : Tell Tomcat we are updated

This should update tomcat, but I found we had to reload it to notice anyway... which is what Step 10 does.

```
echo "----- Reloading index on the search site (Step 9 of $steps) -----"
if [ "$safe" != "yes" ]
then
  touch ${CATALINA_HOME}/webapps/ROOT/WEB-INF/web.xml
  echo Done!
else
  echo runbot: Can not reload index in safe mode.
  echo runbot: Please reload it manually using the following command:
  echo runbot: touch ${CATALINA_HOME}/webapps/ROOT/WEB-INF/web.xml
fi

echo "----- Restarting Tomcat (Step 10 of $steps) -----"
sudo /etc/init.d/tomcat5.5 start

echo "runbot: FINISHED: Crawl completed!"
```

Thats about it. I find the part that is most likely to screw up is always the merge command... And it usually screws up because the parse failed which is likely because one of your segments failed to finish fetching. Or if you have parsing during fetching enabled it failed to finish parsing.

Comments and stuff

Please add comments / corrections to this document. 'cause I don't know what the heck I'm doing yet. 🙂 One thing I want to figure out, is if I can inject just a subset of urls of pages that I know have changed since the last crawl and refetch/index only those pages. I think there is a way to do this using the adddays parameter maybe? anyone have any insight?

How to refetch/index a subset of urls

My solution to this common question is to use a filter on the URL we want to refetch and have those expire using the -adddays option of 'nutch generate' command. In nutch-site.xml you should enable a filter plugin such as urlfilter-regex and specify the file which contains the regex filter rules:

<property>

<name>plugin.includes</name>

<value>protocol-http|parse-(xml|text|html|js|pdf)|index-basic|query-(basic|site|url|more)|summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|feed |***urlfilter-regex***</value>

</property>

<property>
<name>urlfilter.regex.file</name>

<value>regex-urlfilter.txt</value>
</property>

The file regex-urlfilter.txt can contain any regular expression, including one or more specific URLs we want to refetch/index, e.g.:

+http://myhostname/myurl.html

At this stage we can use the command "$NUTCH_HOME/bin/nutch generate crawl/crawldb crawl/segments -adddays 31" to generate a segment and the output should look like:

Fetcher: starting

Fetcher: segment: crawl/segments/20080518090826

Fetcher: threads: 50

fetching http://myhostname/myurl.html

redirectCount=0

Any comments/feedback welcome!