Table of Contents

Installing a Frontier squid cache server

NOTE: this page describes instructions for an old version of frontier-squid, specifically frontier-sqiud-2. Instructions for the current version are at InstallSquid. There is also a supported version called frontier-squid2 which has the same software as that described on this page but which has a "2" suffix on all the paths. Instructions for frontier-squid2 are at InstallSquid2.

The frontier-squid software package is a patched version of the standard squid http proxy cache software, pre-configured for use by the Frontier distributed database caching system. This installation is recommended for use by Frontier in the LHC CMS & ATLAS projects, and also works well with the CernVM FileSystem. Many people also use it for other applications as well; if you have any questions or comments about general use of this package contact `frontier-talk@cern.ch`.

Note to Open Science Grid users: this same package is also available from the Open Science Grid so it will probably be more convenient to you to follow the OSG frontier-squid installation instructions.

Note to users of EGI's UMD repository: the same package is also available in UMD so it might be easier for you to get it from there.

After completing a squid installation and configuration, CMS users should follow these further instructions for CMS squids. All WLCG users should register their squids with the WLCG.

Here is what is on this page:

Support

If you have any problems with the software or installation, or would like to suggest an improvement to the documentation, please submit a support request to the Frontier Application Development JIRA.

For rapid response to configuration questions, send e-mail to `cms-frontier-support@cern.ch` or `atlas-frontier-support@cern.ch` if you are in CMS or ATLAS, respectively. If you're not on either of the projects, send your questions to `wlcg-squidmon-support@cern.ch`. Most questions can also be answered on the user's mailing list `frontier-talk@cern.ch`.

Why use frontier-squid instead of regular squid?

The most important feature of frontier-squid is that it correctly supports the HTTP standard headers Last-Modified and If-Modified-Since better than other distributions of squid. The Frontier distributed database caching system, which is used by the LHC projects ATLAS and CMS, depends on proper working of this feature, so that is the main reason why that project maintains this squid distribution. Older versions of squid2 (including the one distributed with Red Hat EL 5) and all versions of squid3 (including the one on Red Hat EL 6) prior to squid3.5 (which is now in pre-release) do not correctly support this feature, as documented in the infamous squid bug #7. Also, the frontier-squid package contains a couple of related patches that are not in any standard squid distribution. Details are in the beginning paragraph of the MyOwnSquid twiki page. Although this package expressly supports If-Modified-Since, it also works well with applications that do not require If-Modified-Since including CVMFS. The collapsed_forwarding feature is also missing from most versions of squid, and it is important for the most common grid applications that use squid and is included in the frontier-squid package.

In addition, the package has several additional features:

1. A configuration file generator, so configuration customizations can be preserved across package upgrades even when the complicated standard configuration file changes.

2. The ability to easily run multiple squid processes listening on the same port, in order to support more networking throughput than can be handled by a single CPU core (squid2 is single-threaded).
3. Automatic cleanup of the old cache files in the background when starting squid, to avoid problems with cache corruption.
4. Default access control lists to permit remote performance monitoring from shared WLCG squid monitoring servers at CERN.
5. The default log format is more human readable and includes contents of client-identifying headers.
6. Access logs are rotated throughout the day if they reach a configured size, to avoid filling up disks of heavily used squids. The logs are also compressed by default.
7. It chooses default options found to be important by years of operational experience on the WLCG.

Hardware

The first step is to decide what hardware you want to run the squid cache server on. These are some FAQs.

1) Do I need to dedicate a node to squid and only squid?

This is up to you. It is a strongly recommended. It depends on how many jobs try to access the squid simultaneously and what else the machine is used for (see question 2). Large sites may need more than one squid (see question 4). The node needs to have network access to the internet, and be visible to the worker nodes. Virtual machines can help isolate other uses of a physical machine, but it doesn't isolate disk and especially network usage.

2) What hardware specs (CPU, memory, disk cache)?

For most purposes 2 cores at 2GHZ, 2GB memory, and 100 GB for the disk cache should be adequate. This excludes the space needed for log files which is determined by how heavily the system is used and what the clean up schedule is. The default in the rpm always rotates the logs every day and removes the oldest log after 10 rotates, and four times an hour it will also rotate if the access log is bigger than 5GB. By default logs are compressed after rotate and typically are reduced to less than 15% of their original size, so allowing 12GB for logs should be sufficient. On heavily used systems the default will most likely keep logs for too short of a time, however, so it's better to change the default (instructions below) and allow at least 25GB for logs.

From what we have seen, the most critical resource is the memory. If the machine serves other purposes, make sure the other tasks don't use up all the memory. Squid runs as a single thread, so if that is the only use of the machine, having more than 2 cores is a waste (unless you are running multiple squid processes). You should also avoid network filesystems such as AFS and NFS for the disk cache.

Here is a description of squid memory usage: If you have a decent amount of spare memory, the kernel will use that as a disk cache, so it's a good chance that frequenty-requested items will, in fact, be served from RAM (via the disk cache) even if it's not squid's RAM. There's also a design bottleneck in squid that limits cpu efficiency of large cache_mem objects, so resist the urge to give squid all your available memory. Let cache_mem handle your small objects and the kernel handle the larger ones.

3) What network specs?

The latencies will be lower to the worker nodes if you have a large bandwidth. The network is almost always the bottleneck for this system, so at least a gigabit each is highly recommended. If you have many job slots, 2 bonded gigabit network connections is even better, and squid on one core of a modern CPU can pretty much keep up with 2 gigabits. Squid is single-threaded so if you're able to supply more than 2 gigabits, multiple squid processes on the same machine need to be used to serve the full throughput. This is supported in the frontier-squid package (instructions below) but each squid needs its own memory and disk space.

4) How many squids do I need?

Sites with over 500 job slots should have at least 2 squids for reliability. We currently estimate that sites should have one gigabit on a squid per 1000 grid job slots. A lot depends on how quickly jobs start; an empty batch queue that suddenly fills up will need more squids. The number of job slots that can be safely handled per gigabit increases as the number of slots increase because the chances that they all start at once tends to go down.

5) How should squids be load-balanced?

There are many ways to configure multiple squids: round-robin DNS, load-balancing networking hardware, LVS, etc. The simplest thing to do is just set up two or more squid machines independently and let Frontier handle it by making a small addition to the frontier client configuration to have the client do the load balancing (described for CMS in the section on multiple squid servers). If there are many thousands of job slots, hardware-based load balancers can be easily overloaded, so DNS-based or client-based load balancing will probably be called for.

6) Can I put squid behind a NAT?

Possibly, but if so it should not be the same NAT shared by the worker nodes, otherwise if the squid fails it becomes very difficult to tell on the upstream servers whether it is a badly performing squid or direct connections from the worker nodes. It is much better for the squid to be on a machine with its own public IP address.

Software

The instructions below are for the frontier-squid rpm version >= 2.7STABLE9-23.1 on a Scientific Linux version 5 or 6 based system. The rpm is based on the frontier-squid source tarball, and there are also instructions for installing directly from the frontier-squid tarball available. Please see the rpm Release Notes for details on what has changed in recent versions. If, for some reason, you prefer to use a non frontier-squid distribution of squid, see MyOwnSquid.

Puppet

A puppet module for configuring frontier-squid is available on puppet-forge which understands a lot of the following instructions. If you're using puppet, check there first.

Preparation

By default the frontier-squid rpm installs files with a "squid" user id and group. If they do not exist, the rpm will create them. If your system has its own means of creating logins you should create the login and group before installing the rpm. If you want the squid process to use a different user id (historically it has been "dbfrontier"), then for example before installing the rpm create the file `/etc/squid/squidconf` with the following contents:

```
export FRONTIER_USER=dbfrontier
export FRONTIER_GROUP=dbfrontier
```

where you can fill in whichever user and group id you choose.

Installation

First, if you have not installed any frontier rpm before, execute the following command as the root user:

```
# rpm -Uvh http://frontier.cern.ch/dist/rpms/RPMS/noarch/frontier-release-1.1-1.noarch.rpm
```

If it warns about creating `/etc/yum.repos.d/cern-frontier.repo.rpmnew`, then move that file into place:

```
# mv /etc/yum.repos.d/cern-frontier.repo.rpmnew /etc/yum.repos.d/cern-frontier.repo
```

Next, install the package with the following command:

```
# yum install "frontier-squid-2*"
```

Set it up to start at boot time with this command:

```
# chkconfig frontier-squid on
```

Configuration

Custom configuration is done in `/etc/squid/customize.sh`. That script invokes functions that edit a supplied default squid.conf source file to generate the final squid.conf that squid sees when it runs. Comments in the default installation of customize.sh give more details on what can be done with it. Whenever `/etc/init.d/frontier-squid` runs it generates a new squid.conf if customize.sh has been modified.

It is very important for security that squid not be allowed to proxy requests from everywhere to everywhere. The default customize.sh allows incoming connections only from standard private network addresses and allows outgoing connections to anywhere. If the machines that will be using squid are not on a private network, change customize.sh to include the network/maskbits for your network. For example:

```
setoption("acl NET_LOCAL src", "131.154.0.0/16")
```

The script allows specifying many subnets - just separate them by a blank. If you would like to limit the outgoing connections please see the section below on restricting the destination.

If you want to, you can change the cache_mem option to set the size squid reserves for caching small objects in memory, but don't make it more than 1/8th of your hardware memory. The default 128 MB should be fine, leaving a lot of memory for disk caching by the OS, because squid performs better for large objects in disk cache buffers than in its own internal memory cache.

Change the size of the cache_dir (the third parameter) to your desired size in MB. The default is only 10 GB which is rather stingy. For example, for 100 GB set it to this:

```
setoptionparameter("cache_dir", 3, "100000")
```

Now that the configuration is set up, start squid with this command:

```
# service frontier-squid start
```

To have a change to customize.sh take affect while squid is running, run the following command:

```
# service frontier-squid reload
```

Moving disk cache and logs to a non-standard location

Often the filesystems containing the default locations for the disk cache (`/var/cache/squid`) and logs (`/var/log/squid`) isn't large enough and there's more space available in another filesystem. To move them to a new location, simply change the directories into symbolic links to the new locations while the service is stopped. Make sure the new directories are created and writable by the user id that squid is running under. For example if /data is a separate filesystem:

```
# service frontier-squid stop
```

```
# mv /var/log/squid /data/squid_logs
# ln -s /data/squid_logs /var/log/squid
# rm -rf /var/cache/squid/*
# mv /var/cache/squid /data/squid_cache
# ln -s /data/squid_cache /var/cache/squid
# service frontier-squid start
```

Alternatively, instead of creating symbolic links you can set the `cache_log` and `coredump_dir` options, the second parameter of the `cache_dir` option, and the first parameter of the `access_log` option in /etc/squid/customize.sh. For example:

```
setoption("cache_log", "/data/squid_logs/cache.log")
setoption("coredump_dir", "/data/squid_cache")
setoptionparameter("cache_dir", 2, "/data/squid_cache")
setoptionparameter("access_log", 1, "daemon:/data/squid_logs/access.log")
```

It's recommended to use the "daemon:" prefix on the access_log path because that causes squid to use a separate process for writing to logs, so the main process doesn't have to wait for the disk. It is on by default for those who don't set the access_log path.

Changing the size of log files retained

The access.log is rotated each night, and also if it is over a given size (default 5 GB) when it checks each hour. You can change that value by exporting the environment variable SQUID_MAX_ACCESS_LOG in `/etc/sysconfig/frontier-squid` to a different number of bytes. You can also append M for megabytes or G for gigabytes. For example for 20 gigabytes each you can use:

```
export SQUID_MAX_ACCESS_LOG=20G
```

By default, frontier-squid compresses log files when they are rotated, and saves up to 9 access.log.N.gz files where N goes from 1 to 9. In order to estimate disk usage, note that the rotated files are typically compressed to a bit under 15% of their original size, and that the uncompressed size can go a bit above $SQUID_MAX_ACCESS_LOG because the cron job only checks four times per hour. For example, for SQUID_MAX_ACCESS_LOG=20G the maximum size will be a bit above 20GB plus 9 times 3GB, so allow 50GB to be safe.

If frontier-awstats is installed (typically only on central servers), an additional uncompressed copy is also saved in access.log.0.

An alternative to setting the maximum size of each log file, you can leave each log file at the default size and change the number of log files retained, for example for 50 files (about 6GB total space) set the following in `/etc/squid/customize.sh`:

```
setoption("logfile_rotate", "50")
```

It is highly recommended to keep at least 3 days worth of logs, so that problems that happen on a weekend can be investigated during working hours. If you really do not have enough disk space for logs, the log can be disabled with the following in `/etc/squid/customize.sh`:

```
setoption("access_log", "none")
```

Then after doing `service frontier-squid reload` (or `service frontier-squid start` if squid was stopped) remember to remove all the old access.log* files.

On the other hand, the compression of large rotated logs can take a considerably long time to process, so if you have plenty of disk space and don't want to have the additional disk I/O and cpu resources taken during rotation, you can disable rotate compression by putting the following in `/etc/sysconfig/frontier-squid`:

```
export SQUID_COMPRESS_LOGS=false
```

That uses the old method of telling squid to do the rotation, which keeps access.log.N where N goes from 0 to 9, for a total of 11 files including access.log. When compression is turned off, the default SQUID_MAX_ACCESS_LOG is reduced from 5GB to 1GB, so override that to set your desired size. When converting between compressed and uncompressed format, all the files of the old format are automatically deleted the first time the logs are rotated.

See also the section Log compression interfering with squid operation below.

Enabling monitoring

The functionality and performance of your squid should be monitored from CERN using SNMP. The monitoring site is http://wlcg-squid-monitor.cern.ch/.

To enable this, your site should open incoming firewall(s) to allow UDP requests to port 3401 from 128.142.0.0/16, 188.184.128.0/17, and 188.185.128.0/17. When that is ready, register the squid with WLCG to start the monitoring. If you run multiple squid processes, each one will need to be separately monitored. They listen on increasing port numbers, the first one on port 3401, the second on 3402, etc. In order to monitor the extra ports, an exception has to be configured on the wlcg-squid-monitor.cern.ch machine, so please contact the squid support team to have that done.

Note: some sites are tempted to not allow requests from the whole range of IP addresses listed above, but we do not recommend that because the monitoring IP addresses can and will change without warning. Opening the whole CERN range of addresses has been cleared by security experts on the OSG and CMS security teams, because the information that can be collected is not sensitive information. If your site security experts still won't allow it, the next best thing you can do is to allow the aliases wlcgsquidmon1.cern.ch and wlcgsquidmon2.cern.ch. Most firewalls do not automatically refresh DNS entries, so you will also have to be willing to do that manually whenever the values of the aliases change.

Testing the installation

Download the following python script fnget.py (Do a right-click on the link and save the file as fnget.py)

Test access to a Frontier server at CERN with the following commands:

```
$ chmod +x fnget.py #(only first time)
$ ./fnget.py --url=http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier --sql="select 1 from
```

The response should be similar to this:

```
Using Frontier URL:  http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier
Query:  select 1 from dual
Decode results:  True
Refresh cache:  False

Frontier Request:
http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier?type=frontier_request:1:DEFAULT&encoding=BL

Query started:  10/30/12 20:04:09 CET
Query ended:  10/30/12 20:04:09 CET
Query time: 0.0179278850555 [seconds]

Query result:
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE frontier SYSTEM "http://frontier.fnal.gov/frontier.dtd">
<frontier version="3.29" xmlversion="1.0">
 <transaction payloads="1">
```

```
  <payload type="frontier_request" version="1" encoding="BLOBzip">
   <data>eJxjY2BgYDRkA5JsfqG+Tq5B7GxgEXYAGs0CVA==</data>
   <quality error="0" md5="5544fd3e96013e694f13d2e13b44ee3c" records="1" full_size="25"/>
  </payload>
 </transaction>
</frontier>


Fields:
     1     NUMBER

Records:
     1
```

This will return whatever you type in the select statement, for example change 1 to 'hello'. The "dual" table is a special debugging feature of Oracle that just returns what you send it.

Now to test your squid, replace yoursquid.your.domain in the following command with the name of your squid machine

```
$ export http_proxy=http://yoursquid.your.domain:3128
```

and perform the fnget.py test twice again. It should pass through your squid, and cache the response. To confirm that it worked, look at the squid access log (in /var/log/squid/access.log if you haven't moved it). The following is an excerpt:

```
128.220.233.179 - - [22/Jan/2013:08:33:17 +0000] "GET http://cmsfrontier.cern.ch:8000/Frontie
128.220.233.179 - - [22/Jan/2013:08:33:19 +0000] "GET http://cmsfrontier.cern.ch:8000/Frontie
```

Notice the second entry has a "TCP_MEM_HIT", that means the object was cached in the memory. Any subsequent requests for this object will come from the squid cache until the cached item expires.

Log file contents

Error messages are written to cache.log (in /var/log/squid if you haven't moved it) and are generally either self-explanatory or an explanation can be found with google.

Logs of every access are written to access.log (also in /var/log/squid if you haven't moved it) and the default frontier-squid format contains these fields:

1. Source IP address
2. User name from ident if any (usually just a dash)
3. User name from SSL if any (usually just a dash)
4. Date/timestamp query finished in local time, and +0000, surrounded by square brackets
5. The request method, URL, and protocol version, all surrounded by double quotes
6. The http status (result) code
7. Reply size including http headers
8. Squid request status (e.g. TCP_MISS) and heirarchy status (e.g. DEFAULT_PARENT) separated by a colon
9. Response time in milliseconds
10. The contents of the X-Frontier-Id header or a dash if none, then a space, then the contents of the cvmfs-info header, or a dash if none, all surrounded by double quotes (no client sends both so entries will always either start with "- " or end with " -")
11. The contents of the Referer header or a dash if none, surrounded by double quotes
12. The contents of the User-Agent header or a dash if none, surrounded by double quotes

Common issues

SELinux

- SELinux on RHEL 5 does not give the proper context to the default SNMP port (3401) (as of selinux-policy-2.4.6-106.el5). The command (as root):

  ```
  # semanage port -a -t http_cache_port_t -p udp 3401
  ```

  takes care of this problem.

- If squid has difficulty creating cache directories on RHEL 6 or RHEL 7, like for example:

  ```
  # service frontier-squid start

      Generating /etc/squid/squid.conf
      Initializing Cache...
      2014/02/21 14:43:53| Creating Swap Directories
      FATAL: Failed to make swap directory squid/00: (13) Permission denied
      ...
      Starting 1 Frontier Squid...
      Frontier Squid start failed!!!
  ```

  Then if SELinux is enabled and you want to leave it on try the following command:

  ```
  # restorecon -R
  ```

  And start `frontier-squid` again.

- If the suggestions above fail, our recommendation is to disable SELinux.

Inability to reach full network throughput

If you have a CPU that can't quite keep up with full network throughput, we have found that up to an extra 15% throughput can be achieved by binding the single-threaded squid process to a single core, to maximize use of the per-core on-chip caches. This is not enabled by default, but you can enable it by putting the following in `/etc/sysconfig/frontier-squid`:

```
export SETSQUIDAFFINITY=true
```

If that little boost isn't enough, try running multiple squid processes on the same machine. That also enables SETSQUIDAFFINITY option.

Log compression interfering with squid operation

Log compression has been observed on at least one machine to interfere with squid operation. That was an old 10-gbit machine with slow disks, high traffic, and 3 squid processes. These are some possible mitigations. Details of how to do many of these things are in the section Changing the size of log files retained section above.

1. Make sure there's a "daemon:" prefix on the access_log if you have changed its value.
2. Reduce the max log size before compression and increase the number of log files retained, to decrease the length of time of each log compression.
3. Disable compression if you have the space.

4. As root run `ionice -c1 -p PID` for the pid listed in squid.pid (default /var/run/squid/squid.pid) for each squid process run. This raises their I/O priority above ordinary filesystem operations.
5. Disable the access log completely.

Running out of file descriptors

By default, frontier-squid makes sure that there are at least 4096 file descriptors available for squid, which is usually enough. However, under some situations where there are very many clients it might not be enough. When this happens, a message like this shows up in cache.log:

```
WARNING! Your cache is running out of filedescriptors
```

There are two ways to increase the limit:

1. Add a line such as `ulimit -n 16384` in `/etc/sysconfig/frontier-squid`.
2. Set the `nofile` parameter in `/etc/security/limits.conf` or a file in `/etc/security/limits.d`. For example use a line like this to apply to all accounts:

   ```
   * - nofile 16384
   ```

   or replace the '*' with the squid user name if you prefer.

Alternate configurations

Restricting the destination

The default behavior is to allow the squid to be used for any destination. There are some pre-defined access controls commented out for the most common destinations on the WLCG. They are

1. CMS_FRONTIER - CMS Frontier conditions data servers
2. ATLAS_FRONTIER - ATLAS Frontier conditions data servers
3. MAJOR_CVMFS - the major WLCG CVMFS stratum 1 servers

In addition, there are two commented out lines using a general RESTRICT_DEST access control which you can use to set a regular expression that restricts connections to any set of hosts of your choice.

To use one of the pre-defined access controls, use two lines like this (for example with CMS_FRONTIER):

```
uncomment("acl CMS_FRONTIER")
insertline("^# http_access deny !RESTRICT_DEST", "http_access deny !CMS_FRONTIER")
```

To use a combination of two of the pre-defined acls, use "http_access allow" followed by "http_access deny !", for example:

```
uncomment("acl CMS_FRONTIER")
uncomment("acl MAJOR_CVMFS")
insertline("^# http_access deny !RESTRICT_DEST", "http_access allow CMS_FRONTIER")
insertline("^# http_access deny !RESTRICT_DEST", "http_access deny !MAJOR_CVMFS")
```

If for some reason you want to have a different destination or destinations you can instead use a regular expression with the RESTRICT_DEST lines, for example:

```
setoptionparameter("acl RESTRICT_DEST", 3, "^(((cms|atlas).*frontier.*)\\.cern\\.ch)|frontier
uncomment("http_access deny !RESTRICT_DEST")
```

Once you have restricted the destination, it isn't so important anymore to restrict the source. If you want to leave it unrestricted you can change the NET_LOCAL acl to 0.0.0.0/0 (unless you want to restrict both):

```
setoption("acl NET_LOCAL src", "0.0.0.0/0")
```

Running multiple squid processes on the same machine

If you have either a particularly slow machine or a high amount of bandwidth available, you may not be able to get full network throughput out of a single squid process. For example, our measurements with a 10 gigabit interface on a 2010-era machine with 8 cores at 2.27Ghz showed that 3 squids were required for full throughput.

Multiple squids can be enabled very simply by doing these steps:

- Stop frontier-squid and remove the old cache and logs
- Create subdirectories under your cache directory called 'squid0', 'squid1', up to 'squidN-1' for N squids, making sure they are writable by the user id that your squid runs under
- Start frontier-squid again. This will automatically detect the extra subdirectories and start that number of squid processes. It will create corresponding log subdirectories and /var/run/squid subdirectories, and generate a separate squid configuration file for each process in /etc/squid/.squid-N.conf. It will also assign each squid process to a particular core as described above.

When running multiple squids, all of the memory & disk usage is multiplied by the number of squids. For example, if you choose a cache_dir size of 100GB, running 3 squids will require 300GB for cache space. All the squids listen on the same port and take turns handling requests. Only squid0 will contact the upstream servers; the others forward requests to squid0 (this can be changed, see the next section).

If you want to revert to a single squid, reverse the above process including cleaning up the corresponding log directories, /var/run/squid subdirectories, and the generated configuration files.

Running independent squids on the same machine

By default multiple squids are configured so that only one of them will read from upstream servers, and others read from that squid. To disable that feature and instead have each separately read from the upstream server, you can put the following in `/etc/sysconfig/frontier-squid`:

```
export SQUID_MULTI_PEERING=false
```

They still all share the same basic configuration, however they can be used independently by accessing http_port-1, http_port-2, etc. For example if the default http_port is not changed, they all listen on port 3128, but then they each individually listen on port 3127, 3126, etc., so traffic flows can be separated by directly using those ports. A common trick is to set the http_port to 3129, and then don't advertise that port (and perhaps block it in iptables), so one of the squids can be accessed on the usual port 3128.

Note that there is currently no mechanism to have a different administrator-controlled configuration for each of the independent squids.

The frontier-squid2 rpm

In addition to the frontier-squid rpm, there is also a frontier-squid2-2.7 rpm. This is identical to the corresponding frontier-squid-2.7 rpm except that all the squid directories and files in shared directories have a "2" suffix on them, for example there's a `/etc/squid2`, `/var/cache/squid2`, `/var/log/squid2`, and `/etc/init.d/frontier-squid2`. This rpm may be installed on the same machine as the frontier-squid rpm, but one or both must change their http_port and snmp_port options to avoid clashing with the other. Just do

`yum install frontier-squid2` to install, and add the "2" suffix in all the configuration instructions on this page.

Having squid listen on a privileged port

This package runs squid strictly as an unprivileged user, so it is unable to open a privileged TCP port less than 1024. The recommended way to handle that is to have squid listen on an unprivleged port and use iptables to forward a privileged port to the unprivileged port. For example, to forward port 80 to port 8000, use this:

```
# iptables -t nat -A PREROUTING -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8000
```

You can change the port that squid listens on with this in /etc/squid/customize.sh:

```
setoption("http_port","8000")
```

Personal squid on a desktop/laptop

If you want to install a Frontier squid on your personal desktop or laptop, just follow the same instructions as under Software above, except:

- For the NET_LOCAL acl, use "127.0.0.1/32"
- For the cache_dir size you can leave it at the default 10000 or even perhaps cut it down to 5000 if you want to.

Laptop disconnected network operation

If you want to be able to run a laptop disconnected from the network, add the following to customize.sh:

```
setoption("cachemgr_passwd", "none offline_toggle")
```

Then, load up the cache by running your user job once while the network is attached, and run the following command once:

```
squidclient mgr:offline_toggle
```

It should report "offline_mode is now ON" which will prevent cached items from expiring. Then as long as everything was preloaded and the laptop doesn't reboot (because starting squid normally clears the cache) you should be able to re-use the cached data. You can switch back to normal mode with the same command or by stopping and starting squid.

To prevent clearing the cache on start, put the following in `/etc/sysconfig/frontier-squid`:

```
export SQUID_CLEAN_CACHE_ON_START=false
```

If you do that before the first time you start squid (or if you ever want to clear the cache by hand), run this to initialize the cache:

```
# service frontier-squid cleancache
```

Responsible: DaveDykstra

This topic: Frontier > OldInstallSquid
Topic revision: r60 - 2016-12-14 - DaveDykstra