Building DQ2 0.3

Firstly check out the required dashboard modules in the directory you want

```
export CVSROOT=:ext:username@dashboard.cvs.cern.ch:/cvs/dashboard
cd dashboard/
cvs co arda.dashboard arda.dashboard.service-config arda.dashboard.cli arda.dashboard.common
```

Then DQ2

```
export CVSROOT=:ext:username@atlas-sw.cern.ch:/atlascvs
cd ../dq2-HEAD/
cvs co offline/DataManagement/DQ2
```

These are the connection parameters from outside cern, inside you can use eg `:kserver:atlas-sw.cern.ch:/atlascvs`

Set the PYTHONPATH to point to the destination stage directory in the DQ2 area, eg `offline/DataManagement/DQ2/stage`

Now build the dashboard in the correct order, staging to this dir

```
cd arda.dashboard/
python setup.py stage -s ../../dq2-HEAD/offline/DataManagement/DQ2/stage
cd ../arda.dashboard.common/
python setup.py stage -s ../../dq2-HEAD/offline/DataManagement/DQ2/stage
cd ../arda.dashboard.service-config/
python setup.py stage -s ../../dq2-HEAD/offline/DataManagement/DQ2/stage
cd ../arda.dashboard.cli/
python setup.py stage -s ../../dq2-HEAD/offline/DataManagement/DQ2/stage
```

A patch to dashboard's setup.py needs to be applied, this file should be copied from to the DQ2 stage dir

```
cp /afs/cern.ch/user/m/mbranco/public/stage.py stage/lib/dashboard/distutils/stage.py
```

The dq2.common package needs to be staged first with the -s option:

```
cd dq2.common
python setup.py stage -s ../stage
```

Then the other packages can be staged using the -l option to create symlinks, thus meaning no further staging is required after code changes, I use the following python script (stageall.py) to stage all the packages:

```
#!/usr/bin/env python

import os
import commands

dirs = os.listdir('.')

for dir in dirs:
    if dir.find('dq2.') == 0 or dir.find('panda.') == 0:
        print 'staging %s...' % dir

        s,o = commands.getstatusoutput('cd %s; python setup.py clean; python setup.py stage -l; p
        if s != 0:
            print o
            break
```

Then run

```
./stageall.py
```

Tests can be run from the stage area, for example

```
python stage/lib/panda/lrc/client/testcase/LRCClientTestSuite.py
```

Setting the PATH env var to stage/bin gives access to dq2-* and dashb-* commands.

Ubuntu install of RLS catalog service

This service serves as an interface to the Globus RLS used in NDGF for cataloging file replicas. In order that information on replicas in this catalog can be accessed outside NDGF without any dependencies, this http service was developed to handle query requests and contact the RLS service for information.

A test machine in Aalborg (francis.grid.aau.dk) with the Ubuntu 6.0.6 OS is used as a test installation of the service. These are the instructions to get the service running.

Externals

Apache and mod_python

```
apt-get install apache2
apt-get install libapache2-mod-python
```

Grid tools

The Nordugrid standalone client provides all the necessary Globus tools. It was installed following these instructions

http://www.nordugrid.org/documents/ng-client-install.html

I used the 0.6.0 Debian tarball

http://ftp.nordugrid.org/software/nordugrid-arc/releases/latest/debian-3.1/i386/nordugrid-arc-standalone-0.6.0-1.i386.tg

Now there is a bug in Globus threading where it doesn't work properly on some systems, causing commands to hang instead of terminating (only kill -9 can stop them). This is worked around by setting the following environment variable

```
export LD_ASSUME_KERNEL=2.2.5
```

Python

The default python version is 2.4.3

DQ2 installation

The version used was the current HEAD of DQ2 CVS (v0.3 from 16/4/07). To build DQ2 required the dashboard packages to be built. However it proved difficult to find some missing xml modules (eg libxml2) to make the dashboard compile so a tarball was copied over of a pre-compiled DQ2. This was compiled with python 2.3.4 but this did not seem to cause any problems running it with 2.4.3.

The configuration files go into `/opt/dq2/etc`. I put the following into `/opt/dq2/etc/panda.lrc.server/panda.lrc.server.cfg`

```
[panda.lrc.server]
db=
host=rls://atlasrls.nordugrid.org:39281
user=
passwd=
impl=rlscli
dsn=
```

The rlscli implementation uses the globus command line version of the service. To use the mysql version which queries on the LRC views of the RLS tables, set the impl to rls and add the database connection parameters. I think this conf file is the only one required but I copied the whole tree of conf files from another machine so can't be sure.

Apache configuration

The structure of the Debian packaged Apache is slightly different from that of RH4, so the setup on pcatlas264.cern.ch had to be adjusted slightly. The following should go in `/etc/apache2/sites-available/lrc.conf`

```
Listen 8001

NameVirtualHost 193.10.122.10:8001
<VirtualHost 193.10.122.10:8001>

 ServerName rls1.ndgf.org
 DocumentRoot /var/www/apache2-default

 <Directory /var/www/apache2-default/lrc>

    <Limit GET POST PUT DELETE OPTIONS>
      Order allow,deny
      Allow from all
    </Limit>

    # mod_python
    PassEnv             LD_LIBRARY_PATH PYTHONPATH
    PythonPath          "sys.path+['/opt/lrc/lib']"
    SetHandler          mod_python
    PythonHandler       mod_python.publisher
    PythonDebug         On

  </Directory>

</VirtualHost>
```

This runs the service on port 8001. The directory in the PythonPath setting is the stage area of DQ2. A symlink needs to be created to this in the sites-enabled dir

```
ln -s /etc/apache2/sites-available/lrc.conf /etc/apache2/sites-enabled/lrc.conf
```

To put the webservice into action create a symbolic link to the ws in the stage area:

```
mkdir /var/www/apache2-default/lrc
ln -s /opt/lrc/lib/panda/lrc/server/ws_lrc.py /var/www/apache2-default/lrc/ws_lrc.py
```

Setup and running

A Grid proxy cert needs to be in place for the service to call globus-rls-cli, and this needs to be owned by the user www-data and have permissions 600. This is probably not the nicest way to do it...

```
grid-proxy-init -valid [long time]
sudo cp /tmp/x509up_u0000 /etc/apache2/
sudo chown www-data /etc/apache2/x509up_u0000
```

sudo does not take env variables such as LD_LIBRARY_PATH from the user's environment so to start and stop apache we need to use a small script (I called it apache.sh) to set the environment:

```
#!/bin/sh

if [ ! $1 ]; then
  echo "must specify start|stop|restart"
  exit 1
fi

export LD_ASSUME_KERNEL=2.2.5
export X509_USER_PROXY=/etc/apache2/x509up_u1054.apache
export PYTHONPATH=/user/dcameron/stage/lib:/var/www/apache2-default/dq2

cd nordugrid-arc-standalone-0.6.0
. setup.sh

apache2ctl $1
```

To start/stop/restart apache:

```
sudo ./apache.sh start|stop|restart
```

Clients

The client package is the panda LRC client package which can be used for LRC and RLS. In the configuration file (eg `~/.dq2/etc/panda.lrc.client/panda.lrc.client.cfg`) put

```
[panda.lrc.client]
insecure=http://francis.grid.aau.dk:8001/
secure=
```

The RLS service is a service for querying only so there is no secure part to it.

-- DavidCameron - 18 Apr 2007

This topic: Sandbox > DavidCameronSandbox
Topic revision: r10 - 2007-10-05 - DavidCameron