ATLAS computing tricks

A bunch of random tricks, some SLAC-related, many general to ATLAS

If you leave a window open too long, your kerberos ticket will expire, so you can't write to your /afs home directory.
Get a new ticket with:

```
kinit <username>
```

If you need a particular database release, for running over data typically,
you can set the database release you want to use by adding to your cmthome/requirements file:

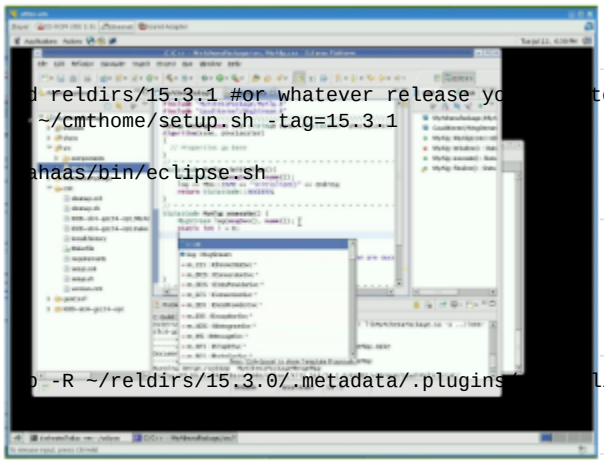
```
 set DBRELEASE_OVERRIDE 7.1.1
```

I still get a "Word too long" message sometimes after setting up an ATLAS release.
It seems to be from the PATH variable getting over a certain length that even bash can't handle.
You can fix it with this, which turns all the /afs/slac.stanford.edu to just /afs/slac, which works just as well:

```
export PATH=`echo $PATH | sed s%.stanford.edu%%g`
```

To kill ALL your batch jobs at SLAC:

```
for j in `bjobs | cut -f 1 -d " "`; do bkill $j; echo $j; done
```



To run eclipse (see this page):

This version at SLAC already has SVN and the latest athena plugin setup!

By the way, when starting a new workspace, you can copy over your old settings (fonts, whatever) using:

If you want to try Andy's settings, just copy them from ~ahaas/reldirs/15.3.0/

(Don't forget to go to Window... Preferences... CMT Plugin... and update the "tag" to the new release after copying settings from an old workspace!)

If you use "konsole" as your terminal, you can make new tabs, and switch tabs with:

```
<shift>+<left arrow> or <shift>+<right arrow>
```

VNC lets you keep open a virtual desktop on a machine, and connect to it remotely. To start a desktop on a machine:

```
vncserver #there are various options for desktop size, etc., like "-geometry 1600x1200"
#vncserver -h (gives a full list of options)
```

To start a normal KDE session, edit your .vnc/xstartup file and add "startkde":

```
xsetroot -solid grey
vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
startkde
```

You should also move your .vnc directory to a place that *doesn't require you to have a kerberos token*

```
mkdir /u1/<username>; mv .vnc /u1/<username>/ ; ln -s /u1/<username>/.vnc .vnc
```

And make sure your .vnc/passwd file is only readable by you (even though it's kind of encrypted):

```
chmod go-r .vnc/passwd
```

Then you use vncviewer (on some other machine, probably!) to connect to your session.
Use the desktop id which it said when you ran vncserver (by default is ":1").

There's a lot more space in /nfs/slac/g/atlas/u01/users:

```
mkdir /nfs/slac/g/atlas/u01/users/<username>
cd; ln -s /nfs/slac/g/atlas/u01/users/<username> nfs2
```

Do this in a release, and then you can always just grep the packages.txt file to see where things are, or what versions are needed:

```
cmt show packages > packages.txt
```

Sometimes a digi job won't work (in 15.3.0?) because "chappy fails" on the input file.
The problem can be fixed by adding the right python directory to your path:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/afs/slac/g/atlas/b/sw/lcg/external/Python/2.5.4/slc4_ia32_gcc34/lib
```

One of my favorites, this will do a "fast" build, if you've only changed a src file:

```
cd cmt; make QUICK=1; cd ..
```

So useful for joining together lots of ROOT files from many jobs into a single ROOT file:

```
hadd -h #show how to use
hadd -f step.root */*step.root #for instance
```

Sometimes when running over data it helps to put a link in your running directory:

```
mkdir sqlite200; ln -s /afs/cern.ch/user/a/atlcond/coolrep/sqlite200/COMP200.db sqlite200/ALLP200.db
```

Can run these on a POOL file to see what StoreGate keys are in there:

```
checkFile.py <file>
checkSG.py <file>
```

This shows the versions of databases and releases used to process a file:

```
dumpVersionTags.py <file>
```

Can put this in a bash script near the top, to check if you have a GRID cert:

```
voms-proxy-info
if [ $? -eq 1 ] ; then echo You need to get a GRID cert; exit; fi
```

To get a list of filenames (to load into athena) from a given dataset (passed in as $1):

```
dq2-ls -f -p -H $1 | sed "s%srm://osgserv04.slac.stanford.edu:8443/srm/v2/server?SFN=/xrootd/atlas/%filelist +=
[\"root://atl-xrdr//atlas/xrootd/%g" | sed "s%$%\"]%g" | grep xrootd
(may also need to add "-L SLACXRD_USERDISK" or wherever there is a complete replica of the dataset...)
```

This gets a ROOT file with info on a given data run (mag field configuration, #events, streams, etc.):

```
#!/bin/bash
#gets a ROOT file with info on a run (takes run number as argument)
wget http://atlas-runquery.cern.ch/query.py?q=find+run+${1}+%2F+show+all+%2F+nodef
sleep 3
wget http://atlas-runquery.cern.ch/data/atlrunquery.root
rm -v query.py\?q\=find+run*
```

There's a few athena options (I like the -s and -c etc.):

```
athena -h #show athena help
```

To specify more than one parameter on the input line:

```
athena -c "DECAY=True; TIMESHIFT=25;" share/jobOptions.pythiaRhad.py
```

Sometimes my JiveXML files get messed up and can't be read, due to a binary character in the trigger string. Fix it with:

```
#!/bin/bash
for f in JiveXML*; do sed -i '/Obeys/d' $f ; done
for f in JiveXML*; do sed -i 's/<trigInfoStreamTag>/<trigInfoStreamTag>fixJive/' $f ; done
```

If a script is expecting a particular ATLAS release version, you can check it with:

```
#!/bin/bash
if [ $AtlasVersion != "15.3.1" ]; then echo "Go to a cmthome and do . setup.sh -tag=15.3.1"; exit; fi
```

Check out all the CSC transforms:

```
csc_<tab> #will show them all... look at csc_atlasG4_trf.py, csc_digi_trf.py, csc_reco_trf.py, etc...
```

This will actually put your files into the catalog, so you don't get annoying warnings:

```
pool_insertFileToCatalog <file>
```

When running on the batch farm, you really should write things out into the /scratch area on the batch node during the job, and then cp it all back at the end of the job, to prevent hammering on NFS. Here's an example script:

```
#!/bin/bash

. /u/at/ahaas/cmthome/setup.sh -tag=15.3.0 #setup the ATLAS release

#make a variable name for the directory which is the number of seconds since 1975
export d=`date +%s`; echo $d

#make a scratch area on the local machine
mkdir /scratch/ahaas; mkdir /scratch/ahaas/${d}; mkdir /scratch/ahaas/${d}/temp; cd /scratch/ahaas/${d}; pwd;

#run your stuff here
athena.py /u/at/ahaas/reldirs/15.3.0/Generators/Pythia_i/share/jobOptions.pythiaRhad.py >  temp/pyth.log.txt
#all outputs of the athena job that are important should get put into the temp directory too...

#copy back results in the temp directory to some nfs directory
pwd; ls -lh temp
export dd=`date +%s` ; echo $dd #this will add the end time of the job to the temp direcory output name
if [ -a /nfs/slac/g/atlas/u01/users/ahaas/temp/rh_production_stripped_files/temp_${d}_${dd} ]
 then echo Destination directory already exists
 else mv -v /scratch/ahaas/${d}/temp /nfs/slac/g/atlas/u01/users/ahaas/temp/rh_production_stripped_files
/temp_${d}_${dd}
fi
cd; pwd; rm -rfv /scratch/ahaas/${d}
echo done
```

You could run this batch script above (put in a file called myjob.sh) with:

```
chmod +x myjob.sh #don't forget to make the script executable
bsub -q xlong -R rhel40 -J myjobname time myjob.sh
```

The xlong queue will kill your job after 177.6 hours of CPU time in "SLAC units"... which is about ~15 hours of real CPU time.
See all queues with "bqueues". You can see the the details of a queue with "bqueues -l xlong".
Note the "-R rhel40" above, which forces your job onto a machine compatible with the ATLAS releases (gcc34, RHEL4). It's "rhel50" for RHEL5 machines.
"bhosts -R rhel40" will show you which batch nodes are in that list.
"lsinfo -r" will show you all resourse lists, like the rhel40 one.
Check your batch jobs with "bjobs".

Here's a much simpler "myjob.sh" script you could use for small things (just run it from a directory with the jobOptions.G4Atlas_Sim.py file in it):

```
#!/bin/bash
. ~/cmthome/setup.sh -tag=15.3.0
athena.py jobOptions.G4Atlas_Sim.py > athena_sim.out.txt
```

Don't forget to set random seeds when you're generating MC with custom scripts! This python code helps set some:

```
import random
random.seed()
R1=random.randint(0,100000000)
R2=random.randint(0,100000000)
R3=random.randint(0,100000000)
R4=random.randint(0,100000000)
PYTHR = "PYTHIA "+str(R1)+" "+str(R2)
PYTHRI = "PYTHIA_INIT "+str(R3)+" "+str(R4)
print PYTHR
print PYTHRI
```

This will print out the MC truth:

```
from AthenaCommon.AlgSequence import AlgSequence
job=AlgSequence()
from TruthExamples.TruthExamplesConf import PrintMC
PrintMC.PrintStyle = "Vertex"
job += PrintMC()
```

To transfer large files from CERN to SLAC (or SLAC to CERN, etc.), use **bbcp**. You can get upto ~1GB/min!
This page descibes its usage.

If a root file wasn't closed, you'll get a lot of "Trying to recover..." messages every time you open it.
To recover what you can, once and for all:

```
root
[0] TFile f("<filename>","update")
[1] .q
```

To find info on MC or data, use ATLAS Metadata Interface (AMI). Click on "AMI Dataset Search", then enter your search term (name or datasetNumber or whatever) to find all datasets, for instance "Zmumu" or "105145".
Under "dataType" you can group them by clicking the little icon. Choose "EVNT" to see generator info for MC.
Click on "details" in the left column to see cross-section info, the jobOptions used, etc. You can see the jobOptions file via (after setting up an ATLAS release):

```
 get_files -jo -list MC8.105145.PythiaZmumu.py
```

(Do "get_files -h" for more info on this useful command...)

The hepix script that gets run when you start bash now sets up the FRONTIER_SERVER variable (to $ATLAS_FRONTIER_CONF).
This *dramatically* improves the geometry and conditions database access speed *in release >=15.40*, especially when processing read data!

For CMT component libraries, like an Analysis_Skeleton package which reads AODs,
you can save some time during the build step (even during the QUICK=1 build!) by
not bothering to build the static library (since you only ever call it from athena.py), by adding the "-no_static" flag in your package/cmt/requirements file:

```
library StoppedGluinoAnalysis -no_static *.cxx components/*.cxx
```

For reading data from real data ESDs that are very recent (more recent than the database release),
you may have to access the CERN POOL data directly. Set this environment variable before you run athena
(in release >15.4.0):

```
export ATLAS_POOLCOND_PATH="/afs/cern.ch/atlas/conditions/poolcond/catalogue/"
```

(This does not slow down the database access for the job significantly...)

If you're running on the batch queue, you don't have access to outside afs however, so you need to use a local copy:

```
export ATLAS_POOLCOND_PATH="/nfs/slac/g/grid/osg/app/atlas_app/atlas_rel/local/conditions"
```

See: https://twiki.cern.ch/twiki/bin/view/Atlas/AthenaDBAccess

---

How to see what tags / conditions are in a database:

```
AtlCoolConsole.py "COOLOFL_Indet/COMP200;readoracle"
>>> ls
>>> tracetags . COMCOND-ES1PST-001-00
>>> cd Indet
>>> ls
>>> listtags Align
>>> usetag InDetAlign-ES1-UPD1-00
>>> more Align
```

This will show you something like:

```
[0,0] - [91396,0] (230) [PoolRef (String4k) : [DB=202EC302-A585-DD11-9F25-0030487C8DC4]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
[91396,0] - [96686,0] (230) [PoolRef (String4k) : [DB=8A61369D-FD95-DD11-8D2E-0030487C8DC4]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
[96686,0] - [140050,0] (230) [PoolRef (String4k) : [DB=92BAC4F9-35B2-DD11-B20D-0030487C8DC4]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
[140050,0] - [142393,0] (230) [PoolRef (String4k) : [DB=74981861-8AD2-DE11-95BD-001CC466D3D3]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
[142393,0] - [145006,0] (230) [PoolRef (String4k) : [DB=ACBDD470-96E8-DE11-8575-001320A52E33]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
[145006,0] - [2147483647,4294967295] (230) [PoolRef (String4k) : [DB=88B921CA-EB05-DF11-ADF2-000423D59BB6]
[CNT=CollectionTreeAlignableTransform][CLID=E779C6B5-3F2A-473E-B35E-6CCB345E0665][TECH=00000202][OID=00000003-
00000021]]
```

which is the POOL conditions file id for each IOV [run,lumiblock] range. You can check your log files to make sure the right conditions file is being loaded in for each folder.

If you find your athena (local) job crash at SLAC due to some db issue, check your environment variable through "echo $FRONTIER_SERVER". If it shows "(proxyurl=http://atl-dbcache1:23128)(serverurl=http://frontier.racf.bnl.gov:8000/frontieratbnl)" then it means you are setting up the frontier server in out-dated way. Try following:

```
export FRONTIER_SERVER="(serverurl=http://atlasfrontier-ai.cern.ch:8000/atlr)(serverurl=http://lcgft-atlas.
gridpp.rl.ac.uk:3128/frontierATLAS)(serverurl=http://frontier-atlas.lcg.triumf.ca:3128/ATLAS_frontier)
(serverurl=http://ccfrontier.in2p3.fr:23128/ccin2p3-AtlasFrontier)"
```

And then re-run your job (and cross your fingers). More details on this can be found in

https://twiki.cern.ch/twiki/bin/view/AtlasComputing/FroNTier#Configuring_local_jobs_to_use_Fr

---

Please add tricks of your own!
