System Administrators

The Setup and Maintenance Guide details hardware and software requirements, how to bring up a new HUB, upgrade system software, etc.

Installation

What is HUBzero?

HUBzero is a platform used to create dynamic web sites for scientific research and educational activities. With HUBzero, you can easily publish your research software and related educational materials on the web. Powerful middleware serves up interactive simulation and modeling tools via your web browser. These tools can connect you with rendering farms and powerful Grid computing resources.

Minimum System Requirements

HUBzero installations require one or more dedicated physical hosts running Debian GNU/Linux 5.0.

Other distributions might theoretically work with some modification, although they would be totally unsupported.

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 16 Gigabytes of RAM and a terabyte of disk

It is possible to run HUBzero inside of a virtual machine such as ones created by VMware and VirtualBox. While fully functional there will would significant performance and resource limitations in such an environment.

Target Audience

This document and the installation of a HUBzero system has a target audience of experienced Linux administrators (preferably experienced with Debian GNU/Linux).

Debian GNU/Linux

Install Basic Operating System

The latest version of Debian GNU/Linux 5.0 (5.0.5 as of this writing) should be installed on each physical host used by a HUBzero installation.

To install Debian GNU/Linux, you can easily obtain a copy, and then follow the installation instructions for your architecture (only 64bit [AMD64] is currently supported).

Installing Debian GNU/Linux using a a small bootable CD is the simplest method.

When installing Debian GNU/Linux be sure to do the following:

- Ensure the disk(s) are partitioned to have at least:
    - A bootable partition at least 100.0 GB in size for the root filesystem.
    - An empty partition at least 50.0 GB in size (note the device name of this partition for later)
    - An appropriately sized swap partition.
- When prompted to select an installation package just select "Standard System", other packages will be added later

When the installation is complete your system will reboot into a minimal Debian GNU/Linux system.

Don't forget to remove your installation media and/or change your server's boot media order if you changed them prior to installation.

Set hostname

Optional. If you didn't specify the fully qualified domain name when running setup you will need to set it here.

HUBzero expects the `hostname` command to return the fully qualified hostname for the system.

```
# hostname myhub.org
```

To make the change permanent you must also edit the file /etc/hostname, this can simply with:

```
# echo "myhub.org" > /etc/hostname
```

Delete local user

If you created a local user account when installing the operating system now would be a good time to delete it before it causes you future problems.

```
# deluser username
```

Configure Networking

Optional. If you didn't configure networking during installation you will need to do so now.

For help with networking setup try this link.

Setting up your IP address.

The IP addresses associated with any network cards you might have are read from the file **/etc/network/interfaces**. This file has documentation you can read with:

```
# man interfaces
```

A sample entry for a machine with a static address would look something like this:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
 address 192.168.1.90
 gateway 192.168.1.1
 netmask 255.255.255.0
 network 192.168.1.0
 broadcast 192.168.1.255
```

Here we've setup the IP addresss, the default gateway, and the netmask.

For a machine running DHCP the setup would look much simpler:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface - use DHCP to find our address
auto eth0
iface eth0 inet dhcp
```

(If you're using a DHCP based setup you must have a DHCP client package installed - usually one of pump or dhcp-client.)

If you make changes to this file you can cause them to take effect by running:

```
# /etc/init.d/networking restart
```

Setting up DNS

Use whatever nameserver and other options as recommended by your ISP. If you used DHCP to set up networking it is likely this has already been set.

When it comes to DNS setup Debian doesn't differ from other distributions. To cause your machine to consult with a particular server for name lookups you simply add their addresses to /etc/resolv.conf.

For example a machine which should perform lookups from the DNS server at IP address 192.168.1.10 would have a resolv.conf file looking like this:

```
nameserver 192.168.1.10
```

Configure Advanced Package Tool

Now configure what debian distribution mirror to use and the location of the HUBzero package repository by editing /etc/apt/sources.list to look like:

```
deb http://ftp.us.debian.org/debian/ lenny main
deb-src http://ftp.us.debian.org/debian/ lenny main
```

```
deb http://security.debian.org/ lenny/updates main
deb-src http://security.debian.org/ lenny/updates main

deb http://volatile.debian.org/debian-volatile lenny/volatile main
deb-src http://volatile.debian.org/debian-volatile lenny/volatile main

deb http://packages.hubzero.org/deb lenny main contrib non-free
deb-src http://packages.hubzero.org/deb lenny main contrib non-free
```

You will need to get and install the hubzero archive key to be able to verify packages from the hubzero archive:

```
# wget http://packages.hubzero.org/deb/hubzero-signing-
key.asc -q -O - | apt-key add -
```

Once the public key for http://packages.hubzero.org has been install you can then upgrade the current packages to their latest releases.

```
# apt-get update
# apt-get upgrade
```

SSH

Next we install fail2ban and ssh

```
# apt-get install fail2ban ssh
```

At this point you can continue configuration and setup remotely if that is more convenient.

Enable OpenVZ

If you are installing this in a VirtualBox VM you must enable PAE/NX support. Go to system ->

processor of your VM, select "Enable PAE/NX".

To use OpenVZ you must use an OpenVZ enabled kernel which is easily installed.

HUBzero makes extensive use of OpenVZ containers so it is recommended to just use the OpenVZ enabled kernel on all HUBzero servers.

```
# apt-get install linux-image-2.6-openvz-amd64
```

You will need to reboot the server to activate the new kernel.

```
# reboot
```

Once you have rebooted you can verify the new kernel is active

```
# uname -a
Linux myhub.hubzero.org 2.6.26-2-openvz-
amd64 #1 SMP Thu Nov 5 03:06:00 UTC 2009 x86_64 GNU/Linux
```

With the new kernel active you can remove the old one

```
# apt-get purge linux-image-2.6.26-2-amd64
```

Prepare Fileystem

The root filesystem ('/') runs with quotas disabled and contains the primary operating system for the server and for each OpenVZ container hosted on the server.

Each HUBzero server may use an addition partition for use appropriate to the function of the server (web document root, project data, home directories, etc).

If you did not create an empty partition during setup, create one now using your favorite disk partitioning tool. Be sure to note the device name for the partition you create as it will be used below.

Once you have an empty partition ready we can install a filesystem. Replace "/dev/PART" with the device name for the empty partition you have created (e.g., /dev/sda2). The command "fdisk -l" will list all paritions the system knows about.

```
# mke2fs -j /dev/PART
# e2fsck -f -C 0 /dev/PART
```

```
# mkdir /export
```

Then make sure the following line appears in /etc/fstab

```
/dev/PART     /export      ext3    defaults,quota,errors=remount-
ro     0     2
```

Then mount the new filesystem

```
# mount /export
```

Bind mount /home

Create a 'home' directory in our new /export filesystem. move the contents of the default home directory to the new location, then bind mount new location over the old.

```
# mkdir -p /export/home/myhub
# mv /home/* /export/home
# mount --bind /export/home /home
```

Bind mount /opt

Currently HUBzero uses the /opt directory for storing subversion and trac data as well as some of hubzero supporting software as well. We recognize this may not be the best organization.

Create a 'opt' directory in our new /export filesystem. move the contents of the default /opt directory to the new location, then bind mount new location over the old.

```
# mkdir /export/opt
# mv /opt/* /export/opt
# mount --bind /export/opt /opt
```

Bind mount /apps

Currently HUBzero uses the /apps directory for storing installed tools and other software that needs to be available to each execution container.

Create a 'apps' directory in our new /export filesystem and in the root filesystem. Then bind mount /export/apps over /apps.

```
# mkdir /export/apps
# mkdir /apps
# mount --bind /export/apps /apps
```

Bind mount /www

HUBzero uses the /www directory for storing the document root and supporting directories needed by the web server.

Create a 'www' directory in our new /export filesystem and in the root filesystem. Then bind mount /export/www over /www.

```
# mkdir -p /export/www
# mkdir /www
# mount --bind /export/www /www
```

Update /etc/fstab

Now edit **/etc/fstab** with the bind mounts we created above by adding the following lines

```
/export/opt  /opt        none    bind,defaults
0     0
/export/apps /apps       none    bind,defaults
0     0
/export/home /home       none    bind,defaults
0     0
/export/www  /www        none    bind,defaults
0     0
```

OpenLDAP

Install OpenLDAP

Install OpenLDAP

```
# apt-get install slapd
```

You will be prompted for an administrative password. This will be the LDAP administrator password and will be used anywhere that write permission to LDAP is required. This will get set again in the next step when we reconfigure OpenLDAP.

Reconfigure OpenLDAP

Debian's default configuration for OpenLDAP is sometimes not quite what you might want. If you want to use an LDAP base DN based off something other than the domain name used when configuring the host you will need to reconfigure the package.

```
# dpkg-reconfigure slapd
```

The reconfiguration script will allow you to change the LDAP base DN to be based on a different domain name. For example, "myhub.org" would become "dc=myhub,dc=org".

The reconfiguration script will then ask for the organization to use. This isn't important to us and can be set arbitrarily.

You will then be asked to enter a password for the admin account. You will need to remember this password for later configuration steps.

Accept the default "HDB" database backend type.

Do not remove database when slapd is purged. Sometimes during maintenance it can be useful to reinstall slapd without wiping out the database.

Move the old database out of the way.

Don't allow LDAPv2 protocol.

Install nscd

The Name Service Cache (nscd) will used later so we go ahead and install it here.

```
# apt-get install nscd
```

Install HUBzero LDAP Schema

```
# apt-get install hubzero-openldap
```

To enable this new schema edit **/etc/ldap/slapd.conf** and add the following line as the last "include" statue under the "Schema and objectClass definitions" comment toward the beginning of the file.

```
include         /etc/ldap/schema/hub.schema
```

Then restart OpenLDAP

```
# /etc/init.d/slapd restart
```

Initialize OpenLDAP Database

Several entries are expected to be prepopulated in OpenLDAP.

There is a script to do this, but the script has to be manually configured.

```
# cp /usr/lib/hubzero/openldap/HUB-INIT-SLAPD.tmpl HUB-INIT-SLAPD
```

Modify HUB-INIT-SLAPD and edit the five configuration lines near the beginning of the file:

```
base_dn=""
admin_pass=""
hubadmin_passhash=""
hubrepo_passhash=""
home_dir=""
```

- **base_dn** should be the base DN of your LDAP (e.g., "dc=myhub,dc=org")

- **admin_pass** should be the clear text password you set for the LDAP administrator.
- **hubadmin_passhash** should be the hashed password for the about to be created hubadmin account. You can hash a password using '/usr/sbin/slappasswd'
- **hubrepo_passhash** should be the hashed password for the about to be created hubrepo account. You can hash a password using '/usr/sbin/slappasswd'
- **home_dir** should be the home directoy you created earlier (eg., "/home/myhub").

Then run the configuration script

```
# sh ./HUB-INIT-SLAPD
```

This should prepopulate the databae enough to bootstrap HUBzero.

Configure PAM to use LDAP

```
# apt-get install libpam-ldap
```

- Use "ldap://127.0.0.1" as the URI
- Set the base DN to match how you configured OpenLDAP (eg., "dc=myhub,dc=org")
- Use LDAP v3
- Accept making local root data admin
- LDAP does not require login
- Specify the DN for the LDAP admin user (eg., "cn=admin,dc=myhub,dc=org")
- Enter admin password for LDAP

Modify **/etc/pam.d/common-auth** by commenting out the existing configuration then adding rules to allow authentication against LDAP.

```
#auth required pam_unix.so nullok_secure

auth sufficient pam_unix.so nullok_secure
auth sufficient pam_ldap.so try_first_pass
auth required pam_deny.so
```

Modify **/etc/pam_ldap.conf** by adding the following section (other mappings in this file should already be commented out).

```
# HUBzero Mappings
nss_base_passwd ou=users,?one
nss_base_shadow ou=users,?one?host=web
pam_filter host=web
pam_password crypt
```

```
nss_map_attribute uniqueMember member
nss_base_group ou=groups,dc=myhub,dc=org?sub
```

Be sure the BASEDN in the above matches that used by your configuration.

/etc/pam_ldap.secret contains the LDAP admin password and should only be readable by root.

Configure NSS to use OpenLDAP

```
# apt-get install libnss-ldap
```

- Specify the DN for the ldap admin account
- Specify the password for the ldap admin account

Modify **/etc/libnss-ldap.conf** by adding the following section. (other mappings in this file should already be commented out).

```
# HUBzero Mappings
nss_base_passwd ou=users,?one
nss_base_shadow ou=users,?one?host=web
pam_filter host=web
pam_password crypt
nss_map_attribute uniqueMember member
nss_base_group ou=groups,dc=myhub,dc=org?sub
```

Be sure the BASEDN in the above matches that used by your configuration.

Modify **/etc/nsswitch.conf**

```
passwd:         compat ldap
group:          compat ldap
shadow:         compat ldap
```

/etc/libnss-ldap.secret contains the LDAP admin password and should only be readable by root.

Test

```
# getent passwd
```

To test configuration. You should see entries for users 'hubrepo' and 'apps' toward the end of the list if everything is working correctly.

Troubleshooting

If you have a problem with the system apparently not recognizing up to date account or group information (eg., in the next section some people report receiving an error about unknown username 'hubadmin') you can nscd to flush it data cache and restart using the following commands:

```
# nscd -i passwd
# nscd -i group
# /etc/init.d/nscd restart
# getent passwd
```

If you still don't see the hubadmin account listed then re-read the instructions and check your work very carefully. These instructions assume a fresh install, if you are working with an existing LDAP/PAM/NSS installation you will have to do more advanced troubleshooting outside the scope of this documentation.

Create home directories

Create a home directory for the apps user

```
# mkdir /home/myhub/apps
# chown apps.public /home/myhub/apps
# chmod 0700 /home/myhub/apps
```

MySQL

Install MySQL Server

Install MySQL Server

```
# apt-get install hubzero-mysql
```

You will be prompted for an administrative password. This will be the MySQL administrator password. This will also fix the GRANT permissions on the default Debian debian-sys-maint mysql account.

Apache

Install Apache Web Server

Install Apache Web Server

```
# apt-get install hubzero-apache2
```

The default apache web site should now work and display "It Works!"

Enable default SSL site

```
# a2ensite default-ssl
# /etc/init.d/apache2 restart
```

The default apache ssl web site should now work (be sure to use https:// prefix) and display "It Works!"

The SSL certficate used by the default-ssl and (see next sections) the hub-ssl sites use the self signed "snakeoil" certificate that was installed by the ssl-cert package. This should only be used for testing and development. A commercial certificate should be purchased and installed for any site put into production.

Enable basic hub site

Enable the hub site, while disabling the apache default site.

```
# a2dissite default
# a2ensite hub
# /etc/init.d/apache2 reload
```

This configuration continues to use the default apache document root so the site should display the standard default "It Works!" page

It would be a good idea to restrict access to the web server via a firewall now. A web based installer will be installed later and it should only be accessed by the person setting up the site.

Enable basic hub SSL site

Enable the hub-ssl site, while disabling the default-ssl site.

```
# a2dissite default-ssl
# a2ensite hub-ssl
# /etc/init.d/apache2 reload
```

This configuration continues to use the default apache document root so the https site should display the standard default "It Works!" page

It would be a good idea to restrict access to the web server via a firewall now. A web based installer will be installed later and it should only be accessed by the person setting up the site.

PHP

Configure PHP

Edit **/etc/php5/apache2/php.ini** and set the display_errors parameter to 'Off' and log_errors to "On".

```
display_errors = Off
log_errors = On
```

then restart apache to enable everything.

```
# /etc/init.d/apache2 restart
```

Test

```
# echo "<?php phpinfo();?>" > /var/www/index.php
```

Go to "/index.php" on your site and you should see a php status page.

Delete the test page when you are done.

```
# rm /var/www/index.php
```

Mail

Install

We need to reconfigure exim4 to enable outgoing email (exim4 got installed earlier as a prerequisite for the mysql server).

```
# dpkg-reconfigure exim4-config
```

- Select "internet site; mail is sent and received directly using SMTP" then configure as appropriate for your site.
- Enter the FQDN of your site when asked
- Listen on all IP addresses (i.e., make list blank)
- Other destinations for which mail is accepted: should be made blank
- Domains to relay mail for: should be made blank
- Machines to relay mail for: should be made blank
- Keep num,ber of DNS-queries minimal (Dial-on-Demand): No
- Delivery method for local mail: mbox format in /var/mail/
- Split configuration into small files? No

This is just an example. Mail should be configured however the site needs. The CMS just expects to be able send outgoing email.

CMS

Global HUBzero Configuration

Edit/create the file **/etc/hubzero.conf**

```
[default]
site=myhub

[myhub]
DocumentRoot=/www/myhub
```

Install

Install the content management system.

```
# apt-get install hubzero-cms
# mkdir /www/myhub
# cp -rp /usr/lib/hubzero/cms/* /www/myhub
# chown -R www-data.www-data /www/myhub
```

Note: /www/myhub will be the document root of your site

Create Database

Create database for the HUBzero CMS to use

```
# /usr/bin/mysql --defaults-file=/etc/mysql/debian.cnf
Welcome to the MySQL monitor.  Commands end with ; or g.
Your MySQL connection id is 33
Server version: 5.0.51a-24+lenny2 (Debian)

Type 'help;' or 'h' for help. Type 'c' to clear the buffer.

mysql> CREATE DATABASE `myhub`;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE `myhub_metrics`;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> GRANT ALL PRIVILEGES ON `myhub`.* TO 'myhub
'@'%' IDENTIFIED BY 'xyzzy#1';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON `myhub_metrics`.* TO 'myhub
'@'%' IDENTIFIED BY 'xyzzy#1';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
```

Configure Apache for CMS

We are now going to tell Apache about the change of document root of the web server.

Modify **/etc/apache2/sites-available/hub *AND* /etc/apache2/sites-available/hub-ssl** replacing three instance of "/var/www" with "/www/myhub": follows

Change

```
DocumentRoot /var/www
...
<Directory /var/www>
...
</Directory>
...
<Directory /var/www/site/protected>
...
</Directory>
```

to

```
DocumentRoot /www/myhub
...
<Directory /www/myhub>
...
</Directory>
...
```

```
        <Directory /www/myhub/site/protected>
        ...
        </Directory>
```

then restart apache

```
# /etc/init.d/apache2 restart
```

Run CMS Installer

The Joomla/HUBzero installer will now appear on your website. Follow the instructions to get the basic site configured. Then disable the installation directory.

You can ignore the LDAP configuration screen as it is currently not used.

Press the "Install HUBzero Sample Data" button to have a basic site layout made [RECOMMENDED]

When you are done with the web installer you need to disable the web installation application:

```
# rm -fr /www/myhub/installation
```

Configure CMS

Some of the initial site configuration still needs be done manually.

```
# cp -p /www/myhub
/hubconfiguration.php-dist /www/myhub/hubconfiguration.php
```

Fill in the dozen or so parameters as needed and the core global site configuration is complete

```
<?php
class HubConfig {
    var $hubLDAPMasterHost = 'ldap://127.0.0.1';
    var $hubLDAPSlaveHosts = '';
    var $hubLDAPBaseDN = 'dc=myhub,dc=org';
    var $hubLDAPNegotiateTLS = '0';
```

```
    var $hubLDAPSearchUserDN = '';
    var $hubLDAPSearchUserPW = '';
    var $hubLDAPAcctMgrDN = 'cn=admin,dc=myhub,dc=org';
    var $hubLDAPAcctMgrPW = 'test';
    var $ipDBDriver = 'mysql';
    var $ipDBHost = 'db.nanohub.org';
    var $ipDBPort = '';
    var $ipDBUsername = 'geodb';
    var $ipDBPassword = 'ge0dbhub';
    var $ipDBDatabase = 'network';
    var $ipDBPrefix = '';
    var $hubShortName = 'myhub';
    var $hubShortURL = 'myhub.org';
    var $hubLongURL = 'http://myhub.org';
    var $hubSupportEmail = 'real@email.address';
    var $hubMonitorEmail = 'real@email.address';
    var $hubHomeDir = '/home/myhub';
    var $forgeName = 'myFORGE';
    var $forgeURL = 'https://myhub.org';
    var $forgeRepoURL = 'http://myhub.org';
    var $svn_user = 'hubrepo';
    var $svn_password = 'test';
}
?>
```

Be sure to use a real email address in hubSupportEmail and hubMonitorEmail, failure to do so will result in a failure of the hub to send out email properly.

Subversion

Install Subversion

```
# install --owner www-data --group www-
data --mode 0770 -d /opt/svn/tools
# touch /etc/apache2/svn.conf /etc/apache2/svn.bak
# chown www-data /etc/apache2/svn.conf /etc/apache2/svn.bak
```

The apache hub site configuration files are preconfigured to support this. Subversion repositories are generated dynamically by the addrepo script (installed later) and are included through the /etc/apache2/svn.conf file. URLs matching !^/tools/[^/]+/svn($|/) excluded from being processed by the CMS and are instead directly handled by the Subversion Apache module.

Test

```
# svnadmin create /opt/svn/tools/test --fs-type fsfs
# chown -R www-data.www-data /opt/svn/tools/test
# echo "<Location /tools/test/svn>
        DAV svn
        SVNPath /opt/svn/tools/test
        AuthType Basic
        AuthBasicProvider ldap
        AuthName "Test"
        AuthzLDAPAuthoritative on
        AuthLDAPGroupAttributeIsDN on
        AuthLDAPGroupAttribute owner
        AuthLDAPGroupAttribute member
        AuthLDAPURL ldap://localhost/ou=users,dc=myhub,dc=org
        Require ldap-group gid=apps,ou=groups,dc=myhub,dc=org
</Location&gt" > /etc/apache2/svn.conf
# /etc/init.d/apache2 restart
```

Be sure to the BASEDN in the above to match that used by your configuration.

Now browse to "/tools/test/svn" using an https connection and you should get prompted for a username and password, use the apps account you created earlier when you installed LDAP. You should see "svn - Revision 0: /".

Delete test file.

```
# echo "" > /etc/apache2/svn.conf
# rm -fr /opt/svn/tools/test
# /etc/init.d/apache2 restart
```

WebDAV

Configure WebDAV

The apache hub site configuration files are preconfigured to support this. Update the LDAP configuration to match the BASEDN of your site:

Edit **/etc/apache2/sites-available/hub-ssl**

```
        <Directory /webdav>
        ...

AuthLDAPURL ldap://localhost/ou=users,dc=myhub,dc=org?uid
        ...
        </Directory>
```

This enables webDAV for the /webdav directory space, rewriting the url to always be under the user's directory. The 'usermap' tool (see next section) is used to map files from the user's home directory into the /webdav space (and mapping ownership to www-data).

Then restart the apache webserver to enable your changes.

```
# /etc/init.d/apache2 restart
```

Test

```
# install --owner www-data --group www-
data --mode 0770 -d /webdav/home/apps
# touch /webdav/home/apps/test
```

Browse to your site's https /webdav address (e.g. https://myhub/webdav). You should get prompted for a username and password. Use the apps account. You should see a directory listing including the file "test".

Now test using a WebDAV client.

```
# apt-get install cadaver
```

```
# cadaver https://myhub.org/webdav
```

You will be prompted to accept self signed certificate (if it is still installed) and then to enter your username and password. Use the 'apps' account again to test. When you get the "dav:/webdav/>" prompt just enter "ls" and it should show the test file.

Finally clean up test case

```
# apt-get purge cadaver
# rm /webdav/home/apps/test
# rmdir /webdav/home/apps /webdav/home /webdav
```

Usermap

Install usermap

Install the WebDAV Usermap Filesystem

```
# apt-get install hubzero-usermap
```

Configure Usermap

```
# apt-get install autofs
```

Edit **/etc/auto.master** by adding the following line

```
/webdav/home /etc/auto.webdav --timeout=60
```

Edit/create **/etc/auto.webdav** so that it has the following content

```
* -fstype=usermap,user=www-data,allow_other :&
```

```
# /etc/init.d/autofs restart
```

Add **fuse** to the **/etc/modules** file so that it is loaded on startup.

This automounts a usermap filesystem of a users home directory inside of /webdav/home on demand. This version of the users home directory is owned and accessible to the user www-data which allows WebDAV to serve its contents.

Test

```
# touch /home/myhub/apps/test
# ls -l /webdav/home/apps
```

You should see a list of files in apps's home directory ("test") which will appear to be owned by www-data.www-data

```
# mount -l
```

You should see something like:
mount.usermap on /webdav/home/apps type fuse.mount.usermap (rw,nosuid,nodev,allow_other)

Finally clean up.

```
# umount -f /webdav/home/apps
# rm /webdav/home/apps/test
```

Troubleshooting

If the test doesn't work, check if the fuse kernel module is loaded

```
#  lsmod | grep fuse
fuse                   54176  0
```

If there is no output then try starting the kernel module manually

```
# modprobe fuse
```

Then try the test again

Trac

Configure Apache for Trac

```
# install --owner www-data --group www-
data --mode 0770 -d /opt/trac/tools
```

The apache hub site configuration files are preconfigured to support this. Update the LDAP configuration to match the BASEDN of your site:

Edit **/etc/apache2/sites-available/hub-ssl**

```
        <LocationMatch /tools/[^/]+/login>
                ...

AuthLDAPURL ldap://localhost/ou=users,dc=myhub,dc=org?uid?sub?(gid=*)
                ...
        </LocationMatch>
```

Then restart apache

```
# /etc/init.d/apache2 restart
```

Install Authentication Plugin

```
# apt-get install hubzero-trac-mysqlauthz
# /etc/init.d/apache2 restart
```

Test

```
# svnadmin create /opt/svn/tools/test --fs-type fsfs
# chown -R www-data.www-data /opt/svn/tools/test
# trac-admin /opt/trac/tools/test initenv "Test" "sqlite:db/trac.db" "
svn" "/opt/svn/tools/test"
# chown -R www-data.www-data /opt/trac/tools/test
```

Now browse to "/tools/test/wiki" using an https connection; you should see a default Trac project page.

Delete test data.

```
# rm -fr /opt/svn/tools/test
# rm -fr /opt/trac/tools/test
# /etc/init.d/apache2 restart
```

addrepo

Install

Install addrepo

```
# apt-get install hubzero-addrepo
```

Configure

The web process needs to be able to run a number of scripts as the "apps" user. To do this it is necessary to configure sudo to allow this:

Edit **/etc/sudoers** and add the following lines

```
www-
data        ALL=(apps)NOPASSW
D:/bin/bash /www/myhub/components/com_contribtool/scripts/*tool.php *
www-
data        ALL=(apps)N
OPASSWD:/usr/bin/expect /www/myhub
/components/com_contribtool/scripts/*tool.php *
www-data        ALL=NOPASSWD:/etc/init.d/apache2
```

Be sure to replace "/www/myhub" with the document root you are using for your hub.

iptables

iptables

```
# apt-get install hubzero-mw-firewall
```

HUBzero requires the use of iptables to route network connections between application sessions and the external network. The scripts controlling this can also be used to manage basic firewall operations for the site. If you use manage iptables with other tools you will have to make sure the rules in these scripts are maintained. /etc/mw/firewall_on and /etc/mw/firewall_off turn the HUBzero firewall on and off respectively. A link from /etc/rc.boot/firewall_on to /etc/mw/firewall_on causes the script to run at startup (this link was created for you). The basic scripts installed here block all access to the host except for those ports required by HUBzero (http,https,http-alt,ldap,ssh.smtp,mysql,submit,etc).

Maxwell Service

Install

```
# apt-get install hubzero-mw-service
```

Configure

```
# /usr/lib/mw/bin/mkvztemplate amd64 lenny
```

Test

```
# /usr/lib/mw/bin/maxwell_service startvnc 1 800x600 24
```

Enter an 8 character password when prompted (e.g., "testtest")

This should result in a newly create OpenVZ session with an instance of a VNC server running inside of it. The output of the above command should look something like:

```
Reading passphrase:
testtest
==================== begin /etc/vz/conf/hub-
session-5.0-amd64.umount ========================

Removing /var/lib/vz/root/1 :root etc var tmp dev/shm dev
==================== end /etc/vz/conf/hub-
session-5.0-amd64.umount ========================
stunnel already running
Starting VE ...
==================== begin /etc/vz/conf/1.mount ====================
=====
Removing and repopulating: root etc var tmp dev
Mounting: /var/lib/vz/template/debian-5.0-amd64-maxwell home apps
==================== end /etc/vz/conf/1.mount ======================
=====
VE is mounted
Setting CPU units: 1000
Configure meminfo: 2000000
```

```
VE start in progress...
TIME: 0 seconds.
Waiting for container to finish booting.
/usr/lib/mw/startxvnc: Becoming nobody.
/usr/lib/mw/startxvnc: Waiting for 8-byte vncpasswd and EOF.
1+0 records in
1+0 records out
8 bytes (8 B) copied, 3.5333e-05 s, 226 kB/s
Got the vncpasswd
Adding auth for 10.51.0.1:0 and 10.51.0.1/unix:0
xauth:  creating new authority file Xauthority-10.51.0.1:0
Adding IP address(es): 10.51.0.1
if-up.d/mountnfs[venet0]: waiting for interface venet0:0 before doing
NFS mounts (warning).
WARNING: Settings were not saved and will be resetted to original valu
es on next start (use --save flag)
```

```
# vzlist
      VEID      NPROC STATUS  IP_ADDR         HOSTNAME

         1          6 running 10.51.0.1       -
```

```
# openssl s_client -connect localhost:4001
```

This should report an SSL connection with a self signed certificate and output text should end with:

```
---
RFB 003.008
```

If you see this then you successfully connected to the VNC server running inside the newly created OpenVZ session.

Clean up

```
# /usr/lib/mw/bin/maxwell_service stopvnc 1
```

Which should give output similar to:

```
Killing 6 processes in veid 1 with signal 1
Killing 7 processes in veid 1 with signal 2
Killing 5 processes in veid 1 with signal 15
Got signal 9
Stopping VE ...
VE was stopped
==================== begin /etc/vz/conf/1.umount ====================
=====
Unmounting /var/lib/vz/root/1/usr
Unmounting /var/lib/vz/root/1/home
Unmounting /var/lib/vz/root/1/apps
Unmounting /var/lib/vz/root/1/.root

Removing /var/lib/vz/root/1 :root etc var tmp dev/shm dev
Removing /var/lib/vz/private/1: apps bin emul home lib lib32 lib64 mnt
 opt proc sbin sys usr .root
==================== end /etc/vz/conf/1.umount ======================
====
VE is unmounted
```

Maxwell Client

Install

```
# apt-get install hubzero-mw-client
```

Configure

Configure SSH to allow maxwell keyed clients to connect as root by copying the contents of /etc/mw/maxwell.key.pub into /root/.ssh/authorized_keys. This allows the maxwell client to run the maxwell service as root.

```
# mkdir -p /root/.ssh
# cat /etc/mw/maxwell.key.pub >>  /root/.ssh/authorized_keys
```

Copy the sample maxwell.conf file

```
#  cp /usr/lib/mw/maxwell.conf-dist  /etc/mw/maxwell.conf
```

Edit /etc/mw/maxwell.conf

```
mysql_host = "localhost"
mysql_user="myhub"
mysql_password="xyzzy#1"
mysql_db="myhub"

default_vnc_timeout=86400
session_suffix="L"

filexfe
r_decoration="
""filexfer_sitelogo { <h1><a
 href="http://myhub.org/" title="myHUB
 home page"><span>myhub.org
: online simulations and more</span></a></h1> }
filexfer_stylesheet http://myhub.org/templates/filexfer/upload.css"
"""
```

```
hub_name="myhub"
hub_url="http://myhub.org/"
hub_homedir="/home/myhub"
hub_template="hubbasic"
```

Test

```
# su www-data
$ ssh -i /etc/mw/maxwell.key root@localhost ls
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is e5:3c:7d:41:71:0b:0f:2a:0c:0e:bb:15:4d:e7:2f:08
.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known host
s.
list of files
$ exit
#
```

VNC Client

Install

```
# apt-get install tightvnc-java
```

To confirm you have the correct version installed:

```
# dpkg --list | grep tightvnc-java
ii  tightvnc-java                              1.3.8-0+hubzero3
    TightVNC java applet and command line progra
```

You should see a "+hubzeroXX" suffix on the version, where XX is some number. This version includes SSL connection support and extra support for resizing the VNC session and a simple client action protocol used by the HUBzero VNC server.

If the wrong version is installed be sure you have added the hubzero package repository to you APT configuration. You may need to run "apt-get update" to get an up to date index of hubzero packages. After updating try installing tightvnc-java again.

vncproxy

Install

```
# apt-get install hubzero-mw-vncproxy
```

expire

Install

```
# apt-get install hubzero-mw-expire
```

telequotad

install

```
# apt-get install hubzero-mw-telequotad
```

App Container

Install

Apps (tools) on HUBzero run inside a Session Container. This is container's filesystem is rooted at /var/lib/vz/template/debian-5.0-amd64-maxwell. To install debian packages into the session container you must use chroot to root your filesystem on the Session Containers. Here we do this and install a number of Session Container related packages.

```
# chroot /var/lib/vz/template/debian-5.0-amd64-maxwell
# apt-get update
# apt-get upgrade
# apt-get install icewm
# apt-get install hubzero-icewm-config
# apt-get install hubzero-icewm-themes
# apt-get install hubzero-use
```

We use a modified version of icewm, to confirm you have the correct version got installed:

```
# dpkg --list | grep icewm
ii  icewm                                    1.2.35-1+hubzero1
    wonderful Win95-OS/2-Motif-like window manag
ii  icewm-common                             1.2.35-1+hubzero1
    wonderful Win95-OS/2-Motif-like window manag
```

You should see a "+hubzeroXX" suffix on the version, where XX is some number. This version fixes a bug when resizing the display when using vnc4server.

This fix should get applied to the upstream package someday, in which case we will be able to remove this version of the package and use version from the Debian Linux distribution.

Be sure to exit the chroot environment when you are done:

```
# exit
```

Configure

Users in the apps group can be granted permission to manage HUBzero Apps by placing them in the 'apps' group. This permission is granted through the sudo, and is configured by adding the following lines to /etc/sudoers:

```
%apps  ALL=NOPASSWD:/bin/su - apps
```

Workspace

Install

```
# apt-get install hubzero-app
# apt-get install hubzero-app-workspace
# hubzero-app setup
# hubzero-
app install --publish /usr/lib/hubzero/apps/workspace-1.0.hza
```

Configure

Go to the administrative web interface and select the Tools component

Click the on the parameters button and fill in values for host,username,password and database for the apps database. In the hub-in-a-box configuration these are the same values as the main HUBzero database.

Test

You should then be able to log in to the site and see the "Workspace" tool in the tool list and launch it in your browser.

Initialization

Setting Component Parameters

Rappture

Install

The Rappture application is install in the apps directory along with proper links.

```
# apt-get install hubzero-rappture
```

Session Installation

Rappture is used from inside a container and needs several other packages installed to allow use of all its features. This process has been simplified by using the hubzero-rappture-session with only contains the dependencies needed to pull in these other packages.

```
# chroot /var/lib/vz/template/debian-5.0-amd64-maxwell
# apt-get update
# apt-get upgrade
# apt-get install hubzero-rappture-session
```

This is also a good time to add some default paths to your session environment so that it doesn't need to be whenever someone logs in. Modify the /etc/profile file as follows:

Change

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games"
fi
```

To

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/apps/rappture/bin:/apps/bin:
/usr/local/bin:/usr/bin:/bin:/usr/games"
```

```
fi
```

Be sure to exit the chroot environment when you are done:

```
# exit
```

A workspace may need to be opened and closed a few times before the changes to the session template appear in a workspace.

Test

Rappture comes with several demostration scripts that can effectively test many parts of the package. These demonstrations must be copied to a user's home directory within a workspace before running.

```
$ mkdir examples
$ cp -r /apps/rappture/examples/* examples/.
$ cd examples
$ ./demo.bash
```

A window should open on the workspace showing that part of the demonstration. Close that window to see the next demonstration. Some demonstrations may need something inputted to work properly (such as the graphing calculator).

filexfer

Install filexfer

Install the filexfer package

```
# apt-get install hubzero-filexfer
```

Configure Apache for filexfer

Modify the hub site file at /etc/apache2/sites-available/hub-ssl

The apache hub site configuration files are not preconfigured to support this. Add the two highlighted lines as shown below. Note that the relative placement of these lines is important.

Edit **/etc/apache2/sites-available/hub-ssl**

```
<VirtualHost *:443>
        RewriteEngine   on
        RewriteMap      xlate           prg:/usr/lib/mw/bin/filexfer-
xlate
        ...
        ...
        ...
        <Directory /www/myhub>
                Options FollowSymLinks
                AllowOverride None
                Order allow,deny
                Allow from all

                RewriteEngine   On

RewriteRule     ^filexfer/(.*) ${xlate:$1|nothing} [P,QSA,L]
```

Then restart apache

```
# /etc/init.d/apache2 restart
```

Submit

Introduction

The submit command provides a means for HUB end users to execute applications on remote resources. The end user is not required to have knowledge of remote job submission mechanics. Jobs can be submitted to traditional queued batch systems including PBS and Condor.

Installation

```
# apt-get install hubzero-app-submit
# apt-get install hubzero-submit-server
# apt-get install hubzero-submit-distributor
```

Local Configuration

The behavior of submit is controlled through a set of configuration files. There are separate files for defining remote sites, staged tools, multiprocessor managers, legal environment variables, remote job monitors, and ssh tunneling.

Sites

Remote sites are defined in the file sites.dat. Each remote site is defined by a stanza indicating an access mechanism and other account and venue specific information. Defined keywords are

- [name] - site name. Used as command line argument (-v/--venue) and in tool.dat (destinations)
- venues - comma separated list of hostnames. If multiple hostnames are listed one site will chosen at random.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
- siteMonitorDesignator - name of site monitor defined in monitors.dat.
- venueMechanism - possible mechanisms are ssh and local.
- remoteUser - login user at remote site.
- remoteBatchSystem - the possible batch submission systems include CONDOR, PBS, and LSF. SCRIPT may also be specified to specify that a script will be executed directly on the remote host.
- remoteBatchQueue - when remoteBatchSystem is PBS the queue name may be specified.
- remoteBatchPartition - slurm parameter to define partition for remote job
- remoteBatchPartitionSize - slurm parameter to define partition size, currently for BG machines.

- remoteBatchConstraints - slurm parameter to define constraints for remote job
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteScratchDirectory - define the top level directory where jobs should be executed. Each job will create a subdirectory under remoteScratchDirectory to isolated jobs from each other.
- remotePpn - set the number of processors (cores) per node. The PPN is applied to PBS and LSF job description files. The user may override the value defined here from the command line.
- remoteManager - site specific multi-processor manager. Refers to definition in managers.dat.
- remoteHostAttribute - define host attributes. Attributes are applied to PBS description files.
- stageFiles - A True/False value indicating whether or not files should be staged to remote site. If the the job submission host and remote host share a file system file staging may not be necessary. Default is True.
- passUseEnvironment - A True/False value indicating whether or not the HUB 'use' environment should passed to the remote site. Default is False. True only makes sense if the remote site is within the HUB domain.
- arbitraryExecutableAllowed - A True/False value indicating whether or not execution of arbitrary scripts or binaries are allowed on the remote site. Default is True. If set to False the executable must be staged or emanate from /apps.
- members - a list of site names. Providing a member list gives a layer of abstraction between the user facing name and a remote destination. If multiple members are listed one will be randomly selected for each job.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.
- failoverSite - specify a backup site if site is not available. Site availability is determined by site probes.
- checkProbeResult - A True/False value indicating whether or not probe results should determine site availability. Default is True.
- restrictedToUsers - comma separated list of user names. If the list is empty all users may garner site access. User restrictions are applied before group restrictions.
- restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner site access.
- logUserRemotely - maintain log on remote site mapping HUB id, user to remote batch job id. If not explicitly set the default value is False.

An example stanza is presented for a site that is accessed through ssh.

```
[cluster]
venues = cluster.university.edu
remotePpn = 8
remoteBatchSystem = PBS
remoteBatchQueue = standby
remoteUser = HUBuser
```

```
remoteManager = mpich-intel64
venueMechanism = ssh
remoteScratchDirectory = /scratch/HUBuser
siteMonitorDesignator = cluster
```

Tools

Staged tools are defined in the file tools.dat. Each staged tool is defined by a stanza indicating an where a tool is staged and any access restrictions. The existence of a staged tool at multiple sites can be expressed with multiple stanzas or multiple destinations within a single stanza. If the tool requires multiprocessors a manager can also be indicated. Defined keywords are

- [name] - tool name. Used as command line argument to execute staged tools. Repeats are permitted to indicate staging at multiple sites.
- destinations - comma separated list of destinations.
- executablePath - path to executable at remote site.
- restrictedToUsers - comma separated list of user names. If the list is empty all users may garner tool access. User restrictions are applied before group restrictions.
- restrictedToGroups - comma separated list of group names. If the list is empty all groups may garner tool access.
- remoteManager - tool specific multi-processor manager. Refers to definition in managers.dat. Overrides value set by site definition.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a staged tool maintained in the HUBuser account on a remote site.

```
[simulator]
destinations = cluster
executablePath = ${HOME}/apps/simulator/bin/simulator.ex
remoteManager = mpi
```

Multi-processor managers

Multiprocessor managers are defined in the file managers.dat. Each manager is defined by a stanza indicating the set of commands used to execute a multiprocessor simulation run. Defined keywords are

- [name] - manager name. Used in sites.dat and tools.dat.
- computationMode - indicate how to use multiple processors for a single job. Recognized

values are mpi, parallel, and matlabmpi. Parallel application request multiprocess have there own mechanism for inter process communication. Matlabmpi is used to enable the an Matlab implementation of MPI.

- preManagerCommands - comma separated list of commands to be executed before the manager command. Typical use of pre manager commands would be to define the environment to include a particular version of MPI amd/or compiler, or setup MPD.
- managerCommand - manager command commonly mpirun. It is possible to include strings that will be sustituted with values defined from the command line.
- postManagerCommands - comma separated list of commands to be executed when the manager command completes. A typical use would be to terminate an MPD setup.
- mpiRankVariable - define environment variable set by manager command to define process rank. Recognized values are: MPIRUN_RANK, GMPI_ID, RMS_RANK, MXMPI_ID, MSTI_RANK, PMI_RANK, and OMPI_MCA_ns_nds_vpid. If no variable is given an attempt is made to determine process rank from command line arguments.
- environment - comma separated list of environment variables in the form e=v.
- moduleInitialize - initialize module script for sh
- modulesUnload - modules to be unloaded clearing way for replacement modules
- modulesLoad - modules to load to define mpi and other libraries

An example stanza is presented for a typical MPI instance.

```
[mpich-intel32]
preManagerCommands = . ${MODULESHOME}/init/sh, module load mpich-
intel32
managerCommand = mpirun -machinefile ${PBS_NODEFILE} -np NPROCESSORS
```

The token NPROCESSORS is replaced by an actual value at runtime.

Environment variables

Legal environment variables are listed in the file environmentwhitelist.dat. The objective is to prevent end users from setting security sensitive environment variables while allowing application specific variables to be passed to the remote site. Environment variables required to define multiprocessor execution should also be included. The permissible environment variables should be entered as a simple list - one entry per line. An example file allowing use of a variable used by openmp is

```
# environment variables listed here can be specified from the command
line with -e/--env option.

OMP_NUM_THREADS
```

Monitors

Remote job monitors are defined in the file monitors.dat. Each remote monitor is defined by a stanza indicating where the monitor is located and to be executed. Defined keywords are

- [name] - monitor name. Used in sites.dat (siteMonitorDesignator)
- venue - hostname upon which to launch monitor daemon. Typically this is a cluster headnode.
- tunnelDesignator - name of tunnel defined in tunnels.dat.
- remoteUser - login user at remote site.
- remoteMonitorCommand - command to launch monitor daemon process.

An example stanza is presented for a remote monitor tool used to report status of PBS jobs.

```
[cluster]
venue = cluster.university.edu
remoteUser = HUBuser
remoteMonitorCommand = ${HOME}/SubmitMonitor/monitorPBS.py
```

Tunnels

In some circumstances access to clusters is restricted such that only a select list of machines is allowed to communicate with the cluster job submission node. The machines that are granted such access are sometimes referred to as gateways. In such circumstances ssh tunneling or port forwarding can be used to submit HUB jobs through the gateway machine. Tunnel definition is specified in the file tunnels.dat. Each tunnel is defined by a stanza indicating gateway host and port information. Defined keywords are

- [name] - tunnel name.
- venue - tunnel target host.
- venuePort - tunnel target port.
- gatewayHost - name of the intermediate host.
- gatewayUser - login user on gatewayHost.
- localPortOffset - local port offset used for forwarding. Actual port is localPortMinimum + localPortOffset

An example stanza is presented for a tunnel between the HUB and a remote venue by way of an accepted gateway host.

```
[cluster]
venue = cluster.university.edu
venuePort = 22
gatewayHost = gateway.university.edu
```

```
gatewayUser = HUBuser
localPortOffset = 1
```

Remote Configuration

For job submission to remote sites via ssh it is necessary to configure a remote job monitor and a set of scripts to perform file transfer and batch job related functions. A set of scripts can be used for each different batch submission system or in some cases they may be combined with appropriate switching based on command line arguments. A separate job monitor is need for each batch submission system. Communication between the HUB and remote resource via ssh requires inclusion of a public key in the authorized_keys file.

Job monitor daemon

A remote job monitor runs a daemon process and reports batch job status to a central job monitor located on the HUB. The daemon process is started by the central job monitor on demand. The daemon terminates after a configurable amount of inactivity time. The daemon code needs to be installed in the location declared in the monitors.dat file. The daemon requires some initial configuration to declare where it will store log and history files. The daemon does not require any special privileges any runs as a standard user. Typical configuration for the daemon looks like this:

```
siteDesignator     = "cluster"
monitorRoot        = "/home/HUBuser/SubmitMonitor"
qstatCommand       = "/usr/pbs/bin/qstat -u HUBuser"
monitorLogLocation = "logs"
```

The directory defined by the combination of monitorRoot and monitorLogLocation needs to be created before the daemon is started. A sample daemon used for PBS batch systems is listed below.

```
#!/usr/bin/env python
#
# Copyright (c) 2004-2010 Purdue University All rights reserved.
#
# Developed by: HUBzero Technology Group, Purdue University
#               http://hubzero.org
#
# HUBzero is free software: you can redistribute it and/or modify it u
nder the terms of the
# GNU Lesser General Public License as published by the Free Software
```

```
Foundation, either
# version 3 of the License, or (at your option) any later version.
#
# HUBzero is distributed in the hope that it will be useful, but WITHO
UT ANY WARRANTY;
# without even the implied warranty of MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.
# See the GNU Lesser General Public License for more details.  You sho
uld have received a
# copy of the GNU Lesser General Public License along with HUBzero.
# If not, see .
#
# GNU LESSER GENERAL PUBLIC LICENSE
# Version 3, 29 June 2007
# Copyright (C) 2007 Free Software Foundation, Inc.
#
# ----------------------------------------------------------------------
--
#  monitorPBS.py
#
#  script which monitors the PBS queue and reports changes in job stat
us
#
import sys
import os
import os.path
import select
import time
import popen2
import re
import signal

siteDesignator     = "pbsHost"
monitorRoot        = os.path.join(os.sep,'home','pbsUser','Submit','pb
sHost')
qstatCommand       = "/usr/pbs/bin/qstat -u pbsUser"
monitorLogLocation = "logs"
monitorLogFileName = "monitorPBS.log"
historyFileName    = "monitorPBS.history"

logFile            = sys.stdout
historyFile        = None
activeJobs         = {}
updates            = []
```

```
def cleanup():
   global historyFile

   if historyFile:
      historyFile.close()


def sigGEN_handler(signal, frame):
   global siteDesignator

   cleanup()
   log("%s monitor stopped" % (siteDesignator))
   sys.exit(1)

def sigINT_handler(signal, frame):
   log("Received SIGINT!")
   sigGEN_handler(signal, frame)

def sigHUP_handler(signal, frame):
   log("Received SIGHUP!")
   sigGEN_handler(signal, frame)

def sigQUIT_handler(signal, frame):
   log("Received SIGQUIT!")
   sigGEN_handler(signal, frame)

def sigABRT_handler(signal, frame):
   log("Received SIGABRT!")
   sigGEN_handler(signal, frame)

def sigTERM_handler(signal, frame):
   log("Received SIGTERM!")
   sigGEN_handler(signal, frame)

signal.signal(signal.SIGINT, sigINT_handler)
signal.signal(signal.SIGHUP, sigHUP_handler)
signal.signal(signal.SIGQUIT, sigQUIT_handler)
signal.signal(signal.SIGABRT, sigABRT_handler)
signal.signal(signal.SIGTERM, sigTERM_handler)


def openLog(logName):
   global logFile

   try:
      logFile = open(logName,"a")
```

```
   except:
      logFile = sys.stdout


def log(message):
   global logFile

   if message != "":
      logFile.write("[%s] %s\n" % (time.ctime(),message))
      logFile.flush()


def openHistory(historyName,
                accessMode):
   global historyFile

   if accessMode == "r":
      if os.path.isfile(historyName):
         historyFile = open(historyName,accessMode)
      else:
         historyFile = None
   else:
      historyFile = open(historyName,accessMode)


def recordHistory(id):
   global updates
   global activeJobs

   historyFile.write("%s:%s %s %s\n" % (siteDesignator,str(id),activeJ
obs[id][0],activeJobs[id][1]))
   historyFile.flush()
   updates.append(str(id) + " " + activeJobs[id][0] + " " + activeJobs
[id][1])


def getCommandOutput(command,
                     streamOutput=False):
   child = popen2.Popen3(command,1)
   child.tochild.close() # don't need to talk to child
   childout = child.fromchild
   childoutFd = childout.fileno()
   childerr = child.childerr
   childerrFd = childerr.fileno()

   outEOF = errEOF = 0
```

```
   BUFSIZ = 4096

   outData = []
   errData = []

   while 1:
      toCheck = []
      if not outEOF:
         toCheck.append(childoutFd)
      if not errEOF:
         toCheck.append(childerrFd)
      ready = select.select(toCheck,[],[]) # wait for input
      if childoutFd in ready[0]:
         outChunk = os.read(childoutFd,BUFSIZ)
         if outChunk == '':
            outEOF = 1
         outData.append(outChunk)
         if streamOutput:
            sys.stdout.write(outChunk)
            sys.stdout.flush()

      if childerrFd in ready[0]:
         errChunk = os.read(childerrFd,BUFSIZ)
         if errChunk == '':
            errEOF = 1
         errData.append(errChunk)
         if streamOutput:
            sys.stderr.write(errChunk)
            sys.stderr.flush()

      if outEOF and errEOF:
         break

   err = child.wait()
   if err != 0:
      log("%s failed w/ exit code %d" % (command,err))
      if not streamOutput:
         log("%s" % ("".join(errData)))

   return  err,"".join(outData),"".join(errData)


if __name__ == '__main__':

   if monitorLogFileName != "stdout":
      openLog(os.path.join(monitorRoot,monitorLogLocation,monitorLogFi
```

```
leName))

   log("%s monitor started" % (siteDesignator))

   sleepTime = 10
   pauseTime = 5.
   maximumConsectutiveEmptyQueues = 30*60/sleepTime

   openHistory(os.path.join(monitorRoot,historyFileName),"r")
   if historyFile:
#  lPBS:6760 R
#  --------------------
      records = historyFile.readlines()
      for record in records:
         colon = record.find(":")
         if colon > 0:
            jobState = record[colon+1:].split()
            id = jobState[0]
            status = jobState[1]
            stage  = "Simulation"
            activeJobs[id] = (status,stage)
      historyFile.close()

      completedJobs = []
      for activeJob in activeJobs:
         if activeJobs[activeJob][0] == "D":
            completedJobs.append(activeJob)

      for completedJob in completedJobs:
         del activeJobs[completedJob]

   openHistory(os.path.join(monitorRoot,historyFileName),"a")
   consectutiveEmptyQueues = 0

   toCheck = []
   toCheck.append(sys.stdin.fileno())
   while 1:
      updates       = []
      currentJobs   = {}
      completedJobs = []

      delayTime = 0
      while delayTime  0:
         updateMessage = str(len(updates)) + " " + siteDesignator + ":
" + ":".join(updates)
         sys.stdout.write("%s\n" % (updateMessage))
```

```
         sys.stdout.flush()

      del updates

      if consectutiveEmptyQueues == maximumConsectutiveEmptyQueues:
         cleanup()
         log("%s monitor stopped" % (siteDesignator))
         sys.exit(0)
```

File transfer and batch job scripts

The simple scripts are used to manage file transfer and batch job launching and termination. Examples of scripts suitable for use with PBS are listed here.

File transfer - input files

receiveinput.sh - receive compressed tar file containing input files required for the job. The file .__fileTimeMarker is used to determine what files should be returned to the HUB.

```
#!/bin/sh
#
rm -rf $1
mkdir $1
exitStatus=$?

if [ $exitStatus -eq 0 ] ; then
   cd $1
   exitStatus=$?

   if [ $exitStatus -eq 0 ] ; then
      tar xvzf -
      exitStatus=$?

      touch .__fileTimeMarker
      sleep 1

      date +"%s" > $2
   fi
fi

exit $exitStatus
```

Batch job script - submission

submitbatchjob.sh - submit batch job using supplied description file. If arguments beyond job working directory and batch description file are supplied an entry is added to the remote site log file. The log file provides a record relating the HUB end user to the remote batch job. The log file should be placed at a location agreed upon by the remote site and HUB.

```
#!/bin/sh
#
cd $1
exitStatus=$?

if [ $exitStatus -eq 0 ] ; then
   case $2 in
      *.pbs)
         JOBID=`qsub $2`
         exitStatus=$?
         if [ $# -gt 2 ] ; then
            logRecord=`date`
            shift 2
            while [ $# -gt 0 ] ; do
               logRecord=${logRecord}'\t'$1
               shift 1
            done
            logRecord=${logRecord}'\t'${JOBID}
            echo -e ${logRecord} >> pbslog
         fi
         ;;
      *)
         echo "Invalid job class $2"
         exitStatus=23
         ;;
   esac
fi

if [ $exitStatus -eq 0 ] ; then
   echo ${JOBID}
else
   echo "-1"
fi
exit $exitStatus
```

File transfer - output files

transmitresults.sh - return compressed tar file containing job output files.

```
#!/bin/sh
#
cd $1
exitStatus=$?

if [ $exitStatus -eq 0 ] ; then
   tar czf - `find . -newer .__fileTimeMarker -not -name . -print`
   exitStatus=$?
fi

exit $exitStatus
```

Batch job script - termination

killbatchjob.sh - terminate the batch job

```
#!/bin/sh
#
case $2 in
   PBS)
      qdel $1
      exitStatus=$?
      ;;
   *)
      echo "Invalid job class $2"
      exitStatus=23
      ;;
esac

exit $exitStatus
```

File transfer - cleanup

cleanupjob.sh - remove job specific directory and any other dangling files

```
#!/bin/sh
#
rm -rf $1
exitStatus=$?

exit $exitStatus
```

Access Control Mechanisms

By default tools and sites are configured so that access is granted to all HUB members. In some cases it is desired to restrict access to either a tool or site to a subset of the HUB membership. The keywords restrictedToUsers and restrictedToGroups provide a mechanism to apply restrictions accordingly. Each keyword should be followed by a list of comma separated values of userids (logins) or groupids (as declared when creating a new HUB group). If user or group restrictions have been declared upon invocation of submit a comparison is made between the restrictions and userid and group memberships. If both user and group restrictions are declared the user restriction will be applied first, followed by the group restriction.

In addition to applying user and group restrictions another mechanism is provided by the boolean keyword arbitraryExecutableAllowed in the sites configuration file. In cases where the executable program is not pre-staged at the remote sites the executable needs to be transferred along with the user supplied inputs to the remote site. Published tools will have their executable program located in the /apps/tools/revision/bin directory. For this reason submitted programs that reside in /apps are assumed to be validated and approved for execution. The same cannot be said for programs in other directories. The common case where such a situation arises is when a tool developer is building and testing within the HUB workspace environment. To grant a tool developer the permission to submit such arbitrary applications the site configuration must allow arbitrary executables and the tool developer must belong the system group submit.