System Administration

Target Audience

This document and the installation and maintenance of a HUBzero system has a target audience of **experienced** Linux administrators.

Minimum System Requirements

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 24 Gigabytes of RAM and a terabyte of disk.

Production systems should try to not limit hardware resources, HUBzero is designed to run on systems with many CPU cores and lots of RAM. If you are looking for a system to run a small site with limited physical or virtual resources this is probably not the system for you. However, for demonstration or development purposes we often create VM images with less than a gigabyte of RAM and 5 gigabytes of disk. While fully functional, these virtual machines would only be suitable for a single user doing development or testing.

System Architecture

All hardware, filesystem partitions, RAID configurations, backup models, security models, etc. and base configurations of the hosts (SSH server, network, etc.) are the responsibility of the system administrator managing the host.

The HUBzero software expects to be installed on a headless server from a minimal ISO with only one network interface (required by OpenVZ) with an MTU no less than '1500'. System accounts must not be created with an id of 1000 or greater - more about that in a forthcoming section.

End of Life for Debian Linux Distribution

HUBzero support for the Debian Linux distribution is no longer available, effective August 27, 2020.

You may continue to use Debian for a hub installation, however, support for HUBzero operating system packages and other related software will no longer receive updates. Maintenance and support for installed software, including security updates are the responsibility of the individual hub system administrators.

HUBzero will continue to offer its suite of open-source software on the CentOS Linux

distribution, allowing us to focus our efforts on maintaining and improving our code-base, as well as bringing you additional features and upgrades such as Docker which replaces OpenVZ for hub tool session containers.

For support or help migrating a hub to CentOS, please contact info@hubzero.org.

Installation

Target Audience

This document and the installation and maintenance of a HUBzero system has a target audience of **experienced** Linux administrators (preferably experienced with RedHat or CentOS distributions).

Minimum System Requirements

HUBzero (RedHat) installations require one or more dedicated hosts running RedHat or CentOS version 6.

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 24 Gigabytes of RAM and a terabyte of disk.

Production systems should try to not limit hardware resources, HUBzero is designed to run on systems with many CPU cores and lots of RAM. If you are looking for a system to run a small site with limited physical or virtual resources this is probably not the system for you. However, for demonstration or development purposes we often create VM images with less than a gigabyte of RAM and 5 gigabytes of disk. While fully functional, these virtual machines would only be suitable for a single user doing development or testing.

System Architecture

All hardware, filesystem partitions, RAID configurations, backup models, security models, etc. and base configurations of the hosts email server, SSH server, network, etc. are the responsibility of the system administrator managing the host.

The Hubzero software expects to be installed on a headless server from a minimal ISO with only one network interface (required by OpenVZ) with an MTU no less than '1500'. System accounts must not be created with an id of 1000 or greater - more about that in a forthcoming section.

Linux

Install Basic Operating System

Advanced Linux system administrator skills are required, please read carefully. Selecting all the default configurations during the operating system installation may not always be suitable.

The latest version of RedHat Enterprise Linux 6 or CentOS 6 (x86_64 is the only architecture supported) should be downloaded and installed. Do not install a default LAMP environment or other server packages.

System reboots are required to complete the installation. Be sure to remove the install disk or otherwise reset your server's boot media before rebooting.

The precise server configuration (such as disk partitioning, networking, etc) is dependent on how the hub is to be used and what hardware is being used, all the possible configuration options are not specifically outlined here. This installation guide outlines a very basic configuration but may not be suitable for larger sites. For larger sites, it is generally expected that the hub will be managed by an experienced Linux administrator who can help setup your site to meet your specific requirements.

All hardware, filesystem partitions, RAID configurations, backup models, security models, etc. and base configurations of the hosts email server, SSH server, network, etc. are the responsibility of the system administrator managing the host.

The following instructions only instruct how to install Hubzero software. At a minimum a "Basic Server" host, ideally from a 'minimal' ISO image, is required with network access.

The Hubzero software expects to be installed on a headless server without a Graphical User Interface.

Configure Networking and DNS

Configure your host's network as desired. A registered domain name and SSL certificate is required and should be obtained prior to installation. A static IP address is highly recommended as well.

By default, the Hubzero middleware uses IP addresses in the 192.168.0.0/16 subnet. Do not use an IP address in this range for your host.

Set hostname

Throughout this documentation you will see specific instructions for running commands, with

part of the text highlighted. The highlighted text should be modified to your local configuration choices. (e.g. replace "example.com" with the fully qualified hostname of your machine).

HUBzero expects the `hostname` command to return the fully qualified hostname for the system. This step may be skipped if previously configured.

```
sudo hostname hubdomain.org
```

Make the change permanent (or manually edit /etc/sysconfig/network):

```
sudo sed -i "s/HOSTNAME=.*/HOSTNAM
E=hubdomain.org/g" /etc/sysconfig/network
```

Delete local Users

HUBzero reserves all user ids from 1000 up for hub accounts. As part of the app middleware every account must map to a corresponding system account. Therefore when starting up a hub it is required to remove all accounts that have user ids 1000 or greater. New RedHat/CentOS installations typically do not setup a non root account during setup, but if you have any accounts added to the system, those accounts can be removed as follows:

```
sudo userdel username
sudo rm -fr /home/username
```

If you require additional system accounts, they should use user and group ids in the range of 500-999 (these will not interfere with hub operations).

Update the initial OS install

```
sudo yum update -y
```

Disable SELinux

Hubzero does not currently support SELinux. Since the default install of RHEL turns it on, we have to turn it off.

```
sudo sed -i 's/^SELINUX=.*/SELINUX=disabled/g' /etc/selinux/config
```

Reboot the system for this change to take effect

```
sudo reboot
```

Yum repository setup

Configure the hubzero repository configuration package

For RedHat Enterprise Linux 6

```
sudo subscription-manager repos --enable rhel-6-server-optional-rpms

sudo subscription-manager repos --enable rhel-6-server-extras-rpms

sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

sudo subscription-manager repos --enable rhel-server-rhscl-6-rpms

sudo rpm -Uvh https://packages.hubzero.org/rpm/julian-el6/hubzero-julian-repo-2.2.5-1.el6.noarch.rpm
```

For CentOS 6

```
sudo yum install -y epel-release

sudo yum install -y centos-release-scl-rh

sudo rpm -Uvh http://packages.hubzero.org/rpm/julian-el6/hubzero-julian-repo-2.2.5-1.el6.noarch.rpm
```

Firewall

Install

```
sudo yum remove -y firewalld
sudo yum install -y hubzero-iptables-basic

sudo service hubzero-iptables-basic start
sudo chkconfig hubzero-iptables-basic on
```

If installing the Hubzero tool infrastructure:

```
sudo yum install -y hubzero-mw2-iptables-basic

sudo service hubzero-mw2-iptables-basic start
sudo chkconfig hubzero-mw2-iptables-basic on
```

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. 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).

Database

MySQL Database Installation

HUBzero should work with any MySQL 5.5 compatible database. We have used MySQL 5.5.x, 5.6.x, MariaDB 5.5.x, 10.1.x, 10.3.x, Percona XtraDB Cluster 5.7.x.

For CentOS 6 and Redhat Enterprise 6 we recommend MariaDB 5.5.x directly from the MariaDB rpm repositories as they are maintaining longer term support. (https://downloads.mariadb.org/mariadb/repositories).

CentOS 6 - MariaDB Database Installation

```
cat << EOF > /etc/yum.repos.d/mariadb-5.5.repo
# MariaDB 5.5 CentOS repository list
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/centos6-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF

sudo yum install MariaDB-server
sudo service mysql start
sudo chkconfig mysql on
```

RedHat Enterprise Linux 6 - MariaDB Database Installation

```
sudo cat << EOF > /etc/yum.repos.d/mariadb-5.5.repo
# MariaDB 5.5 RedHat repository list
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/rhel6-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF

sudo yum install MariaDB-server
sudo service mysql start
sudo chkconfig mysql on
```

Configure

Default configuration works well for starters. But for optimal performance you will need a database administrator capable of tuning your database to your hardware configuration and site usage.

Mail

Install Postfix

```
sudo yum install -y postfix

sudo service postfix start
sudo chkconfig postfix on
```

Test

```
sudo postfix check
```

If the 'postfix check' command returns anything, resolve the reported issues with the Postfix installation before continuing.

Configure Postfix

Configure Postfix as desired. The default installation may only handle mail on the localhost. HUBzero expects to be able to send mail to registered user's email address to confirm registration and also to send group and support ticket related messages. Incoming mail is also expected to work (see Mailgateway section) in order to receive support ticket updates and email replies to group forum messages. Setting up an appropriate mail configuration is up to the site administrator. The Mailgateway service expects postfix to be the Mail Transfer Agent on CentOS and RedHat systems.

Web Server

Install Apache Httpd Web Server

```
sudo yum install -y hubzero-apache2

sudo service httpd start
sudo chkconfig httpd on
```

PHP

Install

HUBzero still requires the use of PHP 5.6 (7.x support coming soon). For CentOS/RedHat 6 you used to be able to get 5.6 via the software collections library and when that was removed we mirrored a copy. These packages are no longer being maintained so are not recommended. The current recommendation for RedHat/CentOS 6 is to use the php56 packages from the Remi Repository (https://rpms.remireponet). These packages are still receiving security updates for serious issues despite PHP 5.6 reaching upstream EOL.

```
sudo rpm -Uvh https://rpms.remirepo.net/enterprise/remi-release-6.rpm
sudo yum install -y hubzero-php56-remi
sudo service php56-php-fpm start
sudo chkconfig php56-php-fpm on
```

CMS

Installation

```
sudo yum install -y hubzero-cms-2.2
sudo yum install -y hubzero-texvc
sudo yum install -y hubzero-textifier
sudo yum install -y wkhtmltopdf
```

Configuration

```
sudo hzcms install hubname
```

It is necessary to immediately run the updater to apply fixes that have not been incorporated into the initial installation.

```
sudo hzcms update
```

SSL Configuration

The default SSL certificate is a self signed certificate (aka snakeoil) meant for evaluation purposes only. Some browers will not accept this certificate and will not allow access to the site without special configuration (https://support.mozilla.org/en-US/questions/1012036). For a production hub you will need to obtain a valid SSL certificate. A certificate may contain two or three pieces: a public certificate, a private key, and sometimes an intermediate certificate. All files are expected to be in PEM format.

Once you obtain the certificate, copy the SSL certificate files to the httpd SSL configuration directories and restart httpd.

```
sudo cp [your certificate pem file]
 /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo cp
[your certificate private key p
em file] /etc/httpd/conf/ssl.key/hubname-privkey.pem
```

```
sudo cp
[your certif
icate intermediate certi
ficate chain pem file] /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo chown root:root /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo chown root:root /etc/httpd/conf/ssl.key/hubname-privkey.pem
sudo chown root:root /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo chmod 0640 /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo chmod 0640 /etc/httpd/conf/ssl.key/hubname-privkey.pem
sudo chmod 0640 /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo hzcms reconfigure hubname
sudo service httpd restart
```

If you are using the HTML5 VNC Proxy Server, you must update your certificate settings there as well.

Mailgateway

Install the Hubzero Mailgateway

```
sudo yum install -y hubzero-mailgateway
```

Configure the Hubzero Mailgateway

```
sudo hzcms configure mailgateway --enable
```

OpenLDAP

Install hubzero-openldap

```
sudo yum install -y hubzero-openldap

sudo service slapd start
sudo chkconfig slapd on
sudo chkconfig sssd on
```

Configure OpenLDAP database

```
sudo hzldap init dc=hubname,dc=org
sudo hzcms configure ldap --enable
sudo hzldap syncusers
```

Test

```
sudo getent passwd
```

You should see an entry for user 'admin' toward the end of the list if everything is working correctly.

WebDAV

Install WebDAV

```
sudo yum install -y hubzero-webdav
```

Configure WebDAV

```
sudo hzcms configure webdav --enable
```

Test

Test the fuse filesystem mount. Create the file mytest and then view the directory contents. You should see no errors and the file "mytest" should appear in the directory listing

```
sudo touch /webdav/home/admin/mytest
sudo ls -l /webdav/home/admin
```

Now test using a WebDAV client.

```
sudo yum install -y cadaver
sudo cadaver https://localhost/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 'admin' account to test. The admin credentials are located in /etc/hubzero.secrets if you have forgetten them. When you get the "dav:/webdav/>" prompt just enter "ls" and you should see the "mytest" file listed.

Finally clean up test case

```
sudo yum remove cadaver
sudo rm /webdav/home/admin/mytest
```

Troubleshooting

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

```
sudo lsmod | grep fuse
fuse                   54176  0
```

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

```
sudo modprobe fuse
```

Then try the test again

Subversion

Install

```
sudo yum install -y hubzero-subversion
```

Configure

```
sudo hzcms configure subversion --enable
```

Trac

Install

```
sudo yum install -y hubzero-trac
```

Configure

```
sudo hzcms configure trac --enable
```

Forge

Install

```
sudo yum install -y hubzero-forge
```

Configure

```
sudo hzcms configure forge --enable
```

OpenVZ

Install

```
sudo curl -s -o /etc/yum.repos.d/openvz.repo https://download.openvz.o
rg/openvz.repo
sudo rpm --import http://download.openvz.org/RPM-GPG-Key-OpenVZ
```

Then install

```
sudo yum install -y hubzero-openvz
```

Configure

```
sudo hzcms configure openvz --enable
```

If configuration is successful it should prompt you to reboot the server to activate the new kernel.

```
sudo reboot
```

Test

```
sudo vzlist
Container(s) not found
```

Or it will list the containers currently running if you check this on a running hub. The salient point being that the command doesn't issue any kind of error message.

Maxwell Client

Install

```
sudo yum install -y hubzero-mw2-client
sudo yum install -y hubzero-expire-sessions
```

Configure

```
sudo hzcms configure mw2-client --enable
sudo service expire-sessions start
sudo chkconfig expire-sessions on
```

Maxwell File Service

Install

```
sudo yum install -y hubzero-mw2-file-service
```

Maxwell Service

Install

```
sudo yum install -y hubzero-mw2-exec-service
sudo yum install -y hubzero-mw2-iptables-basic
sudo service hubzero-mw2-iptables-basic start
sudo chkconfig hubzero-mw2-iptables-basic on
```

Configure

```
sudo mkvztemplate amd64 wheezy ellie
```

```
sudo hzcms configure mw2-service --enable
sudo hzcms mw-
host add localhost up openvz pubnet sessions workspace fileserver
```

Test

```
sudo 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)
```

```
sudo vzlist
      VEID      NPROC STATUS  IP_ADDR         HOSTNAME

         1          6 running 10.51.0.1       -
```

```
sudo 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

```
sudo 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
```

VNC Proxy Server

Install

```
sudo yum install -y hubzero-vncproxyd-ws
```

Configure

```
sudo hzvncproxyd-ws-config configure --enable
sudo service hzvncproxyd-ws start
sudo chkconfig hzvncproxyd-ws on
```

Install SSL certificate files

Copy your Apache SSL certificate files to /etc/hzvncproxyd-ws/ssl-cert-hzvncproxyd-ws.pem and /etc/hzvncproxyd-ws/ssl-cert-hzvncproxyd-ws.key and make sure they are readable by the user "hzvncproxy" to be found automatically by the proxy service.

If you are using a self-signed or otherwise invalid certificate the tool viewer will likely reject it and not work. If you are using the same certificate as your website and you allowed Chrome to use the invalid cert then the tool viewer will probably accept it. If you are using Firefox the tool viewer will always reject the invalid certificate. Always use a valid SSL certificate with hzvncproxyd-ws.

telequotad

Install

```
sudo yum install -y hubzero-telequotad

sudo service telequotad start
sudo chkconfig telequotad on
```

Configure

In order for filesystems quotas to work they must be enabled when they are mounted. Determine which filesystem contains your home directories and add "quota" to the mount option of the corresponding entry in the /etc/fstab file. Only the filesystem with /home on it matters to telequotad.

If quotas weren't already in affect, the run something like the following (depending on your filesystem configuration) to start up the quota system. The following example assumes you want to enable quotas at the root level

```
sudo mount -oremount /home
sudo quotacheck -cugm /home
sudo quotacheck -avugm
sudo quotaon -u /home
```

Test

```
sudo repquota -a
```

Should show disk usage for all users.

Workspace

Install

```
sudo yum install -y hubzero-app
sudo yum install -y hubzero-app-workspace
sudo hubzero-
app install --publish /usr/share/hubzero/apps/workspace-1.3.hza
```

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.

Filexfer

Install

```
sudo yum install -y hubzero-filexfer-xlate
```

Configure

```
sudo hzcms configure filexfer --enable
```

Rappture

Install

```
sudo yum install -y hubzero-rappture-deb7
```

Configure

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 which only contains the dependencies needed to pull in these other packages.

```
sudo chroot /var/lib/vz/template/debian-7.0-amd64-maxwell
apt-get update; apt-get upgrade
apt-get install -y hubzero-rappture-session
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

A user must setup their runtime environment in order to use the Rappture toolkit. Run the following command inside a Workspace tool session before attempting to run any Rappture tests.

```
use rappture
```

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/share/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).

Submit

Introduction

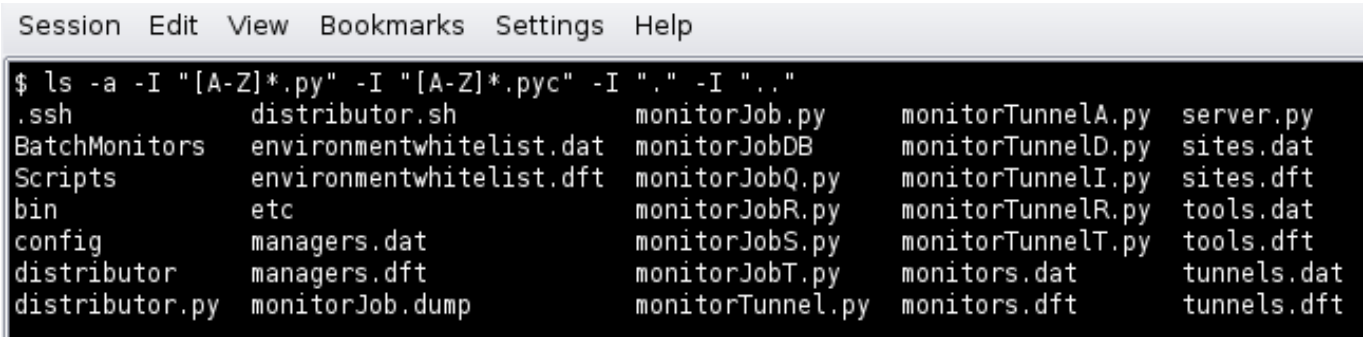
The submit command provides a means for HUB end users to execute applications on remote resources. The end user is not required to have knowledge of remote job submission mechanics. Jobs can be submitted to traditional queued batch systems including PBS and Condor or executed directly on remote resources.

Installation

```
sudo yum install -y hubzero-submit-pegasus
sudo yum install -y hubzero-submit-condor
sudo yum install -y hubzero-submit-common
sudo yum install -y hubzero-submit-server
sudo yum install -y hubzero-submit-distributor
sudo yum install -y hubzero-submit-monitors

sudo hzcms configure submit-server --enable
sudo service submit-server start
sudo chkconfig submit-server on
```

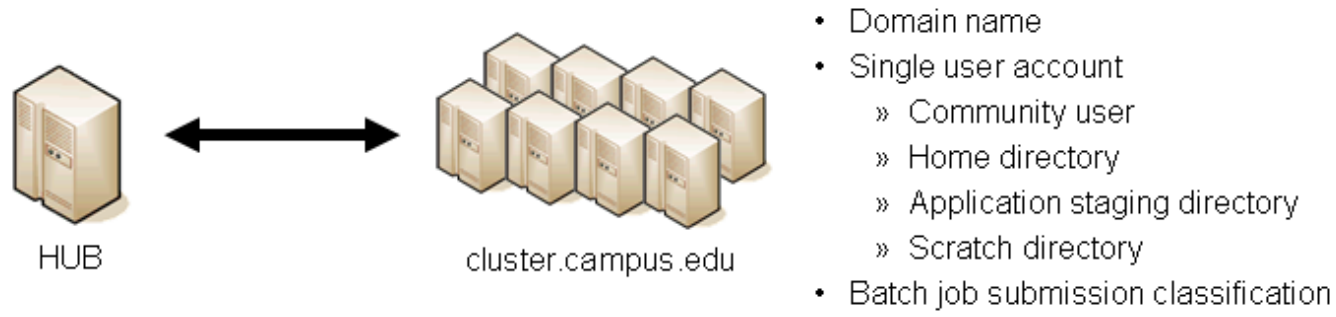
At completion of the yum install commands several files will be located in the directory /opt/submit. Excluding python files, the directory listing should like the following:



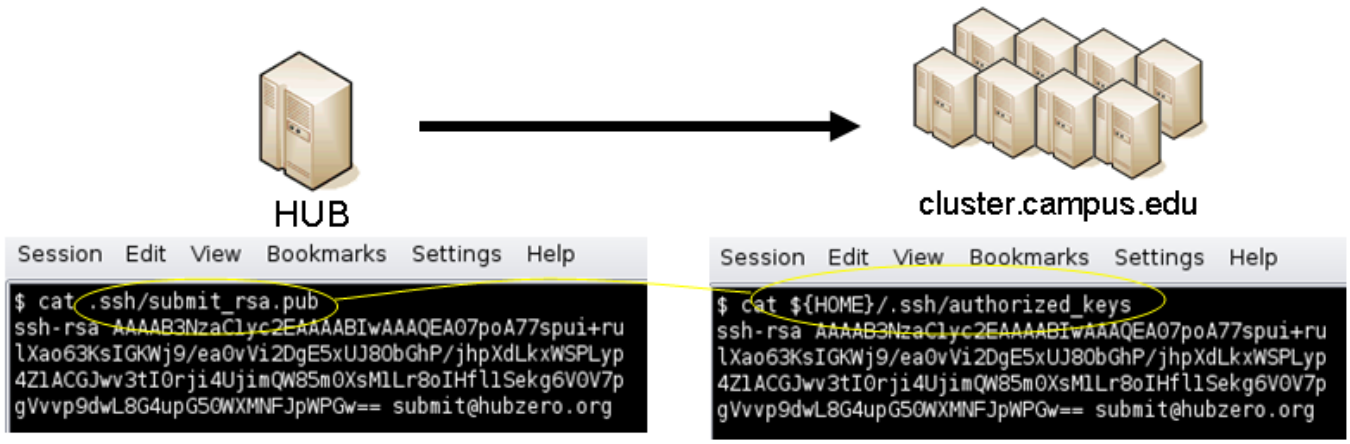
Configuration

Submit provides a mechanism to execute jobs on machines outside the HUB domain. To

accomplish this feat, some configuration is required on the HUB and some additional software must be installed and configured on hosts in remote domains. Before attempting to configure submit it is necessary to obtain access to the target remote domain(s). The premise is that a single account on the remote domain will serve as an execution launch point for all HUB end users. It is further assumes that access to this account can be made by direct ssh login or using an ssh tunnel (port forwarding).



Having attained account access to one or more remote domains, it is possible to proceed with submit configuration. To get started, the ssh public generated by the installation should be transferred to the remote domain host(s).



HUB Configuration

The behavior of submit is controlled through a set of configuration files. The configuration files contain descriptions of the various parameters required to connect to a remote domain, exchange files, and execute simulation codes. There are separate files for defining remote sites, staged tools, multiprocessor managers, file access controls, permissible environment variables, remote job monitors, and ssh tunneling. Most parameters have default values and it is not required that all parameters be explicitly defined in the configuration files. A simple example is given for each category of configuration file.

- Remote sites
- Staged tools
- Remote job monitors
- Multi-processor managers
- Permissible environment variables
- ssh tunnels

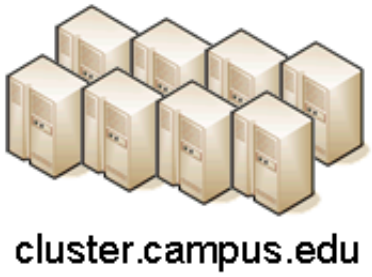
Sites

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 tools.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.
- remoteBatchAccount - some batch systems requirement that an account be provided in addition to user information.
- remoteBatchSystem - the possible batch submission systems include CONDOR, PBS, SGE, 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
- parallelEnvironment - sge parameter
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteApplicationRootDirectory - define directory where application executables are located.
- 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. (deprecated)
- executableClassificationsAllowed - classifications accepted by site. Classifications are set in appaccess.dat
- 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.
- undeclaredSiteSelectionWeight - used when no site is specified to choose between sites where selection weight > 0.
- minimumWallTime - minimum walltime allowed for site or queue. Time should be expressed in minutes.
- maximumWallTime - maximum walltime allowed for site or queue. Time should be expressed in minutes.
- minimumCores - minimum number of cores allowed for site or queue.
- maximumCores - maximum number of cores allowed for site or queue.
- pegasusTemplates - pertinent pegasus templates for site, rc, and transaction files.

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




```
Session  Edit  View  Bookmarks  Settings  Help
$ hostname -f
cluster.campus.edu
$ whoami
yourhub
$ echo ${HOME}
/home/yourhub
$ printenv | SCRATCH
CLUSTER_SCRATCH=/scratch/yourhub
```

```
[cluster]
venues = cluster.campus.edu
remotePpn = 8
remoteBatchSystem = PBS
remoteBatchQueue = standby
remoteUser = yourhub
remoteManager = mpich-intel64
venueMechanism = ssh
remoteScratchDirectory = /scratch/yourhub
siteMonitorDesignator = clusterPBS
```

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. Destination may exist in sites.dat or be a grid site defined by a ClassAd file.
- executablePath - path to executable at remote site. The path may be given as an absolute path on the remote site or a path relative to remoteApplicationRootDirectory defined in sites.dat.
- 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.
- environment - comma separated list of environment variables in the form e=v.
- 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 yourhub account on a remote site.


```
Session  Edit  View  Bookmarks  Settings  Help
$ cd ${HOME}
$ ls -R
.:
apps

./apps:
planets  stars

./apps/planets:
bin

./apps/planets/bin:
earth.x  jupiter.x  mars.x  mercury.x  neptune.x  saturn.x  uranus.x  venus.x

./apps/stars:
bin

./apps/stars/bin:
antares.x  betelgeuse.x  polaris.x  sun.x
```

```
[earth]
destinations = cluster
executablePath = ${HOME}/apps/planets/bin/earth.x
remoteManager = mpich-intel

[sun]
destinations = cluster
executablePath = ${HOME}/apps/stars/bin/sun.x
remoteManager = mpich-intel
```

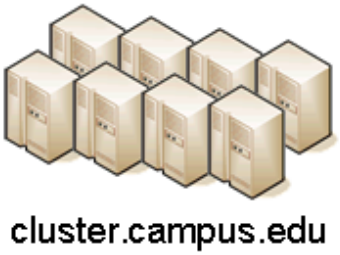
Monitors

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.
- venueMechanism - monitoring job launch process. The default is ssh.
- tunnelDesignator - name of tunnel defined in tunnels.dat.

- remoteUser - login user at remote site.
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteMonitorCommand - command to launch monitor daemon process.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

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




```
Session  Edit  View  Bookmarks  Settings  Help
$ qstat -u yourhub

cluster.campus.edu:
                                                                  Req'd   Elap
Job ID          Username Queue   Jobname SessID NDS TSK Time  S Time
--------------- -------- ------- ------- ------ --- --- ----- - -----
9823388.steele- yourhub  standby earth    3508   2  16  04:00 R 02:15
9824065.steele- yourhub  standby sun      9975   1   1  04:00 R 01:26
```

```
[clusterPBS]
venue = cluster.campus.edu
remoteUser = yourhub
remoteMonitorCommand = ${HOME}/SubmitMonitor/monitorPBS.py
```

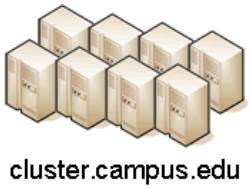
Multi-processor managers

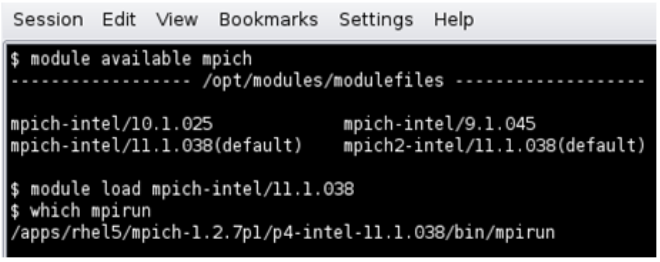
Multiprocessor managers are defined in the file managers.dat. Each manager is defined by a stanza indicating the set of commands used to execute a multiprocessor simulation run. Defined keywords are

- [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
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a typical MPI instance. The given command should be suitable for /bin/sh execution.






```
[mpich-intel]
preManagerCommands = . ${MODULESHOME}/init/sh, module load mpich-
intel/11.1.038
managerCommand = mpirun -machinefile ${PBS_NODEFILE} -np NPROCESSORS
```

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

File access controls

Application or file level access control is described by entries listed in the file appaccess.dat. The ability to transfer files from the HUB to remote sites is granted on a group basis as defined by white and black lists. Each list is given a designated priority and classification. In cases where a file appears on multiple lists, the highest priority takes precedence. Simple wildcard operators are allowed the in the filename declaration allowing for easy listing of entire directories. Each site lists acceptable classification(s) in sites.dat. Defined keywords are

- [group] - group name.
- whitelist - comma separated list of paths. Wildcards allowed.

- blacklist - comma separated list of paths. Wildcards allowed.
- priority - higher priority wins
- classification - apps or user. user class are treated are arbitrary executables.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example file giving permissions reminiscent of those defined in earlier submit releases is presented here

```
[public]
whitelist = /apps/.*
priority = 0
classification = apps

[submit]
whitelist = ${HOME}/.*
priority = 0
classification = home
```

The group public is intended to include all users. Your system may use a different group such as users for this purpose. The definitions shown here allow all users access to files in /apps where applications are published. Additionally members of the submit group are allowed to send files from their $HOME directory.

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 variables used by openmp and mpich is presenter here.

```
# environment variables listed here can be specified from the command
line with -e/--env option. Attempts to specify other environment varia
bles will be ignored and the values will not be passed to the remote s
ite.
```

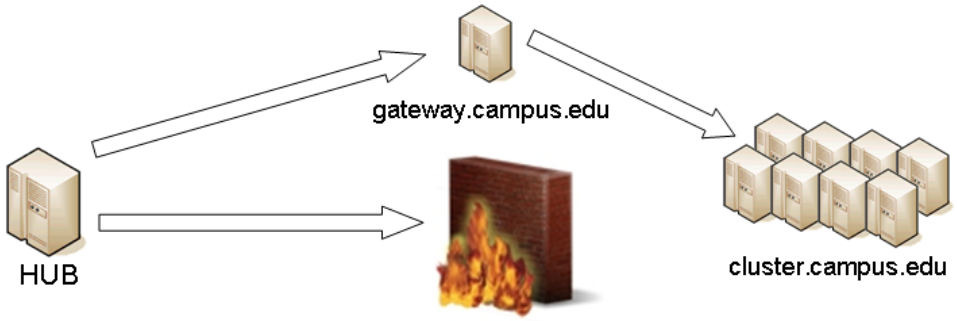
```
OMP_NUM_THREADS
MPICH_HOME
```

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.campus.edu
venuePort = 22
gatewayHost = gateway.campus.edu
gatewayUser = yourhub
localPortOffset = 1
```

Initialization Scripts and Log Files

The submit server and job monitoring server must be started as daemon processes running on the the submit host. If ssh tunneling is going to be used an addition server must be started as a daemon process. Each daemon process writes to a centralized log file facilitating error recording and debugging.

Initialize daemon scripts

Scripts for starting the server daemons are provided and installed in /etc/init.d. The default settings for when to start and terminate the scripts are adequate.

Log files

Submit processes log information to files located in the /var/log/submit directory tree. The exact location varies depending on the vintage of the installation. Each process has its own log file. The three most important log files are submit-server.log, distributor.log, and monitorJob.log.

submit.log

The submit-server.log file tracks when the submit server is started and stopped. Each connection from the submit client is logged with the command line and client ip address reported. All log entries are timestamped and reported by submit-server process ID (PID) or submit ID (ID:) once one has been assigned. Entries from all jobs are simultaneously reported and intermingled. The submit ID serves as a good search key when tracing problems. Examples of startup, job execution, and termination are given here. The job exit status and time metrics are also recorded in the MyQSL database JobLog table.

```
[Sun Aug 26 17:28:24 2012] 0: ########################################
###########
[Sun Aug 26 17:28:24 2012] 0: Backgrounding process.
[Sun Aug 26 17:28:24 2012] 0: Listening: protocol='tcp', host='', port
=830
```

```
[Sun Sep 23 12:33:28 2012] (1154) ==================================
===============
```

```
[Sun Sep 23 12:33:28 2012] (1154) Connection to tcp://:830 from ('192.
168.224.14', 38770)
[Sun Sep 23 12:33:28 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:28 2012] 0: Cumulative job load is 0.84.  (Max: 510.
00)
[Sun Sep 23 12:33:28 2012] 1670: Args are:['/usr/bin/submit', '--local
', '-p', '@@iv=-3:1.5:3', '/home/hubzero/user/hillclimb/bin/hillclimb1
.py', '--seed', '10', '--initialvalue', '@@iv', '--lowerbound', '-3',
'--upperbound', '3', '--function', 'func2', '--solutionslog', 'solutio
ns.dat', '--bestresultlog', 'best.dat']
[Sun Sep 23 12:33:28 2012] 1670: Server stopping.
[Sun Sep 23 12:33:28 2012] 1670: Server(JobExecuter) exiting(2).
[Sun Sep 23 12:33:38 2012] (1154) ================================
================
[Sun Sep 23 12:33:38 2012] (1154) Connection to tcp://:830 from ('192.
168.224.14', 38774)
[Sun Sep 23 12:33:38 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=1:local status=0 cp
u=0.030000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=2:local status=0 cp
u=0.040000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=3:local status=0 cp
u=7.050000 real=7.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=4:local status=0 cp
u=0.080000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=5:local status=0 cp
u=0.020000 real=1.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue= status=0 cpu=10.42
8651 real=9.561828 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Server(JobExecuter) exiting(0).
[Sun Sep 23 12:48:44 2012] (1154) ================================
================
```

```
[Sun Aug 26 17:28:17 2012] 0: Server(10836) was terminated by a signal
 2.
[Sun Aug 26 17:28:17 2012] 0: Server(Listener) exiting(130).
```

distributor.log

The distributor.log file tracks each job as it progresses from start to finish. Details of remote site

assignment, queue status, exit status, and command execution are all reported. All entries are timestamped and reported by submit ID. The submit ID serves as the key to join data reported in submit-server.log. An example for submit ID 1659 is listed here. Again the data for all jobs are intermingled.

```
[Sun Sep 23 00:04:21 2012] 0: quotaCommand = quota -w | tail -n 1
[Sun Sep 23 00:04:21 2012] 1659: command = tar vchf 00001659_01_input.
tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L .__lo
cal_jobid.00001659_01 sayhiinquire.dax
[Sun Sep 23 00:04:21 2012] 1659: remoteCommand pegasus-
plan --dax ./sayhiinquire.dax
[Sun Sep 23 00:04:21 2012] 1659: workingDirectory /home/hubzero/user/d
ata/sessions/3984L
[Sun Sep 23 00:04:21 2012] 1659: command = tar vrhf 00001659_01_input.
tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L/00001
659/01 00001659_01.sh
[Sun Sep 23 00:04:21 2012] 1659: command = nice -n 19 gzip 00001659_01
_input.tar
[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/receiveinpu
t.sh /home/hubzero/user/data/sessions/3984L/00001659/01 /home/hubzero/
user/data/sessions/3984L/00001659/01/.__timestamp_transferred.00001659
_01
[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/submitbatch
job.sh /home/hubzero/user/data/sessions/3984L/00001659/01 ./00001659_0
1.pegasus
[Sun Sep 23 00:04:23 2012] 1659: remoteJobId = 2012.09.23 00:04:22.996
 EDT:   Submitting job(s).
2012.09.23 00:04:23.002 EDT:   1 job(s) submitted to cluster 946.
2012.09.23 00:04:23.007 EDT:
2012.09.23 00:04:23.012 EDT:   ----------------------------------------
--------------------------------
2012.09.23 00:04:23.017 EDT:   File for submitting this DAG to Condor
          : sayhi_inquire-0.dag.condor.sub
2012.09.23 00:04:23.023 EDT:   Log of DAGMan debugging messages
          : sayhi_inquire-0.dag.dagman.out
2012.09.23 00:04:23.028 EDT:   Log of Condor library output
          : sayhi_inquire-0.dag.lib.out
2012.09.23 00:04:23.033 EDT:   Log of Condor library error messages
          : sayhi_inquire-0.dag.lib.err
2012.09.23 00:04:23.038 EDT:   Log of the life of condor_dagman itself
          : sayhi_inquire-0.dag.dagman.log
2012.09.23 00:04:23.044 EDT:
2012.09.23 00:04:23.049 EDT:   ----------------------------------------
--------------------------------
2012.09.23 00:04:23.054 EDT:
2012.09.23 00:04:23.059 EDT:   Your Workflow has been started and runs
```

```
 in base directory given below
2012.09.23 00:04:23.064 EDT:
2012.09.23 00:04:23.070 EDT:   cd /home/hubzero/user/data/sessions/398
4L/00001659/01/work/pegasus
2012.09.23 00:04:23.075 EDT:
2012.09.23 00:04:23.080 EDT:   *** To monitor the workflow you can run
 ***
2012.09.23 00:04:23.085 EDT:
2012.09.23 00:04:23.090 EDT:   pegasus-status -l /home/hubzero/user/da
ta/sessions/3984L/00001659/01/work/pegasus
2012.09.23 00:04:23.096 EDT:
2012.09.23 00:04:23.101 EDT:   *** To remove your workflow run ***
2012.09.23 00:04:23.106 EDT:   pegasus-remove /home/hubzero/user/data/
sessions/3984L/00001659/01/work/pegasus
2012.09.23 00:04:23.111 EDT:
2012.09.23 00:04:23.117 EDT:   Time taken to execute is 0.993 seconds
[Sun Sep 23 00:04:23 2012] 1659: confirmation: S(1):N Job
[Sun Sep 23 00:04:23 2012] 1659: status:Job N WF-DiaGrid
[Sun Sep 23 00:04:38 2012] 1659: status:DAG R WF-DiaGrid
[Sun Sep 23 00:10:42 2012] 0: quotaCommand = quota -w | tail -n 1
[Sun Sep 23 00:10:42 2012] 1660: command = tar vchf 00001660_01_input.
tar --exclude='*.svn*' -C /home/hubzero/clarksm .__local_jobid.0000166
0_01 noerror.sh
[Sun Sep 23 00:10:42 2012] 1660: remoteCommand ./noerror.sh
[Sun Sep 23 00:10:42 2012] 1660: workingDirectory /home/hubzero/clarks
m
[Sun Sep 23 00:10:42 2012] 1660: command = tar vrhf 00001660_01_input.
tar --exclude='*.svn*' -C /home/hubzero/clarksm/00001660/01 00001660_0
1.sh
[Sun Sep 23 00:10:42 2012] 1660: command = nice -n 19 gzip 00001660_01
_input.tar
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/receiveinpu
t.sh /home/hubzero/clarksm/00001660/01 /home/hubzero/clarksm/00001660/
01/.__timestamp_transferred.00001660_01
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/submitbatch
job.sh /home/hubzero/clarksm/00001660/01 ./00001660_01.condor
[Sun Sep 23 00:10:42 2012] 1660: remoteJobId = Submitting job(s).
1 job(s) submitted to cluster 953.
[Sun Sep 23 00:10:42 2012] 1660: confirmation: S(1):N Job
[Sun Sep 23 00:10:42 2012] 1660: status:Job N DiaGrid
[Sun Sep 23 00:11:47 2012] 1660: status:Simulation I DiaGrid
[Sun Sep 23 00:12:07 2012] 1660: Received SIGINT!
[Sun Sep 23 00:12:07 2012] 1660: waitForBatchJobs: nCompleteRemoteJobI
ndexes = 0, nIncompleteJobs = 1, abortGlobal = True
[Sun Sep 23 00:12:07 2012] 1660: command = /opt/submit/bin/killbatchjo
b.sh 953.0 CONDOR
```

```
[Sun Sep 23 00:12:07 2012] 1660: Job 953.0 marked for removal

[Sun Sep 23 00:12:07 2012] 1660: status:Simulation I DiaGrid
[Sun Sep 23 00:12:52 2012] 1660: status:Simulation D DiaGrid
[Sun Sep 23 00:12:52 2012] 1660: venue=1:localCONDOR:953.0:DiaGrid sta
tus=258 cputime=0.000000 realtime=0.000000 waittime=0.000000 ncpus=1
[Sun Sep 23 00:28:14 2012] 1659: status:DAG D WF-DiaGrid
[Sun Sep 23 00:28:14 2012] 1659: waitForBatchJobs: nCompleteRemoteJobI
ndexes = 1, nIncompleteJobs = 0, abortGlobal = False
[Sun Sep 23 00:28:14 2012] 1659: command = /opt/submit/bin/cleanupjob.
sh /home/hubzero/user/data/sessions/3984L/00001659/01
[Sun Sep 23 00:28:15 2012] 1659:
*******************************************SUMMARY*****************
*****************************

Job instance statistics           : /home/hubzero/user/data/sessions/3
984L/00001659/01/work/pegasus/statistics/jobs.txt

****************************************************************************
*****************************

[Sun Sep 23 00:28:15 2012] 1659: venue=1:localPEGASUS:946.0:WF-DiaGrid
 status=0 cputime=1.430000 realtime=2.000000 waittime=0.000000 ncpus=1
[Sun Sep 23 00:28:15 2012] 1659: venue=2:PEGASUS:952.0:DiaGrid status=
0 cputime=0.003000 realtime=0.000000 waittime=681.000000 ncpus=1 event
=/sayhi_inquire-sayhi-1.0
[Sun Sep 23 00:28:15 2012] 1659: venue=3:PEGASUS:954.0:DiaGrid status=
0 cputime=0.003000 realtime=0.000000 waittime=631.000000 ncpus=1 event
=/sayhi_inquire-inquire-1.0
```

monitorJob.log

The monitorJob.log file tracks the invocation and termination of each remotely executed job monitor. The remote job monitors are started on demand when job are submitted to remote sites. The remote job monitors terminate when all jobs complete at a remote site and no new activity has been initiated for a specified amount of time - typically thirty minutes. A typical report should look like:

```
[Sun Aug 26 17:29:16 2012] (1485) ***********************************
[Sun Aug 26 17:29:16 2012] (1485) * distributor job monitor started *
[Sun Aug 26 17:29:16 2012] (1485) ***********************************
[Sun Aug 26 17:29:16 2012] (1485) loading active jobs
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from DB file
```

```
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from dump file
[Sun Aug 26 17:29:16 2012] (1485) 4 jobs purged
[Sun Aug 26 17:29:16 2012] (1485) 11 monitored jobs
[Sun Aug 26 18:02:04 2012] (24250) Launching wf-diagrid
[Sun Aug 26 18:02:04 2012] (1485) 12 monitored jobs
[Sun Aug 26 18:02:15 2012] (1485) Update message received from wf-
diagrid
[Sun Aug 26 18:03:15 2012] (1485) Update message received from wf-
diagrid
[Sun Aug 26 18:06:43 2012] (1485) 13 monitored jobs
...
[Thu Sep 17 17:32:51 2011] (21095) Received SIGTERM!
[Thu Sep 17 17:32:51 2011] (21095) Send TERM to child ssh process
[Thu Sep 17 17:32:51 2011] (21095) distributor site monitor stopped
[Thu Sep 17 17:32:51 2011] (17348) Send TERM to child site steele proc
ess
[Thu Sep 17 17:32:51 2011] (17348) **********************************
[Thu Sep 17 17:32:51 2011] (17348) * distributor job monitor stopped *
[Thu Sep 17 17:32:51 2011] (17348) **********************************
```

It is imperative that the job monitor be running in order for notification of job progress to occur. If users report that their job appears to hang check to make sure the job monitor is running. If necessary take corrective action and restart the daemon.

monitorTunnel.log

The monitorTunnel.log file tracks invocation and termination of each ssh tunnel connection. If users report problems with job submission to sites accessed via an ssh tunnel this log file should be checked for indication of any possible problems.

Remote Domain 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:

The directory defined by MONITORLOGLOCATION needs to be created before the daemon is started. Sample daemon scripts used for PBS, LSF, SGE, Condor, Load Leveler, and Slurm batch systems are included in directory BatchMonitors.

File transfer and batch job scripts

The simple scripts are used to manage file transfer and batch job launching and termination. The location of the scripts is entered in sites.dat.

Examples scripts suitable for use with PBS, LSF, Condor, Load Leveler, and Slurm are included in directory Scripts. After modifications are made to monitors.dat the central job monitor must be notified. This can be accomplished by stopping and starting the submon daemon or a HUP signal can be sent to the monitorJob.py process.

File transfer - input files

Receive compressed tar file containing input files required for the job on stdin. The file

transferredTimestampFile is used to determine what newly created or modified files should be returned to the HUB.

```
receiveinput.sh jobWorkingDirectory jobScratchDirectory  transferredTi
mestampFile
```

Batch job script - submission

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 identifier. The log file should be placed at a location agreed upon by the remote site and HUB.

```
submitbatchjob.sh jobWorkingDirectory jobScratchDirectory jobDescripti
onFile
```

The jobId is returned on stdout if job submission is successful. For an unsuccessful job submission the returned jobId should be -1.

File transfer - output files

Return compressed tar file containing job output files on stdout.

```
transmitresults.sh jobWorkingDirectory
```

File transfer - cleanup

Remove job specific directory and any other dangling files

```
cleanupjob.sh jobWorkingDirectory jobScratchDirectory jobClass
```

Batch job script - termination

Terminate given remote batch job. Command line arguments specify job identifier and batch system type.

```
killbatchjob.sh jobId jobClass
```

Batch job script - post process

For some jobClassses it is appropriate to preform standard post processing actions. An example of such a jobClass is Pegasus.

```
postprocessjob.sh jobWorkingDirectory jobScratchDirectory jobClass
```

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 executableClassificationsAllowed keyword 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 be granted permission to send files from their $HOME directory. Discrete permission can be granted on a file by file basis in appaccess.dat.

Updates

The host operating system should be updated on a regular basis to ensure operating system security updates are promptly installed.

```
# yum upgrade
```

The above will also update HUBzero packages but they won't all take effect until they are applied to your site. To apply updates to your site run

This will regenerate your apache configuration files. If you modified them directly they will be overwritten. Be sure to apply apache configuration changes to /etc/httpd/sites-m4/hub.m4 and hub-ssl.m4 files in order to retain the changes between updates

```
# hzcms update
```

Add-ons

Introduction

Add-ons for HUBzero are available here. Currently these consist of a couple projects that have not yet been fully integrated into the the HUBzero packaging and installation process.

Solr-powered Search

Introduction

Apache Solr is a search engine platform which is relatively mature and has a lot of powerful and flexible configurations. There has been extensive work to implement it into the HUBzero CMS and is currently a work-in-progress.

Solr is an open-source, mature, and stable searching service that is built upon the Apache Lucene search engine. The service provides features which lend itself to scaling and has a rich open source community. It is a Java-based service which provides search results through HTTP. Many companies such as Instagram, eBay, and StubHub rely on Solr to provide advanced searching capabilities.

The integration with Solr is currently under heavy development. It is **strongly recommended** to test on a QA / Stage host before using in a production environment.

Installation & First Time Configuration

Step 1: Install the hubzero-solr package

A system administrator must install the hubzero-solr RedHat or Debian Package using a package manager such as yum or aptitude. The package contains a version of Apache Solr and the configuration necessary for Solr to integrate with the CMS.

```
For RedHat / CentOS:
$    sudo yum install hubzero-solr
```

```
For Debian:
$    sudo apt-get install hubzero-solr
```

Once installed the service will need to be enabled.

```
$    sudo service hubzero-solr start
```

Step 2: Configure Search Service in the CMS

The HUBzero CMS needs to know to use Solr Search instead of Basic Search. To do this, a Hub administrator will need to log into the Administrative Backend and Configure the Search Component.

You will need to set **Engine** to *Apache Solr.* Then click the "Solr tab".

The Solr tab's default settings will work for the open-source distribution.

HUBzero-hosted hubs are configured with different ports! The following scheme is used:

```
Development (dev.hub.org): 2090
Stage (stage.hub.org): 2091
Scan / QA (qa.hub.org): 2092
Production (hub.org): 2093
```

Click "Save and Close" to save the settings. If the hubzero-solr service is started and the correct settings were set in the steps above, the status screen should indicate that the search engine is responding.

If there were any issues with configuration, the following screen will appear.

This would be a point where a support ticket is filed for the system administrator to confirm that the service is running. Please include all configuration parameters contained in Step #3 when

filing the ticket.

Step 3: Enable the Search Background Worker

In order to keep the search index fresh, a background worker is implemented to process data from the CMS and push it into the Solr service.

Currently the background worker is implemented as a Cron task that is called once a minute. There is work being done to develop a daemon which listens to CMS events and processes data without relying on Cron.

To setup the Cron-based worker a Hub administrator must go into the Administrative Backend, go to Components, Cron, and add the Task as shown below:

Click "Save and Close".

Step 4: Build the Initial Index

This implementation of Solr has hooks into the CMS which updates the index when a new record is added or marked for deletion. It will be necessary to add items which have been added before Solr was activated.

This operation should only need to be completed once. You will be unable to start this operation until it finishes for the first time.

The "Full Index" button populates a Queue which is periodically serviced by a worker. The worker will process the records and format for consumption by the Solr service. **This may take several hours to fully complete if the Hub has a lot of content.**

If an error with the worker occurs, a warning message such as this will appear.

Search Breadth

Te question is “What can I search for?”. The answer is “anything you have access to contained within the list in Search Categories. To see all content within these categories perform a simple query using the wildcard character “*” as shown below.

A better of what is currently inside Solr’s index can be viewed on the administrative backend by going to Components >> Search >> Search Index Tab. The number of index items is located to next to each type. Clicking on the name of the hub type will perform a search on that type, displaying all items that are within the index of that type.

For instance clicking “Resources” shows the following screen:

One can perform additional searching using the “Filter” bar on top of the results listing.

Shibboleth authentication

Shibboleth Authentication is an unsupported feature of HUBzero. This documentation has not been verified against HUBzero 2.2 or RedHat. It is included here for completeness. Please send any corrections or feedback to support@hubzero.org

Installation

If installing on Debian GNU/Linux

```
# apt-get install hubzero-shibboleth
```

If installing on Redhat/CentOS Linux

```
# yum install hubzero-shibboleth
```

If you installed the hubzero-shibboleth package on Debian, you're set. The relevant packages were included as dependencies. The packages are:

shibboleth2-sp-utils shibboleth2-sp-schemas libapache2-mod-shib2

At the time of this writing, Shibboleth is not distributed in the core repositories for Redhat/CentOS. You can read about how to add a repo that has what you need here:

https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLinuxRPMInstall

Generate a private key

Use shib-keygen to generate /etc/shibboleth/sp-key.pem. Note that this utility may not be on your path unless you are root.

Configure Shibboleth

Shibboleth official quickstart documentation reference

The main configuration file is located at /etc/shibboleth/shibboleth2.xml. There are some other files that might be of interest to you here, but the defaults are acceptable to get your hub working with InCommon.

In shibboleth2.xml:

- Update <ApplicationDefaults entityID="{url}" ...> so {url} is https://{your hostname}/Shibboleth.sso. This is your Shibboleth endpoint, designated later by the Apache configuration as the location where the shib2 module will manage communication with ID providers.
- Update <Sessions ...handlerSSL="false" ...> to handlerSSL="true", if it is not already

Configure Apache

Shibboleth official Apache configuration reference

Ensure that Apache is loading the module. Typically this means that there is a link in mods-enabled to shib2.load in mods-available

ln -s /etc/apache2/mods-available/shib2.load /etc/apache2/mods-enabled

If you do not have this directory structure you can also enable the module directly in the next step by adding this to your Apache configuration file:

LoadModule mod_shib /usr/lib/apache2/modules/mod_shib2.so

In the conf file defining your SSL host, (usually located in /etc/apache2/sites-enabled):

- If not already set in the SSL <VirtualHost> UseCanonicalName on;
- To enable shibd's endpoint, add: <Location /Shibboleth.sso> SetHandler shib </Location>

- HUBzero CMS routing will stomp on /Shibboleth.sso unless you change the mod_rewrite rules a bit.
    - You should have a line like: RewriteRule (.*) index.php probably preceded by a few 'RewriteCond's. Add a new condition to exempt the shib2-controlled path:

RewriteCond %{REQUEST_URI} !/Shibboleth.sso/.*$ [NC]

Restart apache: /etc/init.d/apache2 restart

Verify

From the same host (this is IP-restricted):

wget -q --no-check-certificate https://localhost/Shibboleth.sso/Metadata -O - | tee /etc/shibboleth/sp-metadata.xml

This command should write XML to the listed file (and stdout) wrapped in <md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" ...>

If it does not, review the references above to troubleshoot.

You may skip to "Configuring HUBzero CMS!" if you do not want to test interop more thoroughly with the TestShib ID provider, but I would recommend you do this test.

Upload metadata to TestShib.org

- Copy the metadata generated above to some unique name, for example:

cp /etc/shibboleth/sp-metadata ~/{your hostname}-sp-metadata.xml

- Upload that file here: https://www.testshib.org/register.html. Uploading a file of the same name will overwrite it on the testshib server, should you need to make any adjustments.

Change your local configuration to accept TestShib as an ID provider

- Visit this URL to get an appropriate test configuration XML file:

https://www.testshib.org/cgi-bin/sp2config.cgi?dist=Others&hostname={your hostname from the Shibboleth Configuration step above})

- Assuming that looks OK, copy the output over your existing

/etc/shibboleth/shibboleth2.xml:

wget -q --no-check-certificate "https://www.testshib.org/cgi-bin/sp2config.cgi?dist=Others&hostname={your hostname}" -O /etc/shibboleth/shibboleth2.xml

Restart services: /etc/init.d/shibd restart && /etc/init.d/apache2 restart

Configuring HUBzero CMS

If you do not already have plg_authentication_shibboleth installed, this package installs a tarball in $(PREFIX)/usr/lib/hubzero that you may install using HUBzero CMS's package management interface at /administrator.

Manage ID providers on HUBzero CMS's admin page

In Extensions->Plugins, select Authentication - Shibboleth

Ideally it should look a lot like the screenshot in that it found testshib in your XML configuration. If so, you can click the down arrow by that entry to move it into your active provider list.

This may fail if, for example, shibboleth2.xml is not readable by the web user, or if you changed your configuration so that the file is located somewhere unexpected.

It is not necessary, however, for the web server to read this file. If you'd like you can simply enter the EntityID for testshib (https://idp.testshib.org/idp/shibboleth) in the white box with the button labeled "Add ID provider". Enter something, eg "TestShib" for the label.

Quick run-down of the fields here:

- Entity id: (required) corresponds to the corresponding entityId in shibboleth2.xml and must match exactly for things to work out.
- Label: (required) name to show on the log-in button of your hub for this provider
- Initialism: (optional) if you have more than ten supported ID providers, the log-in list becomes searchable, and in this case you can add a short name for institutions so that they will come up when the user types that as well as when they type a portion of the label. (For example, if you federated with the National Science Foundation you might add "NSF" here)
- Host: (optional) institutions may be pre-selected if the IP address of the user looks like it is in a particular network, eg, to follow the previous example, nsf.gov to pre-select the National Science Foundation
- Logo: (optional) also shown on the button. Enter a URL here to make a iconified copy of it. You may have better results in some cases if you resize to no more than 28px in either extent yourself.

Finally, you can select the order in which you would like the button to appear on the login page here. When you're done, click "Save & Close" in the top right. This will take you back to the screen where you can click the icon in the "Status" column to enable the plug-in.

Try logging in!

(If you run into any problems here, there might be a clue in TestShib's logs of its ID provider actions)

Since TestShib doesn't release any attributes, you'll have to enter a name when you log in. Hopefully you can negotiate to get names and emails released to your hub with "real" ID providers, which you're now clear to do if everything worked out.

Help?

- If you have a problem that you can't resolve that appears to be related to Shibboleth's machinery, please consult the official documentation carefully.

- If you can't resolve the problem, there is a mailing list:

https://www.shibboleth.net/community/lists/

- If the problem you are experiencing appears to be related not to the Shibboleth interchange mechanism but to something in the hub's implementation of the log-in procedure, visit https://help.hubzero.org/support to enter a support ticket describing the situation.

- Test that you can access https://idp.purdue.edu/idp/shibboleth or similar directly.

InCommon

InCommon Authentication is an unsupported feature of HUBzero. This documentation has not been verified against HUBzero 2.2 or Debian 8+. It is included here for completeness. Please send any corrections or feedback to support@hubzero.org

Introduction

This plugin provides some code necessary to allow your hub to accept credentials using the Shibboleth system. Most commonly, this implies membership in the InCommon network.

Shibboleth has some particular architectural demands, namely that it will install a new daemon and a new Apache module on your system. InCommon has some administrative demands, in that you will need to negotiate to get your hub added to their XML manifest as a service provider.

Installation

Debian

```
# apt-get install -y libapache2-mod-shib2
```

Redhat Enterprise Linux & other distributions

See Shibboleth wiki entry on service provider installation for information on how to add the Shibboleth software to your list of repositories so that it can be installed and upgraded through yum, or, failing that, how to install from SRPMS.

Configuration

Shibboleth

Certificates

As root, run the script shib-keygen, which was installed as part of the package. This will generate a key pair for your service provider to use. No further configuration is required for this; the software will find the keys when the shibd service is restarted.

output

```
Generating a 2048 bit RSA private key
.....................................................................
..................+++
.....................+++
writing new private key to '/etc/shibboleth/sp-key.pem'
-----
```

/etc/shibboleth/attribute-map.xml

This file controls which attributes (bits of user information) the software will extract during login when the identity provider makes them available.

Make sure the following pertinent attributes are not commented out in both forms of the “name” attribute.

eppn (username, probably already enabled in the shipped configuration):

```
    <Attribute name="urn:mace:dir:attribute-
def:eduPersonPrincipalName" id="eppn">
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
    </Attribute>
    <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" id="eppn">
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
    </Attribute>
```

Name & email (probably not enabled by default):

```
    <Attribute name="urn:mace:dir:attribute-def:sn" id="sn"/>
    <Attribute name="urn:mace:dir:attribute-
def:givenName" id="givenName"/>
    <Attribute name="urn:mace:dir:attribute-
def:displayName" id="displayName"/>
    <Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
    <Attribute name="urn:oid:2.5.4.4" id="sn"/>
    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
    <Attribute name="urn:oid:2.16.840.1.113730.3.1.241" id="displayNam
e"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

/etc/shibboleth/shibboleth2.xml

This is the main configuration, which controls how the software federates with identity providers.

First, replace $YOUR_HOSTNAME with, uh, your hostname, in the entityID attribute near the top of the file:

```
       <ApplicationDefaults entityID="https://$YOUR_HOSTNAME/login/shi
bboleth" REMOTE_USER="eppn persistent-id targeted-id">
```

In the block, delete or comment-out any SSO or SessionInitiator blocks that shipped, and add the two listed below, again interpolating your real hostname. This tells the software to check with the HUBzero CMS plugin about where to redirect for a given authentication request, and allows the HUBzero CMS to selectively enable providers.

```
        <Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
                  checkAddress="true" handlerSSL="true" cookieProps="h
ttps">
            <SSO discoveryProtocol="SAMLDS" ECP="true" discoveryURL="h
ttps://$YOUR_HOSTNAME/login?authenticator=shibboleth&wayf">
                   SAML2 SAML1
            </SSO>
            <SessionInitiator type="Chaining" Location="/login/shibbol
eth" isDefault="true" id="Login">
                <SessionInitiator type="SAML2" template="bindingTempla
te.html"/>
                <SessionInitiator type="Shib1"/>
                <SessionInitiator type="SAMLDS" URL="https://$YOUR_HOS
TNAME/login?authenticator=shibboleth&wayf"/>
            </SessionInitiator>
            <!-- Default <Handler> tags not pictured, but they should
stay -->
       </Sessions>
```

If you run into issues where you seem to be stuck in a redirect loop between the idp and the sp, changing the cookie properties to use a less specific path may help.

```
cookieProps="; path=/; secure; HttpOnly"
```

If this is a production machine you will want to set a real email for the support contact:

```
        <Errors supportContact="support@$YOUR_HOSTNAME"
             helpLocation="/about.html"
             styleSheet="/shibboleth-sp/main.css"/>
```

Finally, you will need to configure how and where the software looks for metadata about identity providers. This is just a list of providers you can support, including some helpful annotations like where the service URLs and what public key to use when communicating with it.

Metadata provider: TestShib

For development and test machines it is often useful to use TestShib, and its configuration looks like this, below the Sessions tag and at the same scope:

```
      <MetadataProvider type="XML" uri="http://www.testshib.org/metada
ta/testshib-providers.xml" backingFilePath="testshib-two-idp-
metadata.xml" reloadInterval="180000"/>
```

Visit the TestShib site for more information about how to set this up, if you're interested. Hopefully you do not need the "Install" selection, but pick up from "Register". During "Configure" it recommends replacing your whole shibboleth2.xml with one it generated. Make a backup if you do, or else just add the MetadataProvider above to your existing configuration.

When you reach "Test", see below for the HUBzero CMS configuration that will add TestShib to the list of available identity providers on your hub.

Metadata provider: InCommon

If your plans include membership in the InCommon consortium, this is the incantation, below the Sessions tag and at the same scope:

```
        <MetadataProvider type="XML" uri="https://wayf.incommonfederat
ion.org/InCommon/InCommon-metadata.xml" backingFilePath="federation-
metadata.xml" reloadInterval="7200">
            <MetadataFilter type="RequireValidUntil" maxValidityInterv
al="2419200"/>
            <MetadataFilter type="Signature" certificate="inc-md-
cert.pem"/>
        </MetadataProvider>
```

Install https://ds.incommon.org/certs/inc-md-cert.pem as /etc/shibboleth/inc-md-cert.pem so it's available for this provider.

Metadata provider: others?

If you are doing one-on-one negotiations with identity providers the metadata situation gets a bit more hairy, but the identity providers in question will probably be able to guide your configuration.

Apache

Quoth the Shibboleth wiki entry on service provider installation:

- UseCanonicalName On
- Ensure that the ServerName directive is properly set, and that Apache is being started with SSL enabled.

Make sure installing the software enabled both the module shib2 and the support daemon shibd.

Typically this means that there is a symlink /etc/apache2/mods-enabled/shib2.load that points to /etc/apache2/mods-available/shib2.load and that this report works:

```
# service shibd status
[ ok ] shibd is running.
```

/etc/apache2/sites-enabled/{your-ssl-enabled-config-file}

Your EntityID is something like https://hostname/login/shibboleth, but the actual URL to pick up the login process again in HUBzero CMS terms is more complicated, so we rewrite it. I recommend putting this statement as high as possible in the config (after RewriteEngine on) so that the "L"ast last triggers and you can be assured the URL is not subsequently rewritten by anything else you're doing.

```
        RewriteCond     %{REQUEST_URI}          ^/login/shibboleth
        RewriteRule     (.*) /index.php?option=com_users&authenticator
```

```
=shibboleth&task=user.login [NC,L]
```

Bind an endpoint to the module. This is used during the login process and is also useful to get a basis for your service provider's metadata, which is served at /Shibboleth.sso/Metadata when the request comes from localhost.

```
        <Location /Shibboleth.sso>
                SetHandler shib
        </Location>
```

You probably have a rule that directs all requests that appear to be for HUBzero CMS content to the index.php bootstrap, and we need to note that /Shibboleth.sso isn't HUBzero CMS business, so make sure you have a RewriteCond like this:

```
        RewriteCond     %{REQUEST_URI}          !^/Shibboleth.sso/.*$
[NC]
        RewriteRule     (.*)                    index.php
```

Finally, we actually protect the entityID location /login/shibboleth. We can redirect a user to this path to require them to make a Shibboleth login. Shibboleth won't know specifically how to do that so it will make a request to the wayf location defined above in shibboleth2.xml. This is part of the HUBzero CMS that knows already which provider the user selected from the login page, so it spits back the appropriate identity provider entityId. From there the metadata is referenced to find the endpoint associated with that institution, and the user is sent to the login page. They come back to /login/shibboleth upon submission, but now the requirement to have a Shibboleth session is satisified, and the rewritten URL referencing user.login is served to complete the process.

```
        <Location /login/shibboleth>
                AuthType shibboleth
                ShibRequestSetting requireSession 1
                Require valid-user
        </Location>
```

Restart the shibd and apache2 services when satisified with this configuration.

HUBzero CMS

Plugin

Log in to /administrator, choose Extensions and then Plugin Manager, and locate the Shibboleth plugin in the Authentication category.

If you would like to selectively hide the Shibboleth login options for testing, enter something in the “Testing mode key” field. This term must appear in the query string for the controls of the plugin to render. For example, if you enter “incommon” you should test the login page by visiting “/login?reset=1&incommon” (reset=1 in case it remembers your logging in with a different method, in which case you’ll only see the controls for that method anyway).

The links section is useful only for testing, but it can be used to destroy a link between your test account and a particular institution so that you can try it again.

The Institutions section is where you manage which options are presented on the login page.

An example of an entry here, for the TestShib public identity provider test mechanism described above:

```
Entity ID: https://idp.testshib.org/idp/shibboleth
Label: TestShib
Host: testshib.org
```

The entity ID must strictly match what you have in your metadata provider, but the label is free-form and the host is optional. The login page attempts to do a reverse-DNS of the user’s IP to see if they are on a particular network. If it turned out in this case that the client was from *.testshib.org this option would be pre-selected in the plugin’s controls.

Save your settings with the button near the top right of the page when you’re done.

Centos / RHEL 7

The HUBzero Platform

Installation instructions for Red Hat Enterprise Linux and CentOS Linux. The Hubzero tools infrastructure is not available at this time.

Please use the side menu to navigate this documentation.

Upgrading from CentOS 6 to CentOS 7

Upgrading from CentOS 6 to CentOS 7

1. Make sure your current host is fully upgraded with the latest OS packages and latest Hubzero CMS code. See https://help.hubzero.org/documentation/22/installation/redhat/updates
2. CentOS / RHEL recommends migrating to a CentOS 7 host and not a in-place upgrade. Setup a new host with the CentOS 7 installed and follow the Hubzero installation instructions at https://help.hubzero.org/documentation/22/installation/centos7/install .
3. Migrate data and some configurations. It is the responsibility of the system administrator of the host to decide what should and shouldn't be migrated to the new host. Not everything can be migrated en mass to the new operating system as it will not be compatible.
    1. CMS data migration
        1. Copy the following directories and its sub-directories to the new host at the same location. Do not copy all of /var/www/[hubname]/app/ - it will squash some of the new CMS configuration. Other directories in app/ may copied as needed if they are being used, com_component directories, in the app/components directory, for example.
            1. /var/www/[hubname]/app/site/
            2. /var/www/[hubname]/app/template/[yourtemplate]
        2. Export and Import the CMS database.
            1. Make sure the version of MariaDb are the same on the old and new host.
            2. Export the current database in it's entirety as root.
            3. On the Centos7 host, as root, import the database.
            4. Update the mysql database password in /etc/hubzero.secrets/ and in /var/www/[hubname]/app/config/database.php
        3. At the discretion of the system administrator, you may want to copy log files or other data on the current host to the new host.
        4. If modifications have been made to the current host in the /etc/ directory or other conf, we recommend merging the needed configuration into the new hosts rather than a direct copy and overwrite.

Updates

The host operating system should be updated on a regular basis to ensure operating system security updates are promptly installed.

```
# yum upgrade
```

The above will also update HUBzero packages but they won't all take effect until they are applied to your site. To apply updates to your site run

This will regenerate your apache configuration files. If you modified them directly they will be overwritten. Be sure to apply apache configuration changes to /etc/httpd/sites-m4/hub.m4 and hub-ssl.m4 files in order to retain the changes between updates

```
# hzcms update
```

Installation

Target Audience

This document and the installation and maintenance of a HUBzero system has a target audience of **experienced** Linux administrators (preferably experienced with RedHat or CentOS distributions).

Minimum System Requirements

HUBzero (RedHat) installations require one or more dedicated hosts running RedHat or CentOS version 7.

A typical starter HUBzero installation might consist of a single physical server with dual 64-bit quad-core CPUs, 24 Gigabytes of RAM and a terabyte of disk.

Production systems should try to not limit hardware resources, HUBzero is designed to run on systems with many CPU cores and lots of RAM. If you are looking for a system to run a small site with limited physical or virtual resources this is probably not the system for you. However, for demonstration or development purposes we often create VM images with less than a gigabyte of RAM and 5 gigabytes of disk. While fully functional, these virtual machines would only be suitable for a single user doing development or testing.

System Architecture

All hardware, filesystem partitions, RAID configurations, backup models, security models, etc. and base configurations of the hosts email server, SSH server, network, etc. are the responsibility of the system administrator managing the host.

The Hubzero software expects to be installed on a headless server from a minimal ISO with only one network interface (required by OpenVZ) with an MTU no less than '1500'. System accounts must not be created with an id of 1000 or greater - more about that in a forthcoming section.

Linux

Install Basic Operating System

Advanced Linux system administrator skills are required, please read carefully. Selecting all the default configurations during the operating system installation may not always be suitable.

The latest version of RedHat Enterprise Linux 7 or CentOS 7 (x86_64 is the only architecture supported) should be downloaded and installed. Do not install a default LAMP environment or other server packages.

System reboots are required to complete the installation. Be sure to remove the install disk or otherwise reset your server's boot media before rebooting.

The precise server configuration (such as disk partitioning, networking, etc) is dependent on how the hub is to be used and what hardware is being used, all the possible configuration options are not specifically outlined here. This installation guide outlines a very basic configuration but may not be suitable for larger sites. For larger sites, it is generally expected that the hub will be managed by an experienced Linux administrator who can help setup your site to meet your specific requirements.

All hardware, filesystem partitions, RAID configurations, backup models, security models, etc. and base configurations of the hosts email server, SSH server, network, etc. are the responsibility of the system administrator managing the host.

The following instructions only instruct how to install Hubzero software. At a minimum a "Basic Server" host, ideally from a 'minimal' ISO image, is required with network access.

The Hubzero software expects to be installed on a headless server without a Graphical User Interface.

Configure Networking and DNS

Configure your host's network as desired. A registered domain name and SSL certificate is required and should be obtained prior to installation. A static IP address is highly recommended as well.

By default, the Hubzero middleware uses IP addresses in the 192.168.0.0/16 subnet. Do not use an IP address in this range for your host.

Set hostname

Throughout this documentation you will see specific instructions for running commands, with part of the text highlighted. The highlighted text should be modified to your local configuration choices. (e.g. replace "example.com" with the fully qualified hostname of your machine).

HUBzero expects the `hostname` command to return the fully qualified hostname for the system. This step may be skipped if previously configured.

```
sudo hostname hubdomain.org
```

Make the change permanent:

```
sudo hostnamectl set-hostname hubdomain.org
```

Delete local Users

HUBzero reserves all user ids from 1000 up for hub accounts. As part of the app middleware every account must map to a corresponding system account. Therefore when starting up a hub it is required to remove all accounts that have user ids 1000 or greater. New RedHat/CentOS installations typically do not setup a non root account during setup, but if you have any accounts added to the system, those accounts can be removed as follows:

```
sudo userdel username
sudo rm -fr /home/username
```

If you require additional system accounts, they should use user and group ids in the range of 500-999 (these will not interfere with hub operations).

Update the initial OS install

```
sudo yum update -y
```

Disable SELinux

Hubzero does not currently support SELinux. Since the default install of RHEL turns it on, we have to turn it off.

```
sudo sed -i 's/^SELINUX=.*/SELINUX=disabled/g' /etc/selinux/config
```

Reboot the system for this change to take effect

```
sudo reboot
```

Yum repository setup

Configure the hubzero repository configuration package

For RedHat Enterprise Linux 7

```
sudo subscription-manager repos --enable rhel-7-server-optional-rpms

sudo subscription-manager repos --enable rhel-7-server-extras-rpms

sudo rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-
latest-7.noarch.rpm

sudo subscription-manager repos --enable rhel-server-rhscl-7-rpms

sudo rpm -Uvh http://packages.hubzero.org/rpm/julian-el7/hubzero-
release-julian-2.2.7-1.el7.noarch.rpm
```

For CentOS 7

```
sudo yum install -y epel-release

sudo yum install -y centos-release-scl-rh

sudo rpm -Uvh http://packages.hubzero.org/rpm/julian-el7/hubzero-
release-julian-2.2.7-1.el7.noarch.rpm
```

Firewall

Install

```
sudo yum remove -y firewalld
sudo yum install -y hubzero-iptables-basic

sudo service hubzero-iptables-basic start
sudo chkconfig hubzero-iptables-basic on
```

If installing the Hubzero tool infrastructure:

```
sudo yum install -y hubzero-mw2-iptables-basic

sudo service hubzero-mw2-iptables-basic start
sudo chkconfig hubzero-mw2-iptables-basic on
```

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. 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).

Web Server

Install Apache Httpd Web Server

```
sudo yum install -y hubzero-apache2

sudo service httpd start
sudo chkconfig httpd on
```

PHP

Install

HUBzero still requires the use of PHP 5.6 (7.x support coming soon). For CentOS/RedHat 7 you used to be able to get 5.6 via the software collections library and when that was removed we mirrored a copy. These packages are no longer being maintained so are not recommended. The current recommendation for RedHat/CentOS 7 is to use the php56 packages from the Remi Repository (https://rpms.remireponet). These packages are still receiving security updates for serious issues despite PHP 5.6 reaching upstream EOL.

```
sudo rpm -Uvh https://rpms.remirepo.net/enterprise/remi-release-7.rpm
sudo yum install -y hubzero-php56-remi
sudo service php56-php-fpm start
sudo chkconfig php56-php-fpm on
```

Database

MySQL Database Installation

HUBzero should work with any MySQL 5.5 compatible database. We have used MySQL 5.5.x, 5.6.x, MariaDB 5.5.x, 10.1.x, 10.3.x, Percona XtraDB Cluster 5.7.x.

For CentOS 7 and Redhat Enterprise 7 we recommend MariaDB 5.5.x directly from the MariaDB rpm repositories as they are maintaining longer term support. (https://downloads.mariadb.org/mariadb/repositories).

CentOS 7 - MariaDB Database Installation

```
cat << EOF > /etc/yum.repos.d/mariadb-5.5.repo
# MariaDB 5.5 CentOS repository list
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF

sudo yum install -y MariaDB-server
sudo service mysql start
sudo chkconfig mysql on
```

RedHat Enterprise Linux 7 - MariaDB Database Installation

```
sudo cat << EOF > /etc/yum.repos.d/mariadb-5.5.repo
# MariaDB 5.5 RedHat repository list
# http://downloads.mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/rhel7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF

sudo yum install -y MariaDB-server
sudo service mysql start
sudo chkconfig mysql on
```

Configure

Default configuration works well for starters. But for optimal performance you will need a database administrator capable of tuning your database to your hardware configuration and site usage.

Mail

Install Postfix

```
sudo yum install -y postfix

sudo service postfix start
sudo chkconfig postfix on
```

Test

```
sudo postfix check
```

If the 'postfix check' command returns anything, resolve the reported issues with the Postfix installation before continuing.

Configure Postfix

Configure Postfix as desired. The default installation may only handle mail on the localhost. HUBzero expects to be able to send mail to registered user's email address to confirm registration and also to send group and support ticket related messages. Incoming mail is also expected to work (see Mailgateway section) in order to receive support ticket updates and email replies to group forum messages. Setting up an appropriate mail configuration is up to the site administrator. The Mailgateway service expects postfix to be the Mail Transfer Agent on CentOS and RedHat systems.

CMS

Installation

```
sudo yum install -y hubzero-cms-2.2
sudo yum install -y hubzero-texvc
sudo yum install -y hubzero-textifier
sudo yum install -y wkhtmltopdf
```

Configuration

```
sudo hzcms install hubname
```

It is necessary to immediately run the updater to apply fixes that have not been incorporated into the initial installation.

```
sudo hzcms update
```

SSL Configuration

The default SSL certificate is a self signed certificate (aka snakeoil) meant for evaluation purposes only. Some browers will not accept this certificate and will not allow access to the site without special configuration (https://support.mozilla.org/en-US/questions/1012036). For a production hub you will need to obtain a valid SSL certificate. A certificate may contain two or three pieces: a public certificate, a private key, and sometimes an intermediate certificate. All files are expected to be in PEM format.

Let's Encrypt and Certbot may be a viable option to obtain SSL certificates - https://certbot.eff.org/lets-encrypt/centosrhel7-apache.html
**Do NOT use Snap. "yum install -y certbot python-certbot-apache", then follow the rest of the instructions to run the 'certbot' commands - start with step #7 'just get a certificate' "certbot certonly --apache".

Once you obtain the certificate, copy the SSL certificate files to the httpd SSL configuration directories and restart httpd.

```
sudo cp [your certificate pem file]
 /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo cp
[your certificate private key p
em file] /etc/httpd/conf/ssl.key/hubname-privkey.pem
sudo cp
[your certif
icate intermediate certi
ficate chain pem file] /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo chown root:root /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo chown root:root /etc/httpd/conf/ssl.key/hubname-privkey.pem
sudo chown root:root /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo chmod 0640 /etc/httpd/conf/ssl.crt/hubname-cert.pem
sudo chmod 0640 /etc/httpd/conf/ssl.key/hubname-privkey.pem
sudo chmod 0640 /etc/httpd/conf/ssl.crt/hubname-chain.pem
sudo hzcms reconfigure hubname
sudo service httpd restart
```

If you are using the HTML5 VNC Proxy Server, you must update your certificate settings there as well.

First Hub Account Creation

You should now be able to create a new user account for yourself via hub web registration at "/register". Depending on your email hosting you may not receive the confirmation email. After your new user account is created, you can confirm, approve, and promote the new account as the installing admin by logging into /administrator with the admin account credentials. The 'admin' user password is stored in /etc/hubzero.secrets and can only be read by root user. The admin account is only to be used once to approve and promote your individual user account and should be disabled in the CMS afterwards.

SOLR Search

Install

```
sudo yum install -y hubzero-solr
```

Mailgateway

Install the Hubzero Mailgateway

```
sudo yum install -y hubzero-mailgateway
```

Configure the Hubzero Mailgateway

```
sudo hzcms configure mailgateway --enable
```

Submit

Introduction

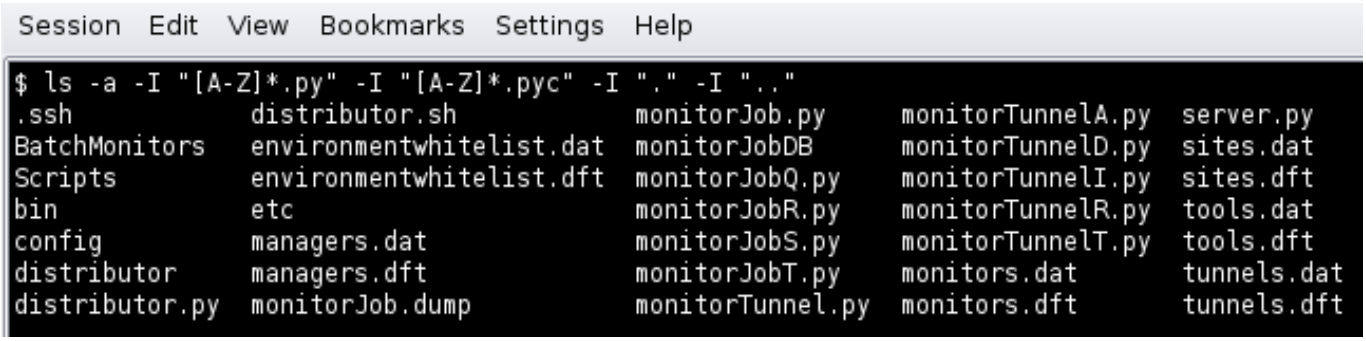
The submit command provides a means for HUB end users to execute applications on remote resources. The end user is not required to have knowledge of remote job submission mechanics. Jobs can be submitted to traditional queued batch systems including PBS and Condor or executed directly on remote resources.

Installation

```
sudo yum install -y hubzero-submit-pegasus
sudo yum install -y hubzero-submit-condor
sudo yum install -y hubzero-submit-common
sudo yum install -y hubzero-submit-server
sudo yum install -y hubzero-submit-distributor
sudo yum install -y hubzero-submit-monitors

sudo hzcms configure submit-server --enable
sudo service submit-server start
sudo chkconfig submit-server on
```

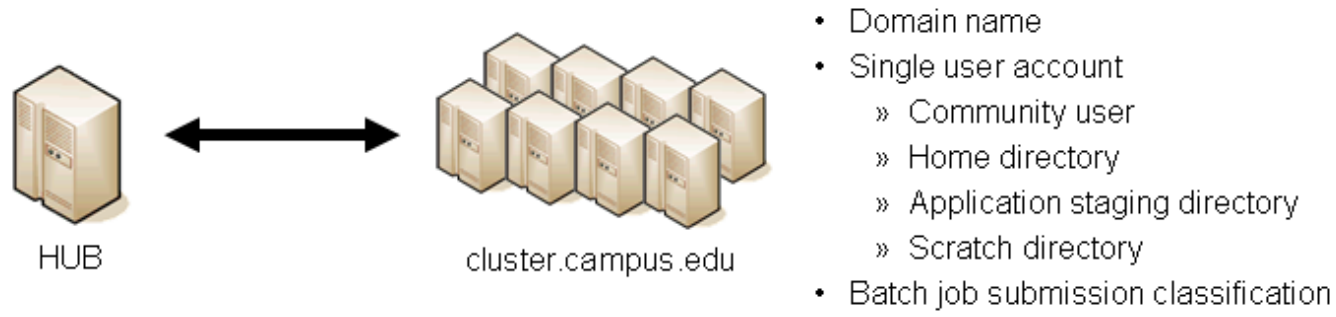
At completion of the yum install commands several files will be located in the directory /opt/submit. Excluding python files, the directory listing should like the following:



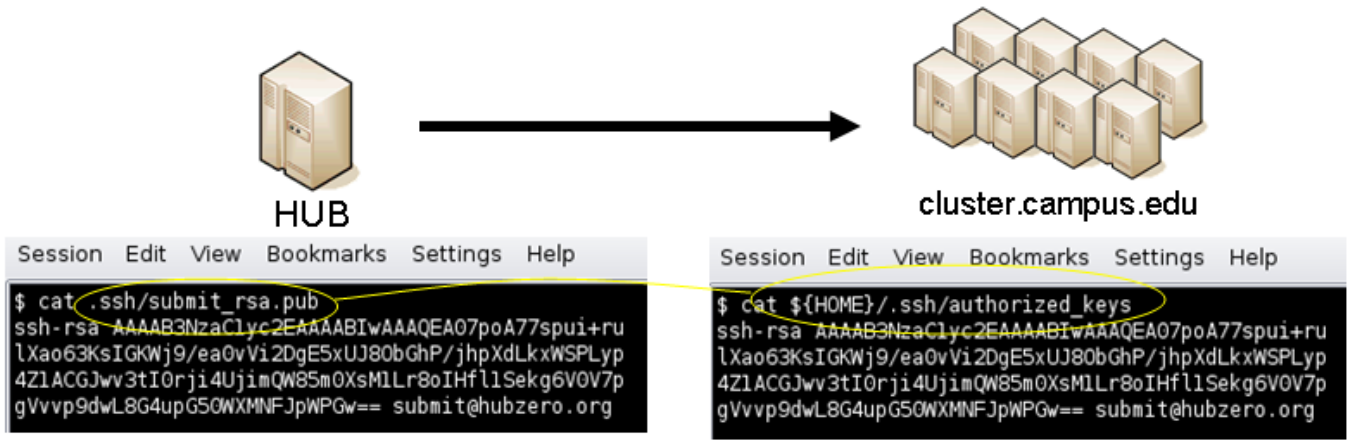
Configuration

Submit provides a mechanism to execute jobs on machines outside the HUB domain. To

accomplish this feat, some configuration is required on the HUB and some additional software must be installed and configured on hosts in remote domains. Before attempting to configure submit it is necessary to obtain access to the target remote domain(s). The premise is that a single account on the remote domain will serve as an execution launch point for all HUB end users. It is further assumes that access to this account can be made by direct ssh login or using an ssh tunnel (port forwarding).



Having attained account access to one or more remote domains, it is possible to proceed with submit configuration. To get started, the ssh public generated by the installation should be transferred to the remote domain host(s).



HUB Configuration

The behavior of submit is controlled through a set of configuration files. The configuration files contain descriptions of the various parameters required to connect to a remote domain, exchange files, and execute simulation codes. There are separate files for defining remote sites, staged tools, multiprocessor managers, file access controls, permissible environment variables, remote job monitors, and ssh tunneling. Most parameters have default values and it is not required that all parameters be explicitly defined in the configuration files. A simple example is given for each category of configuration file.

- Remote sites
- Staged tools
- Remote job monitors
- Multi-processor managers
- Permissible environment variables
- ssh tunnels

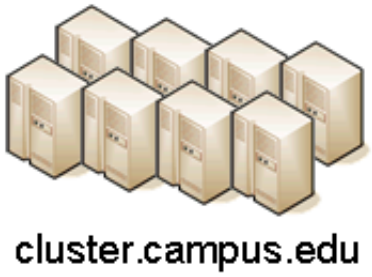
Sites

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 tools.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.
- remoteBatchAccount - some batch systems requirement that an account be provided in addition to user information.
- remoteBatchSystem - the possible batch submission systems include CONDOR, PBS, SGE, 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
- parallelEnvironment - sge parameter
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteApplicationRootDirectory - define directory where application executables are located.
- 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. (deprecated)
- executableClassificationsAllowed - classifications accepted by site. Classifications are set in appaccess.dat
- 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.
- undeclaredSiteSelectionWeight - used when no site is specified to choose between sites where selection weight > 0.
- minimumWallTime - minimum walltime allowed for site or queue. Time should be expressed in minutes.
- maximumWallTime - maximum walltime allowed for site or queue. Time should be expressed in minutes.
- minimumCores - minimum number of cores allowed for site or queue.
- maximumCores - maximum number of cores allowed for site or queue.
- pegasusTemplates - pertinent pegasus templates for site, rc, and transaction files.

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




```
Session  Edit  View  Bookmarks  Settings  Help
$ hostname -f
cluster.campus.edu
$ whoami
yourhub
$ echo ${HOME}
/home/yourhub
$ printenv | SCRATCH
CLUSTER_SCRATCH=/scratch/yourhub
```

```
[cluster]
venues = cluster.campus.edu
remotePpn = 8
remoteBatchSystem = PBS
remoteBatchQueue = standby
remoteUser = yourhub
remoteManager = mpich-intel64
venueMechanism = ssh
remoteScratchDirectory = /scratch/yourhub
siteMonitorDesignator = clusterPBS
```

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. Destination may exist in sites.dat or be a grid site defined by a ClassAd file.
- executablePath - path to executable at remote site. The path may be given as an absolute path on the remote site or a path relative to remoteApplicationRootDirectory defined in sites.dat.
- 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.
- environment - comma separated list of environment variables in the form e=v.
- 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 yourhub account on a remote site.


```
Session  Edit  View  Bookmarks  Settings  Help
$ cd ${HOME}
$ ls -R
.:
apps

./apps:
planets  stars

./apps/planets:
bin

./apps/planets/bin:
earth.x  jupiter.x  mars.x  mercury.x  neptune.x  saturn.x  uranus.x  venus.x

./apps/stars:
bin

./apps/stars/bin:
antares.x  betelgeuse.x  polaris.x  sun.x
```

```
[earth]
destinations = cluster
executablePath = ${HOME}/apps/planets/bin/earth.x
remoteManager = mpich-intel

[sun]
destinations = cluster
executablePath = ${HOME}/apps/stars/bin/sun.x
remoteManager = mpich-intel
```

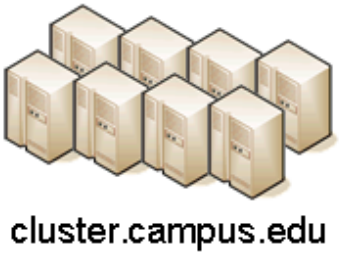
Monitors

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.
- venueMechanism - monitoring job launch process. The default is ssh.
- tunnelDesignator - name of tunnel defined in tunnels.dat.

- remoteUser - login user at remote site.
- remoteBinDirectory - define directory where shell scripts related to the site should be kept.
- remoteMonitorCommand - command to launch monitor daemon process.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

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




```
Session  Edit  View  Bookmarks  Settings  Help
$ qstat -u yourhub

cluster.campus.edu:
                                                                  Req'd   Elap
Job ID          Username Queue   Jobname SessID NDS TSK Time  S Time
--------------- -------- ------- ------- ------ --- --- ----- - -----
9823388.steele- yourhub  standby earth    3508   2  16  04:00 R 02:15
9824065.steele- yourhub  standby sun      9975   1   1  04:00 R 01:26
```

```
[clusterPBS]
venue = cluster.campus.edu
remoteUser = yourhub
remoteMonitorCommand = ${HOME}/SubmitMonitor/monitorPBS.py
```

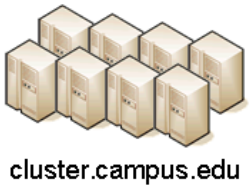
Multi-processor managers

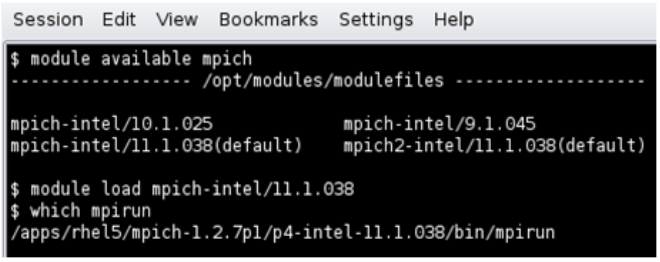
Multiprocessor managers are defined in the file managers.dat. Each manager is defined by a stanza indicating the set of commands used to execute a multiprocessor simulation run. Defined keywords are

- [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
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example stanza is presented for a typical MPI instance. The given command should be suitable for /bin/sh execution.






```
[mpich-intel]
preManagerCommands = . ${MODULESHOME}/init/sh, module load mpich-
intel/11.1.038
managerCommand = mpirun -machinefile ${PBS_NODEFILE} -np NPROCESSORS
```

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

File access controls

Application or file level access control is described by entries listed in the file appaccess.dat. The ability to transfer files from the HUB to remote sites is granted on a group basis as defined by white and black lists. Each list is given a designated priority and classification. In cases where a file appears on multiple lists, the highest priority takes precedence. Simple wildcard operators are allowed the in the filename declaration allowing for easy listing of entire directories. Each site lists acceptable classification(s) in sites.dat. Defined keywords are

- [group] - group name.
- whitelist - comma separated list of paths. Wildcards allowed.

- blacklist - comma separated list of paths. Wildcards allowed.
- priority - higher priority wins
- classification - apps or user. user class are treated are arbitrary executables.
- state - possible values are enabled or disabled. If not explicitly set the default value is enabled.

An example file giving permissions reminiscent of those defined in earlier submit releases is presented here

```
[public]
whitelist = /apps/.*
priority = 0
classification = apps

[submit]
whitelist = ${HOME}/.*
priority = 0
classification = home
```

The group public is intended to include all users. Your system may use a different group such as users for this purpose. The definitions shown here allow all users access to files in /apps where applications are published. Additionally members of the submit group are allowed to send files from their $HOME directory.

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 variables used by openmp and mpich is presenter here.

```
# environment variables listed here can be specified from the command
line with -e/--env option. Attempts to specify other environment varia
bles will be ignored and the values will not be passed to the remote s
ite.
```

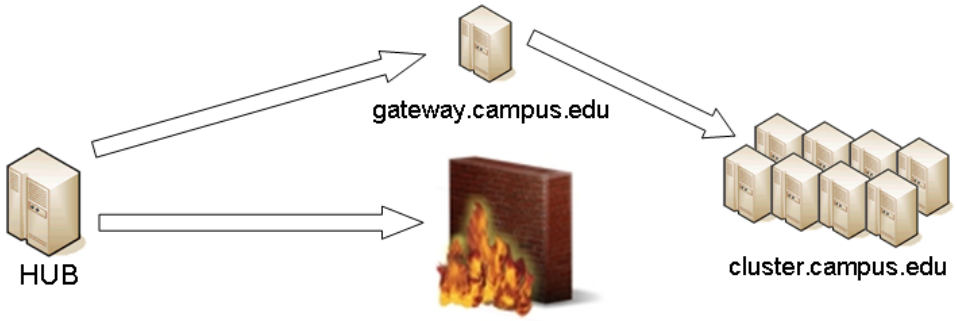
```
OMP_NUM_THREADS
MPICH_HOME
```

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.campus.edu
venuePort = 22
gatewayHost = gateway.campus.edu
gatewayUser = yourhub
localPortOffset = 1
```

Initialization Scripts and Log Files

The submit server and job monitoring server must be started as daemon processes running on the the submit host. If ssh tunneling is going to be used an addition server must be started as a daemon process. Each daemon process writes to a centralized log file facilitating error recording and debugging.

Initialize daemon scripts

Scripts for starting the server daemons are provided and installed in /etc/init.d. The default settings for when to start and terminate the scripts are adequate.

Log files

Submit processes log information to files located in the /var/log/submit directory tree. The exact location varies depending on the vintage of the installation. Each process has its own log file. The three most important log files are submit-server.log, distributor.log, and monitorJob.log.

submit.log

The submit-server.log file tracks when the submit server is started and stopped. Each connection from the submit client is logged with the command line and client ip address reported. All log entries are timestamped and reported by submit-server process ID (PID) or submit ID (ID:) once one has been assigned. Entries from all jobs are simultaneously reported and intermingled. The submit ID serves as a good search key when tracing problems. Examples of startup, job execution, and termination are given here. The job exit status and time metrics are also recorded in the MyQSL database JobLog table.

```
[Sun Aug 26 17:28:24 2012] 0: ########################################
###########
[Sun Aug 26 17:28:24 2012] 0: Backgrounding process.
[Sun Aug 26 17:28:24 2012] 0: Listening: protocol='tcp', host='', port
=830
```

```
[Sun Sep 23 12:33:28 2012] (1154) ==================================
===============
```

```
[Sun Sep 23 12:33:28 2012] (1154) Connection to tcp://:830 from ('192.
168.224.14', 38770)
[Sun Sep 23 12:33:28 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:28 2012] 0: Cumulative job load is 0.84.  (Max: 510.
00)
[Sun Sep 23 12:33:28 2012] 1670: Args are:['/usr/bin/submit', '--local
', '-p', '@@iv=-3:1.5:3', '/home/hubzero/user/hillclimb/bin/hillclimb1
.py', '--seed', '10', '--initialvalue', '@@iv', '--lowerbound', '-3',
'--upperbound', '3', '--function', 'func2', '--solutionslog', 'solutio
ns.dat', '--bestresultlog', 'best.dat']
[Sun Sep 23 12:33:28 2012] 1670: Server stopping.
[Sun Sep 23 12:33:28 2012] 1670: Server(JobExecuter) exiting(2).
[Sun Sep 23 12:33:38 2012] (1154) ================================
================
[Sun Sep 23 12:33:38 2012] (1154) Connection to tcp://:830 from ('192.
168.224.14', 38774)
[Sun Sep 23 12:33:38 2012] 0: Server will time out in 60 seconds.
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=1:local status=0 cp
u=0.030000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=2:local status=0 cp
u=0.040000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=3:local status=0 cp
u=7.050000 real=7.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=4:local status=0 cp
u=0.080000 real=0.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue=5:local status=0 cp
u=0.020000 real=1.000000 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Job Status: venue= status=0 cpu=10.42
8651 real=9.561828 wait=0.000000
[Sun Sep 23 12:33:38 2012] 1670: Server(JobExecuter) exiting(0).
[Sun Sep 23 12:48:44 2012] (1154) ================================
================
```

```
[Sun Aug 26 17:28:17 2012] 0: Server(10836) was terminated by a signal
 2.
[Sun Aug 26 17:28:17 2012] 0: Server(Listener) exiting(130).
```

distributor.log

The distributor.log file tracks each job as it progresses from start to finish. Details of remote site

assignment, queue status, exit status, and command execution are all reported. All entries are timestamped and reported by submit ID. The submit ID serves as the key to join data reported in submit-server.log. An example for submit ID 1659 is listed here. Again the data for all jobs are intermingled.

```
[Sun Sep 23 00:04:21 2012] 0: quotaCommand = quota -w | tail -n 1
[Sun Sep 23 00:04:21 2012] 1659: command = tar vchf 00001659_01_input.
tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L .__lo
cal_jobid.00001659_01 sayhiinquire.dax
[Sun Sep 23 00:04:21 2012] 1659: remoteCommand pegasus-
plan --dax ./sayhiinquire.dax
[Sun Sep 23 00:04:21 2012] 1659: workingDirectory /home/hubzero/user/d
ata/sessions/3984L
[Sun Sep 23 00:04:21 2012] 1659: command = tar vrhf 00001659_01_input.
tar --exclude='*.svn*' -C /home/hubzero/user/data/sessions/3984L/00001
659/01 00001659_01.sh
[Sun Sep 23 00:04:21 2012] 1659: command = nice -n 19 gzip 00001659_01
_input.tar
[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/receiveinpu
t.sh /home/hubzero/user/data/sessions/3984L/00001659/01 /home/hubzero/
user/data/sessions/3984L/00001659/01/.__timestamp_transferred.00001659
_01
[Sun Sep 23 00:04:21 2012] 1659: command = /opt/submit/bin/submitbatch
job.sh /home/hubzero/user/data/sessions/3984L/00001659/01 ./00001659_0
1.pegasus
[Sun Sep 23 00:04:23 2012] 1659: remoteJobId = 2012.09.23 00:04:22.996
 EDT:   Submitting job(s).
2012.09.23 00:04:23.002 EDT:   1 job(s) submitted to cluster 946.
2012.09.23 00:04:23.007 EDT:
2012.09.23 00:04:23.012 EDT:   ----------------------------------------
--------------------------------
2012.09.23 00:04:23.017 EDT:   File for submitting this DAG to Condor
          : sayhi_inquire-0.dag.condor.sub
2012.09.23 00:04:23.023 EDT:   Log of DAGMan debugging messages
          : sayhi_inquire-0.dag.dagman.out
2012.09.23 00:04:23.028 EDT:   Log of Condor library output
          : sayhi_inquire-0.dag.lib.out
2012.09.23 00:04:23.033 EDT:   Log of Condor library error messages
          : sayhi_inquire-0.dag.lib.err
2012.09.23 00:04:23.038 EDT:   Log of the life of condor_dagman itself
          : sayhi_inquire-0.dag.dagman.log
2012.09.23 00:04:23.044 EDT:
2012.09.23 00:04:23.049 EDT:   ----------------------------------------
--------------------------------
2012.09.23 00:04:23.054 EDT:
2012.09.23 00:04:23.059 EDT:   Your Workflow has been started and runs
```

```
 in base directory given below
2012.09.23 00:04:23.064 EDT:
2012.09.23 00:04:23.070 EDT:   cd /home/hubzero/user/data/sessions/398
4L/00001659/01/work/pegasus
2012.09.23 00:04:23.075 EDT:
2012.09.23 00:04:23.080 EDT:   *** To monitor the workflow you can run
 ***
2012.09.23 00:04:23.085 EDT:
2012.09.23 00:04:23.090 EDT:   pegasus-status -l /home/hubzero/user/da
ta/sessions/3984L/00001659/01/work/pegasus
2012.09.23 00:04:23.096 EDT:
2012.09.23 00:04:23.101 EDT:   *** To remove your workflow run ***
2012.09.23 00:04:23.106 EDT:   pegasus-remove /home/hubzero/user/data/
sessions/3984L/00001659/01/work/pegasus
2012.09.23 00:04:23.111 EDT:
2012.09.23 00:04:23.117 EDT:   Time taken to execute is 0.993 seconds
[Sun Sep 23 00:04:23 2012] 1659: confirmation: S(1):N Job
[Sun Sep 23 00:04:23 2012] 1659: status:Job N WF-DiaGrid
[Sun Sep 23 00:04:38 2012] 1659: status:DAG R WF-DiaGrid
[Sun Sep 23 00:10:42 2012] 0: quotaCommand = quota -w | tail -n 1
[Sun Sep 23 00:10:42 2012] 1660: command = tar vchf 00001660_01_input.
tar --exclude='*.svn*' -C /home/hubzero/clarksm .__local_jobid.0000166
0_01 noerror.sh
[Sun Sep 23 00:10:42 2012] 1660: remoteCommand ./noerror.sh
[Sun Sep 23 00:10:42 2012] 1660: workingDirectory /home/hubzero/clarks
m
[Sun Sep 23 00:10:42 2012] 1660: command = tar vrhf 00001660_01_input.
tar --exclude='*.svn*' -C /home/hubzero/clarksm/00001660/01 00001660_0
1.sh
[Sun Sep 23 00:10:42 2012] 1660: command = nice -n 19 gzip 00001660_01
_input.tar
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/receiveinpu
t.sh /home/hubzero/clarksm/00001660/01 /home/hubzero/clarksm/00001660/
01/.__timestamp_transferred.00001660_01
[Sun Sep 23 00:10:42 2012] 1660: command = /opt/submit/bin/submitbatch
job.sh /home/hubzero/clarksm/00001660/01 ./00001660_01.condor
[Sun Sep 23 00:10:42 2012] 1660: remoteJobId = Submitting job(s).
1 job(s) submitted to cluster 953.
[Sun Sep 23 00:10:42 2012] 1660: confirmation: S(1):N Job
[Sun Sep 23 00:10:42 2012] 1660: status:Job N DiaGrid
[Sun Sep 23 00:11:47 2012] 1660: status:Simulation I DiaGrid
[Sun Sep 23 00:12:07 2012] 1660: Received SIGINT!
[Sun Sep 23 00:12:07 2012] 1660: waitForBatchJobs: nCompleteRemoteJobI
ndexes = 0, nIncompleteJobs = 1, abortGlobal = True
[Sun Sep 23 00:12:07 2012] 1660: command = /opt/submit/bin/killbatchjo
b.sh 953.0 CONDOR
```

```
[Sun Sep 23 00:12:07 2012] 1660: Job 953.0 marked for removal

[Sun Sep 23 00:12:07 2012] 1660: status:Simulation I DiaGrid
[Sun Sep 23 00:12:52 2012] 1660: status:Simulation D DiaGrid
[Sun Sep 23 00:12:52 2012] 1660: venue=1:localCONDOR:953.0:DiaGrid sta
tus=258 cputime=0.000000 realtime=0.000000 waittime=0.000000 ncpus=1
[Sun Sep 23 00:28:14 2012] 1659: status:DAG D WF-DiaGrid
[Sun Sep 23 00:28:14 2012] 1659: waitForBatchJobs: nCompleteRemoteJobI
ndexes = 1, nIncompleteJobs = 0, abortGlobal = False
[Sun Sep 23 00:28:14 2012] 1659: command = /opt/submit/bin/cleanupjob.
sh /home/hubzero/user/data/sessions/3984L/00001659/01
[Sun Sep 23 00:28:15 2012] 1659:
*******************************************SUMMARY*****************
*****************************

Job instance statistics           : /home/hubzero/user/data/sessions/3
984L/00001659/01/work/pegasus/statistics/jobs.txt

****************************************************************************
*****************************

[Sun Sep 23 00:28:15 2012] 1659: venue=1:localPEGASUS:946.0:WF-DiaGrid
 status=0 cputime=1.430000 realtime=2.000000 waittime=0.000000 ncpus=1
[Sun Sep 23 00:28:15 2012] 1659: venue=2:PEGASUS:952.0:DiaGrid status=
0 cputime=0.003000 realtime=0.000000 waittime=681.000000 ncpus=1 event
=/sayhi_inquire-sayhi-1.0
[Sun Sep 23 00:28:15 2012] 1659: venue=3:PEGASUS:954.0:DiaGrid status=
0 cputime=0.003000 realtime=0.000000 waittime=631.000000 ncpus=1 event
=/sayhi_inquire-inquire-1.0
```

monitorJob.log

The monitorJob.log file tracks the invocation and termination of each remotely executed job monitor. The remote job monitors are started on demand when job are submitted to remote sites. The remote job monitors terminate when all jobs complete at a remote site and no new activity has been initiated for a specified amount of time - typically thirty minutes. A typical report should look like:

```
[Sun Aug 26 17:29:16 2012] (1485) ***********************************
[Sun Aug 26 17:29:16 2012] (1485) * distributor job monitor started *
[Sun Aug 26 17:29:16 2012] (1485) ***********************************
[Sun Aug 26 17:29:16 2012] (1485) loading active jobs
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from DB file
```

```
[Sun Aug 26 17:29:16 2012] (1485) 15 jobs loaded from dump file
[Sun Aug 26 17:29:16 2012] (1485) 4 jobs purged
[Sun Aug 26 17:29:16 2012] (1485) 11 monitored jobs
[Sun Aug 26 18:02:04 2012] (24250) Launching wf-diagrid
[Sun Aug 26 18:02:04 2012] (1485) 12 monitored jobs
[Sun Aug 26 18:02:15 2012] (1485) Update message received from wf-
diagrid
[Sun Aug 26 18:03:15 2012] (1485) Update message received from wf-
diagrid
[Sun Aug 26 18:06:43 2012] (1485) 13 monitored jobs
...
[Thu Sep 17 17:32:51 2011] (21095) Received SIGTERM!
[Thu Sep 17 17:32:51 2011] (21095) Send TERM to child ssh process
[Thu Sep 17 17:32:51 2011] (21095) distributor site monitor stopped
[Thu Sep 17 17:32:51 2011] (17348) Send TERM to child site steele proc
ess
[Thu Sep 17 17:32:51 2011] (17348) **********************************
[Thu Sep 17 17:32:51 2011] (17348) * distributor job monitor stopped *
[Thu Sep 17 17:32:51 2011] (17348) **********************************
```

It is imperative that the job monitor be running in order for notification of job progress to occur. If users report that their job appears to hang check to make sure the job monitor is running. If necessary take corrective action and restart the daemon.

monitorTunnel.log

The monitorTunnel.log file tracks invocation and termination of each ssh tunnel connection. If users report problems with job submission to sites accessed via an ssh tunnel this log file should be checked for indication of any possible problems.

Remote Domain 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:

The directory defined by MONITORLOGLOCATION needs to be created before the daemon is started. Sample daemon scripts used for PBS, LSF, SGE, Condor, Load Leveler, and Slurm batch systems are included in directory BatchMonitors.

File transfer and batch job scripts

The simple scripts are used to manage file transfer and batch job launching and termination. The location of the scripts is entered in sites.dat.

Examples scripts suitable for use with PBS, LSF, Condor, Load Leveler, and Slurm are included in directory Scripts. After modifications are made to monitors.dat the central job monitor must be notified. This can be accomplished by stopping and starting the submon daemon or a HUP signal can be sent to the monitorJob.py process.

File transfer - input files

Receive compressed tar file containing input files required for the job on stdin. The file

transferredTimestampFile is used to determine what newly created or modified files should be returned to the HUB.

```
receiveinput.sh jobWorkingDirectory jobScratchDirectory  transferredTi
mestampFile
```

Batch job script - submission

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 identifier. The log file should be placed at a location agreed upon by the remote site and HUB.

```
submitbatchjob.sh jobWorkingDirectory jobScratchDirectory jobDescripti
onFile
```

The jobId is returned on stdout if job submission is successful. For an unsuccessful job submission the returned jobId should be -1.

File transfer - output files

Return compressed tar file containing job output files on stdout.

```
transmitresults.sh jobWorkingDirectory
```

File transfer - cleanup

Remove job specific directory and any other dangling files

```
cleanupjob.sh jobWorkingDirectory jobScratchDirectory jobClass
```

Batch job script - termination

Terminate given remote batch job. Command line arguments specify job identifier and batch system type.

```
killbatchjob.sh jobId jobClass
```

Batch job script - post process

For some jobClassses it is appropriate to preform standard post processing actions. An example of such a jobClass is Pegasus.

```
postprocessjob.sh jobWorkingDirectory jobScratchDirectory jobClass
```

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 executableClassificationsAllowed keyword 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 be granted permission to send files from their $HOME directory. Discrete permission can be granted on a file by file basis in appaccess.dat.

Virtualbox CMS Development Appliance

The VM Appliance is based on
http://mirror.umd.edu/centos/7.9.2009/isos/x86_64/CentOS-7-x86_64-Minimal-2009.iso

Download Appliance File

https://help.hubzero.org/app/site/vmimages/Hubzero_CMS_Development_Environment.ova

Import into Virtualbox

https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html

VirtualBox Configuration

The VM Appliance expects to be configured as a server. Do NOT configure shared folders or mount directories to a local workstation filesystem, it's likely permission and write access will be problematic.

Networking in the appliance is configured in Bridged network mode. This mode will obtain a DHCP IP from your local DHCP server and will be on the same network as your laptop for easy access.

Note: You may want to consider using NAT when connecting to a public network where you do not want your development hub exposed to other users of the network.

If you prefer a NAT network configuration:

https://www.howtogeek.com/122641/how-to-forward-ports-to-a-virtual-machine-and-use-it-as-a-server/

Chosen Hub Configuration

The following configuration options have been chosen during the VM appliance creation process and may be different on a production hubs.

hub name is 'hub'

The webroot is /var/www/hub

Finding the Hub Appliance IP address

From the VirtualBox terminal as the root user:

```
ip addr show
```

** The IP address may change if the VM is restarted

How to access the hub VM

It may be useful to add a short name for the VM to your hosts file (/etc/hosts for Linux and Mac or C:\Windows\System32\drivers\etc\hosts on Windows). If your IP address changes you will need to update it, but if you have sftp, DBeaver, and bookmarks set to use the name you will only have to update one location. If you add a name to hosts, use that name below where it says <ip_address_from_above>

SSH/SFTP

ssh root@<ip_address_from_above>

root password: hubzero2021

Web port 443

https://<ip_address_from_above>

** Accept the invalid / Self Signed Cert message. If you don't see that message, please use Firefox or add the default snake-oil certificate to your browser.

One-time use CMS admin credentials

Once your new hub VM is up and running create a new account via hub registration at /register. To email confirm, approve and promote the new account to Super User/Admin, you may use the included admin account once. The username is "admin" and you can find the password in the VM at /etc/hubzero.secrets "CMS-ADMIN". You must be root on the VM to access this file. ** Do not use this account after promoting your individual account to a CMS Super User - it is no longer needed.

CMS Cron Job Configuration

Add the <ip_address_from_above> to the CMS Cron component IP Whitelist configuration.

The full list should be "127.0.0.1,<ip_address_from_above>"

Updating the CMS code

As root:

```
webroot="hub" && \
cd /var/www/${webroot} && \
git stash && git pull && \
cd /var/www/${webroot}/core && ./bin/composer install && \
cd /var/www/${webroot} && php muse migration -i -f && \
chown -R apache:apache /var/www/${webroot} && \
hzcms update
```

Reference: https://help.hubzero.org/documentation/22/webdevs/index/devenvironment

Debugging the Hub

As root:

```
yum install php56-php-pecl-xdebug
```

Add the following to /opt/remi/php56/etc/php.ini:

```
xdebug.remote_autostart = 1
xdebug.remote_connect_back = 1
xdebug.remote_enable = 1
xdebug.remote_host = <ip_address_from_above>
xdebug.remote_port = 9003
xdebug.idekey=IDEKEY
xdebug.remote_log="/var/log/hubzero/xdebug.log"
```

CLI

Download an appropriate version of [dbgpClient](https://xdebug.org/download#dbgpClient). Possibly you will need to look into the [archive](https://xdebug.org/download/historical). Once downloaded you should be able to run:

```
dbgpClient -p 9003
```

and debug on the CLI as needed.

IDE (VS Code example)

In VS Code, as an example, install PHP Debug and click "Run", "Add Configurations". In the resulting launch.json you will need to minimally edit the "Listen for Xdebug" configuration to include the following:

```
{
  "name": "Listen for Xdebug",
  "type": "php",
  "request": "launch",
  "port": 9003,
  "pathMappings": {
    "/var/www/hub/": "${workspaceFolder}/",
  },
},
```