Managing unix commands with Grouper permissions example

Summary

In this example from University of Pennsylvania, Grouper manages permissions for actions used in Linux commands. The Grouper permissions are exported to a text file and kept in sync (batch and real time) via the grouperClient on the linux server.

Using

For someone to restart a tomcat, call a script that uses sudo to call the command to restart a tomcat:

```
[mchyzer@lukes clusterLinux]$ clusterTomcat tomcat_directory restart
```

To onboard someone, add a user to the sudoers file. e.g. allow user mchyzer to restart tomcats (that are allowed in Grouper).

```
mchyzer ALL=(appadmin) /opt/appserv/binMgmt/clusterTomcatHelper.sh *
```

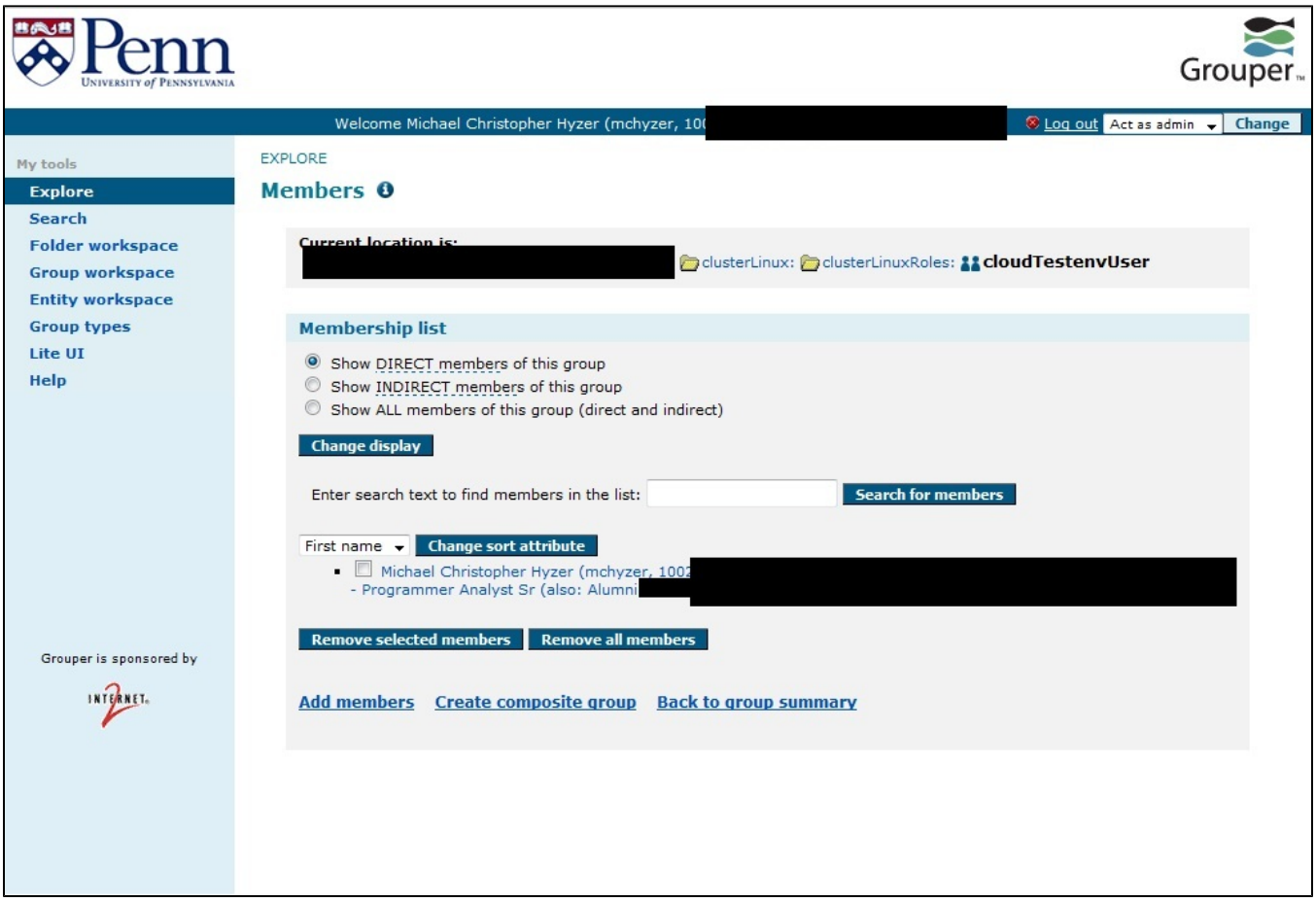
Maybe this should be a wildcard so anyone can do this since permissions are in Grouper and you dont have to do this for each user... either way

```
ALL ALL=(appadmin) /opt/appserv/binMgmt/clusterTomcatHelper.sh *
```

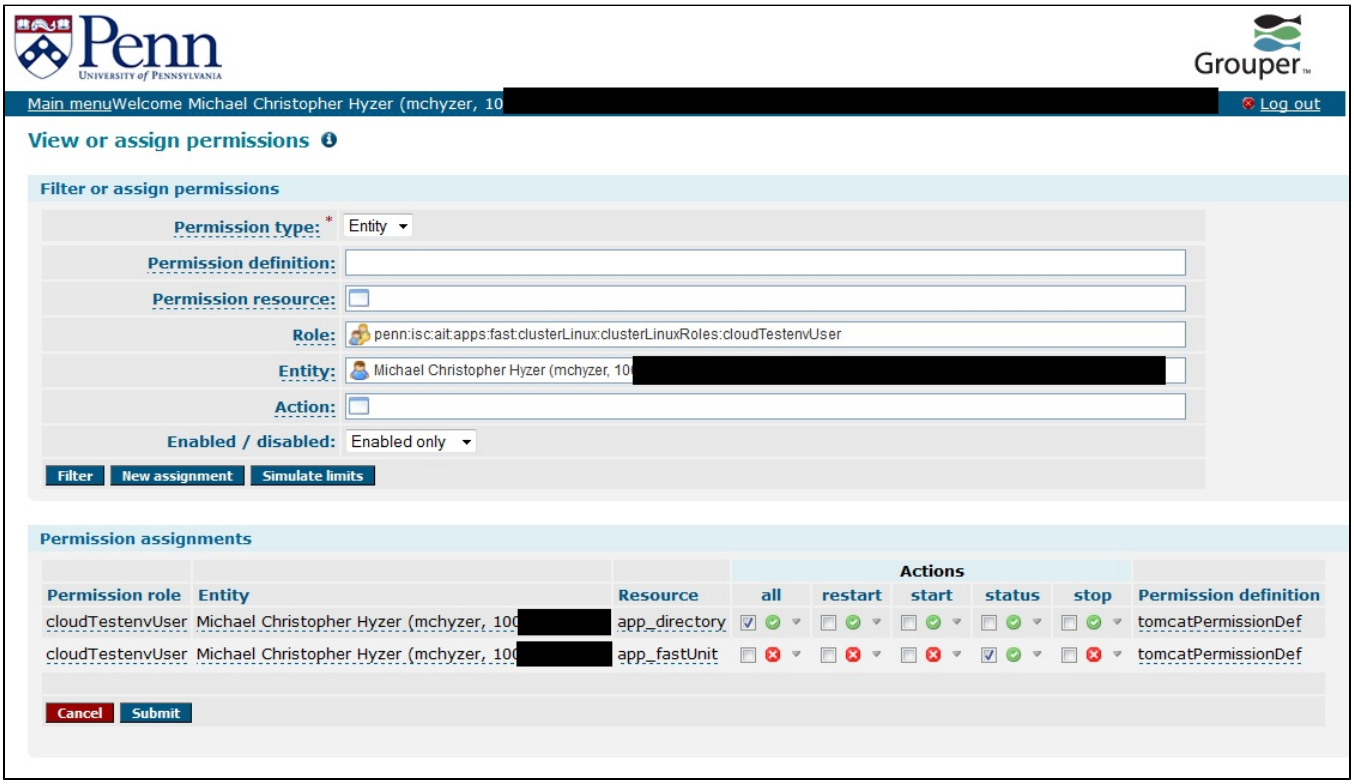
At this point the user is not allowed to do anything, since they have no permissions in Grouper, and clusterTomcatHelper.sh (below) uses Grouper permissions to decide if the user can perform the action.

Go to the Grouper permissions screen, and add the user to the role for this server

---



Assign permissions to the user. Note, the resource is which tomcat the user is allowed to perform actions on



Wait max two minutes, and the permissions file on the server will be updated, e.g.

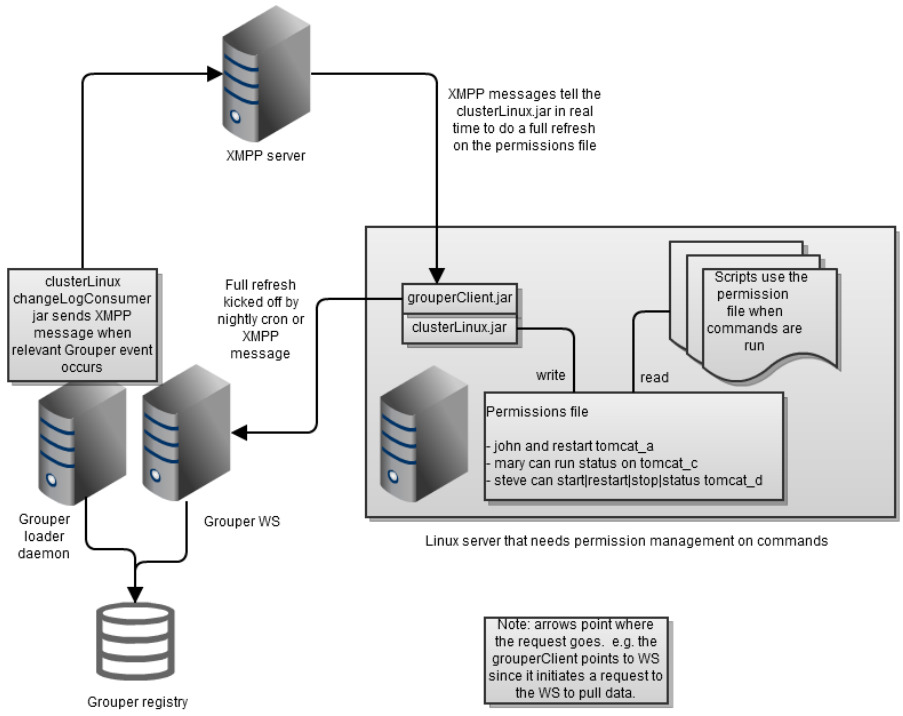
```
[appadmin@server clusterLinux]$ more  /opt/appserv/binMgmt/clusterLinux/clusterLinuxRules.sh
# File automatically generated from PennGroups...

clusterLinux__mchyzer__app_directory__all=true
clusterLinux__mchyzer__app_directory__restart=true
clusterLinux__mchyzer__app_directory__start=true
clusterLinux__mchyzer__app_directory__status=true
clusterLinux__mchyzer__app_directory__stop=true
clusterLinux__mchyzer__app_fastUnit__status=true
[appadmin@lukes clusterLinux]$
```

Note the file is concerned with the user, resource (which tomcat), and action (start|stop|restart|status). If there are other rollup actions (e.g. "all") then it will be ignored by the calling script since you cant perform "all" on the tomcat service.

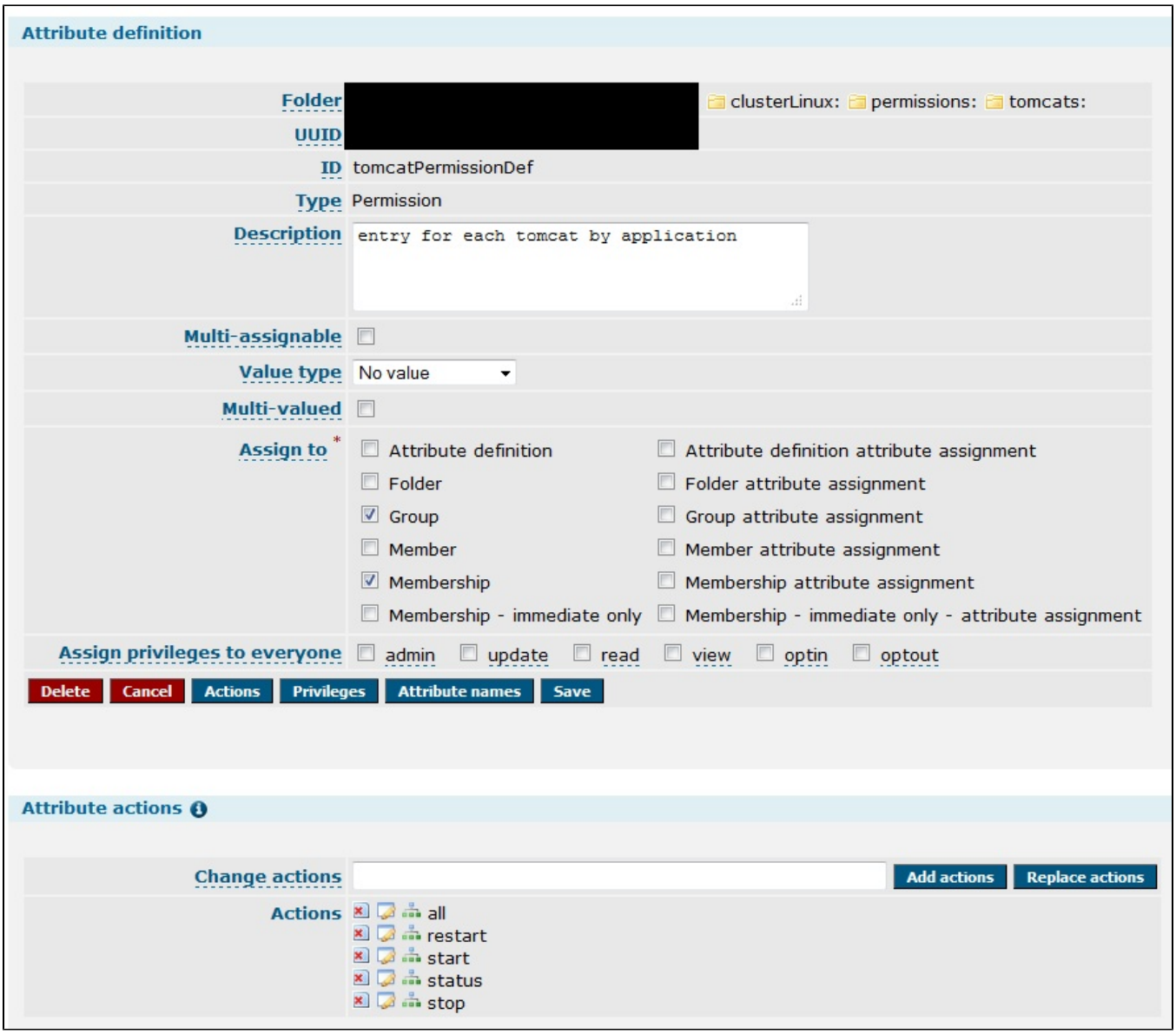
Architecture

Diagram of permissions in use, cron updates, and XMPP real time updates

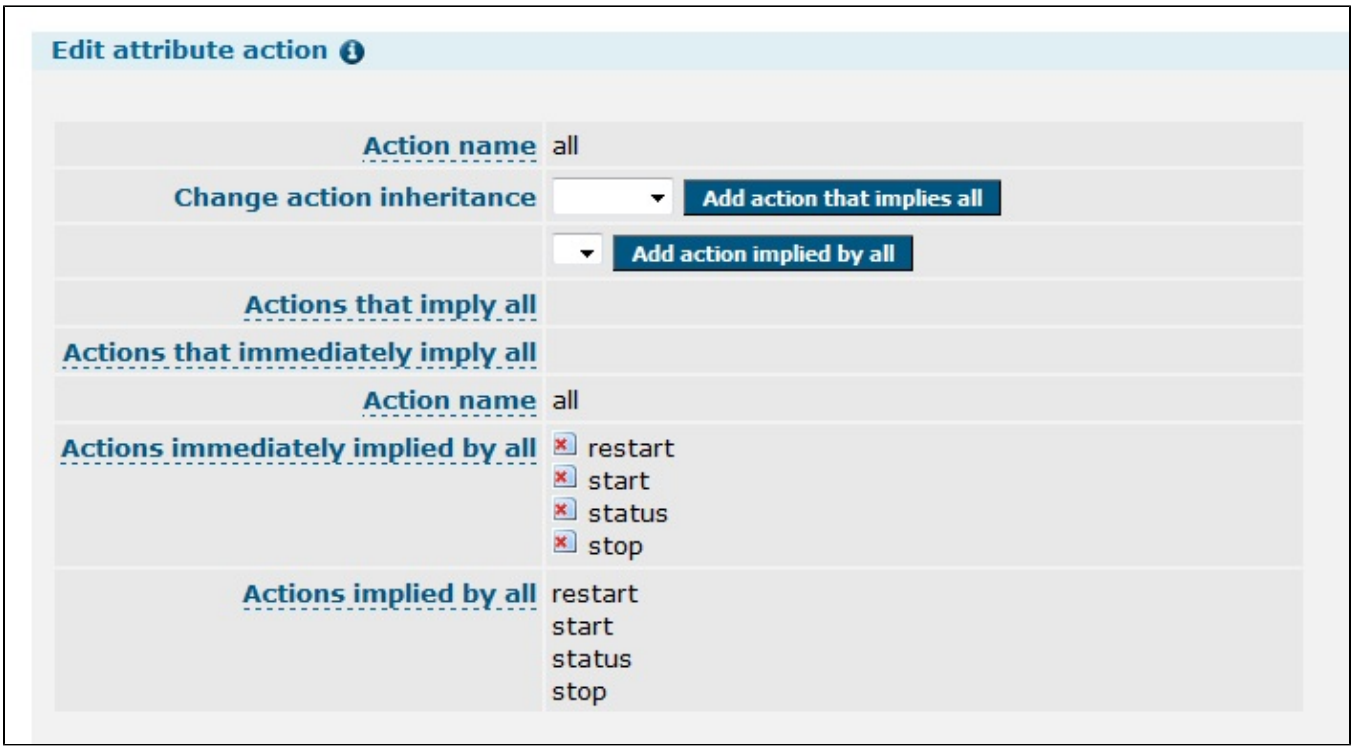


One-time setup on the server where the permissions are needed

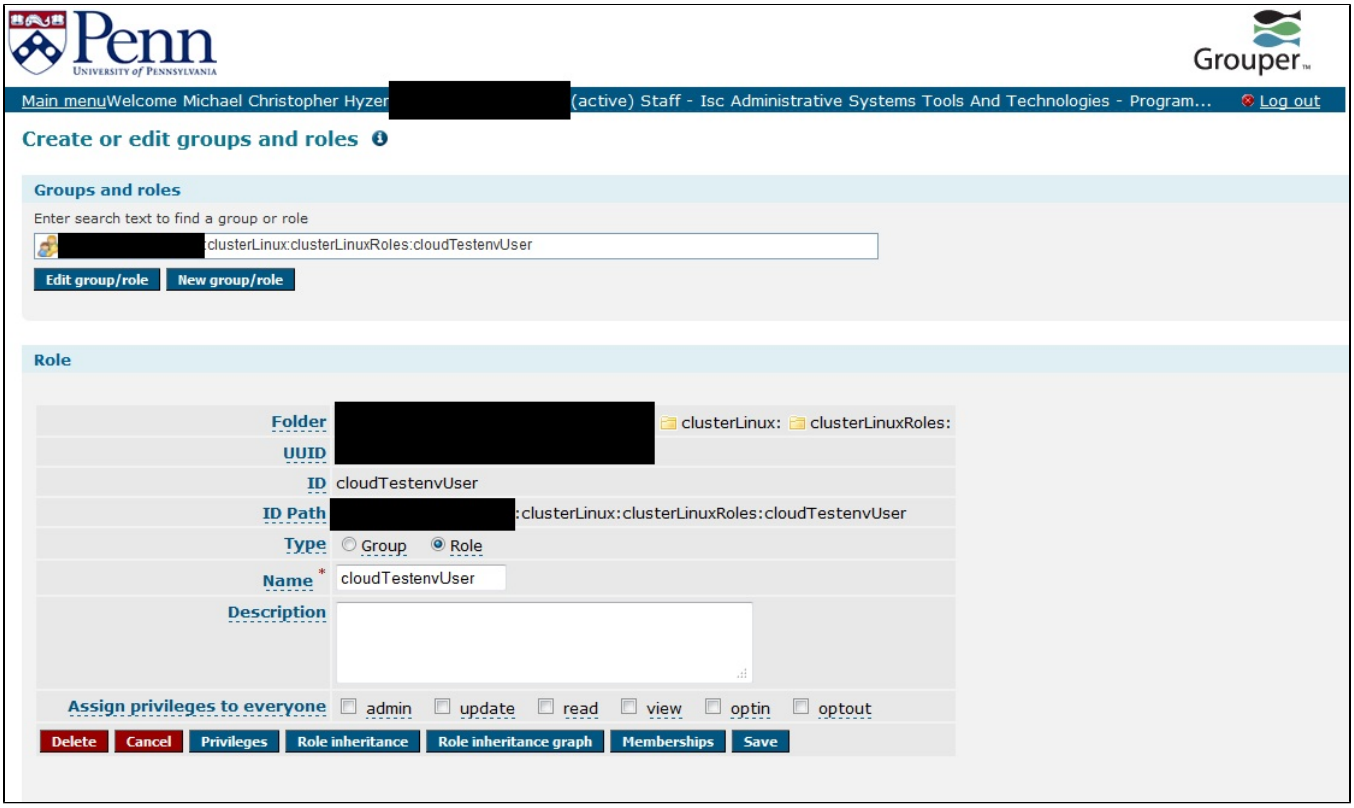
Create a permission definition:



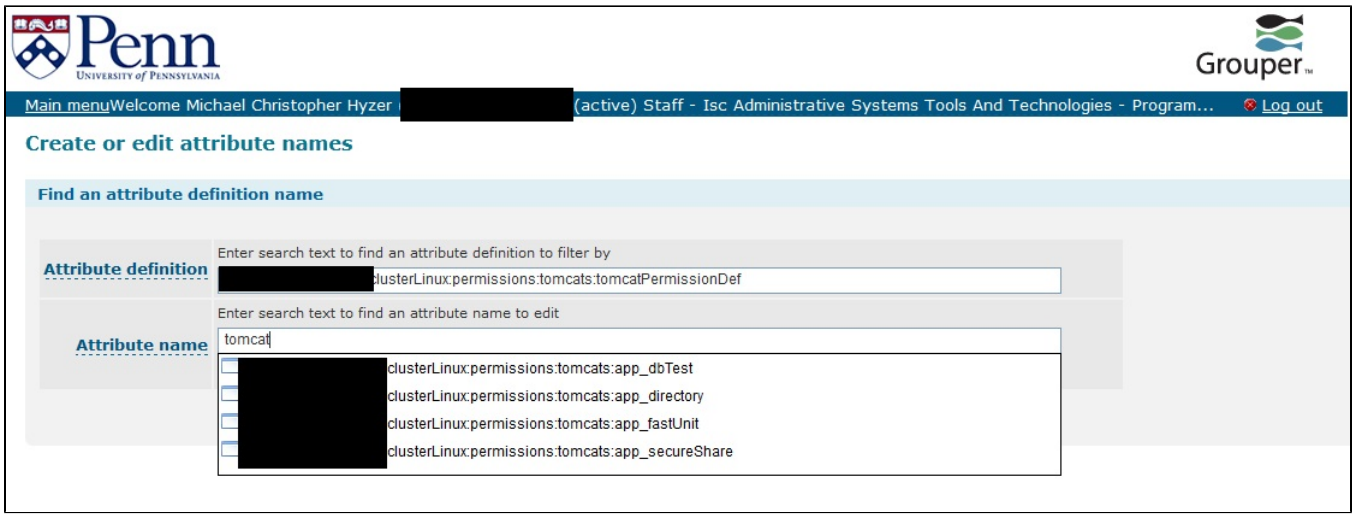
Add action inheritance so "all" implies the others



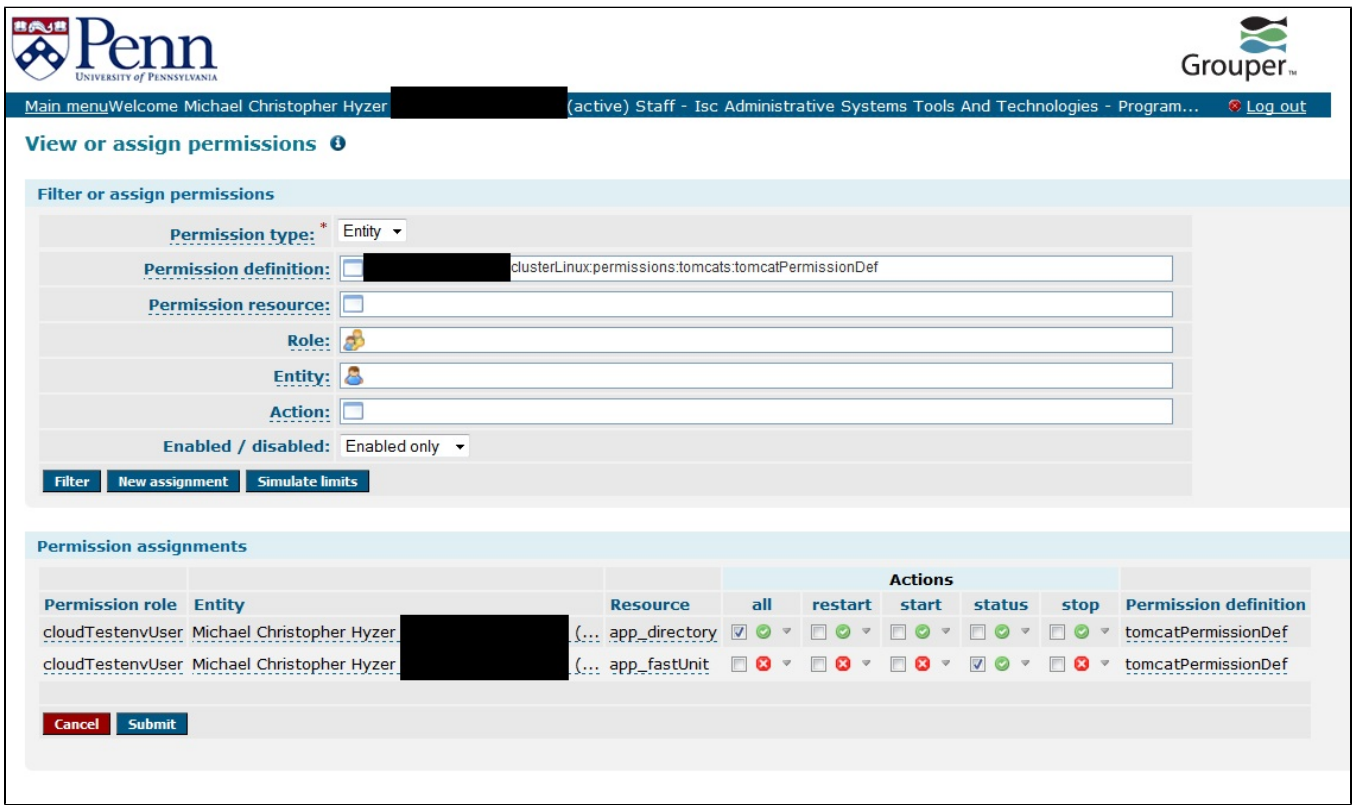
Create a role for an environment



Add permission names for each application, in this case prefix the real app name with "app_"



Assign permissions to a user in the role



Permissions client real time and daemon process

Here is the source file:

```
/**
 * @author mchyzer
 * $Id: ClusterLinuxLogic.java,v 1.4 2012/05/21 17:01:24 mchyzer Exp $
 */
package edu.upenn.isc.clusterLinux;

import java.io.File;
import java.sql.Timestamp;
```

```
import java.util.Date;
import java.util.HashSet;
import java.util.Set;
import java.util.Timer;
import java.util.TimerTask;
import java.util.TreeSet;

import org.jivesoftware.smack.packet.Message;
import org.quartz.Job;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import org.quartz.StatefulJob;

import edu.internet2.middleware.grouperClient.api.GcGetPermissionAssignments;
import edu.internet2.middleware.grouperClient.api.GcGetSubjects;
import edu.internet2.middleware.grouperClient.collections.MultiKey;
import edu.internet2.middleware.grouperClient.util.GrouperClientUtils;
import edu.internet2.middleware.grouperClient.ws.beans.WsGetPermissionAssignmentsResults;
import edu.internet2.middleware.grouperClient.ws.beans.WsGetSubjectsResults;
import edu.internet2.middleware.grouperClient.ws.beans.WsPermissionAssign;
import edu.internet2.middleware.grouperClient.ws.beans.WsSubject;
import edu.internet2.middleware.grouperClient.ws.beans.WsSubjectLookup;
import edu.internet2.middleware.grouperClientExt.org.apache.commons.logging.Log;
import edu.internet2.middleware.grouperClientExt.xmpp.GrouperClientXmppMain;
import edu.internet2.middleware.grouperClientExt.xmpp.GrouperClientXmppMessageHandler;

/**
 *
 */
public class ClusterLinuxLogic implements Job, StatefulJob {

  /**
   * @param args
   */
  public static void main(String[] args) {

    log.debug("Starting cluster linux permissions daemon");

    //schedule full refresh job on startup
    performFullRefresh();

    scheduleQuartzJob();

    //do xmpp loop
    xmppLoop();

  }

  /**
   * schedule the full refresh job, e.g. nightly
   */
  private static void scheduleQuartzJob() {

    String jobName = "clusterLinuxFullRefreshJob";

    String quartzCronString = GrouperClientUtils.propertiesValue("clusterLinux.fullRefreshQuartzCron", false);

    log.debug("Scheduling cluster linux permissions daemon for quartzCron string: " + quartzCronString);

    GrouperClientXmppMain.scheduleJob(jobName, quartzCronString, ClusterLinuxLogic.class);

  }

  /** timer */
  private static Timer timer = null;

  /** timer scheduled for */
  private static long timerScheduledFor = -1;

  /**
   * do a full refresh in one minute (batch subsequent requests)
```

```
 */
public static synchronized void scheduleFullRefresh() {

  //if it is already scheduled, then we are all good
  if (timer != null) {

    log.debug("Job is already scheduled at " + new Date(timerScheduledFor).toString() + ", exiting");

    return;
  }
  timer = new Timer(true);

  int timeInFuture = 1000*60;

  timerScheduledFor = System.currentTimeMillis() + timeInFuture;

  //schedule in 60 seconds
  timer.schedule(new TimerTask() {

    @Override
    public void run() {
      performFullRefresh();
    }
  }, timeInFuture);
}

/**
 * base folder for grouper
 * @return the folder, e.g. school:apps:clusterLinux
 */
public static String grouperFolderBase() {
  return GrouperClientUtils.propertiesValue("clusterLinux.grouperFolderBase", true);
}

/**
 * do a full refresh in one minute (batch subsequent requests)
 */
public static synchronized void performFullRefresh() {

  try {
    //let another timer be scheduled, whether from schedule or xmpp
    timer = null;

    String nameOfAttributeDef = grouperFolderBase() + ":permissions:tomcats:tomcatPermissionDef";

    String theRoleName = grouperFolderBase() + ":clusterLinuxRoles:"
        + GrouperClientUtils.propertiesValue("clusterLinux.environmentRoleExtension", true);

    WsGetPermissionAssignmentsResults wsGetPermissionAssignmentsResults =
      new GcGetPermissionAssignments().addAttributeDefName(nameOfAttributeDef)
        .addRoleName(theRoleName).addSubjectAttributeName("PENNNAME").execute();

    WsSubject[] wsSubjects = wsGetPermissionAssignmentsResults.getWsSubjects();

    //note, there is a bug where subjects arent returned...  go get them
    wsSubjects = ClusterLinuxLogic.retrieveSubjectsFromServer(wsGetPermissionAssignmentsResults);

    //lets make the permissions file
    StringBuilder fileContents = new StringBuilder("# File automatically generated from PennGroups...\n\n");

    //lets sort these so they are easier to manage
    Set<String> lines = new TreeSet<String>();

    if (log.isDebugEnabled()) {
      log.debug("Received "
          + GrouperClientUtils.length(wsGetPermissionAssignmentsResults.getWsPermissionAssigns())
          + " permission entries from Grouper");
    }

    for (WsPermissionAssign wsPermissionAssign : GrouperClientUtils.nonNull(
        wsGetPermissionAssignmentsResults.getWsPermissionAssigns(), WsPermissionAssign.class)) {
```

```
        //only worried about personal access at this time
        if (!GrouperClientUtils.equals(wsPermissionAssign.getSourceId(), "pennperson")) {
          continue;
        }

        //get the subject
        WsSubject wsSubject = ClusterLinuxLogic.retrieveSubject(
            wsSubjects, wsPermissionAssign.getSourceId(),
            wsPermissionAssign.getSubjectId());

        if (wsSubject == null) {
          throw new RuntimeException("Why is wsSubject null??? " + wsPermissionAssign.getSourceId()
            + wsPermissionAssign.getSubjectId());
        }

        String pennname = GrouperClientUtils.subjectAttributeValue(wsSubject,
            wsGetPermissionAssignmentsResults.getSubjectAttributeNames(), "PENNNAME");

        //not sure why there wouldnt be a pennname, but if not, then skip

        if (GrouperClientUtils.isBlank(pennname)) {
          continue;
        }

        //ok, we have pennname, extension, action, lets write the line
        String extension = GrouperClientUtils.extensionFromName(wsPermissionAssign.getAttributeDefNameName());
        String action = wsPermissionAssign.getAction();

        lines.add("clusterLinux__" + pennname + "__" + extension + "__" + action + "=true");

      }

      for (String line : lines) {
        fileContents.append(line).append("\n");
      }

      if (log.isDebugEnabled()) {
        log.debug("Generated file has "
            + GrouperClientUtils.length(lines)
            + " lines");
      }

      File fileToSave = new File(GrouperClientUtils.propertiesValue("clusterLinux.fileToSave", true));

      //save this to file (try 3 times)
      for (int i=0;i<3;i++) {

        try {

          boolean updated = GrouperClientUtils.saveStringIntoFile(fileToSave, fileContents.toString(), true,
true);
          if (updated) {
            log.debug("File: " + fileToSave.getAbsolutePath() + " was saved since there were updates from
Grouper");
          } else {
            log.debug("File: " + fileToSave.getAbsolutePath() + " was not saved since there were no changes
from Grouper");
          }

          //we are done
          break;
        } catch (Exception e) {
          log.error("Error saving file", e);
        }
        GrouperClientUtils.sleep(2000);

      }

    } catch (Exception e) {
      log.error("Error in full refresh", e);
```

```
  }

}

/**
 * logger
 */
private static Log log = GrouperClientUtils.retrieveLog(ClusterLinuxLogic.class);

/**
 * this will be run when the quertz nightly job fires
 * @see org.quartz.Job#execute(org.quartz.JobExecutionContext)
 */
// @Override
public void execute(JobExecutionContext context) throws JobExecutionException {

  String jobName = null;
  try {
    jobName = context.getJobDetail().getName();
    if (log.isDebugEnabled()) {
      log.debug("Scheduling full refresh on job: " + jobName);
    }
    ClusterLinuxLogic.scheduleFullRefresh();
  } catch (RuntimeException re) {
    log.error("Error in job: " + jobName, re);
    throw re;
  }

}

/**
 * connect to xmpp, listen for messages from a certain sender (the grouper server)
 */
 private static void xmppLoop() {

   GrouperClientXmppMain.xmppLoop(new GrouperClientXmppMessageHandler() {

    @Override
    public void handleMessage(Message message) {

      log.debug("Received message: " + message.getBody());

      //whatever message we get, we know it is from the right sender based on
      //config, so just schedule a full refresh a minute from now if its not already scheduled
      scheduleFullRefresh();

    }
  });

 }


 /**
  * lookup a subject by subject id and source id
  * @param subjects
  * @param sourceId
  * @param subjectId
  * @return probably shouldnt be null, but if cant be found, then will be null
  */
 public static WsSubject retrieveSubject(WsSubject[] subjects, String sourceId, String subjectId) {

   for (WsSubject wsSubject : GrouperClientUtils.nonNull(subjects, WsSubject.class)) {
     if (GrouperClientUtils.equals(sourceId, wsSubject.getSourceId())
         && GrouperClientUtils.equals(subjectId, wsSubject.getId())) {
       return wsSubject;
     }
   }
   return null;
 }

 /**
```

```
    * until Penn is upgraded to Grouper 2.1, we need another query to get subjects based on bug in grouper.
    * Once this is fixed this wont be needed, the subjects will be in the permissions response
    * @param wsGetPermissionAssignmentsResults
    * @return subjects
    */
   public static WsSubject[] retrieveSubjectsFromServer(WsGetPermissionAssignmentsResults
wsGetPermissionAssignmentsResults) {

     //make sure no dupes
     Set<MultiKey> subjectSourceAndIds = new HashSet<MultiKey>();

     for (WsPermissionAssign wsPermissionAssign : wsGetPermissionAssignmentsResults.getWsPermissionAssigns()) {
       if (GrouperClientUtils.equals("pennperson", wsPermissionAssign.getSourceId())) {
         subjectSourceAndIds.add(new MultiKey(wsPermissionAssign.getSourceId(), wsPermissionAssign.
getSubjectId()));
       }
     }
     GcGetSubjects gcGetSubjects = new GcGetSubjects().addSubjectAttributeName("PENNNAME");

     //add those to the get subjects request
     for (MultiKey subjectSourceAndId : subjectSourceAndIds) {
       gcGetSubjects.addWsSubjectLookup(new WsSubjectLookup((String)subjectSourceAndId.getKey(1),
           (String)subjectSourceAndId.getKey(0), null));
     }

     WsGetSubjectsResults wsGetSubjectsResults = gcGetSubjects.execute();

     return wsGetSubjectsResults.getWsSubjects();

   }


}
```

Compile that into a jar:

```
<project name="clusterLinux" default="build" basedir=".">

  <!-- declare the ant-contrib tasks -->
  <taskdef resource="net/sf/antcontrib/antlib.xml" />

  <!-- copy build.properties if not there already -->
  <if><not><available file="build.properties" /></not>
    <then><copy file="build.example.properties" tofile="build.properties" /></then>
  </if>

  <!-- Grouper Global Build Properties -->
  <property file="${basedir}/build.properties"/>

  <target name="build" description="full build" depends="init,clean,compile,jarPrepare,jar">
  </target>

  <target name="init">
    <tstamp />

    <property name="main.source" value="src" />

    <property name="main.lib" value="lib" />

    <property name="main.bin" value="dist/bin" />
    <property name="main.outputDir" value="dist" />

    <property name="main.appName" value="clusterLinux" />
    <property name="main.jarFile" value="${main.outputDir}/${main.appName}.jar" />

    <path id="main.classpath">
      <fileset dir="${main.lib}">
        <include name="**/*.jar" />
      </fileset>
```

```
    </path>

  </target>

  <target name="clean" depends="init">
    <mkdir dir="${main.bin}" />
    <delete dir="${main.bin}" />
    <mkdir dir="${main.bin}" />

  </target>

  <target name="compile">
    <mkdir dir="${main.outputDir}" />
    <mkdir dir="${main.bin}" />

    <javac  target="1.6"
      srcdir="${main.source}" destdir="${main.bin}" debug="true" >
      <classpath refid="main.classpath" />
    </javac>

  </target>

  <target name="jarPrepare">
    <mkdir dir="${main.bin}" />

    <copy todir="${main.bin}">
      <fileset dir="${main.source}">
        <include name="**/*.java"/>      <!-- source -->
      </fileset>
    </copy>

    <mkdir dir="${main.bin}/clusterLinux" />

    <copy todir="${main.bin}/clusterLinux" file="build.xml" />

  </target>

  <target name="jar">
    <tstamp>
        <format property="the.timestamp" pattern="yyyy/MM/dd HH:mm:ss" />
    </tstamp>
    <jar jarfile="${main.jarFile}" duplicate="fail">
      <fileset dir="${main.bin}" />
      <manifest>
        <attribute name="Built-By"               value="${user.name}"/>
        <attribute name="Implementation-Vendor"   value="Penn"/>
        <attribute name="Implementation-Title"    value="clusterLinux"/>
        <attribute name="Build-Timestamp"         value="${the.timestamp}"/>
      </manifest>
     </jar>
     <echo message="Output is: ${main.jarFile}" />
   </target>


</project>
```

Put that jar, and the other required jars from the grouper client for XMPP and quartz cron into a dir on the server:

```
[appadmin@lukes clusterLinux]$ ls -lat
total 6964
-rw-r--r-- 1 appadmin users      11099 May 21 14:29 clusterLinux.jar
drwxrwxr-x 2 appadmin appadmin    4096 May 21 14:04 .
-rw-r--r-- 1 appadmin users     173783 May 21 14:04 commons-beanutils.jar
-rw-r--r-- 1 appadmin users     570463 May 21 14:04 commons-collections.jar
-rw-r--r-- 1 appadmin users     468109 May 21 14:04 commons-lang.jar
-rw-r--r-- 1 appadmin users     131078 May 21 14:04 commons-logging.jar
-rw-r--r-- 1 appadmin users      86542 May 21 14:04 ezmorph.jar
-rw-r--r-- 1 appadmin users    2649669 May 21 14:04 grouperClient.jar
-rw-r--r-- 1 appadmin users     255813 May 21 14:04 json-lib.jar
-rw-r--r-- 1 appadmin users       8374 May 21 14:04 jta.jar
-rw-r--r-- 1 appadmin users     391834 May 21 14:04 log4j.jar
-rw-r--r-- 1 appadmin users     792769 May 21 14:04 quartz.jar
-rw-r--r-- 1 appadmin users    1381464 May 21 14:04 smack.jar
-rw-r--r-- 1 appadmin users        753 May 21 13:48 log4j.properties
-rw-r--r-- 1 appadmin users      17402 May 21 13:47 grouper.client.properties
-rw-r--r-- 1 appadmin users      17390 May 21 13:46 grouper.client.properties~
-rwxr-xr-x 1 appadmin users       3124 May 21 13:44 clusterLinuxPermissions
-rwxr-xr-x 1 appadmin users        306 May 21 13:41 clusterLinuxCommand.sh
```

Have a log4j.properties

```
## Grouper API error logging
log4j.appender.grouper_error                            = org.apache.log4j.RollingFileAppender
log4j.appender.grouper_error.File                       = /opt/appserv/local/clusterLinux/clusterLinux.log
log4j.appender.grouper_error.MaxFileSize                = 1000KB
log4j.appender.grouper_error.MaxBackupIndex             = 1
log4j.appender.grouper_error.layout                     = org.apache.log4j.PatternLayout
log4j.appender.grouper_error.layout.ConversionPattern   = %d{ISO8601}: [%t] %-5p %C{1}.%M(%L) - %x - %m%n

# Loggers

## Default logger; will log *everything*
log4j.rootLogger  = ERROR, grouper_error

## All Internet2 (warn to grouper_error per default logger)
#log4j.logger.edu.internet2.middleware = WARN
log4j.logger.edu.upenn.isc.clusterLinux = DEBUG
```

Configure the grouper.client.properties to point to the Grouper WS, XMPP, and configuration for this daemon (here are snippets)

```
########################################
## Web service Connection settings
########################################

# url of web service, should include everything up to the first resource to access
# e.g. http://groups.school.edu:8090/grouperWs/servicesRest
# e.g. https://groups.school.edu/grouperWs/servicesRest
grouperClient.webService.url = https://grouper.school.edu/grouperWs/servicesRest

# kerberos principal used to connect to web service
# e.g. name/server.whatever.upenn.edu
grouperClient.webService.kerberosPrincipal = clusterLinuxGrouper/school.edu

# password for shared secret authentication to web service
# or you can put a filename with an encrypted password
grouperClient.webService.password = XXXXXXXXXX

...

################################
## XMPP client settings
## Note: you need the smack.jar in your classpath, see the grouper xmpp wiki for usage
## https://spaces.at.internet2.edu/display/Grouper/Grouper+XMPP+notifications+v1.6.0
################################


## general xmpp configuration
grouperClient.xmpp.server.host = jabber.school.edu
grouperClient.xmpp.server.port = 5222
grouperClient.xmpp.user = clusterLinuxProcess@school.edu
# note, pass can be in an external file with morphstring
grouperClient.xmpp.pass = xxxxxxxxxxxx
grouperClient.xmpp.resource = clusterLinuxTest
# note, you need the exact id and resource here or it wont match
grouperClient.xmpp.trustedMessagesFromJabberIds = grouperjabber@school.edu/grouperServer



################################
## Cluster linux settings
################################

# when should the full refresh kick off
clusterLinux.fullRefreshQuartzCron = 0 0 4 * * ?

# cluster linux base folder in grouper
clusterLinux.grouperFolderBase = XXXXXXXX:apps:fast:clusterLinux

# must be a role in the clusterLinuxRoles folder under the base for this env
clusterLinux.environmentRoleExtension = cloudTestenvUser

# file that holds the BASH env variables
clusterLinux.fileToSave = /somefolder/clusterLinuxPermissions.sh
```

Here is the init.d script (which is symlinked to the real location): /opt/appserv/binMgmt/clusterLinux/clusterLinuxPermissions

```
#!/bin/sh
#
# Startup script for the Tomcat Server
#
# chkconfig: - 86 14
# description: Cluster Linux Permissions
# processname:
# pidfile:
# config:
# Tomcat
```

```
# description: Keeps Grouper permissions in sync with file on linux server

runningUser=`/usr/bin/whoami`

# we only want appadmin or root to be able to do this
if [ "$runningUser" != "appadmin" ]; then

  if [ "$runningUser" != "root"  ]; then

    echo "ERROR: only user appadmin or root can run administer the clusterLinuxPermissions service:
'$runningUser'"
    echo
    exit 0

  fi

fi

case "$1" in
  start)

    echo -n "Starting Cluster linux permissions service: "
    echo

    #if there is a pid file, then stop this service
    if [ -f /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid ]; then
      $0 stop
      sleep 5
    fi

    # if not appadmin, then we must be root
    if [ ! "$runningUser" == "appadmin" ]; then
      su appadmin -c /opt/appserv/binMgmt/clusterLinux/clusterLinuxCommand.sh
    else
      # if not root then we must be appadmin
      /opt/appserv/binMgmt/clusterLinux/clusterLinuxCommand.sh
    fi
    echo
    ;;
  stop)

      pid=`cat /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid`
      /bin/kill $pid

    #loop until the program is dead
    waitingForExit=true
    iterator=1

    while [ "$waitingForExit" == "true" ]; do

      processId=`/bin/ps -fp $pid | grep java | cut -c10-15`
      if [ -n "$processId" ]; then
        echo Waiting for exit...
      else
        waitingForExit=false
      fi

      if [ "$iterator" -gt "10" ]; then
        waitingForExit=false
      fi

      let "iterator += 1"
      sleep 1
    done

      rm /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid

    echo
    ;;
  status)
    if [ -f /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid ]; then
```

```
      pid=`cat /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid`
      processId=`/bin/ps -fp $pid | grep java | cut -c10-15`
      if [ -n "$processId" ]; then
        echo "clusterLinuxPermissions is running under process ID $processId"
      else
        echo "clusterLinuxPermissions is not running"
      fi
    else
      echo "clusterLinuxPermissions is not running"
    fi

    ;;
  restart)
    echo -n "Restarting clusterLinuxPermissions: "
    $0 stop
    sleep 5
    $0 start
    echo "done."
    ;;
  *)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
esac
```

Symlink to init.d

```
[root@server init.d]# ln -s /opt/appserv/binMgmt/clusterLinux/clusterLinuxPermissions clusterLinuxPermissions
[root@server init.d]# chkconfig --add clusterLinuxPermissions
[root@server init.d]# chkconfig --levels 345 clusterLinuxPermissions on
```

Start the process:

```
[appadmin@server clusterLinux]$ /sbin/service clusterLinuxPermissions start
Starting Cluster linux permissions service:

[appadmin@server clusterLinux]$ more /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid
25507
[appadmin@server clusterLinux]$ ps -ef | grep clusterLinux | grep -v "grep"
appadmin 25507     1  4 17:06 pts/9    00:00:04 /opt/appserv/common/java6/bin/java -cp /opt/appserv/binMgmt
/clusterLinux:/opt/appserv/binMgmt/clusterLinux/* edu.upenn.isc.clusterLinux.ClusterLinuxLogic
[appadmin@server clusterLinux]$
```

Stop the process:

```
[appadmin@lukes clusterLinux]$ /sbin/service clusterLinuxPermissions stop
Waiting for exit...
[appadmin@lukes clusterLinux]$ more /opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid
/opt/appserv/local/clusterLinux/clusterLinuxDaemon.pid: No such file or directory
[appadmin@lukes clusterLinux]$ ps -ef | grep clusterLinux | grep -v "grep"
[appadmin@lukes clusterLinux]$
```

At this point the permissions file should exist

```
[appadmin@server clusterLinux]$ more  /opt/appserv/binMgmt/clusterLinux/clusterLinuxRules.sh
# File automatically generated from PennGroups...

clusterLinux__mchyzer__app_directory__all=true
clusterLinux__mchyzer__app_directory__restart=true
clusterLinux__mchyzer__app_directory__start=true
clusterLinux__mchyzer__app_directory__status=true
clusterLinux__mchyzer__app_directory__stop=true
clusterLinux__mchyzer__app_fastUnit__status=true
[appadmin@lukes clusterLinux]$
```

That script is what the end user script calls

```
[mchyzer@lukes clusterLinux]$ clusterTomcat tomcat_directory restart
```

The command should sudo su as another user:

```
/usr/bin/sudo -u appadmin /somedir/whatever/clusterTomcatHelper.sh $1 $2
```

Another shell script (clusterTomcatHelper.sh) can be coded to use the rules from the permissions file:

```
#!/bin/bash

# this script *must* run as appadmin or root

runningUser=`/usr/bin/whoami`

# sudo should set this variable
loggedInUser=$SUDO_USER

if [ "$runningUser" != "appadmin" ]; then
  echo "ERROR: only user appadmin can run this command: '$runningUser'"
  echo
  exit 0
fi

# this script will start / stop / get status of tomcat

if [[ $# -ne "2" && $# -ne "3" ]]; then
  echo
  echo "ERROR: pass in the tomcat or application to affect (appName, tomcat_b, ... etc) and"
  echo "an argument to tomcat as the command line arg"
  echo "e.g. status, start, stop, restart"
  echo "optionally you can give a server to run on, or blank for all in cluster"
  echo "or 'all' to run on all nodes in every cluster (if tomcat has changed clusters)"
  echo "e.g. clusterTomcatHelper.sh tomcat_b status"
  echo "e.g. clusterTomcatHelper.sh secureShare status theprince"
  echo
  exit 1
fi

argAppName=$1
argAction=$2
argServer=$3

# make sure the argument has no spaces or special chars
function validateArg() {
  if [[ "${1}" == "" ]]; then
    return;
  fi
  #echo checking "'$1'"
```

```
  if [[ ! $1 =~ ^[a-zA-Z0-9_-]*$ ]]; then
    echo "Invalid arg '$1'"
    exit 1;
  fi
}

validateArg "$argAppName"
validateArg "$argAction"
validateArg "$argServer"



# convert the arg to the application name, e.g. secureShare
appName=`/opt/appserv/common/binGroovy/clusterGetApplicationName.groovy $argAppName`
#echo "'$appName'"

appTomcatName=tomcat_"$appName"
#echo $appTomcatName

# get the env var name
# e.g. clusterLinux__mchyzer__app_directory__all=true
securityVarName=clusterLinux__"$loggedInUser"__app_"$appName"__"$argAction"
#echo $securityVarName

# see if it is set, if it is, it is a security problem since the ACL file should set it
eval securityVarValue=\$$securityVarName

if [[ ! "${securityVarValue}" == "" ]]; then
  echo "Why is this set? $securityVarValue???"
  exit 1;
fi

source /opt/appserv/binMgmt/clusterLinuxRules.sh

eval securityVarValue=\$$securityVarName

if [[ ! "$securityVarValue" == "true" ]]; then
  echo "User is not allowed to run the command, contact the sysadmin to get permission: $loggedInUser,
$appName, $argAction"
  exit 1
fi

# ok, we are allowed to execute the command, go for it
```

...

sdf

Change log consumer

This is the part that is a jar installed in the Grouper loader and an edit in the grouper-loader.properties to send permissions to XMPP

```
/**
 * @author mchyzer
 * $Id: ClusterLinuxChangeLogConsumer.java,v 1.2 2012/05/21 17:01:24 mchyzer Exp $
 */
package edu.upenn.isc.clusterLinuxClc;

import java.util.List;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;

import edu.internet2.middleware.grouper.app.loader.GrouperLoaderConfig;
import edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase;
import edu.internet2.middleware.grouper.changeLog.ChangeLogEntry;
import edu.internet2.middleware.grouper.changeLog.ChangeLogLabels;
```

```
import edu.internet2.middleware.grouper.changeLog.ChangeLogProcessorMetadata;
import edu.internet2.middleware.grouper.changeLog.ChangeLogTypeBuiltin;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.grouper.xmpp.XmppConnectionBean;


/**
 * change log consumer to listen for specific gruoper actions that should tell linux to refresh
 * their permissions
 */
public class ClusterLinuxChangeLogConsumer extends ChangeLogConsumerBase {

  /** */
  private static final Log LOG = GrouperUtil.getLog(ClusterLinuxChangeLogConsumer.class);

  /**
   * @see edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerBase#processChangeLogEntries(java.util.
List, edu.internet2.middleware.grouper.changeLog.ChangeLogProcessorMetadata)
   */
  @Override
  public long processChangeLogEntries(List<ChangeLogEntry> changeLogEntryList,
      ChangeLogProcessorMetadata changeLogProcessorMetadata) {

    //identifies which config is running, multiple could be running
    String consumerName = changeLogProcessorMetadata.getConsumerName();

    long currentId = -1;

    XmppConnectionBean xmppConnectionBean = null;
    String recipient = null;

    String grouperFolderBase = null;

    {
      String grouperFolderBaseConfigName = "changeLog.consumer." + consumerName + ".clusterLinux.
grouperFolderBase";

      grouperFolderBase = GrouperLoaderConfig.getPropertyString(grouperFolderBaseConfigName, true);
    }

    boolean sendMessage = false;

    for (ChangeLogEntry changeLogEntry : changeLogEntryList) {

      //try catch so we can track that we made some progress
      try {

        currentId = changeLogEntry.getSequenceNumber();

        //only need to send once
        if (sendMessage) {
          continue;
        }

        //this might be a little aggressive, but basically if a permission or member changes in the
clusterLinux folder of
        //grouper then send a refresh message
        //note: not using constants for permissions so it works in 2.0 and 2.1...
        if (StringUtils.equals("permission", changeLogEntry.getChangeLogType().getChangeLogCategory())
            && (StringUtils.equals("addPermission", changeLogEntry.getChangeLogType().getActionName())
                || StringUtils.equals("deletePermission", changeLogEntry.getChangeLogType().getActionName()))) {

          String permissionName = changeLogEntry.retrieveValueForLabel("attributeDefNameName");

          if (permissionName != null && permissionName.startsWith(grouperFolderBase)) {
            sendMessage = true;
          }
          if (LOG.isDebugEnabled()) {
            LOG.debug("Processing changeLog #" + currentId + ", permissionName: " + permissionName
                + ", grouperFolderBase: " + grouperFolderBase
                + ", message: "
```

```
            + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
            + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
            );
        }
      } else if (changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_ADD)
          || changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_DELETE)
          || changeLogEntry.equalsCategoryAndAction(ChangeLogTypeBuiltin.MEMBERSHIP_UPDATE)) {

        String roleName = changeLogEntry.retrieveValueForLabel(ChangeLogLabels.MEMBERSHIP_ADD.groupName);

        if (roleName != null && roleName.startsWith(grouperFolderBase)) {
          sendMessage = true;
        }
        if (LOG.isDebugEnabled()) {
          LOG.debug("Processing changeLog #" + currentId + ", roleName: " + roleName
            + ", grouperFolderBase: " + grouperFolderBase
            + ", message: "
            + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
            + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
            );
        }

      } else if (StringUtils.equals("permission", changeLogEntry.getChangeLogType().getChangeLogCategory())
          && StringUtils.equals("permissionChangeOnRole", changeLogEntry.getChangeLogType().getActionName()))
{

        //note, this is 2.1+

        String roleName = changeLogEntry.retrieveValueForLabel("roleName");

        if (roleName != null && roleName.startsWith(grouperFolderBase)) {
          sendMessage = true;
        }
        if (LOG.isDebugEnabled()) {
          LOG.debug("Processing changeLog #" + currentId + ", roleName: " + roleName
            + ", grouperFolderBase: " + grouperFolderBase
            + ", message: "
            + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
            + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
            );
        }
      } else {
        if (LOG.isDebugEnabled()) {
          LOG.debug("Processing changeLog #" + currentId + ", "
            + changeLogEntry.getChangeLogType().getChangeLogCategory() + "."
            + changeLogEntry.getChangeLogType().getActionName() + ", sendMessage: " + sendMessage
            );
        }
      }

      //is there something to send?
      if (sendMessage) {

        String message = "Update permissions";

        if (xmppConnectionBean == null) {

          recipient = GrouperLoaderConfig.getPropertyString("changeLog.consumer."
              + consumerName + ".publisher.recipient", "");

          String xmppServer = GrouperLoaderConfig.getPropertyString("xmpp.server.host");
          int port = GrouperLoaderConfig.getPropertyInt("xmpp.server.port", -1);
          String username = GrouperLoaderConfig.getPropertyString("xmpp.user", "");
          String password = GrouperLoaderConfig.getPropertyString("xmpp.pass", "");
          String resource = GrouperLoaderConfig.getPropertyString("xmpp.resource", "");

          xmppConnectionBean = new XmppConnectionBean(xmppServer, port, username, resource, password);

        }

        xmppConnectionBean.sendMessage(recipient, message);
```

```
        }
      } catch (Exception e) {
        //we unsuccessfully processed this record... decide whether to wait, throw, ignore, log, etc...
        LOG.error("problem with id: " + currentId, e);
        changeLogProcessorMetadata.setHadProblem(true);
        changeLogProcessorMetadata.setRecordException(e);
        changeLogProcessorMetadata.setRecordExceptionSequence(currentId);
        //stop here
        return currentId;
        //continue
      }
    }

    return currentId;
  }

}
```

Build that consumer into a jar:

```
<project name="clusterLinuxChangeLogConsumer" default="build" basedir=".">

  <!-- declare the ant-contrib tasks -->
  <taskdef resource="net/sf/antcontrib/antlib.xml" />

  <!-- copy build.properties if not there already -->
  <if><not><available file="build.properties" /></not>
    <then><copy file="build.example.properties" tofile="build.properties" /></then>
  </if>

  <!-- Grouper Global Build Properties -->
  <property file="${basedir}/build.properties"/>

  <target name="build" description="full build" depends="init,clean,compile,jarPrepare,jar">
  </target>

  <target name="init">
    <tstamp />

    <property name="main.sourceChangeLogDir" value="../sourceChangeLogConsumer" />

    <property name="main.lib" value="../lib" />

    <property name="main.binChangeLogDir" value="../dist/binChangeLog" />
    <property name="main.outputDir" value="../dist" />

    <property name="main.appName" value="clusterLinuxChangeLog" />
    <property name="main.jarFileChangeLog" value="${main.outputDir}/${main.appName}.jar" />

    <path id="main.changeLogClasspath">
      <fileset dir="${main.lib}">
        <include name="**/*.jar" />
      </fileset>
      <fileset file="${grouper.jar.location}" />
    </path>

  </target>

  <target name="clean" depends="init">
    <mkdir dir="${main.binChangeLogDir}" />
    <delete dir="${main.binChangeLogDir}" />
    <mkdir dir="${main.binChangeLogDir}" />

  </target>

  <target name="compile">
    <mkdir dir="${main.outputDir}" />
    <mkdir dir="${main.binChangeLogDir}" />
```

```
    <javac  target="1.6"
      srcdir="${main.sourceChangeLogDir}" destdir="${main.binChangeLogDir}" debug="true" >
      <classpath refid="main.changeLogClasspath" />
    </javac>

  </target>

  <target name="jarPrepare">
    <mkdir dir="${main.binChangeLogDir}" />

    <copy todir="${main.binChangeLogDir}">
      <fileset dir="${main.sourceChangeLogDir}">
        <include name="**/*.java"/>      <!-- source -->
      </fileset>
    </copy>

    <mkdir dir="${main.binChangeLogDir}/clusterLinux" />

    <copy todir="${main.binChangeLogDir}/clusterLinux" file="build.xml" />

  </target>

  <target name="jar">
    <tstamp>
        <format property="the.timestamp" pattern="yyyy/MM/dd HH:mm:ss" />
    </tstamp>
    <jar jarfile="${main.jarFileChangeLog}" duplicate="fail">
      <fileset dir="${main.binChangeLogDir}" />
      <manifest>
        <attribute name="Built-By"                value="${user.name}"/>
        <attribute name="Implementation-Vendor"   value="Penn"/>
        <attribute name="Implementation-Title"    value="clusterLinuxChangeLog"/>
        <attribute name="Build-Timestamp"         value="${the.timestamp}"/>
      </manifest>
     </jar>
     <echo message="Output is: ${main.jarFileChangeLog}" />
   </target>


</project>
```

If you want to log the change log, add this to the log4j.properties:

```
log4j.logger.edu.upenn.isc.clusterLinuxClc.ClusterLinuxChangeLogConsumer = DEBUG
```

If you need more debugging, try these:

```
log4j.logger.edu.internet2.middleware.grouper.changeLog = DEBUG

log4j.logger.edu.internet2.middleware.grouper.app.loader = DEBUG
```

Make the jar with ant, copy to the grouper loader lib dir, and edit the grouper-loader.properties

```
#####################################
## CLUSTER LINUX CONSUMER
#####################################

changeLog.consumer.clusterLinuxTest.class = edu.upenn.isc.clusterLinuxClc.ClusterLinuxChangeLogConsumer

changeLog.consumer.clusterLinuxTest.publisher.recipient = someuser@school.edu/clusterLinuxTest,
anotheruser@school.edu, someuser@school.edu/clusterLinuxTestLegacy

changeLog.consumer.clusterLinuxTest.clusterLinux.grouperFolderBase = xxxxxxx:apps:fast:clusterLinux

changeLog.consumer.clusterLinuxTest.quartzCron =
```

Restart the loader and add/remove someone from the role, or change permissions, see an XMPP message being sent

```
(4:35:01 PM) PennGroups: Update permissions
```

Note, the body of the message isn't important... if the message comes from Grouper and goes to the grouperClient keeping the file in sync, then it will do a 1 minute delay full refresh if one isnt already scheduled (to batch up multiple changes)

If you have DEBUG logging on for the change log consumer, you will see entries like this:

```
120521 163500.260 [DefaultQuartzScheduler_Worker-5] DEBUG edu.upenn.isc.clusterLinuxClc.
ClusterLinuxChangeLogConsumer -
Processing changeLog #3593567, permissionName: XXXXXXXXX:apps:fast:clusterLinux:permissions:tomcats:
app_directory,
grouperFolderBase: XXXXXXXXX:apps:fast:clusterLinux, message: attributeAssign.deleteAttributeAssign,
sendMessage: true
```

sdf