Monitoring Your OpenMRS Implementations to Ensure It's Running

This is meant to be a page to describe how to monitor your different instances of OpenMRS (and perhaps other hardware or software) to ensure it's up and running.

First approach

Many folks monitor their IT infrastructure with Nagios, an enterprise-level, open-source monitoring system.

For a simple script, you could try something like this:

```bash
#!/bin/sh
url='http://localhost:8080/openmrs/index.htm';
timeout=5;  # seconds allowed before giving up
status=IFS=' ' r=$(curl -Is --connect-timeout $(timeout) "$url" || echo 1 500);echo ${r[1]};
if [ $status -eq 500 ]; then
  /etc/init.d/tomcat6 restart
fi
```

For ubuntu, if you save that script (setting URL to the exact URL of the page you want to test) into /etc/init.d/cron.hourly, then tomcat6 should get restarted if the page doesn't load. You could play with cron and make it run more frequently, but beware that this could try to restart tomcat6 each time it is run if OpenMRS isn't working. Also note that the URL should be a final URL and not a redirect (e.g., http://localhost:8080/openmrs/ is not a final URL; rather, it redirects you to another URL).

Cheers, -Burke

For those interested in how this line works:

```bash
status=IFS=' ' r=$(curl -Is --connect-timeout $(timeout) "$url" || echo 1 500);echo ${r[1]};
```

I'll break it down...working from the inside out:

```bash
curl -Is --connect-timeout $(timeout) "$url"
```

uses the curl command to fetch the URL, waiting up to TIMEOUT seconds for it to respond. Option -I makes it return HTTP headers and -s silences output like a progress bar. The HTTP output starts with something like "HTTP/1.1 200 OK", where the second word is the status code that we want.

```
|| echo 1 500
```

follows the curl command, using the scripting "or" operator (||) to return "1 500" if the curl command comes up empty. The "1" is a dummy value, since we want "500" as the second word to mimic a HTTP internal server error code 500. The above is wrapped in a $() to execute the command and its output is assigned to the variable r. The preceding IFS=' ' sets the input field separator to split the subsequent command's output on spaces -- in other words, thanks to the IFS=' ', the output of the curl command is broken apart into separate words. Finally, the echo $r[1] command at the end takes the value of variable r (our curl output, broken into words), retrieves the second word (we geeks start numbering at zero, so "1" is the second word), and echoes it into the status variable. When the line is done executing, we either have the HTTP status code (200 is good, 500 is bad) in the status variable.

Another approach

Not sure if this is providing a deeper check, but I was invoking a full login request to make sure the whole stack is working. Together with some mail notification and a double check if the system is really down I wrote down a few more lines. Hourly invoked through a cronjob it does want I needed for now. Just in case someone wants to have an old-school alternative to the fancy curl statement...
#!/bin/bash

MAIL=cneumann@pih.org
PATH=$PATH:/bin:/usr/bin # make sure wget and mailx commands are in the crontab path
BASEDIR=$(dirname $0)

# copy these 3 lines into a file openmrs.config and make sure it never gets checked in/published anywhere
# BASE_URL=http://localhost:8080/openmrs
# USER=admin
# PW=<change me>

source $BASEDIR/openmrs.config

wget --quiet --post-data "uname=$USER&pw=$PW" $BASE_URL/loginServlet

if [ $? -ne 0 ]; then
    # do it again to make sure this system wasn't just too busy in this moment
    sleep 300
    wget --quiet --post-data "uname=$USER&pw=$PW" $BASE_URL/loginServlet
    if [ $? -ne 0 ]; then
        # stop and start tomcat, might/will require root privileges
        # run script from root crontab or give sudo permissions to user (sudo visudo)
        sudo /etc/init.d/tomcat5.5 stop
        sleep 300
        sudo /etc/init.d/tomcat5.5 start
        mailx -s "emr: OpenMRS did not respond. Restarted at `date +%Y%m%d-%H%M'" "$MAIL" <<EOF
(intentionally empty)
EOF
        fi
    else
        #mailx -s "emr: OpenMRS running. All ok at `date +%Y%m%d-%H%M'" "$MAIL" <<EOF
(intentionally empty)
EOF
        fi
fi

rm index.htm*