Upwatch Administration Guide

Ron Arts

Upwatch Administration Guide
by Ron Arts

Table of Contents

Preface ..... **i**

1. About UpWatch **1**

1.1. History 1

1.2. Features 1

2. Installation **3**

2.1. Getting upwatch 3

2.2. Requirements 3

2.2.1. Run-time requirements 3

2.2.2. Build requirements 3

2.3. Compiling upwatch 4

2.4. Actual Installation 4

2.5. PHP pages 4

2.6. Security considerations 4

2.7. Database 5

3. Configuration **6**

3.1. Probe configuration 6

3.2. Tailoring uw_sysstat 6

3.3. Database configuration 7

4. Administration **8**

4.1. Logging 8

4.2. Managing daemons 8

4.3. Queues 8

Index **9**

Preface

People, especially managers, like to have facts and figures when taking decisions, either because a lot of money may be involved, or their job (or both). If you want to prove your website (or switch, or basically any other device) was available, showed the proper performance, or just want to know current and past CPU load, you've come to the right place.

Chapter 1. About UpWatch

1.1. History

UpWatch is born from the loins of Netland Internet Services BV, Amsterdam, The Netherlands. We are a hosting company which started in 1993 (when even Bill Gates knew nothing of the internet). We started doing managed hosting in 1995, and that's when we found out about monitoring. The hard way.

It became clear that customers can easily bring down their own server on impossible moments, and that it doesn't look very good if you both find out about that the monday after. So we started doing SLA's and limit customer rights on their own server.

Initially we used Big Brother (bb4.com) for monitoring. This is an outstanding and useful package, and we have been using it for many years. But it has a few downsides. One is scalability. It does not scale well to hundreds of hosts. Also it has a geek-like look, we felt we couldn't give the URL to our customers. Third problem was integration with our backoffice.

At the same time yours truly was thinking about setting up a commercial service for monitoring servers remotely. All this culminated into UpWatch. So lets get straight to the ..

1.2. Features

This is the full list of upwatch features:

- Scalable: designed for monitoring tens of thousands of hosts
- Resilient: autorestarts after database failures, forgiving for operator errors
- SuSE, RedHat and Fedora RPM's generated from sourcetree for easy installation
- Extensive and complete documentation, partly generated from source
- Multi-tenanting: multiple companies can run monitoring services for network of multiple client-companies using the same backend+probe serverpark
- OS support: clients available for Linux, Windows, FreeBSD, Solaris, server runs on linux or freeBSD, remote monitoring is linux only.
- Monitoring results are in XML, and can be pre- and postprocessed
- Secure: run as ordinary user, developed with security in mind
- Compatible with all Big Brother clients, imports bb-hosts file
- GUI is multi-language enabled (uses gettext)
- GUI has mobile client support
- Generates realtime graphs from the database
- Notifications by email or SMS.
- Clients for: HTTP GET, IMAP, MSSQL, MySQL, PING, POP3, PostgreSQL, SMTP, SNMP GET, TCP connect (any port)
- Local client detects: CPU load, loadavg, swap use, I/O use, memory use, and where supported hardware info like CPU temperature, fan speed and Power voltages. Also you can set it up to scan any logfile using regular expressions you supply.

- Fully opensource: GUI built on Apache/PHP, Backend on C/Perl, Database is MySQL. Uses GNU configure.

Chapter 2. Installation

2.1. Getting upwatch

Currently, upwatch is not released, and is not allowed to be distributed. The only way to get it, is through written permission of UpWatch BV.

If you aquired that, you will either receive access to CVS, or will receive a tar.gz file, or .RPM's.

Building and installing upwatch is not for the faint of heart. It uses lots of external libraries which may or may not be available on your platform. I myself use Redhat 8/9 for development, and test compilation on RH7, Fedora Core 1/2, SuSE8.2, Solaris, FreeBSD, and Yellowdog Linux 3.0.

2.2. Requirements

2.2.1. Run-time requirements

First ensure that the time/date on all hosts is correctly set.

Run time requirements differ per probe. Look in the corresponding .def file (or in the spec file for the probe), here's a list of everything we expect on a machine running all probes, and the database (I'll also list the version we use ourselves):

- glib2 >= 2.0.4
- xml2 (any version will do)
- freetds >= 0.6.0 compiled with --enable-threadsafe
- mysql 3.23.49
- postgreSQL 7.1.0
- net-snmp 5.0.6
- -lcrypto
- libpcap 0.6.2
- libpcre 3.9.10
- libncurses 5.2
- libesmtp 1.0.1
- libreadline 4.3

Delivered with upwatch are libstatgrab (0.7), xmbmon 2.03, and the State Threads Library (1,4).

2.2.2. Build requirements

You probably don't want to build upwatch yourself. Most likely you'll grab the RPM packages and issue rpm -Uvh upwatch*rpm. Then skip to Configuration.

But on the other hand: you can build the software yourself. Apart from the normal GNU compilation tools, and the development versions of the above mentioned packages, you'll need the following on your system to build upwatch:

- autogen 5.3.6 (autogen.sourceforge.net)
- if you also want to rebuild the manual: lynx 2.8.4, libxslt 1.0.15 and docbook 1.48, including the entire toolchain: openjade, jadetex, tetex, netpbm, perl-SGMLSpm.
- RPM tools, if you want to build RPM's

If you run RedHat, Debian or SuSE, don't forget to install the *-devel packages if there are any.

2.3. Compiling upwatch

Just in case you really want to (or need to) compile upwatch yourself, it's pretty easy:

```
$ tar xzvf upwatch-x.x.tar.gz
$ cd upwatch-x.x
$ ./configure
$ make
```

Nothing to it... In case of problems, you're probably missing some library or header files, or they are in unexpected places. Look at the last parts of config.log.

You can optionally specify --enable-monitors, --enable-iptraf --enable-server or --enable-all to configure. Default configure only builds the client, docs, and utils.

2.4. Actual Installation

Before you install the software decide on the architecture. If you know in advance you'll have to monitor thousands of hosts, or the probes will exhaust your machine otherwise, you may have to split your installation across several machines. There may be more reasons to do that. Consult 'Scaling up' and 'How it all works' in the Programmers Guide.

For simplicity we assume you run everything on the same host. Type the following command as root:

```
$ make install
```

2.5. PHP pages

The PHP pages can just be copied to any directory. There is an include directory. Copy that a some location ouside the web root, and enter its location in the .htaccess file in the web root dir. Also enter the database details in config.php.

2.6. Security considerations

All upwatch directories are readable and writable by members of the group upwatch. Most all executables run as user upwatch. Some probes need root-access, most notable uw_ping, and they will be installed suid root. These probes drop root privileges wherever possible. Further you can assign each probe its own database user and grant that user access rights to its own database tables. The probes itself don't write to the database, they only read from the `pr_xxx_def` tables.

The PHP web user should have SELECT, UPDATE, DELETE access to all tables.

uw_access, and **uw_accessbb** are the programs most vulnerable to crackers, as they wait for incoming connections on a TCP port (1985/1984). If possible, use **chroot** and firewall rules to limit connections to real probes only. Something similar holds for **mysql**. Most probes will want access, and passwords can be sniffed. For real security use ssh-tunnels.

2.7. Database

You actually need to create two databases, one for the probes to read from, and one for the results to be written to. The first database should be called upwatch, the other one can be any name (we use netland). Create the databases as follows. (You DO have a root password set for mysql don't you?)

```
$ mysqladmin -u root --password=PASSWORD create upwatch
$ mysql -u root --password=PASSWORD upwatch < upwatch-base.mysql
$ mysqladmin -u root --password=PASSWORD create netland
$ mysql -u root --password=PASSWORD netland < upwatch-full.mysql
```

Of course you need to assign users and GRANT them access. We start with the probes. Mosts probes need read access to their definition table. In many situations you can use just one user for that. Give that user access with:

```
$ mysql -u root --PASSWORD=PASSWORD mysql
mysql> GRANT SELECT ON upwatch.% TO probe@'192.168.1.23' IDENTIFIED BY 'PASSWD'
```

Do this for every host that runs probes.

Next user is for the background processing and the php access:

```
$ mysql -u root --PASSWORD=PASSWORD mysql
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON 'netland'.* TO 'upwatch'@'localhost' IDENTIFIED BY 'PASSWD'
```

Chapter 3. Configuration

3.1. Probe configuration

Each probe first reads the general configuration file `/etc/upwatch.conf` and then its own configurationfile in `/etc/upwatch.d` if it exists. Normally some general things like the debug and logging level, and the database access are specified in the first file, and any probe-specific setting in the second file. You can also override settings from the generic file in the probe-specific file.

Each program has a manual page that documents options. Every long commandine option can also be entered in a configuration file.

3.2. Tailoring uw_sysstat

uw_sysstat is special in that it allows you to scan every (line-oriented) logfile you want. It uses regular expressions to set a yellow or red state. It works as follows:

On startup it reads `/etc/upwatch.d/uw_sysstat.conf`, and searches for **logfile** statements. Say it encounters the statement:

```
logfile errlog /var/log/messages
```

what it does is it reads all files in the directory `/etc/upwatch.d/uw_sysstat.d/errlog` (except `rmacros.txt` and `macros.txt`). These files should contain regular expressions prefixed by one of the keywords **green**, **yellow**, or **red**. Next uw_sysstat starts scanning `/var/log/messages`. It reads a line from the logfile and the following happens:

- Check against the red list. If match found, flag red condition, and send the offending line to the upwatch server
- Check line against the yellow list. If matches, flag yellow and send to server
- Check agains green list. If it matches, ignore this line and go the next line in the logfile. If the current line does not match any of the green list, flag yellow, and send line to server

The regular expressions may (for readability) contain macros, they should be entered in `/etc/upwatch.d/uw_sysstat.d/syslog/macros.txt`.

You can easily add a directory of youw own, containing regular expressions for you own logfiles. In fact upwatch includes a handy utility **chklog** to help you create regular expression lists. Here is an example how to do it. Suppose you plan to scan the logfile for the imaginary 'timtim' navigational system. It resides in `/var/log/timtim.log`.

- First create the directory:

```
# cd /etc/upwatch.d/uw_sysstat.d
# mkdir timtim
# cp syslog/rmacros.txt timtim
# cp syslog/general timtim
# chown -R root:upwatch timtim
# chmod 770 timtim
# chmod 660 timtim/*
```

- Next look at `rmacros.txt` and tailor it to accomodate specifics for the timtim logfile. It might for example contain entries for zipcodes, or latitudes/longitudes for which you would like to create macros.
- Next step: extract regular expressions from an example logfile:

  ```
  # chklog -t timtim -r /var/log/timtim.log | sort -u > /tmp/timtim
  ```

- edit this file. Maybe it will contain dupes, you should try to keep the number of regular expressions low. If you are satisfied you can try copying it to `/etc/upwatch.d/uw_sysstat/timtim` using any filename, and use **chklog** to test it:

  ```
  # chklog -t timtim -m /var/log/timtim.log
  ```

  Now you should only see the lines you want to be reported by uw_sysstat. Repeat steps until you are satisfied

- Finally tell uw_sysstat that you want it to start scanning by adding **errlog timtim /var/log/timtim.log** to its configuration file. That's it.

3.3. Database configuration

First things first. Depending on the size of your installation you may run out of database or record space. It happened to me on the iptraf probe. I was measuring traffic for 4000 IP addresses and ran out of space after a month on the pr_iptraf_raw table - it hit the max_data_length limit. I had to issue the following commands:

```
$ mysql -u root --password=PASSWORD
mysql> alter table pr_iptraf_raw max_rows = 1000000000;
```

and this took almost two hours! So you better look at your own situation and adjust the settings MAX_ROWS and AVG_RECORD_SIZE accordingly for each table.

Chapter 4. Administration

4.1. Logging

The upwatch package contains various ways of logging errors. The standard way is to its own logfile `/var/log/upwatch/upwatch.log`. Other ways are logging to stderr (probably not practical) and to the syslog. Tweak the `debug` to increase the amount of logging. Setting the debug level higher than 2 should only be used for debugging serious problems, for example it causes daemons to stay always in the foreground. In debuglevel 0 only errors are logged, in debuglevel 1 some progress information is logged.

The website has its own logfile in `log/error.log`

4.2. Managing daemons

In most Linux distributions you can start/stop daemons using the scripts in `/etc/init.d`. Don't forget: you will miss sample data in the database if a probe is not running. You can watch what a probe is running if you run **ps ax**.

4.3. Queues

Queues play an important part in upwatch. The queues are so-called `maildir` queues. This means that while the queuefile is written, it is written to a temporary directory, and when it's closed it is hardlinked to the actual queue directory. This way you can be absolutely sure that if you find a file in the queue, it is complete and nobody has the file open. Only one process reads from the queue and deletes the file when done.

Index