urlwatch

Christian Külker

2024-06-19

Contents

1 Introduction

Urlwatch is a handy utility for monitoring changes on web pages and notifying system administrators of updates. It's particularly useful for keeping track of important information on websites that don't provide RSS feeds or other forms of notification. Developed as a script in Python, urlwatch is designed to be simple, yet powerful, offering extensive customization options for system administrators.

With urlwatch, you can:

- Monitor multiple URLs for changes.
- Receive notifications via email or other configurable methods
- Customize the way changes are detected using filters and hooks.

This step-by-step guide will walk you through the installation, configuration, and usage of urlwatch on a Debian system.

2 Installation

Installing urlwatch on a Debian system is straightforward. Follow these steps to get urlwatch up and running:

1. **Update Your System:**

   Before installing any new package, it's a good practice to update your package list to ensure you have the latest information on available packages.

   ```
   aptitude update
   ```

2. **Install urlwatch:**

   Urlwatch can be installed directly from the Debian repositories using the APT package manager.

   ```
   aptitude install urlwatch
   ```

   This will install `/usr/bin/urlwatch` and files in standard locations like `/usr/lib/python3/dist-packages/urlwatch` and `/usr/share/doc/urlwatch`.

3. **Verify the Installation:**

   After the installation is complete, you can verify that urlwatch is installed correctly by checking its version. However unlike the bare evocation of `urlwatch` this step will not create any files.

   ```
   urlwatch --version
   ```

By following these steps, you'll have urlwatch installed on your Debian system, ready to configure and use for monitoring web page changes.

2.1 Basic Configuration

Once you have urlwatch installed, the next step is to configure it to monitor web pages. The basic configuration involves setting up a list of URLs to watch and running the tool to check for changes. Follow these steps for a simple configuration:

1. **Initial Setup:**

   Run one of the following two commands to create the necessary configuration files in your home directory:

   ```
   urlwatch
   ```

```
urlwatch --edit
```

Either of this commands will generate a default configuration for urlwatch located in the following files under the `~/.urlwatch` directory.

- `~/.cache/urlwatch/cache.db`
- `~/.config/urlwatch/urlwatch.yaml`
- `~/.config/urlwatch/urls.yaml`

Other files like `hooks.py` are not created automatically. The `cache.db` is a SQLite3 database. It is possible to use other database, but this is not covered in this section.

While it is nice that there many entries in `urlwatch.yaml` to give ideas what is possible to configure, it is too much for a basic configuration. Therefore the best way for now is either to delete the content and start from scratch or comment out all if not most of its content. This section will not touch the default example configuration content.

2. **Edit the URL List:**

   Open the `urls.yaml` file in your preferred text editor. This file contains the list of URLs you want to monitor. Here's an example of how to add a URL to this file:

   ```
   name: "urlwatch version"
   url: "https://thp.io/2008/urlwatch/"
   filter:
     - html2text
     - grep: "urlwatch-.*.tar.gz"
     - strip
   ---
   ```

   You can add as many URLs as you need, each defined with its own `url` and an optional `name` for identification.

   The strip filter in the configuration is used to remove any leading or trailing whitespace from the lines that remain after the html2text and grep filters are applied. This helps in cleaning up the output by ensuring there are no unnecessary spaces at the beginning or end of each line, making the output more readable and precise.

3. **Run urlwatch:**

   To understand if the configuration is correct, list the jobs:

   ```
   urlwatch --list
   1: urlwatch webpage ( https://thp.io/2008/urlwatch/ )
   ```

Then to understand if the filter is actually filtering, do:

```
urlwatch --test-filter 1
urlwatch-2.28.tar.gz (2022-05-03)
```

After adding your URLs, run urlwatch to start monitoring, first in verbose mode.

First run:

```
urlwatch -v
2024-05-27 15:55:37,292 cli INFO: turning on verbose logging mode
2024-05-27 15:55:37,298 minidb DEBUG: PRAGMA table_info(CacheEntry)
2024-05-27 15:55:37,298 minidb DEBUG: CREATE TABLE CacheEntry (id
↪  INTEGER \
PRIMARY KEY, guid TEXT, timestamp INTEGER, data TEXT, tries INTEGER,
↪  etag \
TEXT)
2024-05-27 15:55:37,328 main INFO: Using \
/home/USER/.config/urlwatch/urls.yaml as URLs file
2024-05-27 15:55:37,328 main INFO: Using \
/home/USER/.config/urlwatch/hooks.py for hooks
2024-05-27 15:55:37,328 main INFO: Using \
/home/USER/.cache/urlwatch/cache.db as cache database
2024-05-27 15:55:37,331 main INFO: Found 1 jobs
2024-05-27 15:55:37,331 worker DEBUG: Processing 1 jobs
2024-05-27 15:55:37,332 handler INFO: Processing: \
<url url='https://thp.io/2008/urlwatch/' name='urlwatch webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']>
2024-05-27 15:55:37,332 minidb DEBUG: SELECT data, timestamp, tries,
↪  etag \
FROM CacheEntry WHERE guid = ? ORDER BY timestamp DESC, tries DESC
↪  LIMIT ? \
['5545f2b73c613bb02228f42413767c5aecac6a8e', 1]
2024-05-27 15:55:37,334 connectionpool DEBUG: Starting new HTTPS \
connection (1): thp.io:443
2024-05-27 15:55:37,408 connectionpool DEBUG: https://thp.io:443
↪  "GET \
/2008/urlwatch/ HTTP/1.1" 200 2462
2024-05-27 15:55:37,409 filters INFO: Applying filter 'html2text', \
subfilter {} to https://thp.io/2008/urlwatch/
2024-05-27 15:55:37,409 filters INFO: Applying filter 'grep',
↪  subfilter \
{'re': 'urlwatch-.*.tar.gz'} to https://thp.io/2008/urlwatch/
2024-05-27 15:55:37,410 filters INFO: Applying filter 'strip',
↪  subfilter \
{} to https://thp.io/2008/urlwatch/
2024-05-27 15:55:37,410 worker DEBUG: Job finished: \
<url url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  \
webpage' filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'},
 ↪  'strip' ]>
2024-05-27 15:55:37,410 worker DEBUG: Using max_tries of 0 for <url \
url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']>
2024-05-27 15:55:37,410 minidb DEBUG: INSERT INTO CacheEntry (guid, \
timestamp, data, tries, etag) VALUES (?, ?, ?, ?, ?) \
['5545f2b73c613bb02228f42413767c5aecac6a8e', 1716818137,
 ↪  'urlwatch-2.
28.tar.gz (2022-05-03)', 0, '"3976154372"']
2024-05-27 15:55:37,419 reporters INFO: Submitting with stdout
 ↪  (<class \
'urlwatch.reporters.StdoutReporter'>)
===========================================================================
01. NEW: urlwatch webpage
===========================================================================

--------------------------------------------------------------------
 ↪  -------
NEW: urlwatch webpage ( https://thp.io/2008/urlwatch/ )
--------------------------------------------------------------------
 ↪  -------



--
urlwatch 2.22, Copyright 2008-2020 Thomas Perl
Website: https://thp.io/2008/urlwatch/
watched 1 URLs in 0 seconds
2024-05-27 15:55:37,420 minidb DEBUG: VACUUM
```

Second run:

This will generate a **NotModifiedError** if the page did not change.

```
urlwatch -v
2024-05-27 15:58:12,047 cli INFO: turning on verbose logging mode
2024-05-27 15:58:12,053 minidb DEBUG: PRAGMA table_info(CacheEntry)
2024-05-27 15:58:12,053 main INFO: Using \
/home/USER/.config/urlwatch/urls.yaml as URLs file
2024-05-27 15:58:12,053 main INFO: Using \
/home/USER/.config/urlwatch/hooks.py for hooks
2024-05-27 15:58:12,053 main INFO: Using \
/home/USER/.cache/urlwatch/cache.db as cache database
2024-05-27 15:58:12,055 main INFO: Found 1 jobs
2024-05-27 15:58:12,055 worker DEBUG: Processing 1 jobs
2024-05-27 15:58:12,055 handler INFO: Processing: \
<url url='https://thp.io/2008/urlwatch/' name='urlwatch webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']>
2024-05-27 15:58:12,055 minidb DEBUG: SELECT data, timestamp, tries,
 ↪  etag \
FROM CacheEntry WHERE guid = ? ORDER BY timestamp DESC, tries DESC
 ↪  LIMIT ? \
['5545f2b73c613bb02228f42413767c5aecac6a8e', 1]
2024-05-27 15:58:12,057 connectionpool DEBUG: Starting new HTTPS \
connection (1): thp.io:443
2024-05-27 15:58:12,134 connectionpool DEBUG: https://thp.io:443
 ↪  "GET \
/2008/urlwatch/ HTTP/1.1" 304 0
2024-05-27 15:58:12,135 worker DEBUG: Job finished: <url \
url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']>
2024-05-27 15:58:12,135 worker DEBUG: Using max_tries of 0 for <url \
url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']>
2024-05-27 15:58:12,135 worker INFO: Job <url \
url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  webpage' \
filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'}, 'strip']> has
 ↪  not
changed (HTTP 304)
2024-05-27 15:58:12,135 handler DEBUG: Got exception while
 ↪  processing \
<url url='https://thp.io/2008/urlwatch/' method='GET' name='urlwatch
 ↪  \
webpage' filter=['html2text', {'grep': 'urlwatch-.*.tar.gz'},
 ↪  'strip']>
Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/urlwatch/handler.py", line
   ↪  113, in \
  process
    data = self.job.retrieve(self)
  File "/usr/lib/python3/dist-packages/urlwatch/jobs.py", line 294,
   ↪  in \
  retrieve
    raise NotModifiedError()
urlwatch.jobs.NotModifiedError
2024-05-27 15:58:12,135 reporters INFO: Submitting with stdout
 ↪  (<class \
'urlwatch.reporters.StdoutReporter'>)
2024-05-27 15:58:12,135 minidb DEBUG: VACUUM
```

The tool will check the listed URLs for changes and output the results to the terminal in colored diff format.

Unfortunately, if you are using this on external web pages that are changing seldom, you will not see the diff output. For this example a new urlwatch release would be needed to be announced on that page. However we can have 3 different outcomes from a urlwatch run:

1. New
2. NotModifiedError aka nothing has changed
3. Changed

The following grep on my web page will for example show a diff output, because I changed the page in-between 2 runs of urlwatch.

```
name: "urlwatch cki"
url: "https://christian.kuelker.info"
filter:
  - html2text
  - grep: "Last modified:.*"
  - strip
---
```

```
urlwatch
===========================================================================
01. CHANGED: urlwatch cki
===========================================================================


--------------------------------------------------------------------
 ↪  -------
CHANGED: urlwatch cki ( https://christian.kuelker.info )
```

```
---------------------------------------------------------------------
↪  -------
--- @   Mon, 27 May 2024 16:46:07 +0200
+++ @   Mon, 27 May 2024 16:46:29 +0200
@@ -1 +1 @@
-Last modified: 2024-02-18
+Last modified: 2024-05-27
---------------------------------------------------------------------
↪  -------
```

However, this would actually not be a good method to understand if a random sub page under https://christian.kuelker.info has changed, as this modified date changes per page update. Therefore it is utmost important to understand what the meaning of parts of the web page actually are. In this case the side bar under "Latest" would update, if there is a new page, but not the page specific modification date.

4. **View Notifications:**

   By default, urlwatch will output the changes directly to the terminal. If there are changes detected on the monitored web pages, they will be displayed in the command output in diff format..

This basic setup will get you started with urlwatch, allowing you to monitor web pages for changes with minimal configuration. For more advanced options, you can explore additional filters and hooks in the configuration files.

3 History

Version	Date	Notes
0.1.0	2024-06-19	Initial release

4 Disclaimer of Warranty

5 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.