User manual

This page documents basic usage of Hydrilla. The instructions assume a POSIX shell and a UNIX-like system are being used.

Installation

Using Python wheel

You can install Hydrilla server and(or) builder using .whl files from the Releases page. Please consider verifying the downloads using provided cryptographic signatures.

Installing dependencies

Python3

Hydrilla requires Python interpreter in at least version 3.7. You'd typically use Python3 as provided by you operating system distribution. For example, on Debian-based systems (including Trisquel) you can install it with:

```
sudo apt install python3
```

pip

Pip is the package manager for Python. While not a direct dependency of Hydrilla, it is needed to utilize .whl files. You most likely also want to install pip as provided by your distro, e.g. for APT-based ones:

```
sudo apt install python3-pip
```

Python libraries

Hydrilla relies on the following Python packages:

- jsonschema
- click
- flask (needed for Hydrilla server only)
- reuse (optional, only needed for Hydrilla builder to generate SPDX report)

If you don't have those dependencies installed, pip will automatically pull them from PyPI (except for reuse which would need to be installed separately with a command like python3 -m pip install reuse).

Nevertheless, you are encouraged to instead install the respective packages from your operating system's official repositories because those usually have stricter policies on stability, security and free licensing. In case of APT-based distributions the packages to install would be python3-jsonschema, python3-click, python3-flask and reuse[1].

Installing Hydrilla

Let's assume you want to install version 1.0 of Hydrilla server. First, download and verify both[2] hydrilla.builder-1.0-py3-none-any.whl and hydrilla-1.0-py3-none-any.whl. Then, run:

```
python3 -m pip install
path/to/downloaded/hydrilla.builder-1.0-py3-none-any.whl path/to/downloaded/hydrilla-1.0-py3-none-
any.whl
```

This will install Hydrilla **for the current user**. The commands hydrilla and hydrilla-builder will be made available in ~/.local/bin/.

Installing in virtualenv

If for example you don't want pip to install things under ~/.local/, you might choose to create a virtual Python environment. First, make sure you have the virtualenv tool installed[3] (for example from APT package python3-virtualenv). Then, choose the folder in which you'd like to install the environment and run:

```
virtualenv -p python3 --system-site-packages path/to/chosen/folder --no-download
```

The --system-site-packages flag is not strictly necessary for it to work but is needed if you want packages inside the virtual environment to be able to see globally-installed dependencies. The --no-download flag instructs the command not to pull some basic tools like wheel and setuptools from PyPI.

Once the environment is created, you need to enter it by sourcing a script created by the virtualenv command, e.g.:

```
source path/to/chosen/folder/bin/activate
```

Afterwards, the python3 -m pip commands you enter in this shell will install packages inside this virtual environment. You can learn more about Python virtual environments from online tutorials and the virtualenv documentation.

Using APT

Hydrilla APT repository is hosted at https://hydrillarepos.koszko.org/apt2/ and is signed with Wojtek's PGP key (fingerprint **E9727060E3C5637C8A4F4B424BC5221C5A79FD1A**). It is expected to work with modern releases of most APT-based distributions (including Debian bullseye and Trisquel nabia).

This APT repository can be used to install Hydrilla server and builder system-wide and to later update the installation. It has to be said that this also requires you to trust Wojtek's repository with your system's safety (a malicious APT repository could easily take over a system that uses it).

If you've decided you want to install the APT repository on your system, the easiest way to do so is by copy-pasting the following script into your POSIX shell (and then confirming with your password). You can of course modify it according to your needs.

```
__install_hydrilla_apt_repo() {
    local TMP="$1"
    local LISTS="$(cat <<EOF
deb     https://hydrillarepos.koszko.org/apt2/ koszko/
deb-src https://hydrillarepos.koszko.org/apt2/ koszko/
EOF
)"

    if ! wget -O "$TMP/koszko-keyring.gpg"
 https://hydrillarepos.koszko.org/apt2/koszko-keyring.gpg; then
    echo "Error! Failed to download keyring file!" >&2
    return 1
    elif ! gpg --no-default-keyring --keyring "$TMP/koszko-keyring.gpg" --list-key
 E9727060E3C5637C8A4F4B424BC5221C5A79FD1A; then
    echo "Error! Invalid keyring file! Someone might be doing something nasty!" >&2
    return 1
    elif ! sudo cp "$TMP/koszko-keyring.gpg" /etc/apt/trusted.gpg.d/; then
    echo "Error!" >&2
    return 1
    elif ! printf %s "$LISTS" | sudo tee /etc/apt/sources.list.d/hydrillarepos.list > /dev/null;
then
    echo "Error!" >&2
    return 1
    fi

    sudo apt-get update
}

install_hydrilla_apt_repo() {
    local TMP="$(mktemp -d)"
```

```
    __install_hydrilla_apt_repo "$TMP"
    local RESULT="$?"

    rm -r "$TMP"

    return "$RESULT"
}

install_hydrilla_apt_repo
```

This snippet is idempotent (i.e. it can be run multiple times and the effect will be as if it was run once). In addition, it executes apt-get update command at the end so that your APT is immediately aware of the new repository and its contents.

After installing the repository you can install Hydrilla builder and server using the following commands:

```
sudo apt install python3-hydrilla.builder
```

```
sudo apt install python3-hydrilla # this alone will also pull the builder as a dependency
```

The packages install their modules under /usr/lib/python3/dist-packages/ which is seen by Python3 interpreters installed from APT. The hydrilla and hydrilla-builder commands get placed in /usr/bin/.

In addition, the python3-hydrilla package also includes sample WSGI script and Apache2 config files for Hydrilla under /usr/share/doc/python3-hydrilla/examples/.

Understanding the concepts

Hydrilla serves Haketilo packages through an HTTP interface, as described in Repository API. It takes the package files to serve from a specific directory in the system as configured by the administrator. The package files stored in that directory conform to Hydrilla on-disk data format. Since that format is inconvenient for humans to operate on, there also exists another one - the Hydrilla source package format. One would typically prepare a Haketilo site resource as a source package and then use the hydrilla-builder command to convert it to Hydrilla on-disk format.

Hydrilla builder takes a directory with a source package, processes it and (if no errors are encountered) writes the "built" package files into the requested directory. Somewhat counterintuitively, the "build" does not involve actual compilation of sources nor any similar task (in future versions of Hydrilla all these will be delegated to other software packaging systems like Guix). Rather, the purpose of this step is to save files under the desired names (which involve files' hash sums) and to generate complete JSON definitions of packages being processed.

The serveable directory can be populated by invoking Hydrilla builder multiple times to put the files of different Haketilo packages in it. However, it is also possible to "build" multiple source packages into separate directories and then combine them. It is up to you, as the administrator, to choose how you are going to manage built packages, Hydrilla doesn't impose anything in this regard. Just keep in mind that there is no facility to remove a package from a serveable directory and that brutally deleting one package's files could break the other ones.

Running

Hydrilla builder

Hydrilla software includes a hydrilla-builder command that can be used to convert Hydrilla source package into its serveable form. The command has an associated manpage (also included in the APT package) as well as a --help option.

Assuming you have both Hydrilla builder and git installed, you can clone and "build" a sample Haketilo package with the following:

```
git clone https://git.koszko.org/hydrilla-source-package-example
mkdir /tmp/tmprepo
hydrilla-builder -s ./hydrilla-source-package-example -d /tmp/tmprepo
```

If you then run:

```
find /tmp/tmprepo/
```

you should see the following list of files written by the builder:

```
/tmp/tmprepo/
/tmp/tmprepo/mapping
/tmp/tmprepo/mapping/helloapple
/tmp/tmprepo/mapping/helloapple/2021.11.10
/tmp/tmprepo/resource
/tmp/tmprepo/resource/hello-message
/tmp/tmprepo/resource/hello-message/2021.11.10
/tmp/tmprepo/resource/helloapple
/tmp/tmprepo/resource/helloapple/2021.11.10
/tmp/tmprepo/file
/tmp/tmprepo/file/sha256
/tmp/tmprepo/file/sha256/cee8d88cf5e5346522fd9ee5e1f994b335fb68d2f460b27b2a2a05f0bcd2b55b
/tmp/tmprepo/file/sha256/a6b9425a80963373d90d8fa22fca07e8626291a4f4bf5f17a4487c16ea1b8dac
/tmp/tmprepo/file/sha256/18dd7d7ac9f74a5ba871791ac6aa4d55530a336ae1b373538b6dd5320b330414
/tmp/tmprepo/file/sha256/a2010f343487d3f7618affe54f789f5487602331c0a8d03f49e9a7c547cf0499
/tmp/tmprepo/file/sha256/fd3d332844be0923b29206b32b9111767d73dd995e3ead7b7f06f72020bce650
/tmp/tmprepo/source
/tmp/tmprepo/source/hello.json
/tmp/tmprepo/source/hello.zip
```

In some cases hydrilla-builder command may fail with a message about the REUSE tool being unavailable. This will only happen when a source package actually requires REUSE (for generation of an SPDX report). The error means that either you don't have it installed or it's somewhere where Hydrilla software cannot see its Python modules.

Hydrilla development server

Hydrilla repository software includes a hydrilla command that can be used to quickly spawn a local repository server. This is unsuitable for deployment of a publicly visible Hydrilla instance but very suitable for testing of both Hydrilla itself and Haketilo packages being developed. The command has an associated manpage (also included in the APT package) as well as a --help option.

For a sample run, you're going to need a directory with some Haketilo packages. You can clone the source package example repository and perform something along the lines of:

```
mkdir /tmp/tmprepo/
hydrilla-builder -s path/to/cloned/hydrilla-source-package-example/ -d /tmp/tmprepo/
```

Then comes a typical invocation of hydrilla command:

```
hydrilla -m /tmp/tmprepo/ -p 0
```

It causes Haketilo packages from /tmp/tmprepo/ directory to be served on a random free port on localhost. Sample run generated this output:

```
 * Serving Flask app "hydrilla.server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
```

```
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:46485/ (Press CTRL+C to quit)
```

One could then (in another shell) try running some commands like the ones below to confirm that the just-spawned local server is responding:

```
# The following assume that Hydrilla is loaded with the sample Haketilo package
curl http://127.0.0.1:46485/mapping/helloapple.json
curl http://127.0.0.1:46485/resource/hello-message/2021.11.10
curl http://127.0.0.1:46485/query?url=https://hydrillabugs.koszko.org/a/b/c
```

Hydrilla as WSGI application under Apache2

This section describes how to configure an Apache2 virtual host to serve a Hydrilla repository. This guide is mostly meant to be useful to people running their own web servers.

You're going to need:

- root access on the machine[4] (for writing to /etc/apache2/sites-available/ directory)
- Apache2 with mod_wsgi installed and enabled
- Hydrilla installed

First, choose a directory where you want to store your serveable Haketilo packages. The default is /var/lib/hydrilla/malcontent. You can override this by saving the following file as /etc/hydrilla/config.json:

```
{
    // Path to directory from which Hydrilla will load packages metadata and serve files.
    "malcontent_dir": "/your/chosen/dir"
}
```

Fill the directory with some package files. You might for example clone the source package example repository and build it with something along the lines of:

```
sudo hydrilla-builder -s path/to/cloned/hydrilla-source-package-example/ -d
 /var/lib/hydrilla/malcontent/
```

Once done, grab Hydrilla's sample WSGI script and save it in your chosen location (the suggested one is /var/lib/hydrilla/wsgi/hydrilla.wsgi). Follow the comments in this script to modify it according to your needs.

Now, get the sample Apache2 configuration (there is also one for TLS-less deployment), also modify it according to your needs (in particular, you'll likely want to change hydrilla.example.com to some real domain of yours) and save under /etc/apache2/sites-available/your.chosen.config.name.conf.

You can now enable the configuration with:

```
sudo a2ensite your.chosen.config.name
```

You also need to reload or restart the Apache daemon for the configuration to be picked up (the command to do that varies between init systems). Once you do so, you can verify that the server is running properly. Consider running something like the following (replacing hydrilla.example.com with the domain name you used):

```
# The following assume that Hydrilla is loaded with the sample Haketilo package
curl http://hydrilla.example.com/mapping/helloapple.json
# -v flag will let us verify that the "Content-Type: application/json" header is present
curl -v http://hydrilla.example.com/resource/hello-message/2021.11.10
curl -v http://hydrilla.example.com/query?url=https://hydrillabugs.koszko.org/a/b/c
```

If everything is working as expected (i.e. JSON documents are properly served by Hydrilla&Apache2), you can start populating the "malcontent directory" with built packages of your choice.

---

1. Reuse tool was first packaged for Debian Bookworm and is not yet available in Debian Bullseye nor in Trisquel Nabia. ↩

2. Hydrilla server also depends on Hydrilla builder. ↩

3. Since Python 3.3 a virtual environment can also be created without this tool. ↩

4. If you want to run a Hydrilla server on shared hosting without root access, this might be achievable using a .htaccess file but is not documented right now. ↩