ArchiveBox

Release 0.7.1

Nick Sweeting

Nov 20, 2023

CONTENTS

1 Documentation **3**

1.1 Intro . . . 3
1.2 Getting Started . . . 19
1.3 General . . . 28
1.4 API Reference . . . 70
1.5 Meta . . . 109

Python Module Index **127**

Index **129**

Welcome to ArchiveBox!

Just getting started?
Check out the Quickstart guide.

Need help with something?
Ping us on Twitter or Github.

Want to join the community?
See our Community Wiki page.

ARCHIVEBOX

"The open-source self-hosted internet archive."

Website | Github | Source | Bug Tracker

```
mkdir my-archive; cd my-archive/
pip install archivebox

archivebox init
archivebox add https://example.com
archivebox info
```

CHAPTER

ONE

DOCUMENTATION

1.1 Intro

Quickstart | Demo | GitHub | Documentation | Info & Motivation | Community | Roadmap

ArchiveBox is a powerful, self-hosted internet archiving solution to collect, save, and view websites offline.

Without active preservation effort, everything on the internet eventually dissapears or degrades. Archive.org does a great job as a free central archive, but they require all archives to be public, and they can't save every type of content.

ArchiveBox is an open source tool that helps you archive web content on your own (or privately within an organization): save copies of browser bookmarks, preserve evidence for legal cases, backup photos from FB / Insta / Flickr, download your media from YT / Soundcloud / etc., snapshot research papers & academic citations, and more...

Use ArchiveBox as a command-line package *and/or* self-hosted web app *on Linux, macOS, or in* Docker.

You can feed ArchiveBox URLs one at a time, or schedule regular imports from browser bookmarks or history, feeds like RSS, bookmark services like Pocket/Pinboard, and more. See input formats for a full list.

It saves snapshots of the URLs you feed it in several redundant formats. It also detects any content featured *inside* each webpage & extracts it out into a folder:

- `HTML/Generic Websites -> HTML, PDF, PNG, WARC, Singlefile`
- `YouTube/SoundCloud/etc. -> MP3/MP4 + subtitles, description, thumbnail`
- `news articles -> article body TXT + title, author, featured images`
- `github/gitlab/etc. links -> git cloned source code`
- and more...

It uses normal filesystem folders to organize archives (no complicated proprietary formats), and offers a CLI + web UI.

ArchiveBox is used by many *professionals and hobbyists* who save content off the web, for example:

- **Individuals:** `backing up browser bookmarks/history`, `saving FB/Insta/etc. content`, `shopping lists`
- **Journalists:** `crawling and collecting research`, `preserving quoted material`, `fact-checking and review`
- **Lawyers:** `evidence collection`, `hashing & integrity verifying`, `search, tagging, & review`
- **Researchers:** `collecting AI training sets`, `feeding analysis / web crawling pipelines`

The goal is to sleep soundly knowing the part of the internet you care about will be automatically preserved in durable, easily accessible formats *for decades* after it goes down.

Get ArchiveBox with `docker` / `apt` / `brew` / `pip3` / `nix` / etc. (*see Quickstart below*).

```
# Get ArchiveBox with Docker or Docker Compose (recommended)
docker run -v $PWD/data:/data -it archivebox/archivebox:dev init --setup

# Or install with your preferred package manager (see Quickstart below for apt, brew,␣
↪and more)
pip3 install archivebox

# Or use the optional auto setup script to install it
curl -sSL 'https://get.archivebox.io' | sh
```

Example usage: adding links to archive.

```
archivebox add 'https://example.com'                                    # add URLs one at␣
↪a time
archivebox add < ~/Downloads/bookmarks.json                             # or pipe in URLs␣
↪in any text-based format
archivebox schedule --every=day --depth=1 https://example.com/rss.xml  # or auto-import␣
↪URLs regularly on a schedule
```

Example usage: viewing the archived content.

```
archivebox server 0.0.0.0:8000            # use the interactive web UI
archivebox list 'https://example.com'     # use the CLI commands (--help for more)
ls ./archive/*/index.json                 # or browse directly via the filesystem
```

1.1.1 Key Features

- **Free & open source**, doesn't require signing up online, stores all data locally
- **Powerful, intuitive command line interface** with *modular optional dependencies*
- **Comprehensive documentation**, active development, and rich community
- **Extracts a wide variety of content out-of-the-box**: *media (yt-dlp), articles (readability), code (git), etc.*
- **Supports scheduled/realtime importing** from *many types of sources*
- ***Uses standard, durable, long-term formats*** like HTML, JSON, PDF, PNG, MP4, TXT, and WARC
- **Usable as a oneshot CLI**, **self-hosted web UI**, Python API (BETA), REST API (ALPHA), or desktop app (ALPHA)
- **Saves all pages to archive.org as well** by default for redundancy (can be disabled for local-only mode)
- Advanced users: support for archiving content requiring login/paywall/cookies (see wiki security caveats!)
- Planned: support for running JS during archiving to adblock, autoscroll, modal-hide, thread-expand

1.1.2 Professional Integration

Contact us if your non-profit institution/org wants to use ArchiveBox professionally.

- setup & support, team permissioning, hashing, audit logging, backups, custom archiving etc.
- for **individuals**, **NGOs**, **academia**, **governments**, **journalism**, **law**, and more…

All our work is open-source and primarily geared towards non-profits.Support/consulting pays for hosting and funds new ArchiveBox open-source development.

1.1.3 Quickstart

Supported OSs: Linux/BSD, macOS, Windows (Docker) **CPUs:** `amd64` (`x86_64`), `arm64` (`arm8`), `arm7` (raspi>=3) Note: On `arm7`, the `playwright` package, provides easy `chromium` management, is not yet available. Do it manually with alternative methods.

Easy Setup

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See setup.sh for the source code of the auto-install script. See "Against curl | sh as an install method" blog post for my thoughts on the shortcomings of this install method.

Package Manager Setup

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See the debian-archivebox repo for more details about this distribution.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See the homebrew-archivebox repo for more details about this distribution.

See below for more usage examples using the CLI, Web UI, or filesystem/SQL/Python to manage your archive. See the pip-archivebox repo for more details about this distribution.

[!WARNING]*These are contributed by external volunteers and may lag behind the official* `pip` *channel.*

Other Options

For more discussion on managed and paid hosting options see here: Issue #531.

Next Steps

- Import URLs from some of the supported *Input Formats* or view the supported *Output Formats*...
- Tweak your UI or archiving behavior *Configuration* or read about some of the *Caveats* and troubleshooting steps...
- Read about the *Dependencies* used for archiving, the Upgrading Process, or the *Archive Layout* on disk...
- Or check out our full *Documentation* or *Community Wiki*...

Usage

CLI Usage

```
# archivebox [subcommand] [--args]
# docker-compose run archivebox [subcommand] [--args]
# docker run -v $PWD:/data -it [subcommand] [--args]

archivebox init --setup      # safe to run init multiple times (also how you update␣
↪versions)
archivebox --version
archivebox help
```

- `archivebox setup/init/config/status/manage` to administer your collection
- `archivebox add/schedule/remove/update/list/shell/oneshot` to manage Snapshots in the archive
- `archivebox schedule` to pull in fresh URLs regularly from *bookmarks/history/Pocket/Pinboard/RSS/etc.*

Web UI Usage

```
archivebox manage createsuperuser  # set an admin password
archivebox server 0.0.0.0:8000     # open http://127.0.0.1:8000 to view it

# you can also configure whether or not login is required for most features
archivebox config --set PUBLIC_INDEX=False
archivebox config --set PUBLIC_SNAPSHOTS=False
archivebox config --set PUBLIC_ADD_VIEW=False
```

SQL/Python/Filesystem Usage

```
sqlite3 ./index.sqlite3    # run SQL queries on your index
archivebox shell           # explore the Python API in a REPL
ls ./archive/*/index.html  # or inspect snapshots on the filesystem
```

1.1.4 Overview

Input Formats

ArchiveBox supports many input formats for URLs, including Pocket & Pinboard exports, Browser bookmarks, Browser history, plain text, HTML, markdown, and more!

Click these links for instructions on how to prepare your links from these sources:

- TXT, RSS, XML, JSON, CSV, SQL, HTML, Markdown, or any other text-based format...
- Browser history or browser bookmarks (see instructions for: Chrome, Firefox, Safari, IE, Opera, and more...)
- Browser extension `archivebox-exporter` (realtime archiving from Chrome/Chromium/Firefox)
- Pocket, Pinboard, Instapaper, Shaarli, Delicious, Reddit Saved, Wallabag, Unmark.it, OneTab, and more...

```
# archivebox add --help
archivebox add 'https://example.com/some/page'
archivebox add < ~/Downloads/firefox_bookmarks_export.html
archivebox add --depth=1 'https://news.ycombinator.com#2020-12-12'
echo 'http://example.com' | archivebox add
echo 'any_text_with [urls](https://example.com) in it' | archivebox add

# if using Docker, add -i when piping stdin:
# echo 'https://example.com' | docker run -v $PWD:/data -i archivebox/archivebox add
# if using Docker Compose, add -T when piping stdin / stdout:
# echo 'https://example.com' | docker compose run -T archivebox add
```

See the Usage: CLI page for documentation and examples.

It also includes a built-in scheduled import feature with `archivebox schedule` and browser bookmarklet, so you can pull in URLs from RSS feeds, websites, or the filesystem regularly/on-demand.

Output Formats

Inside each Snapshot folder, ArchiveBox saves these different types of extractor outputs as plain files:

`./archive/<timestamp>/*`

- **Index:** `index.html` & `index.json` HTML and JSON index files containing metadata and details
- **Title**, **Favicon**, **Headers** Response headers, site favicon, and parsed site title
- **SingleFile:** `singlefile.html` HTML snapshot rendered with headless Chrome using SingleFile
- **Wget Clone:** `example.com/page-name.html` wget clone of the site with `warc/<timestamp>.gz`
- Chrome Headless
  - **PDF:** `output.pdf` Printed PDF of site using headless chrome
  - **Screenshot:** `screenshot.png` 1440x900 screenshot of site using headless chrome
  - **DOM Dump:** `output.html` DOM Dump of the HTML after rendering using headless chrome
- **Article Text:** `article.html/json` Article text extraction using Readability & Mercury
- **Archive.org Permalink:** `archive.org.txt` A link to the saved site on archive.org
- **Audio & Video:** `media/` all audio/video files + playlists, including subtitles & metadata with youtube-dl (or yt-dlp)

- **Source Code:** `git/` clone of any repository found on GitHub, Bitbucket, or GitLab links
- *More coming soon! See the Roadmap...*

It does everything out-of-the-box by default, but you can disable or tweak individual archive methods via environment variables / config.

Configuration

ArchiveBox can be configured via environment variables, by using the `archivebox config` CLI, or by editing the `ArchiveBox.conf` config file directly.

```
archivebox config                               # view the entire config
archivebox config --get CHROME_BINARY           # view a specific value

archivebox config --set CHROME_BINARY=chromium  # persist a config using CLI
# OR
echo CHROME_BINARY=chromium >> ArchiveBox.conf  # persist a config using file
# OR
env CHROME_BINARY=chromium archivebox ...       # run with a one-off config
```

These methods also work the same way when run inside Docker, see the Docker Configuration wiki page for details.

The config loading logic with all the options defined is here: `archivebox/config.py`.

Most options are also documented on the **Configuration Wiki page**.

Most Common Options to Tweak

```
# e.g. archivebox config --set TIMEOUT=120

TIMEOUT=120                # default: 60    add more seconds on slower networks
CHECK_SSL_VALIDITY=True    # default: False True = allow saving URLs w/ bad SSL
SAVE_ARCHIVE_DOT_ORG=False # default: True  False = disable Archive.org saving
MAX_MEDIA_SIZE=1500m       # default: 750m  raise/lower youtubedl output size

PUBLIC_INDEX=True          # default: True  whether anon users can view index
PUBLIC_SNAPSHOTS=True      # default: True  whether anon users can view pages
PUBLIC_ADD_VIEW=False      # default: False whether anon users can add new URLs
```

Dependencies

For better security, easier updating, and to avoid polluting your host system with extra dependencies, **it is strongly recommended to use the official Docker image** with everything pre-installed for the best experience.

To achieve high-fidelity archives in as many situations as possible, ArchiveBox depends on a variety of 3rd-party tools and libraries that specialize in extracting different types of content. These optional dependencies used for archiving sites include:

- `chromium` / `chrome` (for screenshots, PDF, DOM HTML, and headless JS scripts)
- `node` & `npm` (for readability, mercury, and singlefile)
- `wget` (for plain HTML, static files, and WARC saving)

- `curl` (for fetching headers, favicon, and posting to Archive.org)
- `youtube-dl` or `yt-dlp` (for audio, video, and subtitles)
- `git` (for cloning git repos)
- and more as we grow...

You don't need to install every dependency to use ArchiveBox. ArchiveBox will automatically disable extractors that rely on dependencies that aren't installed, based on what is configured and available in your `$PATH`.

If not using Docker, make sure to keep the dependencies up-to-date yourself and check that ArchiveBox isn't reporting any incompatibility with the versions you install.

```
# install python3 and archivebox with your system package manager
# apt/brew/pip/etc install ... (see Quickstart instructions above)

archivebox setup       # auto install all the extractors and extras
archivebox --version   # see info and check validity of installed dependencies
```

Installing directly on **Windows without Docker or WSL/WSL2/Cygwin is not officially supported** (I cannot respond to Windows support tickets), but some advanced users have reported getting it working.

For detailed information about upgrading ArchiveBox and its dependencies, see: https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives

Archive Layout

All of ArchiveBox's state (including the index, snapshot data, and config file) is stored in a single folder called the "ArchiveBox data folder". All `archivebox` CLI commands must be run from inside this folder, and you first create it by running `archivebox init`.

The on-disk layout is optimized to be easy to browse by hand and durable long-term. The main index is a standard `index.sqlite3` database in the root of the data folder (it can also be exported as static JSON/HTML), and the archive snapshots are organized by date-added timestamp in the `./archive/` subfolder.

```
./
    index.sqlite3
    ArchiveBox.conf
    archive/
        ...
        1617687755/
            index.html
            index.json
            screenshot.png
            media/some_video.mp4
            warc/1617687755.warc.gz
            git/somerepo.git
            ...
```

Each snapshot subfolder `./archive/<timestamp>/` includes a static `index.json` and `index.html` describing its contents, and the snapshot extractor outputs are plain files within the folder.

Static Archive Exporting

You can export the main index to browse it statically without needing to run a server.

> **Note** These exports are not paginated, exporting many URLs or the entire archive at once may be slow. Use the filtering CLI flags on the `archivebox list` command to export specific Snapshots or ranges.

```
# archivebox list --help
archivebox list --html --with-headers > index.html     # export to static html table
archivebox list --json --with-headers > index.json     # export to json blob
archivebox list --csv=timestamp,url,title > index.csv  # export to csv spreadsheet

# (if using Docker Compose, add the -T flag when piping)
# docker compose run -T archivebox list --html --filter-type=search snozzberries > index.
↪json
```

The paths in the static exports are relative, make sure to keep them next to your `./archive` folder when backing them up or viewing them.

Caveats

Archiving Private Content

If you're importing pages with private content or URLs containing secret tokens you don't want public (e.g Google Docs, paywalled content, unlisted videos, etc.), **you may want to disable some of the extractor methods to avoid leaking that content to 3rd party APIs or the public**.

```
# don't save private content to ArchiveBox, e.g.:
archivebox add 'https://docs.google.com/document/d/12345somePrivateDocument'
archivebox add 'https://vimeo.com/somePrivateVideo'

# without first disabling saving to Archive.org:
archivebox config --set SAVE_ARCHIVE_DOT_ORG=False  # disable saving all URLs in Archive.
↪org

# restrict the main index, Snapshot content, and Add Page to authenticated users as-
↪needed:
archivebox config --set PUBLIC_INDEX=False
archivebox config --set PUBLIC_SNAPSHOTS=False
archivebox config --set PUBLIC_ADD_VIEW=False

# if extra paranoid or anti-Google:
archivebox config --set SAVE_FAVICON=False          # disable favicon fetching (it calls␣
↪a Google API passing the URL's domain part only)
archivebox config --set CHROME_BINARY=chromium      # ensure it's using Chromium instead␣
↪of Chrome
```

Security Risks of Viewing Archived JS

Be aware that malicious archived JS can access the contents of other pages in your archive when viewed. Because the Web UI serves all viewed snapshots from a single domain, they share a request context and **typical CSRF/CORS/XSS/CSP protections do not work to prevent cross-site request attacks**. See the Security Overview page and Issue #239 for more details.

```
# visiting an archived page with malicious JS:
https://127.0.0.1:8000/archive/1602401954/example.com/index.html

# example.com/index.js can now make a request to read everything from:
https://127.0.0.1:8000/index.html
https://127.0.0.1:8000/archive/*
# then example.com/index.js can send it off to some evil server
```

The admin UI is also served from the same origin as replayed JS, so malicious pages could also potentially use your ArchiveBox login cookies to perform admin actions (e.g. adding/removing links, running extractors, etc.). We are planning to fix this security shortcoming in a future version by using separate ports/origins to serve the Admin UI and archived content (see Issue #239).

Note: Only the `wget` & `dom` extractor methods execute archived JS when viewing snapshots, all other archive methods produce static output that does not execute JS on viewing. If you are worried about these issues ^ you should disable these extractors using `archivebox config --set SAVE_WGET=False SAVE_DOM=False`.

Saving Multiple Snapshots of a Single URL

First-class support for saving multiple snapshots of each site over time will be added eventually (along with the ability to view diffs of the changes between runs). For now **ArchiveBox is designed to only archive each unique URL with each extractor type once**. The workaround to take multiple snapshots of the same URL is to make them slightly different by adding a hash:

```
archivebox add 'https://example.com#2020-10-24'
...
archivebox add 'https://example.com#2020-10-25'
```

The button in the Admin UI is a shortcut for this hash-date workaround.

Storage Requirements

Because ArchiveBox is designed to ingest a firehose of browser history and bookmark feeds to a local disk, it can be much more disk-space intensive than a centralized service like the Internet Archive or Archive.today. **ArchiveBox can use anywhere from ~1gb per 1000 articles, to ~50gb per 1000 articles**, mostly dependent on whether you're saving audio & video using `SAVE_MEDIA=True` and whether you lower `MEDIA_MAX_SIZE=750mb`.

Disk usage can be reduced by using a compressed/deduplicated filesystem like ZFS/BTRFS, or by turning off extractors methods you don't need. You can also deduplicate content with a tool like fdupes or rdfind. **Don't store large collections on older filesystems like EXT3/FAT** as they may not be able to handle more than 50k directory entries in the `archive/` folder. **Try to keep the `index.sqlite3` file on local drive (not a network mount)** or SSD for maximum performance, however the `archive/` folder can be on a network mount or slower HDD.

Screenshots

1.1.5 Background & Motivation

ArchiveBox aims to enable more of the internet to be saved from deterioration by empowering people to self-host their own archives. The intent is for all the web content you care about to be viewable with common software in 50 - 100 years without needing to run ArchiveBox or other specialized software to replay it.

Vast treasure troves of knowledge are lost every day on the internet to link rot. As a society, we have an imperative to preserve some important parts of that treasure, just like we preserve our books, paintings, and music in physical libraries long after the originals go out of print or fade into obscurity.

Whether it's to resist censorship by saving articles before they get taken down or edited, or just to save a collection of early 2010's flash games you love to play, having the tools to archive internet content enables to you save the stuff you care most about before it disappears.

The balance between the permanence and ephemeral nature of content on the internet is part of what makes it beautiful. I don't think everything should be preserved in an automated fashion–making all content permanent and never removable, but I do think people should be able to decide for themselves and effectively archive specific content that they care about.

Because modern websites are complicated and often rely on dynamic content, ArchiveBox archives the sites in **several different formats** beyond what public archiving services like Archive.org/Archive.is save. Using multiple methods and the market-dominant browser to execute JS ensures we can save even the most complex, finicky websites in at least a few high-quality, long-term data formats.

Comparison to Other Projects

Check out our community page for an index of web archiving initiatives and projects.

A variety of open and closed-source archiving projects exist, but few provide a nice UI and CLI to manage a large, high-fidelity archive collection over time.

ArchiveBox tries to be a robust, set-and-forget archiving solution suitable for archiving RSS feeds, bookmarks, or your entire browsing history (beware, it may be too big to store), ~~including private/authenticated content that you wouldn't otherwise share with a centralized service~~ (this is not recommended due to JS replay security concerns).

Comparison With Centralized Public Archives

Not all content is suitable to be archived in a centralized collection, whether because it's private, copyrighted, too large, or too complex. ArchiveBox hopes to fill that gap.

By having each user store their own content locally, we can save much larger portions of everyone's browsing history than a shared centralized service would be able to handle. The eventual goal is to work towards federated archiving where users can share portions of their collections with each other.

Comparison With Other Self-Hosted Archiving Options

ArchiveBox differentiates itself from similar self-hosted projects by providing both a comprehensive CLI interface for managing your archive, a Web UI that can be used either independently or together with the CLI, and a simple on-disk data format that can be used without either.

ArchiveBox is neither the highest fidelity nor the simplest tool available for self-hosted archiving, rather it's a jack-of-all-trades that tries to do most things well by default. It can be as simple or advanced as you want, and is designed to do everything out-of-the-box but be tuned to suit your needs.

If you want better fidelity for very complex interactive pages with heavy JS/streams/API requests, check out ArchiveWeb.page and ReplayWeb.page.

If you want more bookmark categorization and note-taking features, check out Archivy, Memex, Polar, or LinkAce.

If you need more advanced recursive spider/crawling ability beyond `--depth=1`, check out Browsertrix, Photon, or Scrapy and pipe the outputted URLs into ArchiveBox.

For more alternatives, see our list here…

Internet Archiving Ecosystem

Whether you want to learn which organizations are the big players in the web archiving space, want to find a specific open-source tool for your web archiving need, or just want to see where archivists hang out online, our Community Wiki page serves as an index of the broader web archiving community. Check it out to learn about some of the coolest web archiving projects and communities on the web!

- Community Wiki
  - The Master Lists*Community-maintained indexes of archiving tools and institutions.*
  - Web Archiving Software*Open source tools and projects in the internet archiving space.*
  - Reading List*Articles, posts, and blogs relevant to ArchiveBox and web archiving in general.*
  - Communities*A collection of the most active internet archiving communities and initiatives.*
- Check out the ArchiveBox Roadmap and Changelog
- Learn why archiving the internet is important by reading the "On the Importance of Web Archiving" blog post.
- Reach out to me for questions and comments via @ArchiveBoxApp or @theSquashSH on Twitter

Need help building a custom archiving solution?

> **Hire the team that built Archivebox to work on your project.** (@ArchiveBoxApp)

(We also offer general software consulting across many industries)

1.1.6 Documentation

We use the GitHub wiki system and Read the Docs (WIP) for documentation.

You can also access the docs locally by looking in the `ArchiveBox/docs/` folder.

Getting Started

- Quickstart
- Install
- Docker

Reference

- Usage
- Configuration
- Supported Sources
- Supported Outputs
- Scheduled Archiving
- Publishing Your Archive
- Chromium Install
- Security Overview
- Troubleshooting
- Upgrading or Merging Archives
- Python API (alpha)
- REST API (alpha)

More Info

- Tickets
- Roadmap
- Changelog
- Donations
- Background & Motivation
- Web Archiving Community

1.1.7 ArchiveBox Development

All contributions to ArchiveBox are welcomed! Check our issues and Roadmap for things to work on, and please open an issue to discuss your proposed implementation before working on things! Otherwise we may have to close your PR if it doesn't align with our roadmap.

For low hanging fruit / easy first tickets, see: ArchiveBox/Issues `#good first ticket` `#help wanted`.

Python API Documentation: https://docs.archivebox.io/en/dev/archivebox.html#module-archivebox.main

Setup the dev environment

1. Clone the main code repo (making sure to pull the submodules as well)

```
git clone --recurse-submodules https://github.com/ArchiveBox/ArchiveBox
cd ArchiveBox
git checkout dev  # or the branch you want to test
git submodule update --init --recursive
git pull --recurse-submodules
```

2. Option A: Install the Python, JS, and system dependencies directly on your machine

```
# Install ArchiveBox + python dependencies
python3 -m venv .venv && source .venv/bin/activate && pip install -e '.[dev]'
# or: pipenv install --dev && pipenv shell

# Install node dependencies
npm install
# or
archivebox setup

# Check to see if anything is missing
archivebox --version
# install any missing dependencies manually, or use the helper script:
./bin/setup.sh
```

2. Option B: Build the docker container and use that for development instead

```
# Optional: develop via docker by mounting the code dir into the container
# if you edit e.g. ./archivebox/core/models.py on the docker host, runserver
# inside the container will reload and pick up your changes
docker build . -t archivebox
docker run -it \
    -v $PWD/data:/data \
    archivebox init --setup
docker run -it -p 8000:8000 \
    -v $PWD/data:/data \
    -v $PWD/archivebox:/app/archivebox \
    archivebox server 0.0.0.0:8000 --debug --reload
```

(continues on next page)

(continued from previous page)

```
# (remove the --reload flag and add the --nothreading flag when profiling with the␣
↪django debug toolbar)
# When using --reload, make sure any files you create can be read by the user in the␣
↪Docker container, eg with 'chmod a+rX'.
```

Common development tasks

See the `./bin/` folder and read the source of the bash scripts within. You can also run all these in Docker. For more examples see the GitHub Actions CI/CD tests that are run: `.github/workflows/*.yaml`.

Run in DEBUG mode

```
archivebox config --set DEBUG=True
# or
archivebox server --debug ...
```

https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

Install and run a specific GitHub branch

```
# docker-compose.yml:
services:
    archivebox:
        image: archivebox/archivebox:dev
        build: 'https://github.com/ArchiveBox/ArchiveBox.git#dev'
        ...

# docker:
docker build -t archivebox:dev https://github.com/ArchiveBox/ArchiveBox.git#dev
docker run -it -v $PWD:/data archivebox:dev init --setup

# bare metal:
pip install 'git+https://github.com/pirate/ArchiveBox@dev'
npm install 'git+https://github.com/ArchiveBox/ArchiveBox.git#dev'
archivebox init --setup
```

Run the linters

```
./bin/lint.sh
```

(uses `flake8` and `mypy`)

Run the integration tests

```
./bin/test.sh
```

(uses `pytest -s`)

Make migrations or enter a django shell

Make sure to run this whenever you change things in `models.py`.

```
cd archivebox/
./manage.py makemigrations

cd path/to/test/data/
archivebox shell
archivebox manage dbshell
```

(uses `pytest -s`)https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

Contributing a new extractor

ArchiveBox `extractors` are external binaries or Python/Node scripts that ArchiveBox runs to archive content on a page.

Extractors take the URL of a page to archive, write their output to the filesystem `archive/<timestamp>/<extractorname>/...`, and return an `ArchiveResult` entry which is saved to the database (visible on the `Log` page in the UI).

Check out how we added ***`archivebox/extractors/singlefile.py`*** *as an example of the process: Issue #399 + PR #403.*

The process to contribute a new extractor is like this:

1. Open an issue with your propsoed implementation (please link to the pages of any new external dependencies you plan on using)
2. Ensure any dependencies needed are easily installable via a package managers like `apt`, `brew`, `pip3`, `npm` (Ideally, prefer to use external programs available via `pip3` or `npm`, however we do support using any binary installable via package manager that exposes a CLI/Python API and writes output to stdout or the filesystem.)
3. Create a new file in `archivebox/extractors/<extractorname>.py` (copy an existing extractor like `singlefile.py` as a template)
4. Add config settings to enable/disable any new dependencies and the extractor as a whole, e.g. `USE_DEPENDENCYNAME`, `SAVE_EXTRACTORNAME`, `EXTRACTORNAME_SOMEOTHEROPTION` in `archivebox/config.py`
5. Add a preview section to `archivebox/templates/core/snapshot.html` to view the output, and a column to `archivebox/templates/core/index_row.html` with an icon for your extractor
6. Add an integration test for your extractor in `tests/test_extractors.py`
7. Submit your PR for review!
8. Once merged, please document it in these places and anywhere else you see info about other extractors:

- https://github.com/ArchiveBox/ArchiveBox#output-formats
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#archive-method-toggles
- https://github.com/ArchiveBox/ArchiveBox/wiki/Install#dependencies

Build the docs, pip package, and docker image

(Normally CI takes care of this, but these scripts can be run to do it manually)

```
./bin/build.sh

# or individually:
./bin/build_docs.sh
./bin/build_pip.sh
./bin/build_deb.sh
./bin/build_brew.sh
./bin/build_docker.sh
```

Roll a release

(Normally CI takes care of this, but these scripts can be run to do it manually)

```
./bin/release.sh

# or individually:
./bin/release_docs.sh
./bin/release_pip.sh
./bin/release_deb.sh
./bin/release_brew.sh
./bin/release_docker.sh
```

Further Reading

- Home: ArchiveBox.io
- Demo: Demo.ArchiveBox.io
- Docs: Docs.ArchiveBox.io
- Releases: Github.com/ArchiveBox/ArchiveBox/releases
- Wiki: Github.com/ArchiveBox/ArchiveBox/wiki
- Issues: Github.com/ArchiveBox/ArchiveBox/issues
- Discussions: Github.com/ArchiveBox/ArchiveBox/discussions
- Community Chat: Zulip Chat (preferred) or Matrix Chat (old)
- Social Media: Twitter, LinkedIn, YouTube, Alternative.to, Reddit
- Donations: Github.com/ArchiveBox/ArchiveBox/wiki/Donations

(/)

1.2 Getting Started

1.2.1 Quickstart

It only takes about 5 minutes to get up and running with ArchiveBox.

ArchiveBox officially supports **macOS**, **Ubuntu/Debian**, and **BSD**, but likely runs on many other systems. You can run it on any system that supports **Docker** and/or Python. Windows is not officially supported but user have reported success getting it running using Docker, Docker in WSL2, bare WSL/WS2, or batch/powershell (advanced users only).

For more detailed Docker and Docker Compose-specific instructions, see the [[Docker]] page.

1. Set up ArchiveBox

Follow the README Instructions for your platform to get archivebox set up.

2. Get your list of URLs to archive

Follow the links here to find instructions for exporting a list of URLs from each service.

- Pocket
- Pinboard
- Instapaper
- Reddit Saved Posts
- Shaarli
- Unmark.it
- Wallabag
- Chrome Bookmarks
- Firefox Bookmarks
- Safari Bookmarks
- Opera Bookmarks
- Internet Explorer Bookmarks
- Chrome History: `./bin/export-browser-history.sh --chrome`
- Firefox History: `./bin/export-browser-history.sh --firefox`
- Safari History: `./bin/export-browser-history.sh --safari`
- Other File or URL: (e.g. RSS feed url, text file path) pass as second argument in the next step

(If any of these links are broken, please submit an issue and I'll fix it)

3. Add your URLs to the archive

Pass in URLs directly, import a list of links from a file, or import from a feed URL. All via stdin:

```
archivebox add < your_urls.txt

# or if using docker
docker run -v $PWD:/data -it archivebox/archivebox add < your_urls.txt

# or if using docker-compose
docker-compose run archivebox add < your_urls.txt

# any text containing URLs can ingested via stdin or as args
curl https://getpocket.com/users/YOURUSERNAME/feed/all | archivebox add
archivebox add 'https://example.com'
```

Done!

Open `./archive` to view your archive data in the filesystem.

You can also use the interactive Web UI to view/manage/add links to your archive:

```
# with docker:
docker run -v $PWD:/data -it -p 8000:8000 archivebox/archivebox
# or without docker:
archivebox server

open http://127.0.0.1:8000
```

Next Steps:

```
archivebox help   # see info about all the available commands
```

- Read [[Usage]] to learn about the various CLI and web UI functions
- Read [[Configuration]] to learn about the various archive method options
- Read [[Scheduled Archiving]] to learn how to set up automatic daily archiving
- Read [[Publishing Your Archive]] if you want to host your archive for others to access online
- Read [[Troubleshooting]] if you encounter any problems

1.2.2 Install

ArchiveBox only has a few main dependencies apart from `python3`, and they can all be installed using your normal package manager. It usually takes 1min to get up and running if you use the *helper script*, or about 5min if you install everything *manually*.

- *Supported Systems*
- *Dependencies*
- *Automatic Setup*
- *Manual Setup*

- *Docker Setup*

Supported Systems

ArchiveBox officially supports the following operating systems:

- ***macOS:*** >=10.12 (with homebrew)
- ***Linux:*** Ubuntu, Debian, etc (with apt)
- ***BSD:*** FreeBSD, OpenBSD, NetBSD etc (with pkg)

Other systems that are not officially supported but probably work to varying degrees:

- Windows: Via [[Docker]], Docker in WSL2, bare WSL/WSL2, or even directly in batch/powershell with `pip` (if you're adventurous)
- Other Linux distros: Fedora, SUSE, Arch, CentOS, etc.

Platforms other than Linux, BSD, and macOS are untested, but you can probably get it working on them without too much effort.

It's recommended to use a filesystem with compression and/or deduplication abilities (e.g. ZFS or BTRFS) for maximum archive storage efficiency.

You will also need 500MB of RAM (bare minimum), though 2GB or greater recommended. You may be able to reduce the RAM requirements if you disable all the chrome-based archiving methods with `USE_CHROME=False`.

Dependencies

Not all the dependencies are required for all modes. If you disable some archive methods you can avoid those dependencies, for example, if you set `FETCH_MEDIA=False` you don't need to install `youtube-dl`, and if you set `FETCH_[PDF, SCREENSHOT,DOM]=False` you don't need `chromium`.

- `python3 >= 3.7`
- `wget >= 1.16`
- `chromium >= 59` (`google-chrome >= v59` works fine as well)
- `youtube-dl`
- `curl` (usually already on most systems)
- `git` (usually already on most systems)

More info:

- For help installing these, see the *Manual Setup*, [[Troubleshooting]] and [[Chromium Install]] pages.
- To use specific binaries for dependencies, see the Configuration: Dependencies page.
- To disable unwanted dependencies, see the Configuration: Archive Method Toggles page.

Automatic Setup

If you're on Linux with `apt`, or macOS with `brew` there is an automatic setup script provided to install all the dependencies. BSD, Windows, and other OS users should follow the *Manual Setup* or [[Docker]] instructions.

```
# docker or the manual setup are preferred on all platforms now, if you want to use the␣
↪old install script you can run:
curl https://raw.githubusercontent.com/pirate/ArchiveBox/master/bin/setup.sh | sh
```

The script explains what it installs beforehand, and will prompt for user confirmation before making any changes to your system.

After running the setup script, continue with the [[Quickstart]] guide...

Manual Setup

If you don't like running random setup scripts off the internet (:+1:), you can follow these manual setup instructions.

1. Install dependencies

macOS

```
brew tap homebrew-ffmpeg/ffmpeg
brew install homebrew-ffmpeg/ffmpeg/ffmpeg --with-fdk-aac
brew install python3 git wget curl youtube-dl
brew cask install chromium  # Skip this if you already have Google Chrome/Chromium␣
↪installed in /Applications/
```

Ubuntu/Debian

```
apt install python3 python3-pip python3-distutils git wget curl youtube-dl
apt install chromium-browser  # Skip this if you already have Google Chrome/Chromium␣
↪installed
```

BSD

FreeBSD:

```
pkg install python git wget curl youtube-dl
pkg install chromium-browser  # Skip this if you already have Google Chrome/Chromium␣
↪installed
```

OpenBSD:

```
pkg_add python3 git wget curl youtube-dl chromium
```

Install ArchiveBox using pip

```
python3 -m pip install --upgrade archivebox
```

Check that everything worked and the versions are high enough.

```
python3 --version | head -n 1 &&
git --version | head -n 1 &&
wget --version | head -n 1 &&
curl --version | head -n 1 &&
youtube-dl --version | head -n 1 &&
echo "[] All dependencies installed."

archivebox version
```

If you have issues setting up Chromium / Google Chrome, see the [[Chromium Install]] page for more detailed setup instructions.

2. Get your bookmark export file

Follow the [[Quickstart]] guide to download your bookmarks export file containing a list of links to archive.

3. Run archivebox

```
# create a new folder to hold your data and cd into it
mkdir data && cd data
archivebox init
archivebox version
archivebox add < ~/Downloads/bookmarks_export.html
```

You can also use the `update` subcommand to resume the archive update at a specific timestamp `archivebox update --resume=153242424324.123`.

Next Steps

- Read [[Usage]] to learn how to use the ArchiveBox CLI and HTML output
- Read [[Configuration]] to learn about the various archive method options
- Read [[Scheduled Archiving]] to learn how to set up automatic daily archiving
- Read [[Publishing Your Archive]] if you want to host your archive for others to access online
- Read [[Troubleshooting]] if you encounter any problems

Docker Setup

First, if you don't already have docker installed, follow the official install instructions for Linux, macOS, or Windows https://docs.docker.com/install/#supported-platforms.

Then see the [[Docker]] page for next steps.

1.2.3 Docker

Overview

Running ArchiveBox with Docker allows you to manage it in a container without exposing it to the rest of your system. Usage with Docker is similar to usage of ArchiveBox normally, with a few small differences.

Make sure you have Docker installed and set up on your machine before following these instructions. If you don't already have Docker installed, follow the official install instructions for Linux, macOS, or Windows here: https://docs.docker.com/install/#supported-platforms.

- *Overview*
- *Docker Compose* (recommended way)
  - *Setup*
  - Upgrading
  - *Usage*
  - *Accessing the data*
  - *Configuration*
- *Plain Docker*
  - *Setup*
  - Upgrading
  - *Usage*
  - *Accessing the data*
  - *Configuration*

Official Docker Hub image:https://hub.docker.com/r/archivebox/archivebox

Usage:

```
docker run -v $PWD:/data -it archivebox/archivebox init --setup
docker run -v $PWD:/data -it archivebox/archivebox add 'https://example.com'
docker run -v $PWD:/data -p 8000:8000 archivebox/archivebox server 0.0.0.0:8000
```

Docker Compose

An example `docker-compose.yml` config with ArchiveBox and an Nginx server to serve the archive is included in the project root. You can edit it as you see fit, or just run it as it comes out-of-the-box.

Just make sure you have a Docker version that's new enough to support `version: 3` format:

```
docker --version
Docker version 18.09.1, build 4c52b90    # must be >= 17.04.0
```

Setup

```
mkdir archivebox && cd archivebox
curl -O https://raw.githubusercontent.com/ArchiveBox/ArchiveBox/master/docker-compose.yml
docker-compose run archivebox init --setup
docker-compose run archivebox add 'https://example.com'
docker-compose up
```

If you want to use sonic for full text search, download the sonic config file uncomment the sonic service in your `docker-compose.yml` file:

```
curl https://raw.githubusercontent.com/ArchiveBox/ArchiveBox/master/etc/sonic.cfg >␣
↪sonic.cfg
# then uncomment the sonic block in docker-compose.yml

# to backfill previously added snapshots into the full text index, run:
docker-compose run archivebox update --index-only
```

Upgrading

See https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#upgrading-with-docker-compose-%EF%B8%8F

Usage

First, make sure you're `cd`'ed into the same folder as your `docker-compose.yml` file (e.g. the project root) and that your containers have been started with `docker-compose up -d`.

Then open `http://127.0.0.1:8000` or browse `./data/archive` in the filesystem to view the archive.

To add new URLs, you can use docker-compose just like the normal `archivebox <subcommand> [args]` CLI.

To add an individual link or list of links, pass in URLs via stdin.

```
echo "https://example.com" | docker-compose run archivebox add
```

To import links from a file you can either `cat` the file and pass it via stdin like above, or move it into your data folder so that ArchiveBox can access it from within the container.

```
docker-compose run archivebox add 'https://exmaple.com/some/url/here'
docker-compose run archivebox add < ~/Downloads/bookmarks.html
curl https://example.com/some/rss/feed.xml | docker-compose run archivebox add
```

To ingest a feed or remote file and archive all the URLs within, pass the URL or path to the feed or page as an argument using depth=1.

```
docker-compose run archivebox add --depth=1 https://example.com/some/feed.rss
```

The `depth` argument controls if you want to save the links contained in that URL, or only the specified URL.

Accessing the data

The outputted archive data is stored in `data/` (relative to the project root), or whatever folder path you specified in the `docker-compose.yml` `volumes:` section. Make sure the `data/` folder on the host has permissions initially set to `777` so that the ArchiveBox command is able to set it to the specified `OUTPUT_PERMISSIONS` config setting on the first run.

To access your archive, you can open `data/index.html` directly, or you can use the provided Django development server running inside docker on `http://127.0.0.1:8000`.

Configuration

ArchiveBox running with docker-compose accepts all the same environment variables as normal, see the full list on the [[Configuration]] page.

The recommended way to pass in config variables is to edit the `environment:` section in `docker-compose.yml` directly or add an `env_file: ./path/to/ArchiveBox.conf` line before `environment:` to import variables from an env file.

Example of adding config options to `docker-compose.yml`:

```
...

services:
    archivebox:
        ...
        environment:
            - USE_COLOR=False
            - SHOW_PROGRESS=False
            - CHECK_SSL_VALIDITY=False
            - RESOLUTION=1900,1820
            - MEDIA_TIMEOUT=512000
        ...
```

You can also specify an env file via CLI when running compose using `docker-compose --env-file=/path/to/config.env ...` although you must specify the variables in the `environment:` section that you want to have passed down to the ArchiveBox container from the passed env file.

If you want to access your archive server with HTTPS, put a reverse proxy like Nginx or Caddy in front of `http://127.0.0.1:8098` to do SSL termination. You can find many instructions to do this online if you search "SSL reverse proxy".

Docker

Setup

Fetch and run the ArchiveBox Docker image to create your initial archive.

```
echo 'https://example.com' | docker run -i -v $PWD:/data archivebox/archivebox add
```

Replace `~/ArchiveBox` in the command above with the full path to a folder to use to store your archive on the host, or name of a Docker data volume.

Make sure the data folder you use host is either a new, uncreated path, or if it already exists make sure it has permissions initially set to `777` so that the ArchiveBox command is able to set it to the specified `OUTPUT_PERMISSIONS` config setting on the first run.

Upgrading

See https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#upgrading-with-plain-docker

Usage

To add a single URL to the archive or a list of links from a file, pipe them in via stdin. This will archive each link passed in.

```
echo 'https://example.com' | docker run -i -v $PWD:/data archivebox/archivebox add
# or
docker run -i -v $PWD:/data archivebox/archivebox add < bookmarks.html
```

To add a list of pages via feed URL or remote file, pass the URL of the feed as an argument.

```
docker run -it -v $PWD:/data archivebox/archivebox add --depth=1 'https://example.com/
↪some/rss/feed.xml'
```

The `depth` argument controls if you want to save the links contained in that URL, or only the specified URL.

Accessing the data

Using a bind folder

Use the flag:

```
-v /full/path/to/folder/on/host:/data
```

This will use the folder `/full/path/to/folder/on/host` on your host to store the ArchiveBox output.

Configuration

The easiest way is to use the a `.env` file or add your config to your `docker-compose.yml` `environment:` section.

The next easiest way to get/set config is using the archivebox CLI:

```
docker-compose run archivebox config --get RESOLUTION
docker-compose run archivebox config --set RESOLUTION=1440,900
# or
docker run -i -v $PWD:/data archivebox/archivebox config --set MEDIA_TIMEOUT=120
```

ArchiveBox in Docker accepts all the same environment variables as normal, see the list on the [[Configuration]] page.

To set environment variables for a single run, you can use the `env KEY=VAL ...` command, `-e KEY=VAL`, or `--env-file=somefile.env`.

```
echo 'https://example.com' | docker run -i -v $PWD:/data -e FETCH_SCREENSHOT=False␣
↪archivebox/archivebox add
```

```
docker run ... --env-file=ArchiveBox.env archivebox/archivebox ...
```

You can also edit the `data/ArchiveBox.conf` file directly and the changes will take effect on the next run.

1.3 General

1.3.1 Usage

Make sure the dependencies are fully installed before running any ArchiveBox commands.

ArchiveBox API Reference:

- *CLI Usage*: Docs and examples for the ArchiveBox command line interface.
- *Admin UI Usage*: Docs and screenshots for the outputted HTML archive interface.
- *Browser Extension Usage*: Docs and screenshots for the outputted HTML archive interface.
- *Disk Layout*: Description of the archive folder structure and contents.

Related:

- [[Docker]]: Learn about ArchiveBox usage with Docker and Docker Compose
- [[Configuration]]: Learn about the various archive method options
- [[Scheduled Archiving]]: Learn how to set up automatic daily archiving
- [[Publishing Your Archive]]: Learn how to host your archive for others to access
- [[Troubleshooting]]: Resources if you encounter any problems

CLI Usage

All three of these ways of running ArchiveBox are equivalent and interchangeable:

- `archivebox [subcommand] [...args]` *Using the PyPI package via* `pip install archivebox`
- `docker run -it -v $PWD:/data archivebox/archivebox [subcommand] [...args]` *Using the official Docker image*
- `docker-compose run archivebox [subcommand] [...args]` *Using the official Docker image w/ Docker Compose*

You can share a single archivebox data directory between Docker and non-Docker instances as well, allowing you to run the server in a container but still execute CLI commands on the host for example.

For more examples see the [[Docker]] page.

- *Run ArchiveBox with configuration options*
- *Import a single URL*
- *Import a list of URLs from a text file*
- *Import list of links from browser history*

Run ArchiveBox with configuration options

You can set environment variables in your shell profile, a config file, or by using the `env` command.

```
# via the CLI
archivebox config --set TIMEOUT=3600

# by modifying the config file
nano ArchiveBox.conf
# TIMEOUT=3600

# or by using environment variables
env TIMEOUT=3600 archivebox add 'https://example.com'
```

See [[Configuration]] page for more details about the available options and ways to pass config.If you're using Docker, also make sure to read the Configuration section on the [[Docker]] page.

Import a single URL

```
archivebox add 'https://example.com'
# or
echo 'https://example.com' | archivebox add
```

You can also add `--depth=1` to any of these commands if you want to recursively archive the URLs and all URLs one hop away. (e.g. all the outlinks on a page + the page).

Import a list of URLs from a text file

```
cat urls_to_archive.txt | archivebox add
# or
archivebox add < urls_to_archive.txt
# or
curl https://getpocket.com/users/USERNAME/feed/all | archivebox add
```

You can also pipe in RSS, XML, Netscape, or any of the other supported import formats via stdin.

```
archivebox add < ~/Downloads/browser_bookmarks_export.html
# or
archivebox add < ~/Downloads/pinboard_bookmarks.json
# or
archivebox add < ~/Downloads/other_links.txt
```

Import list of links from browser history

Look in the `bin/` folder of this repo to find a script to parse your browser's SQLite history database for URLs. Specify the type of the browser as the first argument, and optionally the path to the SQLite history file as the second argument.

```
./bin/export-browser-history --chrome
archivebox add < output/sources/chrome_history.json
# or
./bin/export-browser-history --firefox
archivebox add < output/sources/firefox_history.json
# or
./bin/export-browser-history --safari
archivebox add < output/sources/safari_history.json
```

UI Usage

```
# configure which areas you want to require login to use vs make publicly available
archivebox config --set PUBLIC_INDEX=False
archivebox config --set PUBLIC_SNAPSHOTS=False
archivebox config --set PUBLIC_ADD_VIEW=False

archivebox manage createsuperuser  # set an admin password to use for any areas␣
↪requiring login
archivebox server 0.0.0.0:8000     # start the archivebox web server

open http://127.0.0.1:8000         # open the admin UI in a browser to view your archive
```

See the Configuration Wiki and Security Wiki for more info…

Or if you prefer to use the static HTML UI instead of the interactive UI provided by the server, you can run `archivebox list --html --with-headers > ./index.html` and then open `./index.html` in a browser. You should see something like this.

You can sort by column, search using the box in the upper right, and see the total number of links at the bottom.

Click the Favicon under the "Files" column to go to the details page for each link.

Explanation of buttons in the web UI - admin snapshots list

A logged-in admin user may perform these operations on one or more snapshots:

- Search Search text in the Snapshot title, URL, tags, or archived content (supports regex with the default ripgrep search backend, or enable the Sonic full-text search backend in `docker-compose.yml` and set `SEARCH_BACKEND_ENGINE=sonic`, `SEARCH_BACKEND_HOST`, `SEARCH_BACKEND_PASSWORD` for full-text fuzzy searching) https://github.com/ArchiveBox/ArchiveBox/issues/956
- Tags - tag or un-tag snapshots
- Title Pull the title (redownload if it was missing, or the title has changed)
- Pull Download missing/failed outputs/extractors methods (pdf, wget… etc). Maybe because download failed or interrupted by a reboot or something. This is the default behavior when you add new URL, they will get pulled automatically. https://github.com/ArchiveBox/ArchiveBox#output-formats
- Re-Snapshot As the name suggests, re-download the page as a separated unique page. Not the same as pull, this one will create a separate entry, and the page is treated as a new URL ending with the date and time #2020-10-24-08:00 https://github.com/ArchiveBox/ArchiveBox#saving-multiple-snapshots-of-a-single-url
- Reset Delete all type of output and redownload them. In the contrary of snapshot, this will overwrite the files.
- Delete Delete a snapshot entirely. This action cannot be undone.

Browser Extension Usage

Get the official `@tjhorner/archivebox-exporter` Browser Extension:

1. Set your Add page to allow non-logged-in access `archivebox config --set PUBLIC_ADD_VIEW=True` (see `PUBLIC_ADD_VIEW` in the wiki)
2. Install the extension in your browser:

- Chrome/Edge/Other Chromium
- Firefox

1. Set the `BASE_URL` in the extension to your ArchiveBox server's URL, e.g. `https://archivebox.example.com:3000`
2. Test it by archiving some pages from your browser and checking `data/logs/*` and `https://archivebox.example.com:3000/admin/core/archiveresult/`

See https://github.com/ArchiveBox/ArchiveBox/issues/577 for more information.

Disk Layout

The `OUTPUT_DIR` folder (usually whatever folder you run the `archivebox` command in), contains the UI HTML and archived data with the structure outlined below.

```
- data/
  - index.sqlite3        # Main index of all archived URLs
  - ArchiveBox.conf      # Main config file in ini format

  - archive/
     - 155243135/        # Archived links are stored in folders by timestamp
        - index.json     # Index/details page for individual archived link
        - index.html

        # Archive method outputs:
        - warc/
        - media/
        - git/
        ...

  - sources/             # Each imported URL list is saved as a copy here
     - getpocket.com-1552432264.txt
     - stdin-1552291774.txt
     ...
```

For more info about ArchiveBox's database/filesystem layout and troubleshooting steps:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#modify-the-archivebox-sqlite3-db-directly
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting

Large Archives

I've found it takes about an hour to download 1000 articles, and they'll take up roughly 1GB.Those numbers are from running it single-threaded on my i5 machine with 50mbps down. YMMV.

Storage requirements go up immensely if you're using `FETCH_MEDIA=True` and are archiving many pages with audio & video.

You can try to run it in parallel by manually splitting your URLs into separate chunks (though this may not work with `database locked` errors on slower filesystems):

```
archivebox add < urls_chunk_1.txt &
archivebox add < urls_chunk_2.txt &
archivebox add < urls_chunk_3.txt &
```

(though this may not be faster if you have a very large collection/main index)

Users have reported running it with 50k+ bookmarks with success (though it will take more RAM while running).

If you already imported a huge list of bookmarks and want to import only new bookmarks, you can use the `ONLY_NEW` environment variable. This is useful if you want to import a bookmark dump periodically and want to skip broken links

which are already in the index.

For more info about troubleshooting filesystem permissions, performance, or issues when running on a NAS:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting

SQL Shell Usage

Explore the SQLite3 DB a bit to see whats available using the SQLite3 shell:

```
cd ~/archivebox
sqlite3 index.sqlite3

# example usage:
SELECT * FROM snapshot;
UPDATE auth_user SET email = 'someNewEmail@example.com' WHERE username =
↪'someUsernameHere';
...
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#-sqlpythonfilesystem-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#modify-the-archivebox-sqlite3-db-directly
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting
- https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

Python Shell Usage

Explore the Python API a bit to see whats available using the archivebox shell:

Python API Documentation: https://docs.archivebox.io/en/master/archivebox.html#module-archivebox.main

```
$ archivebox shell
[i] [2020-09-17 16:57:07] ArchiveBox v0.4.21: archivebox shell
    > /Users/squash/Documents/opt/ArchiveBox/data

# Shell Plus Model Imports
from core.models import Snapshot
from django.contrib.admin.models import LogEntry
from django.contrib.auth.models import Group, Permission, User
from django.contrib.contenttypes.models import ContentType
from django.contrib.sessions.models import Session
# Shell Plus Django Imports
from django.core.cache import cache
from django.conf import settings
from django.contrib.auth import get_user_model
from django.db import transaction
from django.db.models import Avg, Case, Count, F, Max, Min, Prefetch, Q, Sum, When
from django.utils import timezone
from django.urls import reverse
```

(continues on next page)

(continued from previous page)

```
from django.db.models import Exists, OuterRef, Subquery
# ArchiveBox Imports
from archivebox.core.models import Snapshot, User
from archivebox import *
    help
    version
    init
    config
    add
    remove
    update
    list
    shell
    server
    status
    manage
    oneshot
    schedule

[i] Welcome to the ArchiveBox Shell!
    https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#Shell-Usage
    https://docs.archivebox.io/en/master/archivebox.html#module-archivebox.main

    Hint: Example use:
        print(Snapshot.objects.filter(is_archived=True).count())
        Snapshot.objects.get(url="https://example.com").as_json()
        add("https://example.com/some/new/url")

# run Python API queries/function calls directly
>>> print(Snapshot.objects.filter(is_archived=True).count())
24

# get help info on an object or function
>>> help(Snapshot)
...

# show raw SQL queries run
>>> from django.db import connection
>>> print(connection.queries)
```

For more info and example usage:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#example-adding-a-new-user-with-a-hashed-password
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/main.py
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/config.py
- https://github.com/ArchiveBox/ArchiveBox/blob/dev/archivebox/core/models.py
- https://stackoverflow.com/questions/1074212/how-can-i-see-the-raw-sql-queries-django-is-running

Python API Usage

You can interact with ArchiveBox as a Python library from external scripts or programs.

For example you could create and a script `add_archivebox_url.py` like so:

```
import os
DATA_DIR = '~/archivebox/data'
os.chdir(DATA_DIR)

# you must import and setup django first to establish a DB connection
from archivebox.config import setup_django
setup_django()

# then you can import all the main functions
from archivebox.main import add, remove, server

add('https://example.com', index_only=True, out_dir=DATA_DIR)
remove(...)
server(...)
...
```

For more information see:

- ArchiveBox Python API Reference (ReadTheDocs)
- ArchiveBox Developer Documentation
- ArchiveBox Python source code

1.3.2 Configuration

The full ArchiveBox config file definition with defaults can be found here: `archivebox/config.py`.

Configuration of ArchiveBox is done by using the `archivebox config` command, modifying the `ArchiveBox.conf` file in the data folder, or by using environment variables. All three methods work equivalently when using Docker as well.

Some equivalent examples of setting some configuration options:

```
archivebox config --set CHROME_BINARY=google-chrome-stable
# OR
echo "CHROME_BINARY=google-chrome-stable" >> ArchiveBox.conf
# OR
env CHROME_BINARY=google-chrome-stable archivebox add ~/Downloads/bookmarks_export.html
```

Environment variables take precedence over the config file, which is useful if you only want to use a certain option temporarily during a single run.

Available Configuration Options:

- *General Settings:* Archiving process, output format, and timing.
- *Archive Method Toggles:* On/off switches for methods.
- *Archive Method Options:* Method tunables and parameters.
- *Shell Options:* Format & behavior of CLI output.

- *Dependency Options:* Specify exact paths to dependencies.

In case this document is ever out of date, it's recommended to read the code that loads the config directly in `archivebox/config.py`.

General Settings

General options around the archiving process, output format, and timing.

OUTPUT_PERMISSIONS

Possible Values: [`755`]/`644`/...Permissions to set the output directory and file contents to.

This is useful when running ArchiveBox inside Docker as root and you need to explicitly set the permissions to something that the users on the host can access.

ONLY_NEW

Possible Values: [`True`]/`False`Toggle whether or not to attempt rechecking old links when adding new ones, or leave old incomplete links alone and only archive the new links.

By default, ArchiveBox will only archive new links on each import. If you want it to go back through all links in the index and download any missing files on every run, set this to `False`.

Note: Regardless of how this is set, ArchiveBox will never re-download sites that have already succeeded previously. When this is `False` *it only attempts to fix previous pages have* missing *archive extractor outputs, it does not re-archive pages that have already been successfully archived.*

TIMEOUT

Possible Values: [`60`]/`120`/...Maximum allowed download time per archive method for each link in seconds. If you have a slow network connection or are seeing frequent timeout errors, you can raise this value.

Note: Do not set this to anything less than `15` *seconds as it will cause Chrome to hang indefinitely and many sites to fail completely.*

MEDIA_TIMEOUT

Possible Values: [3600]/120/...Maximum allowed download time for fetching media when SAVE_MEDIA=True in seconds. This timeout is separate and usually much longer than TIMEOUT because media downloaded with youtube-dl can often be quite large and take many minutes/hours to download. Tweak this setting based on your network speed and maximum media file size you plan on downloading.

Note: Do not set this to anything less than 10 seconds as it can often take 5-10 seconds for youtube-dl just to parse the page before it starts downloading media files.

Related options:SAVE_MEDIA

ADMIN_USERNAME / ADMIN_PASSWORD

Possible Values: [None]/"admin"/...

Only used on first run / initial setup. ArchiveBox will create an admin user with the specified username and password when these options are found in the environment. Useful for setting up a Docker instance of ArchiveBox without needing to run a shell command to create the admin user.

Related options:PUBLIC_INDEX

PUBLIC_INDEX / PUBLIC_SNAPSHOTS / PUBLIC_ADD_VIEW

Possible Values: [False]/True Configure whether or not login is required to use each area of ArchiveBox.

```
archivebox manage createsuperuser  # set a password before disabling public access

archivebox config --set PUBLIC_INDEX=False
archivebox config --set PUBLIC_SNAPSHOTS=False
archivebox config --set PUBLIC_ADD_VIEW=False
```

https://github.com/ArchiveBox/ArchiveBox#-web-ui-usage

CUSTOM_TEMPLATES_DIR

Possible Values: [None]/./path/to/custom_templates/...Path to a directory containing custom html/css/images for overriding the default UI styling. Files found in the folder at the specified path can override any of the defaults in the TEMPLATES_DIR directory (copy files from that default dir into your custom dir to get started making a custom theme).

If you've used django before, this works exactly the same way that django template overrides work (because it uses django under the hood).

Related options:FOOTER_INFO

SNAPSHOTS_PER_PAGE

Possible Values: [40]/100/...

Maximum number of Snapshots to show per page on Snapshot list pages. Lower this value on slower machines to make the UI faster.

Related options:SEARCH_BACKEND_TIMEOUT

FOOTER_INFO

Possible Values: [Content is hosted for personal archiving purposes only. Contact server owner for any takedown requests.]/Operated by ACME Co./...Some text to display in the footer of the archive index. Useful for providing server admin contact info to respond to takedown requests.

Related options:TEMPLATES_DIR

URL_BLACKLIST

Possible Values: [\.(css|js|otf|ttf|woff|woff2|gstatic\.com|googleapis\.com/css)(\?.*)?$]/.+\.exe$/http(s)?:\/\/(.+)?example.com\/.*/...

A regex expression used to exclude certain URLs from archiving. You can use if there are certain domains, extensions, or other URL patterns that you want to ignore whenever they get imported. Blacklisted URLs wont be included in the index, and their page content wont be archived.

When building your exclusion list, you can check whether a given URL matches your regex expression in `python` like so:

```
>>> import re
>>> URL_BLACKLIST = r'^http(s)?:\/\/(.+\.)?(youtube\.com)|(amazon\.com)\/.*$'  # replace␣
↪this with your regex to test
>>> URL_BLACKLIST_PTN = re.compile(URL_BLACKLIST, re.IGNORECASE | re.UNICODE | re.
↪MULTILINE)

>>> bool(URL_BLACKLIST_PTN.search('https://test.youtube.com/example.php?abc=123'))  #␣
↪replace this with the URL to test
True   # this URL would not be archived because it matches the exclusion pattern
```

Note: all assets required to render each page are still archived, URL_BLACKLIST/URL_WHITELIST do not apply to images, css, video, etc. visible inline within the page.

Note 2: I named these options poorly years ago when I added them and I plan to rename them to `URL_ALLOWLIST` & `URL_DENYLIST` in a future release.

Related options:URL_WHITELIST, SAVE_MEDIA, SAVE_GIT, GIT_DOMAINS

URL_WHITELIST

Possible Values: [None]/^http(s)?:\/\/(.+)?example\.com\/?.*$/...

A regex expression used to exclude all URLs that don't match the given pattern from archiving. You can use if there are certain domains, extensions, or other URL patterns that you want to restrict the scope of archiving to (e.g. to only archive a single domain, subdirectory, or filetype, etc..

When building your whitelist, you can check whether a given URL matches your regex expression in `python` like so:

```
>>> import re
>>> URL_WHITELIST = r'^http(s)?:\/\/(.+)?example\.com\/?.*$'  # replace this with your␣
↪regex to test
>>> URL_WHITELIST_PTN = re.compile(URL_WHITELIST, re.IGNORECASE | re.UNICODE | re.
↪MULTILINE)

>>> bool(URL_WHITELIST_PTN.search('https://test.example.com/example.php?abc=123'))
True      # this URL would be archived

>>> bool(URL_WHITELIST_PTN.search('https://test.youtube.com/example.php?abc=123'))
False     # this URL would be excluded from archiving
```

This option is useful for **recursive archiving** of all the pages under a given domain or subfolder (aka crawling/spidering), without following links to external domains / parent folders.

```
# temporarily enforce a whitelist by setting the option as an environment variable
export URL_WHITELIST='^http(s)?:\/\/(.+)?example\.com\/?.*$'

# then run your archivebox commands in the same shell
archivebox add --depth=1 'https://example.com'
archivebox list https://example.com | archivebox add --depth=1
archivebox list https://example.com | archivebox add --depth=1
archivebox list https://example.com | archivebox add --depth=1   # repeat up to desired␣
↪depth
...
# all URLs that don't match *.example.com will be excluded, e.g. a link to youtube.com␣
↪would not be followed
```

Note: all assets required to render each page are still archived, URL_BLACKLIST/URL_WHITELIST do not apply to images, css, video, etc. visible inline within the page.

Related options:URL_BLACKLIST, SAVE_MEDIA, SAVE_GIT, GIT_DOMAINS

Archive Method Toggles

High-level on/off switches for all the various methods used to archive URLs.

SAVE_TITLE

Possible Values: [True]/FalseBy default ArchiveBox uses the title provided by the import file, but not all types of imports provide titles (e.g. Plain texts lists of URLs). When this is True, ArchiveBox downloads the page (and follows all redirects), then it attempts to parse the link's title from the first <title></title> tag found in the response. It may be buggy or not work for certain sites that use JS to set the title, disabling it will lead to links imported without a title showing up with their URL as the title in the UI.

Related options:ONLY_NEW, *CHECK_SSL_VALIDITY*

SAVE_FAVICON

Possible Values: [True]/FalseFetch and save favicon for the URL from Google's public favicon service: `https://www.google.com/s2/favicons?domain={domain}`. Set this to `FALSE` if you don't need favicons.

Related options:TEMPLATES_DIR, *CHECK_SSL_VALIDITY*, *CURL_BINARY*

SAVE_WGET

Possible Values: [True]/FalseFetch page with wget, and save responses into folders for each domain, e.g. `example.com/index.html`, with `.html` appended if not present. For a full list of options used during the `wget` download process, see the `archivebox/archive_methods.py:save_wget(...)` function.

Related options:TIMEOUT, *SAVE_WGET_REQUISITES*, *CHECK_SSL_VALIDITY*, *COOKIES_FILE*, *WGET_USER_AGENT*, *SAVE_WARC*, *WGET_BINARY*

SAVE_WARC

Possible Values: [True]/FalseSave a timestamped WARC archive of all the page requests and responses during the wget archive process.

Related options:TIMEOUT, *SAVE_WGET_REQUISITES*, *CHECK_SSL_VALIDITY*, *COOKIES_FILE*, *WGET_USER_AGENT*, *SAVE_WGET*, *WGET_BINARY*

SAVE_PDF

Possible Values: [True]/FalsePrint page as PDF.

Related options:TIMEOUT, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_BINARY*

SAVE_SCREENSHOT

Possible Values: [True]/FalseFetch a screenshot of the page.

Related options:RESOLUTION, *TIMEOUT*, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_BINARY*

SAVE_DOM

Possible Values: [True]/FalseFetch a DOM dump of the page.

Related options:TIMEOUT, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_BINARY*

SAVE_SINGLEFILE

Possible Values: [True]/FalseFetch an HTML file with all assets embedded using Single File.

Related options:TIMEOUT, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_BINARY*, *SINGLEFILE_BINARY*

SAVE_READABILITY

Possible Values: [True]/FalseExtract article text, summary, and byline using Mozilla's Readability library. Unlike the other methods, this does not download any additional files, so it's practically free from a disk usage perspective. It works by using any existing downloaded HTML version (e.g. wget, DOM dump, singlefile) and piping it into readability.

Related options:TIMEOUT, *SAVE_WGET*, *SAVE_DOM*, *SAVE_SINGLEFILE*, *SAVE_MERCURY*

SAVE_MERCURY

Possible Values: [True]/FalseExtract article text, summary, and byline using the Mercury library. Unlike the other methods, this does not download any additional files, so it's practically free from a disk usage perspective. It works by using any existing downloaded HTML version (e.g. wget, DOM dump, singlefile) and piping it into Mercury.

Related options:TIMEOUT, *SAVE_WGET*, *SAVE_DOM*, *SAVE_SINGLEFILE*, *SAVE_READABILITY*

SAVE_GIT

Possible Values: [True]/FalseFetch any git repositories on the page.

Related options:TIMEOUT, *GIT_DOMAINS*, *CHECK_SSL_VALIDITY*, *GIT_BINARY*

SAVE_MEDIA

Possible Values: [True]/FalseFetch all audio, video, annotations, and media metadata on the page using youtube-dl. Warning, this can use up *a lot* of storage very quickly.

Related options:MEDIA_TIMEOUT, *CHECK_SSL_VALIDITY*, *YOUTUBEDL_BINARY*

SAVE_ARCHIVE_DOT_ORG

Possible Values: [True]/FalseSubmit the page's URL to be archived on Archive.org. (The Internet Archive)

Related options:TIMEOUT, *CHECK_SSL_VALIDITY*, *CURL_BINARY*

Archive Method Options

Specific options for individual archive methods above. Some of these are shared between multiple archive methods, others are specific to a single method.

CHECK_SSL_VALIDITY

Possible Values: [True]/FalseWhether to enforce HTTPS certificate and HSTS chain of trust when archiving sites. Set this to False if you want to archive pages even if they have expired or invalid certificates. Be aware that when False you cannot guarantee that you have not been man-in-the-middle'd while archiving content, so the content cannot be verified to be what's on the original site.

SAVE_WGET_REQUISITES

Possible Values: [True]/FalseFetch images/css/js with wget. (True is highly recommended, otherwise your won't download many critical assets to render the page, like images, js, css, etc.)

Related options:TIMEOUT, *SAVE_WGET*, *SAVE_WARC*, *WGET_BINARY*

RESOLUTION

Possible Values: [1440,2000]/1024,768/...Screenshot resolution in pixels width,height.

Related options:SAVE_SCREENSHOT

CURL_USER_AGENT

Possible Values: [Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.61 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/) curl/{CURL_VERSION}]/"Mozilla/5.0 ..."/...This is the user agent to use during curl archiving. You can set this to impersonate a more common browser like Chrome or Firefox if you're getting blocked by servers for having an unknown/blacklisted user agent.

Related options:USE_CURL, *SAVE_TITLE*, *CHECK_SSL_VALIDITY*, *CURL_BINARY*, *WGET_USER_AGENT*, *CHROME_USER_AGENT*

WGET_USER_AGENT

Possible Values: [Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.61 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/) wget/{WGET_VERSION}]/"Mozilla/5.0 ..."/...This is the user agent to use during wget archiving. You can set this to impersonate a more common browser like Chrome or Firefox if you're getting blocked by servers for having an unknown/blacklisted user agent.

Related options:SAVE_WGET, *SAVE_WARC*, *CHECK_SSL_VALIDITY*, *WGET_BINARY*, *CHROME_USER_AGENT*

CHROME_USER_AGENT

Possible Values: [Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.61 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)]/"Mozilla/5.0 ..."/...

This is the user agent to use during Chrome headless archiving. If you're experiencing being blocked by many sites, you can set this to hide the `Headless` string that reveals to servers that you're using a headless browser.

Related options:SAVE_PDF, *SAVE_SCREENSHOT*, *SAVE_DOM*, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_HEADLESS*, *CHROME_BINARY*, *WGET_USER_AGENT*

GIT_DOMAINS

Possible Values: [github.com,bitbucket.org,gitlab.com]/git.example.com/...Domains to attempt download of git repositories on using `git clone`.

Related options:SAVE_GIT, *CHECK_SSL_VALIDITY*

COOKIES_FILE

Possible Values: [None]/path/to/cookies.txt/...Cookies file to pass to wget. To capture sites that require a user to be logged in, you can specify a path to a netscape-format `cookies.txt` file for wget to use. You can generate this file by using a browser extension to export your cookies in this format, or by using wget with `--save-cookies`.

Related options:SAVE_WGET, *SAVE_WARC*, *CHECK_SSL_VALIDITY*, *WGET_BINARY*

CHROME_USER_DATA_DIR

Possible Values: [~/.config/google-chrome]/tmp/chrome-profile/...Path to a Chrome user profile directory. To capture sites that require a user to be logged in, you can specify a path to a chrome user profile (which loads the cookies needed for the user to be logged in). If you don't have an existing Chrome profile, create one with `chromium-browser --user-data-dir=/tmp/chrome-profile`, and log into the sites you need. Then set `CHROME_USER_DATA_DIR=/tmp/chrome-profile` to make ArchiveBox use that profile.

Note: Make sure the path does not have `Default` at the end (it should the the parent folder of `Default`), e.g. set it to `CHROME_USER_DATA_DIR=~/.config/chromium` and not `CHROME_USER_DATA_DIR=~/.config/chromium/Default`.

By default when set to `None`, ArchiveBox tries all the following User Data Dir paths in order:https://chromium.googlesource.com/chromium/src/+/HEAD/docs/user_data_dir.md

Related options:SAVE_PDF, *SAVE_SCREENSHOT*, *SAVE_DOM*, *CHECK_SSL_VALIDITY*, *CHROME_HEADLESS*, *CHROME_BINARY*

CHROME_HEADLESS

Possible Values: [True]/FalseWhether or not to use Chrome/Chromium in `--headless` mode (no browser UI displayed). When set to `False`, the full Chrome UI will be launched each time it's used to archive a page, which greatly slows down the process but allows you to watch in real-time as it saves each page.

Related options:SAVE_PDF, *SAVE_SCREENSHOT*, *SAVE_DOM*, *CHROME_USER_DATA_DIR*, *CHROME_BINARY*

CHROME_SANDBOX

Possible Values: [True]/FalseWhether or not to use the Chrome sandbox when archiving.

If you see an error message like this, it means you are trying to run ArchiveBox as root:

```
:ERROR:zygote_host_impl_linux.cc(89)] Running as root without --no-sandbox is not␣
↪supported. See https://crbug.com/638180
```

*Note: **Do not run ArchiveBox as root!** The solution to this error is not to override it by setting `CHROME_SANDBOX=False`, it's to use create another user (e.g. `www-data`) and run ArchiveBox under that new, less privileged user. This is a security-critical setting, only set this to `False` if you're running ArchiveBox inside a container or VM where it doesn't have access to the rest of your system!

Related options: *SAVE_PDF*, *SAVE_SCREENSHOT*, *SAVE_DOM*, *CHECK_SSL_VALIDITY*, *CHROME_USER_DATA_DIR*, *CHROME_HEADLESS*, *CHROME_BINARY*

Shell Options

Options around the format of the CLI output.

USE_COLOR

Possible Values: [`True`]/`False`Colorize console output. Defaults to `True` if stdin is a TTY (interactive session), otherwise `False` (e.g. if run in a script or piped into a file).

SHOW_PROGRESS

Possible Values: [`True`]/`False`Show real-time progress bar in console output. Defaults to `True` if stdin is a TTY (interactive session), otherwise `False` (e.g. if run in a script or piped into a file).

Note: We use asymptotic progress bars because most tasks complete quickly!

Dependency Options

Options for defining which binaries to use for the various archive method dependencies.

CHROME_BINARY

Possible Values: [`chromium-browser`]/`/usr/local/bin/google-chrome/...`Path or name of the Google Chrome / Chromium binary to use for all the headless browser archive methods.

Without setting this environment variable, ArchiveBox by default look for the following binaries in `$PATH` in this order:

- `chromium-browser`
- `chromium`
- `google-chrome`
- `google-chrome-stable`
- `google-chrome-unstable`
- `google-chrome-beta`
- `google-chrome-canary`
- `google-chrome-dev`

You can override the default behavior to search for any available bin by setting the environment variable to your preferred Chrome binary name or path.

The chrome/chromium dependency is *optional* and only required for screenshots, PDF, and DOM dump output, it can be safely ignored if those three methods are disabled.

Related options: *SAVE_PDF*, *SAVE_SCREENSHOT*, *SAVE_DOM*, *SAVE_SINGLEFILE*, *CHROME_USER_DATA_DIR*, *CHROME_HEADLESS*, *CHROME_SANDBOX*

WGET_BINARY

Possible Values: [wget]/usr/local/bin/wget/...Path or name of the wget binary to use.

Related options: *SAVE_WGET*, *SAVE_WARC*

YOUTUBEDL_BINARY

Possible Values: [youtube-dl]/usr/local/bin/youtube-dl/...Path or name of the youtube-dl binary to use.

Related options: *SAVE_MEDIA*

GIT_BINARY

Possible Values: [git]/usr/local/bin/git/...Path or name of the git binary to use.

Related options: *SAVE_GIT*

CURL_BINARY

Possible Values: [curl]/usr/local/bin/curl/...Path or name of the curl binary to use.

Related options: *SAVE_FAVICON*, *SAVE_ARCHIVE_DOT_ORG*

SINGLEFILE_BINARY

Possible Values: [single-file]/./node_modules/single-file/cli/single-file/...Path or name of the SingleFile binary to use.

This can be installed using `npm install --no-audit --no-fund 'git+https://github.com/gildas-lormeau/SingleFile.git'`.

Related options: *SAVE_SINGLEFILE*, *CHROME_BINARY*, *CHROME_USER_DATA_DIR*, *CHROME_HEADLESS*, *CHROME_SANDBOX*

READABILITY_BINARY

Possible Values: [readability-extractor]/./node_modules/readability-extractor/readability-extractor/...Path or name of the Readability extrator binary to use.

This can be installed using `npm install --no-audit --no-fund 'git+https://github.com/ArchiveBox/readability-extractor.git'`.

Related options:SAVE_READABILITY

MERCURY_BINARY

Possible Values: [mercury-parser]/./node_modules/@postlight/mercury-parser/cli.js/...Path or name of the Mercury parser extractor binary to use.

This can be installed using `npm install --no-audit --no-fund '@postlight/mercury-parser'`.

Related options:SAVE_MERCURY

RIPGREP_BINARY

Possible Values: [rg]/rga/...

Path or name of the ripgrep binary to use for full text search.

This can be installed using your system package manager, e.g. `apt install ripgrep` or `brew install ripgrep`.

Optionally switch this to use `ripgrep-all` for full-text search support across more filetypes (e.g. PDF): https://github.com/phiresky/ripgrep-all.

Related options:SEARCH_BACKEND_ENGINE

SINGLEFILE_ARGS

Possible Values: [["--back-end=playwright-firefox","--load-deferred-images-dispatch-scroll-event=true"]]/..

Arguments that are passed to the SingleFile binary. The values should be a valid JSON string.

Related options:SINGLEFILE_BINARY

CURL_ARGS

Possible Values: [["--tlsv1.3","--http2"]]/..

Arguments that are passed to the curl binary. The values should be a valid JSON string.

Related options:CURL_BINARY

WGET_ARGS

Possible Values: `[["--https-only"]]`/..

Arguments that are passed to the wget binary. The values should be a valid JSON string.

Related options:WGET_BINARY

YOUTUBEDL_ARGS

Possible Values: `[["--limit-rate=10M"]]`/..

Arguments that are passed to the youtube-dl binary. The values should be a valid JSON string.

Related options:YOUTUBEDL_BINARY

GIT_ARGS

Possible Values: `[["--depth=1"]]`

Arguments that are passed to the `git clone` subcommand. The values should be a valid JSON string.

Related options:GIT_BINARY

1.3.3 Troubleshooting

If you need help or have a question, you can open an issue or reach out on Twitter.

What are you having an issue with?:

- *Installing ArchiveBox*
- Upgrading ArchiveBox
- Configuring ArchiveBox
- *Archiving content with ArchiveBox*
- *Hosting your collection publicly*
- Database and filesystem issues

Installing

Make sure you've followed the Manual Setup guide in the [[Install]] instructions first. Then check here for help depending on what component you need help with:

Python

Make sure you have at least Python 3.9 installed on your system.

```
python3 --version
pip install --upgrade pip setuptools
```

If you still need help getting Python installed, the official Python docs are a good place to start.

Chromium/Google Chrome

For more info, see the [[Chromium Install]] page.

ArchiveBox depends on being able to access a `chromium-browser`/`google-chrome` executable. The executable used defaults to `chromium-browser` but can be manually specified with the environment variable `CHROME_BINARY`:

```
env CHROME_BINARY=/usr/local/bin/chromium-browser archivebox add ~/Downloads/bookmarks_
↪export.html
```

1. Test to make sure you have Chrome on your `$PATH` with:

```
which chromium-browser || which google-chrome
```

If no executable is displayed, follow the setup instructions to install and link one of them.

1. If a path is displayed, the next step is to check that it's runnable:

```
chromium-browser --version || google-chrome --version
```

If no version is displayed, try the setup instructions again, or confirm that you have permission to access chrome.

1. If a version is displayed and it's <59, upgrade it:

```
apt upgrade chromium-browser -y
# OR
brew cask upgrade chromium-browser
```

1. If a version is displayed and it's >=59, make sure ArchiveBox is running the right one:

```
env CHROME_BINARY=/path/from/step/1/chromium-browser archivebox version   # replace the␣
↪path with the one you got from step 1
```

Wget & Curl

If you're missing `wget` or `curl`, simply install them using `apt` or your package manager of choice. See the "Manual Setup" instructions for more details.

If wget times out or randomly fails to download some sites that you have confirmed are online, upgrade wget to the most recent version with `brew upgrade wget` or `apt upgrade wget`. There is a bug in versions <=`1.19.1_1` that caused wget to fail for perfectly valid sites.

Archiving

No links parsed from export file

Please open an issue with a description of where you got the export, and preferrably your export file attached (you can redact the links). We'll fix the parser to support your format.

Lots of skipped sites

If you ran the archiver once, it wont re-download sites subsequent times, it will only download new links. If you haven't already run it, make sure you have a working internet connection and that the parsed URLs look correct. You can check the ArchiveBox stdout logs or the Web UI to see what links it's downloading.

If you're still having issues, try deleting or moving the `./archive` folder (back it up first!) and running `archivebox init` again.

Lots of errors

Make sure you have all the dependencies installed and that you're able to visit the links from your browser normally. Open an issue with a description of the errors if you're still having problems.

Lots of broken links from the index

Not all sites can be effectively archived with each method, that's why it's best to use a combination of `wget`, PDFs, and screenshots. If it seems like more than 10-20% of sites in the archive are broken, open an issue with some of the URLs that failed to be archived and I'll investigate.

Removing unwanted links from the index

```
archivebox remove --help
```

Hosting the Archive

If you're having issues trying to host the archive via nginx, make sure you already have nginx running with SSL. If you don't, google around, there are plenty of tutorials to help get that set up. Open an issue if you have problem with a particular nginx config.

Other database or filesystem issues

See here for more info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#repairing-a-corrupted-database
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#modify-the-archivebox-sqlite3-db-directly

- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#merge-two-or-more-existing-archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#do-not-run-as-root
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder

1.3.4 Security Overview

Usage Modes

Archiving Public Content [Default]

This is the default (lax) mode, intended for archiving public (non-secret) URLs without authenticating the headless browser. This is the mode used if you're archiving news articles, audio, video, etc. browser bookmarks to a folder published on your webserver. This allows you to access and link to content on `http://your.archive.com/archive...` after the originals go down.

This mode should not be used for archiving entire browser history or authenticated private content like Google Docs, paywalled content, invite-only subreddits, private photo share urls, etc.

Archiving Private Content

```
WARNING! Advanced users only
```

ArchiveBox is able to archive content that requires authentication or cookies, but it comes with some caveats. Create dedicated logins for archiving to access paywalled content, private forums, LAN-only content, etc. then share them with ArchiveBox via Chrome profile + cookies.txt file.

To get started, set `CHROME_USER_DATA_DIR` and `COOKIES_FILE` to point to a Chrome user folder that has your sessions and a wget `cookies.txt` file respectively.

For full instructions on setting up a Chromium user profile see here: https://github.com/ArchiveBox/ArchiveBox/wiki/Chromium-Install#setting-up-a-chromium-user-profile

If you're importing private links or authenticated content, you probably don't want to share your archive folder publicly on a webserver, so don't follow the [[Publishing Your Archive]] instructions unless you are only serving it on a trusted LAN or have some sort of authentication in front of it. Make sure to point ArchiveBox to an output folder with conservative permissions, as it may contain archived content with secret session tokens or pieces of your user data. You may also wish to encrypt the archive using an encrypted disk image or filesystem like ZFS as it will contain all requests and response data, including session keys, user data, usernames, etc.

Things to watch out for:

- any cookies / secret state present in a Chrome user profile or `cookies.txt` file may be reflected in server responses and saved in the Snapshot output (e.g. in `headers.json`) making it visible in cleartext to anyone viewing the Snapshot, (don't use your personal Chrome profile for archiving or people viewing your archive can then authenticate as you!)
- any secret tokens embedded in URLs (e.g. secret invite links, Google Doc URLs, etc.) will be visible on `archive.org` as the URLs are not filtered when saving to `archive.org` (disable submitting to Archive.org entirely with `SAVE_ARCHIVE_DOT_ORG=False`)
- the domain portion in archived URLs is sent to a favicon service in order to retrieve an icon more reliably than a janky internal implementation would be able to (if leaking domains is a concern, you can disable the favicon fetching entirely with `SAVE_FAVICON=False`)
- viewing malicious archived JS could allow an attacker to access your other archive items + the admin interface (JS executes on the same origin as the admin panel right now, fix is pending, set `SAVE_WGET=False SAVE_DOM=False` to disable the risky extractors entirely or avoid viewing their output directly in a browser)

An example of a session cookie reflected in `headers.json` *visible in the archive.*

Do not run as root

Do not run ArchiveBox as root for a number of reasons:

- Chrome will execute as root and fail immediately because Chrome sandboxing is pointless when the data directory is opened as root (do not set `CHROME_SANDBOX=False` just to bypass that error!)
- All dependencies will be run as root, if any of them have a vulnerability that's exploited by sites you're archiving you're opening yourself up to full system compromise
- ArchiveBox does lots of HTML parsing, filesystem access, and shell command execution. A bug in any one of those subsystems could potentially lead to deleted/damaged data on your hard drive, or full system compromise unless restricted to a user that only has permissions to access the directories needed
- Do you really trust a project created by a Github user called `@pirate` ? Why give a random program off the internet root access to your entire system? (I don't have malicious intent, I'm just saying in principle you should not be running random Github projects as root)

Instead, you should run ArchiveBox as your normal user, or create a user with less privileged access:

```
useradd -r -g archivebox -G audio,video archivebox  # the audio & video groups are used␣
↪by chrome
mkdir -p /home/archivebox/data
chown -R archivebox:archivebox /home/archivebox
...
sudo -u archivebox archivebox add ...
```

~~If you absolutely must run it as root for some reason, a footgun is provided: you can set `ALLOW_ROOT=True` via environment variable or in your ArchiveBox.conf file.~~ This footgun option was removed (I'm sorry, the support burden of helping people who messed up their systems by running everything as root was too high).

Output Folder

Database

The ArchiveBox database is an unencrypted, uncompressed SQLite3 `index.sqlite3` file on disk, and such does not require an authenticated admin SQL login to access (like PostgreSQL/MySQL would). Make sure to protect your database file adequately as anyone who can read it can read your entire collection contents. Passwords for the admin users are stored as salted and PBKDF2 hashed strings in the `auth_user` table.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#modify-the-archivebox-sqlite3-db-directly
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#example-adding-a-new-user-with-a-hashed-password

Filesystem

How much are you planning to archive? Only a few bookmarked articles, or thousands of pages of browsing history a day? If it's only 1-50 pages a day, you can probably just stick it in a normal folder on your hard drive, but if you want to go over 100 pages a day, you will likely want to put your archive on a compressed/deduplicated/encrypted disk image or filesystem like ZFS. Other distributed/networked/checksummed filesystems that have also been reported to work (but are not technically officially supported) include SMB, NFS, Ceph, Unraid, and BTRFS. Make sure the filesystem you're using supports FSYNC. Some filesystems are unable to store more than a certain number of directory entries, and your total number of snapshots in `./archive` may be capped as a result. Some other filesystems begin to have performance degradations but continue to function when the directory entry count gets too high. Generally this isn't an issue unless you have more than ~20,000 Snapshot folders in `./archive`.

Purging entries

Unless `--delete` is passed to `archivebox remove`, Snapshots removed from the index remain in the filesystem and their `./archive/<timestamp>` folders need to be deleted manually to be fully removed. Imported URLs are also logged separately in `./sources`, `./logs`, and the Sonic full-text index `./sonic` and should be removed manually as well to clear all traces of a URL added by accident. You can search for a URL on the filesystem you're trying to remove using `grep -a -r "https://example.com/url/to/search/for"`.

Permissions

Consider what permissioning to apply to your archive folder carefully. Limit access to the fewest possible users by checking folder ownership and setting `OUTPUT_PERMISSIONS` accordingly. Generally the `index.sqlite3` file, `archive/` folder, and `ArchiveBox.conf` file must all be owned and writable by the `archivebox` user or a dedicated non-root user.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox#output-formats
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-troubleshooting

- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#filesystem-doesnt-support-fsync-eg-network-mounts

Publishing

Are you publishing your archive? If so, make sure you use the built-in `archivebox server` or only serve the static export as static HTML (don't accidentally serve it as PHP or CGI or you may execute malicious archived files by accident). Regardless of how you serve it, make sure to put it on its own domain not shared with other services. This is done in order to avoid cookies leaking between your main domain and domains hosting content you don't control. A common practice is to put user provided / untrusted archived content on completely separate top-level domains from anything else (like how Google and Github do with `googleusercontent.com` and `github.io`).

Published archives automatically include a `robots.txt` `Dissallow: /` to block search engines from indexing them. You may still wish to publish your contact info in the index footer though using `FOOTER_INFO` so that you can respond to any DMCA and copyright takedown notices if you accidentally rehost copyrighted content.

Make sure to read all the warnings above about the dangers of exposing Chrome profile data, cookies, secret tokens in URLs, and the risks of viewing archived JS on a shared origin before publishing your archive.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive
- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive#security-concerns
- https://github.com/ArchiveBox/ArchiveBox/wiki/Publishing-Your-Archive#copyright-concerns
- https://en.wikipedia.org/wiki/Cross-site_request_forgery
- https://github.com/ArchiveBox/ArchiveBox/issues/239

1.3.5 Publishing Your Archive

There are two ways to publish your archive: using the `archivebox server` or by exporting and hosting it as static HTML.

1. Use the built-in webserver

```
# set the permissions depending on how public/locked down you want it to be
archivebox config --set PUBLIC_INDEX=True
archivebox config --set PUBLIC_SNAPSHOTS=True
archivebox config --set PUBLIC_ADD_VIEW=True

# create an admin username and password for yourself
archivebox manage createsuperuser

# then start the webserver and open the web UI in your browser
archivebox server 0.0.0.0:8000
open http://127.0.0.1:8000
```

This server is enabled out-of-the-box if you're using `docker-compose` to run ArchiveBox, and there is a commented-out example nginx config with SSL set up as well.

2. Export and host it as static HTML

```
archivebox list --html --with-headers > index.html
archivebox list --json --with-headers > index.json

# then upload the entire output folder containing index.html and archive/ somewhere
# e.g. github pages or another static hosting provider

# you can also serve it with the simple python HTTP server
python3 -m http.server --bind 0.0.0.0 --directory . 8000
open http://127.0.0.1:8000
```

Here's a sample nginx configuration that works to serve your static archive folder:

```
location / {
    alias       /path/to/your/ArchiveBox/data/;
    index       index.html;
    autoindex   on;
    try_files   $uri $uri/ =404;
}
```

Make sure you're not running any content as CGI or PHP, you only want to serve static files!

Urls look like: `https://archive.example.com/archive/1493350273/en.wikipedia.org/wiki/Dining_philosophers_problem.html`

Security Concerns

Re-hosting other people's content has security implications for any other sites sharing your hosting domain. Make sure you understand the dangers of hosting untrusted archived HTML/JS/CSS on a shared domain. Due to the security risk of serving some malicious JS you archived by accident, it's best to put this on a domain or subdomain of its own to keep cookies separate and help limit the effectiveness of CSRF attacks and other nastiness.

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#publishing
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#%EF%B8%8F-things-to-watch-out-for-%EF%B8%8F

Copyright Concerns

Be aware that some sites you archive may not allow you to rehost their content publicly for copyright reasons, it's up to you to host responsibly and respond to takedown requests appropriately.

You may also want to blacklist your archive in `/robots.txt` if you don't want to be publicly associated with all the links you archive via search engine results.

Please modify the `FOOTER_INFO` config variable to add your contact info to the footer of your index.

1.3.6 Scheduled Archiving

ArchiveBox contains a built-in scheduler that supports pulling in URLs regularly from the web or from the local filesystem.

```
$ archivebox schedule --help

usage: archivebox schedule [-h] [--quiet] [--add] [--every EVERY] [--depth {0,1}] [--
↪overwrite] [--clear] [--show] [--foreground] [--run-all] [import_path]

Set ArchiveBox to regularly import URLs at specific times using cron

positional arguments:
  import_path       Check this path and import any new links on every run (can be either␣
↪local file or remote URL)

optional arguments:
  -h, --help        show this help message and exit
  --quiet, -q       Dont warn about storage space.
  --add             Add a new scheduled ArchiveBox update job to cron
  --every EVERY     Run ArchiveBox once every [timeperiod] (hour/day/month/year or cron␣
↪format e.g. "0 0 * * *")
  --depth {0,1}     Depth to archive to [0] or 1, see "add" command help for more info
  --overwrite       Re-archive any URLs that have been previously archived, overwriting␣
↪existing Snapshots
  --clear           Stop all ArchiveBox scheduled runs (remove cron jobs)
  --show            Print a list of currently active ArchiveBox cron jobs
  --foreground, -f  Launch ArchiveBox scheduler as a long-running foreground task␣
↪instead of using cron.
  --run-all         Run all the scheduled jobs once immediately, independent of their␣
↪configured schedules, can be used together with --foreground
```

ArchiveBox ignores links that are imported multiple times (keeping the earliest version that it's seen). This means you can add cron jobs that regularly poll the same file or URL for new links, adding only new ones as necessary, or you can pass `--overwrite` to save a fresh copy each time the scheduled task runs.

The list of defined scheduled tasks can be inspected and cleared with `archivebox schedule --show` and `archivebox schedule --clear`.

Many popular sites such as Twitter, Reddit, Facebook, etc. take efforts to block/ratelimit/lazy-load content to avoid being scraped by bots like ArchiveBox. It may be better to use an alternative frontend with minimal JS when archiving those sites:https://github.com/mendel5/alternative-front-ends

The scheduled interval can be passed easily using `--every={day,week,month,year}` or by passing a cron-style schedule e.g. `--every='5 4 * * *'` to run at 04:05 every day.

The scheduler can also be run in `--foreground` mode to avoid relying on your host system's cron scheduler to be running.In foreground mode, it will run all tasks previously added using `archivebox schedule` in a long-running foreground process. This is useful for running scheduled tasks inside docker-compose or supervisord.

Docker Usage

```
docker-compose run --rm archivebox schedule --every=week --depth=1 https://example.com
docker-compose run --rm archivebox schedule --every=day https://example.com
docker-compose run --rm archivebox schedule --show
docker-compose run --rm archivebox schedule --help

# restart the scheduler container to pick up any changes made
docker compose restart archivebox_scheduler
```

docker-compose.yml:

```
services:
  archivebox:
    image: archivebox/archivebox:dev
    command: server --quick-init 0.0.0.0:8000
    ...
    volumes:
      - ./data:/data
      - ./etc/crontabs:/var/spool/cron/crontabs

  archivebox_scheduler:
    image: archivebox/archivebox:dev
    command: schedule --foreground
    ...
    volumes:
      - ./data:/data
      - ./etc/crontabs:/var/spool/cron/crontabs
```

For a full Docker Compose example config see here: https://github.com/ArchiveBox/ArchiveBox/blob/dev/docker-compose.yml#:~:text=schedule

For more examples of plain Docker and Docker Compose usage with scheduling, see: https://github.com/ArchiveBox/ArchiveBox/issues/1155#issuecomment-1590146616

Example: Archive a Twitter user's Tweets and linked content within once a week

```
archivebox schedule --every=week --depth=1 'https://nitter.net/ArchiveBoxApp'
```

Nitter is an alternative frontends recommended Twitter that formats the content better for archiving/bots and avoids ratelimits.

Example: Archive a Reddit subreddit and discussions for every post once a week

```
# optionally limit URLs to Teddit (aka Reddit) to capture discussion and user pages but␣
↪not external outbound URLs
archivebox config --set URL_WHITELIST='^http(s)?:\/\/(.+)?teddit\.net\/?.*$'

archivebox schedule --every=week --overwrite --depth=1 'https://teddit.net/r/DataHoarder/
↪'
```

Teddit is an alternative frontend recommended for Reddit that formats the content better for archiving/bots and avoids ratelimits.`--overwrite` is passed to save a fresh copy each week, otherwise the URL will be ignored as it's already present in the collection after the first time it's added.

Example: Archive the HackerNews front page and some linked articles every 24 hours

```
# optional exclude some URLs you don't want to archive
archivebox config --set URL_BLACKLIST='^http(s)?:\/\/(.+\.)?(youtube\.com)|(amazon\.com)\
↪/.*$'

archivebox schedule --every=day --depth=1 'https://news.ycombinator.com'
```

Example: Archive all URLs in an RSS feed from Pocket every 12 hours

This example imports your Pocket bookmark feed and archives any new links every 12 hours:

First, set your Pocket RSS feed to "public" under https://getpocket.com/privacy_controls.

Then tell ArchiveBox to pull it regularly:

```
archivebox schedule --every=day --depth=1 https://getpocket.com/users/
↪yourusernamegoeshere/feed/all
```

Example: Archive a Github repository's source code only once a month

```
archivebox schedule --every=month --extract=git --overwrite 'https://github.com/
↪ArchiveBox'
```

`--extract=git` tells it to only use the Git source extractor and skip saving the HTML/screenshot/etc. other extractor methods.

Example: Archive a list of URLs pulled from the filesystem every 30 minutes

```
archivebox schedule --every='30 * * * *' /some/path/to/urls.txt
```

Advanced Scheduling Using Cron

To schedule regular archiving you can also use any other task scheduler like `cron`, `at`, `systemd`, etc. aside from the built-in scheduler `archivebox schedule`.

For some example configs, see the `etc/cron.d` and `etc/supervisord` folders.

Example: Export and archive Firefox browser history every 24 hours

This example exports your browser history and archives it once a day, saving a summary to disk:

First download the ArchiveBox helper script for browser history exporting https://github.com/ArchiveBox/ArchiveBox/blob/dev/bin/export_browser_history.sh to `./bin/export_browser_history.sh`

Then create `/home/ArchiveBox/archivebox/bin/scheduled_firefox_import.sh`:

```
#!/bin/bash

cd `/home/ArchiveBox/archivebox
bash ./bin/export_browser_history --firefox ./output/sources/firefox_history.json
archivebox add < ./output/sources/firefox_history.json >> /var/log/ArchiveBox.log
archivebox status >> /var/log/ArchiveBox.log
```

Then tell cron to run your script every 24 hours:

```
echo '0 24 * * * archivebox /home/ArchiveBox/archivebox/bin/scheduled_firefox_import.sh'␣
↪> /etc/cron.d/archivebox_scheduled_firefox_import
```

Example: Import an RSS feed from Pocket every 12 hours

If you need to customize the import process or archive a password-locked RSS feed, you can do it manually with a bash script + cron `/home/ArchiveBox/archivebox/bin/scheduled_imports.sh`:

```
#!/bin/bash

cd /home/ArchiveBox/archivebox
curl --silent https://getpocket.com/users/yourusernamegoeshere/feed/all | archivebox add␣
↪>> /home/ArchiveBox/archivebox/logs/scheduled_imports.log
# you can add additional flags to curl here e.g. to authenticate with HTTP
# curl --silent -u username:password ... | archivebox add >> ...
```

Then create a cronjob telling your system to run the script on your chosen regular interval (e.g. every 12 hours):

```
echo '0 12 * * * archivebox /home/ArchiveBox/archivebox/bin/scheduled_imports.sh' > /etc/
↪cron.d/archivebox_scheduled_imports
```

1.3.7 Chromium Install

By default, ArchiveBox looks for any existing installed version of Chrome/Chromium and uses it if found. You can optionally install a specific version and set the environment variable `CHROME_BINARY` to force ArchiveBox to use that one, e.g.:

- `CHROME_BINARY=google-chrome-beta`
- `CHROME_BINARY=/usr/bin/chromium-browser`
- `CHROME_BINARY='/Applications/Chromium.app/Contents/MacOS/Chromium'`

If you don't already have Chrome installed, I recommend installing Chromium instead of Google Chrome, as it's the open-source fork of Chrome that doesn't send as much tracking data to Google.

Check for existing Chrome/Chromium install:

```
google-chrome --version | chromium-browser --version
Google Chrome 73.0.3683.75 beta     # should be >v59
```

Installing Chromium

macOS

If you already have `/Applications/Chromium.app`, you don't need to run this.

```
brew install chromium
```

Ubuntu/Debian

If you already have `chromium-browser` >= v59 installed (run `chromium-browser --version`, you don't need to run this.

```
apt update
apt install chromium-browser
# or on some systems:
apt install chromium
```

Installing Google Chrome

macOS

If you already have `/Applications/Google Chrome.app`, you don't need to run this.

```
brew install google-chrome
```

Ubuntu/Debian

If you already have `google-chrome` >= v59 installed (run `google-chrome --version`, you don't need to run this.

```
wget -q -O - https://dl-ssl.google.com/linux/linux_signing_key.pub | sudo apt-key add -
sudo sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >>
↪ /etc/apt/sources.list.d/google-chrome.list'
apt update
apt install google-chrome
```

Troubleshooting Chromium Install

If you encounter problems setting up Google Chrome or Chromium, see the Troubleshooting page.

1.3.8 Setting Up a Chromium User Profile

You may choose to set up a Chrome/Chromium user profile in order to use your cookies/sessions to log into sites behind authentication/paywall during archiving.

You must set up the profile using the exact same version of chrome that ArchiveBox is running (which can be found with `archivebox version`). You can download old versions of Chrome in order to match it from https://chromium.cypress.io.

General steps:

1. Install desired Chromium version in new directory `./data/chromium` inside your data folder on the host (outside Docker)
2. Open the Chromium binary directly on the host if possible, or run `vncserver` as `archivebox` user and run chromium in VNC session to generate cookies, then close VNC session (detailed instructions here)
3. Add the config to `docker-compose.yml` to mount the `./data/chromium` volume and environment variables telling ArchiveBox to use it `docker-compose.yml`:

```
services:
  archivebox:
      ...
      environment:
          ...
          - CHROME_USER_DATA_DIR=/data/chromium/.config/chromium
          - CHROME_BINARY=/data/chromium/chrome
      volumes:
          - ./data:/data
          - ./data/chromium:/data/chromium
          ...
...
```

1. Set the permissions on the chromium dir

```
docker-compose run --rm archivebox /bin/bash
# then inside of docker run these:
chown -R archivebox:archivebox /data/chromium/
chmod -R ugo+rwx /data/chromium
```

The new profile is now generated and used by same instance of Chrome on docker host and container.

Each step is crucial, especially the permissions and matching the binary inside of Docker and outside.

More info and troubleshooting steps:

- https://github.com/ArchiveBox/ArchiveBox/issues/952
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#archiving-private-content
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#%EF%B8%8F-things-to-watch-out-for-%EF%B8%8F
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#publishing
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#chrome_user_data_dir
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#chrome_binary
- https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration#cookies_file

1.3.9 Upgrade your ArchiveBox collection to a new version

Note: It's recommended to only upgrade one major version at a time. e.g. if you're on `v0.4.14`*, upgrade to* `v0.5.6` *next, then* `v0.6.3`*, and finally* `v0.7.1` *(as 3 separate steps). You can specify exact versions with pip like so:* `pip install archivebox==0.6.3` *or with docker* `docker pull archivebox/archivebox:0.6.3`*. Upgrading directly across multiple major versions may work in some cases, but is not recommended for maximum data safety.*

Upgrading checklist:

1. Find the version you want to upgrade to on https://github.com/ArchiveBox/ArchiveBox/releases
2. **Read the release notes carefully** for any instructions or extra steps around upgrading for each release you're skipping or installing
3. **Make a full backup** of your `index.sqlite3` and `archive/` content before upgrading!`gzip -9 < index.sqlite3 > "index.sqlite3.$(date +%s).bak"`
4. Follow the steps below depending on your setup to run `archivebox init` (repeating as necessary for each major version if upgrading across multiple major versions)
5. Confirm the upgrade succeeded and check for any orphan/corrupted snapshots with `archivebox status`

Open an issue in our bug tracker if you experience any problems with upgrading/merging/modifying collections.

How it works internally:

The same command is used for initializing a new archive and upgrading an existing one. `archivebox init` is indempotent and safely be run multiple times. Running it will ensure your collection is on the latest version and all the files are in their correct locations. `archivebox status` can be used to check for orphan/corrupted snapshots or invalid index data.

There are three main areas on disk that ArchiveBox modifies during upgrades:

- `index.sqlite3` contains the SQLite3 DB index that gets upgraded automatically by Django based on the changes in `archivebox/core/models.py`.
- `archive/*/index.json` these files are redundant json exports of the data for each Snapshot in `index.sqlite3`, these files are overwritten on every `archivebox update` run or anytime the Snapshot is modified from the GUI or CLI. These files will be lazily updated to the latest schema versions as ArchiveBox accesses them, but are usually not modified in bulk during `archivebox init` when upgrading.

- `archive/*` the Snapshot output files may be moved or renamed by future upgrades (so far they have remained unchanged since v0.1, but future versions reserve the right to change their locations)

The `ArchiveBox.conf` file is not modified by upgrades and should remain forward-compatible across future versions (even when config options are renamed, we check the old names internally to maintain compatibility).

As of v0.4 and above, ArchiveBox uses the Django migrations system for deterministic, atomic, safe upgrades, so your DB should always be left in a consistent state in the event of a failure or power outage. If you need help fixing a corrupted collection, open an issue using the link above.

More info:

- https://docs.djangoproject.com/en/4.0/topics/migrations/
- https://github.com/ArchiveBox/ArchiveBox/wiki/Upgrading-or-Merging-Archives#database-migrations-errors-or-upgrade-issues
- https://github.com/ArchiveBox/ArchiveBox/wiki/Troubleshooting

---

Upgrading with Docker Compose

Using Docker Compose is recommended because it makes upgrading a breeze! Pulling and running the latest version automatically upgrades the ArchiveBox collection and all of ArchiveBox's internal dependencies.

```
cd ~/archivebox         # or wherever your data folder is
docker-compose down     # stop the currently running archivebox containers
docker-compose down     # run twice to clear stopped containers
docker-compose pull     # pull the latest image version from Docker Hub
docker-compose up       # collection will be automatically upgraded as it starts
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#%EF%B8%8F-easy-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#docker-compose
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#setup

Upgrading with plain Docker

Upgrading with plain Docker is similar to the process with Docker Compose, but you have to run `archivebox init` manually at the end to finish the process.

```
docker ps -a -q  --filter ancestor=archivebox/archivebox  # find any currently running␣
↪archivebox containers
docker kill <image>    # stop any currently running archivebox versions

docker pull archivebox/archivebox
docker run -v $PWD:/data -it archivebox/archivebox init --setup  # upgrade the␣
↪collection to the latest version

# restart the archivebox server container if needed
docker run -v $PWD:/data -it -p 8000:8000 archivebox/archivebox server 0.0.0.0:8000
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#%EF%B8%8F-easy-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#docker
- https://github.com/ArchiveBox/ArchiveBox/wiki/Docker#setup-1

Upgrading with a package manager

Package manager releases take a lot of effort to maintain (contributions welcome!) and sometimes lag behind the Docker releases. We make a best effort to have the latest release available through all channels within a reasonable timeframe.

```
cd ~/archivebox        # or wherever your data folder is
killall archivebox     # stop the currently running archivebox version

# upgrade ArchiveBox using the package manager you originally used to install it
pip install --upgrade archivebox
# or
brew upgrade archivebox
# or
apt install --upgrade archivebox
# or with the optional auto-installer script
curl -sSL 'https://get.archivebox.io' | sh

archivebox init        # run init to upgrade the collection to the latest version

archivebox update --index-only  # optionally force an update of the snapshot index files␣
↪(normally done lazily, see issue #962 for more info)

archivebox status      # check that everything succeeded
```

More info:

- https://github.com/ArchiveBox/ArchiveBox#-package-manager-setup
- https://github.com/ArchiveBox/ArchiveBox/wiki/Install#manual-setup
- https://github.com/ArchiveBox/pip-archivebox
- https://github.com/ArchiveBox/homebrew-archivebox
- https://github.com/ArchiveBox/docker-archivebox
- https://github.com/ArchiveBox/debian-archivebox
- https://github.com/ArchiveBox/electron-archivebox
- https://aur.archlinux.org/packages/archivebox
- https://github.com/NixOS/nixpkgs/blob/master/pkgs/applications/misc/archivebox/default.nix

1.3.10 Merge two or more existing archives

Two or more existing ArchiveBox collection dirs can be merged together by simply combining the contents of `archive/*` and re-running `archivebox init` to pull the new Snapshots into the index.

1. Upgrade both old collections to the most recent ArchiveBox version (following instructions above)

```
pip install --upgrade archivebox   # or follow instructions above for upgrading w/ Docker

cd /path/to/archivebox1/data
archivebox init --setup
archivebox status

cd /path/to/archivebox2/data
archivebox init --setup
archivebox status

# ... repeat the same for each collection if merging more than two
```

1. Create a new empty archivebox collection in a new folder somewhere, this will hold the new merged collection

```
mkdir /path/to/archivebox_new
cd /path/to/archivebox_new
archivebox init --setup
```

1. Copy everything under `./archive/*` in each old collection into the new collection's `./archive/` folder

```
rsync --archive --info=progress2 /path/to/archivebox1/data/archive/ /path/to/archivebox_
↪new/data/archive
rsync --archive --info=progress2 /path/to/archivebox2/data/archive/ /path/to/archivebox_
↪new/data/archive
# ...repeat the same for each collection if merging more than two
```

1. Run `archivebox init` in the new merged collection to regenerate the new index

```
cd /path/to/archivebox_new
archivebox init --setup
```

1. The new collection should now contain all the entries from the old collections combined

```
cd /path/to/archivebox_new
archivebox status

# optionally force an update of the snapshot index files (normally done lazily)
archivebox update --index-only
```

For more information about why Snapshot index files are usually updated lazily, see: https://github.com/ArchiveBox/ArchiveBox/issues/962

1.3.11 Modify the ArchiveBox SQLite3 DB directly

If you need to automate changes to the ArchiveBox DB (for example adding a User from an Ansible script), you can modify the SQLite3 DB directly.

Note, this is often unnecessary for modifying ArchiveBox on a host that doesn't have the CLI installed, as you can also copy the `index.sqlite3` to a local machine that has it, do the modifications locally, then copy the modified db back into place on the host. (Docker/CLI/GUI/Web ArchiveBox all share the same DB schema/format)

```
cd ~/archivebox          # cd into your archivebox collection dir
sqlite3 index.sqlite3    # open the db with sqlite3 shell
```

Example: Modifying an existing user's email

```
UPDATE auth_user
SET email = 'someNewEmail@example.com', is_superuser = 1
WHERE username = 'someUsernameHere';
```

Example: Adding a new user with a hashed password

Note: this is just an example to demonstrate direct database usage. If you are trying to create a user on initial setup, use the ADMIN_USERNAME & ADMIN_PASSWORD configuration options.

1. First, generate the hashed password in a Python shell using Django's `make_password` function.

This can be done on any machine with Python 3+, it doesn't have to have ArchiveBox installed.

```
pip3 install django==3.1.3   # install the django version used by ArchiveBox
python3                      # open any python shell with django available, doesn't have␣
↪to be the archivebox shell
```

```
>>> from django.contrib.auth.hashers import make_password
>>> make_password('somePasswordHere', 'someSaltHere', 'pbkdf2_sha256')         # choose␣
↪a password and a salt (can be anything 12 chars long)
'pbkdf2_sha256$216000$someSaltHere$styW1Uoy8SHp3zbSwGRp20C9mPjOHVjP9rl5a8/UOVE='
```

1. Use the generated hashed password to insert a new User row in the SQLite3 database directly:

```
cd ~/archivebox          # cd into your archivebox collection dir
sqlite3 index.sqlite3    # open the db with sqlite3 shell
```

```
INSERT INTO "auth_user" ("password", "last_login", "is_superuser", "username", "first_
↪name", "last_name", "email", "is_staff", "is_active", "date_joined")
VALUES ('pbkdf2_sha256$216000$someSaltHere$+2beZufc3JUXnmn0tG+2peJEBh7MjxPYmT3YfIFzEl0=',
↪ NULL, 0, 'someUsername', '', '', 'someEmail@example.com', 0, 1, '2022-03-22 23:34:02.
↪333042')
```

Replace the values above with the desired username, email, and password hash from python output^.

1. Log in using the new generated user to confirm it works https://localhost:8000/admin/login/ user: `someUsername` pass:`somePasswordHere`

More info:

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage

---

1.3.12 Database Troubleshooting

Database and filesystem issues are uncommon but do come up from time to time (especially when using networked storage, large archives, or multiple ArchiveBox processes for a single collection).

Generally, these commands can help you resolve most issues:

```
archivebox init                 # upgrade the archivebox collection
archivebox init --setup         # upgrade the archivebox collection and all dependencies
archivebox update --index-only  # force an upgrade of some of the archivebox index/
↪collection files
archivebox server --debug       # run the server with more verbose debug log output
archivebox shell                # access the Python API / Django management shell
sqlite3 index.sqlite3           # access the SQLite3 SQL database shell
```

Don't be scared by the volume of content here. Almost all of these issues linked below are duplicates or old resolved bugs, but they contain valuable context and troubleshooting steps if you're trying to figure out the cause of a problem with your setup.

Filesystem doesn't support FSYNC (e.g. network mounts)

The `index.sqlite3` file must be stored on a filesystem that supports FSYNC (most local filesystems) in order to ensure SQLite3 database integrity when multiple ArchiveBox processes may be accessing it simultaneously. However, the `./archive` folder can be on a NAS or other filesystem that does not support FSYNC.

- Archivebox hangs when initializing collection on network drive that doesn't support FSYNC #742
- Question: How to run AB on localhost but store data on NAS? #894
- Question: Docker on Windows archiving to an SMB path that doesn't support FSYNC #722
- Support for network drives or filesystems that don't implement FSYNC #456

More info:

- https://www.geeksforgeeks.org/python-os-fsync-method/
- https://man7.org/linux/man-pages/man2/fdatasync.2.html
- https://www.samba.org/samba/docs/current/man-html/smb.conf.5.html
- https://eclecticlight.co/2022/02/18/how-can-you-trust-a-disk-to-write-data/

Database and filesystem contention issues when running multiple ArchiveBox processes

ArchiveBox can sometimes struggle when archiving many links in parallel with multiple ArchiveBox processes trying to write to the database at the same time, leading to errors like this:

```
Unable to create the django_migrations table (database is locked)
```

These errors can also be encountered when there are permissions, network, or filesystem issues preventing writes to the `index.sqlite3` file.

- Question: Unable to create the django_migrations table (database is locked) - When OUTPUT_DIR to SAMBA share #946
- Question: ...Unable to create the django_migrations table (database is locked) #880
- Database is locked and other weird behavior when doing simultaneous adds #781
- Bugfix: Retry on "database locked" error (or add support for PostgreSQL/MySQL DB backend) #601
- Architecture: Use multiple cores to run link archiving in parallel #91
- ArchiveBox index corruption when running multiple import processes on v0.5.0 #454
- Architecture: Concurrent runs accidentally delete each other's temp files, leaving the index broken #234
- Database is locked and other weird behavior when doing simultaneous adds #781
- Bugfix: Retry on "database locked" error (or add support for PostgreSQL/MySQL DB backend) #601

More info:

- https://www.sqlite.org/lockingv3.html
- https://charlesleifer.com/blog/going-fast-with-sqlite-and-python/
- https://victoria.dev/blog/sqlite-in-production-with-wal/
- https://code.djangoproject.com/ticket/29280
- https://stackoverflow.com/questions/47761570/how-can-i-avoid-database-is-locked-sqlite3-errors-in-django

Database migrations errors or upgrade issues

Migration or upgrade issues happen occasionally with some niche setups or when skipping major versions during archiving.Always backup your archive before upgrading, but know that migrations are deterministic and atomic using Django's migration system, so a failed migration does not mean your archive is unrecoverable, you just have to downgrade to the previous stable major version then continue upgrading.

```
archivebox init  # this usually applies any necesary migrations (atomically and␣
↪indempotently, safe to run multiple times)
```

- Bug: NOT NULL constraint failed: core_archiveresult.output when upgrading v0.4.24 archive to v0.6 #705
- Bugfix: sqlite3.IntegrityError: NOT NULL constraint failed: core_archiveresult.cmd_version and .output #597
- Error: django.db.utils.IntegrityError: UNIQUE constraint failed: core_tag.slug #596
- Bugfix: django.db.utils.IntegrityError: UNIQUE constraint failed: core_snapshot.timestamp #412
- Best Practices for Backup/Restore #341
- Bug: Running archivebox update –index-only doesn't upgrade Snapshot index.{html,json} files #962
- Feature Request: Deduplicate files on archives #704

More info:

- https://docs.djangoproject.com/en/4.0/topics/migrations/
- https://realpython.com/django-migrations-a-primer/
- https://realpython.com/digging-deeper-into-migrations/
- https://www.kite.com/blog/python/django-database-migrations-overview/
- https://markusholtermann.eu/2021/06/writing-safe-database-migrations-in-django/

Repairing a corrupted SQLite3 database file

A corrupted database file can theoretically only happen if an external process or filesystem error corrupts the SQLite3 database (there has only been one report of a user encountering this in real life). If you ever need to repair a corrupted ArchiveBox index you can run the following steps.

Note this is specific to this error, these steps do not apply to other migrations/db errors (see below for other issues):

```
sqlite3.DatabaseError: database disk image is malformed
```

Generally all index issues should be fixable by running `archivebox init`.You can see the status of Snapshots and find any invalid/orphan/missing snapshots with `archivebox status`.

Error output:

```
[i] [2022-03-24 20:37:27] ArchiveBox v0.6.2: archivebox init
    > /data

[^] Verifying and updating existing ArchiveBox collection to v0.6.2...
------------------------------------------------------------------------

[*] Verifying archive folder structure...
    + ./archive, ./sources, ./logs...
    + ./ArchiveBox.conf...

[*] Verifying main SQL index and running any migrations needed...
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/site-packages/django/db/backends/utils.py", line 82, in␣
↪_execute
    return self.cursor.execute(sql)
  File "/usr/local/lib/python3.9/site-packages/django/db/backends/sqlite3/base.py", line␣
↪411, in execute
    return Database.Cursor.execute(self, query)
sqlite3.DatabaseError: database disk image is malformed
```

Steps to fix:

```
cd ~/archivebox
echo '.dump' | sqlite3 index.sqlite3 | sqlite3 repaired_index.sqlite3
mv index.sqlite3 corrupt_index.sqlite3
mv repaired_index.sqlite3 index.sqlite3
```

More info:

- https://github.com/ArchiveBox/ArchiveBox/issues/955

- https://stackoverflow.com/questions/5274202/sqlite3-database-or-disk-is-full-the-database-disk-image-is-malformed

---

1.3.13 Related Documents

- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#disk-layout
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#large-archives
- https://github.com/ArchiveBox/ArchiveBox/wiki/Security-Overview#output-folder
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#python-shell-usage
- https://github.com/ArchiveBox/ArchiveBox/wiki/Usage#sql-shell-usage

1.4 API Reference

1.4.1 archivebox

archivebox package

Subpackages

archivebox.cli package

Submodules

archivebox.cli.archivebox_add module

archivebox.cli.archivebox_add.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Add a new URL or list of URLs to your archive

archivebox.cli.archivebox_config module

archivebox.cli.archivebox_config.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Get and set your ArchiveBox project configuration values

archivebox.cli.archivebox_help module

archivebox.cli.archivebox_help.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print the ArchiveBox help message and usage

archivebox.cli.archivebox_status module

archivebox.cli.archivebox_status.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print out some info and statistics about the archive collection

archivebox.cli.archivebox_init module

archivebox.cli.archivebox_init.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Initialize a new ArchiveBox collection in the current directory

archivebox.cli.archivebox_list module

archivebox.cli.archivebox_list.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

List, filter, and export information about archive entries

archivebox.cli.archivebox_manage module

archivebox.cli.archivebox_manage.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Run an ArchiveBox Django management command

archivebox.cli.archivebox_remove module

archivebox.cli.archivebox_remove.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Remove the specified URLs from the archive

archivebox.cli.archivebox_schedule module

archivebox.cli.archivebox_schedule.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Set ArchiveBox to regularly import URLs at specific times using cron

archivebox.cli.archivebox_server module

archivebox.cli.archivebox_server.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Run the ArchiveBox HTTP server

archivebox.cli.archivebox_shell module

archivebox.cli.archivebox_shell.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Enter an interactive ArchiveBox Django shell

archivebox.cli.archivebox_update module

archivebox.cli.archivebox_update.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Import any new links from subscriptions and retry any previously failed/skipped links

archivebox.cli.archivebox_version module

archivebox.cli.archivebox_version.**main**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print the ArchiveBox version and dependency information

Module contents

archivebox.cli.**list_subcommands**() → Dict[str, str]

find and import all valid archivebox_<subcommand>.py files in CLI_DIR

archivebox.cli.**run_subcommand**(*subcommand: str*, *subcommand_args: List[str] = None*, *stdin: IO | None = None*, *pwd: Path | str | None = None*) → None

Run a given ArchiveBox subcommand with the given list of args

archivebox.cli.**help**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print the ArchiveBox help message and usage

archivebox.cli.**version**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print the ArchiveBox version and dependency information

archivebox.cli.**init**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Initialize a new ArchiveBox collection in the current directory

archivebox.cli.**config**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Get and set your ArchiveBox project configuration values

archivebox.cli.**setup**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Automatically install all ArchiveBox dependencies and extras

archivebox.cli.**add**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Add a new URL or list of URLs to your archive

archivebox.cli.**remove**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Remove the specified URLs from the archive

archivebox.cli.**update**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Import any new links from subscriptions and retry any previously failed/skipped links

archivebox.cli.**list**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

List, filter, and export information about archive entries

archivebox.cli.**status**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Print out some info and statistics about the archive collection

archivebox.cli.**shell**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Enter an interactive ArchiveBox Django shell

archivebox.cli.**server**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Run the ArchiveBox HTTP server

archivebox.cli.**manage**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Run an ArchiveBox Django management command

archivebox.cli.**oneshot**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Create a single URL archive folder with an index.json and index.html, and all the archive method outputs. You can run this to archive single pages without needing to create a whole collection with archivebox init.

archivebox.cli.**schedule**(*args: List[str] | None = None*, *stdin: IO | None = None*, *pwd: str | None = None*) → None

Set ArchiveBox to regularly import URLs at specific times using cron

archivebox.config package

Submodules

Module contents

ArchiveBox config definitons (including defaults and dynamic config options).

Config Usage Example:

> archivebox config –set MEDIA_TIMEOUT=600 env MEDIA_TIMEOUT=600 USE_COLOR=False … archivebox [subcommand] …

Config Precedence Order:

1. cli args (–update-all / –index-only / etc.)
2. shell environment vars (env USE_COLOR=False archivebox add '…')

3. config file (echo "SAVE_FAVICON=False" >> ArchiveBox.conf)
4. defaults (defined below in Python)

Documentation:

https://github.com/ArchiveBox/ArchiveBox/wiki/Configuration

archivebox.config.**get_real_name**(*key: str*) → str

get the current canonical name for a given deprecated config key

archivebox.config.**get_version**(*config*)

archivebox.config.**get_commit_hash**(*config*)

archivebox.config.**load_config_val**(*key: str*, *default: str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[ConfigDict], str | bool | int | None | Pattern | Dict[str, Any]] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]] = None*, *type: Type | None = None*, *aliases: Tuple[str, ...] | None = None*, *config: ConfigDict | None = None*, *env_vars: _Environ | None = None*, *config_file_vars: Dict[str, str] | None = None*) → str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]

parse bool, int, and str key=value pairs from env

archivebox.config.**load_config_file**(*out_dir: str = None*) → Dict[str, str] | None

load the ini-formatted config file from OUTPUT_DIR/Archivebox.conf

archivebox.config.**write_config_file**(*config: Dict[str, str]*, *out_dir: str = None*) → *ConfigDict*

load the ini-formatted config file from OUTPUT_DIR/Archivebox.conf

archivebox.config.**load_config**(*defaults: Dict[str, ConfigDefault]*, *config: ConfigDict | None = None*, *out_dir: str | None = None*, *env_vars: _Environ | None = None*, *config_file_vars: Dict[str, str] | None = None*) → *ConfigDict*

archivebox.config.**stdout**(**args*, *color: str | None = None*, *prefix: str = ''*, *config: ConfigDict | None = None*) → None

archivebox.config.**stderr**(**args*, *color: str | None = None*, *prefix: str = ''*, *config: ConfigDict | None = None*) → None

archivebox.config.**hint**(*text: Tuple[str, ...] | List[str] | str*, *prefix=' '*, *config: ConfigDict | None = None*) → None

archivebox.config.**bin_version**(*binary: str | None*) → str | None

check the presence and return valid version line of a specified binary

archivebox.config.**bin_path**(*binary: str | None*) → str | None

archivebox.config.**bin_hash**(*binary: str | None*) → str | None

archivebox.config.**find_chrome_binary**() → str | None

find any installed chrome binaries in the default locations

archivebox.config.**find_chrome_data_dir**() → str | None

find any installed chrome user data directories in the default locations

archivebox.config.**wget_supports_compression**(*config*)

archivebox.config.**get_code_locations**(*config: ConfigDict*) → Dict[str, str | bool | int | None | Pattern | Dict[str, Any]]

archivebox.config.**get_external_locations**(*config: ConfigDict*) → str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]

archivebox.config.**get_data_locations**(*config: ConfigDict*) → str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]

archivebox.config.**get_dependency_info**(*config: ConfigDict*) → str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]

archivebox.config.**get_chrome_info**(*config: ConfigDict*) → str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]

archivebox.config.**load_all_config**()

archivebox.config.**check_system_config**(*config: ~archivebox.config_stubs.ConfigDict = {'ADMIN_PASSWORD': None, 'ADMIN_USERNAME': None, 'ALLOWED_HOSTS': '*', 'ANSI': {'black': '', 'blue': '', 'green': '', 'lightblue': '', 'lightred': '', 'lightyellow': '', 'red': '', 'reset': '', 'white': ''}, 'ARCHIVEBOX_BINARY': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.1 packages/sphinx/__main__.py', 'ARCHIVE_DIR': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ar 'BIND_ADDR': '127.0.0.1:8000', 'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_OPTIONS': {'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'RESOLUTION': '1440,2000', 'TIMEOUT': 60}, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'CHROME_VERSION': None, 'CODE_LOCATIONS': {'CUSTOM_TEMPLATES_DIR': {'enabled': False, 'is_valid': None, 'path': None}, 'PACKAGE_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ar 'TEMPLATES_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ar 'COMMIT_HASH': '0bd83076db5ddfe23857b0bcfc7028fa72cf2edf', 'CONFIG_FILE': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/A 'COOKIES_FILE': None, 'CURL_ARGS': ['--silent', '--location', '--compressed'], 'CURL_BINARY': 'curl', 'CURL_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/dev (+https://github.com/ArchiveBox/ArchiveBox/) curl/curl 7.81.0 (x86_64-pc-linux-gnu)', 'CURL_VERSION': 'curl 7.81.0 (x86_64-pc-linux-gnu)', 'CUSTOM_TEMPLATES_DIR': None, 'DATA_LOCATIONS': {'ARCHIVE_DIR': {'enabled': True, 'is_mount': False, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ar 'CONFIG_FILE': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/A 'LOGS_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/lo 'OUTPUT_DIR': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')}, 'SOURCES_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/so 'SQL_INDEX': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/in*

archivebox.config.**check_dependencies**(*config: ~archivebox.config_stubs.ConfigDict = {'ADMIN_PASSWORD': None, 'ADMIN_USERNAME': None, 'ALLOWED_HOSTS': '*', 'ANSI': {'black': '', 'blue': '', 'green': '', 'lightblue': '', 'lightred': '', 'lightyellow': '', 'red': '', 'reset': '', 'white': ''}, 'ARCHIVEBOX_BINARY': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.12 packages/sphinx/__main__.py', 'ARCHIVE_DIR': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arc 'BIND_ADDR': '127.0.0.1:8000', 'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_OPTIONS': {'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'RESOLUTION': '1440,2000', 'TIMEOUT': 60}, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'CHROME_VERSION': None, 'CODE_LOCATIONS': {'CUSTOM_TEMPLATES_DIR': {'enabled': False, 'is_valid': None, 'path': None}, 'PACKAGE_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arc 'TEMPLATES_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arc 'COMMIT_HASH': '0bd83076db5ddfe23857b0bcfc7028fa72cf2edf', 'CONFIG_FILE': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Arc 'COOKIES_FILE': None, 'CURL_ARGS': ['--silent', '--location', '--compressed'], 'CURL_BINARY': 'curl', 'CURL_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/dev (+https://github.com/ArchiveBox/ArchiveBox/) curl/curl 7.81.0 (x86_64-pc-linux-gnu)', 'CURL_VERSION': 'curl 7.81.0 (x86_64-pc-linux-gnu)', 'CUSTOM_TEMPLATES_DIR': None, 'DATA_LOCATIONS': {'ARCHIVE_DIR': {'enabled': True, 'is_mount': False, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arc 'CONFIG_FILE': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Arc 'LOGS_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/log 'OUTPUT_DIR': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')}, 'SOURCES_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/sou 'SQL_INDEX': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ind 'DEBUG': False, 'DEPENDENCIES': {'ARCHIVEBOX_BINARY': {'enabled': True, 'hash':*

archivebox.config.**check_data_folder**(*out_dir: str | ~pathlib.Path | None = None, config: ~archivebox.config_stubs.ConfigDict = {'ADMIN_PASSWORD': None, 'ADMIN_USERNAME': None, 'ALLOWED_HOSTS': '*', 'ANSI': {'black': '', 'blue': '', 'green': '', 'lightblue': '', 'lightred': '', 'lightyellow': '', 'red': '', 'reset': '', 'white': ''}, 'ARCHIVEBOX_BINARY': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.12/ packages/sphinx/__main__.py', 'ARCHIVE_DIR': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arch 'BIND_ADDR': '127.0.0.1:8000', 'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_OPTIONS': {'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'RESOLUTION': '1440,2000', 'TIMEOUT': 60}, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'CHROME_VERSION': None, 'CODE_LOCATIONS': {'CUSTOM_TEMPLATES_DIR': {'enabled': False, 'is_valid': None, 'path': None}, 'PACKAGE_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arch 'TEMPLATES_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arch 'COMMIT_HASH': '0bd83076db5ddfe23857b0bcfc7028fa72cf2edf', 'CONFIG_FILE': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Arch 'COOKIES_FILE': None, 'CURL_ARGS': ['--silent', '--location', '--compressed'], 'CURL_BINARY': 'curl', 'CURL_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/dev (+https://github.com/ArchiveBox/ArchiveBox/) curl/curl 7.81.0 (x86_64-pc-linux-gnu)', 'CURL_VERSION': 'curl 7.81.0 (x86_64-pc-linux-gnu)', 'CUSTOM_TEMPLATES_DIR': None, 'DATA_LOCATIONS': {'ARCHIVE_DIR': {'enabled': True, 'is_mount': False, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/arch 'CONFIG_FILE': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Arch 'LOGS_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/logs' 'OUTPUT_DIR': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')}, 'SOURCES_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/sour 'SQL_INDEX': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/inde 'DEBUG': False, 'DEPENDENCIES': {'ARCHIVEBOX_BINARY': {'enabled': True, 'hash':*

archivebox.config.**check_migrations**(*out_dir: str | ~pathlib.Path | None = None, config: ~archivebox.config_stubs.ConfigDict = {'ADMIN_PASSWORD': None, 'ADMIN_USERNAME': None, 'ALLOWED_HOSTS': '*', 'ANSI': {'black': '', 'blue': '', 'green': '', 'lightblue': '', 'lightred': '', 'lightyellow': '', 'red': '', 'reset': '', 'white': ''}, 'ARCHIVEBOX_BINARY': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.12/s packages/sphinx/__main__.py', 'ARCHIVE_DIR': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archiv 'BIND_ADDR': '127.0.0.1:8000', 'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_OPTIONS': {'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'RESOLUTION': '1440,2000', 'TIMEOUT': 60}, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'CHROME_VERSION': None, 'CODE_LOCATIONS': {'CUSTOM_TEMPLATES_DIR': {'enabled': False, 'is_valid': None, 'path': None}, 'PACKAGE_DIR': {'enabled': True, 'is_valid': True, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archiv 'TEMPLATES_DIR': {'enabled': True, 'is_valid': True, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archiv 'COMMIT_HASH': '0bd83076db5ddfe23857b0bcfc7028fa72cf2edf', 'CONFIG_FILE': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Archi 'COOKIES_FILE': None, 'CURL_ARGS': ['--silent', '--location', '--compressed'], 'CURL_BINARY': 'curl', 'CURL_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/dev (+https://github.com/ArchiveBox/ArchiveBox/) curl/curl 7.81.0 (x86_64-pc-linux-gnu)', 'CURL_VERSION': 'curl 7.81.0 (x86_64-pc-linux-gnu)', 'CUSTOM_TEMPLATES_DIR': None, 'DATA_LOCATIONS': {'ARCHIVE_DIR': {'enabled': True, 'is_mount': False, 'is_valid': False, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archiv 'CONFIG_FILE': {'enabled': True, 'is_valid': False, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/Archi 'LOGS_DIR': {'enabled': True, 'is_valid': False, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/logs') 'OUTPUT_DIR': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')}, 'SOURCES_DIR': {'enabled': True, 'is_valid': False, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/source 'SQL_INDEX': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/index. 'DEBUG': False, 'DEPENDENCIES': {'ARCHIVEBOX_BINARY': {'enabled': True, 'hash': 'md5:18bc286a2f948a74a8d0bb67650f7714', 'is_valid': True, 'path':*

archivebox.config.**TERM_WIDTH**()

archivebox.config.**setup_django**(*out_dir: ~pathlib.Path = None, check_db=False, config: ~archivebox.config_stubs.ConfigDict = {'ADMIN_PASSWORD': None, 'ADMIN_USERNAME': None, 'ALLOWED_HOSTS': '*', 'ANSI': {'black': '', 'blue': '', 'green': '', 'lightblue': '', 'lightred': '', 'lightyellow': '', 'red': '', 'reset': '', 'white': ''}, 'ARCHIVEBOX_BINARY': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.12/site-packages/sphinx/__main__.py', 'ARCHIVE_DIR': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archive'), 'BIND_ADDR': '127.0.0.1:8000', 'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_OPTIONS': {'CHECK_SSL_VALIDITY': True, 'CHROME_BINARY': None, 'CHROME_HEADLESS': True, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'RESOLUTION': '1440,2000', 'TIMEOUT': 60}, 'CHROME_SANDBOX': True, 'CHROME_TIMEOUT': 0, 'CHROME_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/{VERSION} (+https://github.com/ArchiveBox/ArchiveBox/)', 'CHROME_USER_DATA_DIR': None, 'CHROME_VERSION': None, 'CODE_LOCATIONS': {'CUSTOM_TEMPLATES_DIR': {'enabled': False, 'is_valid': None, 'path': None}, 'PACKAGE_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archivebox 'TEMPLATES_DIR': {'enabled': True, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archivebox/ 'COMMIT_HASH': '0bd83076db5ddfe23857b0bcfc7028fa72cf2edf', 'CONFIG_FILE': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ArchiveBox 'COOKIES_FILE': None, 'CURL_ARGS': ['--silent', '--location', '--compressed'], 'CURL_BINARY': 'curl', 'CURL_USER_AGENT': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 ArchiveBox/dev (+https://github.com/ArchiveBox/ArchiveBox/) curl/curl 7.81.0 (x86_64-pc-linux-gnu)', 'CURL_VERSION': 'curl 7.81.0 (x86_64-pc-linux-gnu)', 'CUSTOM_TEMPLATES_DIR': None, 'DATA_LOCATIONS': {'ARCHIVE_DIR': {'enabled': True, 'is_mount': False, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/archive')}, 'CONFIG_FILE': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/ArchiveBox 'LOGS_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/logs')}, 'OUTPUT_DIR': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')}, 'SOURCES_DIR': {'enabled': True, 'is_valid': False, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/sources')}, 'SQL_INDEX': {'enabled': True, 'is_mount': False, 'is_valid': True, 'path': PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev/index.sqlite 'DEBUG': False, 'DEPENDENCIES': {'ARCHIVEBOX_BINARY': {'enabled': True, 'hash': 'md5:18bc286a2f948a74a8d0bb67650f7714', 'is_valid': True, 'path': '/home/docs/checkouts/readthedocs.org/user_builds/archivebox/envs/dev/lib/python3.12/site-*

archivebox.config.stubs module

class archivebox.config_stubs.**BaseConfig**(**args*, ***kwargs*)

Bases: dict

class archivebox.config_stubs.**ConfigDict**(**args*, ***kwargs*)

Bases: dict

Regenerate by pasting this quine into *archivebox shell* from archivebox.config import ConfigDict, CONFIG_DEFAULTS print('class ConfigDict(BaseConfig, total=False):') print(' ' + '"""*3 + ConfigDict.__doc__ + '"""*3) for section, configs in CONFIG_DEFAULTS.items():

for key, attrs in configs.items():
Type, default = attrs['type'], attrs['default'] if default is None:

print(f' {key}: Optional[{Type.__name__}]')

else:
print(f' {key}: {Type.__name__}')

print()

IS_TTY: bool

USE_COLOR: bool

SHOW_PROGRESS: bool

IN_DOCKER: bool

PACKAGE_DIR: Path

OUTPUT_DIR: Path

CONFIG_FILE: Path

ONLY_NEW: bool

TIMEOUT: int

MEDIA_TIMEOUT: int

OUTPUT_PERMISSIONS: str

RESTRICT_FILE_NAMES: str

URL_DENYLIST: str

SECRET_KEY: str | None

BIND_ADDR: str

ALLOWED_HOSTS: str

DEBUG: bool

PUBLIC_INDEX: bool

PUBLIC_SNAPSHOTS: bool

FOOTER_INFO: str

SAVE_TITLE: bool

SAVE_FAVICON: bool

SAVE_WGET: bool

SAVE_WGET_REQUISITES: bool

SAVE_SINGLEFILE: bool

SAVE_READABILITY: bool

SAVE_MERCURY: bool

SAVE_PDF: bool

SAVE_SCREENSHOT: bool

SAVE_DOM: bool

SAVE_WARC: bool

SAVE_GIT: bool

SAVE_MEDIA: bool

SAVE_ARCHIVE_DOT_ORG: bool

RESOLUTION: str

GIT_DOMAINS: str

CHECK_SSL_VALIDITY: bool

CURL_USER_AGENT: str

WGET_USER_AGENT: str

CHROME_USER_AGENT: str

COOKIES_FILE: str | Path | None

CHROME_USER_DATA_DIR: str | Path | None

CHROME_TIMEOUT: int

CHROME_HEADLESS: bool

CHROME_SANDBOX: bool

USE_CURL: bool

USE_WGET: bool

USE_SINGLEFILE: bool

USE_READABILITY: bool

USE_MERCURY: bool

USE_GIT: bool

USE_CHROME: bool

USE_YOUTUBEDL: bool

CURL_BINARY: str

GIT_BINARY: str

WGET_BINARY: str

SINGLEFILE_BINARY: str

READABILITY_BINARY: str

MERCURY_BINARY: str

YOUTUBEDL_BINARY: str

CHROME_BINARY: str | None

YOUTUBEDL_ARGS: List[str]

WGET_ARGS: List[str]

CURL_ARGS: List[str]

GIT_ARGS: List[str]

TAG_SEPARATOR_PATTERN: str

class archivebox.config_stubs.**ConfigDefault**(**args*, ***kwargs*)

Bases: dict

default: str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[*ConfigDict*], str | bool | int | None | Pattern | Dict[str, Any] | Dict[str, str | bool | int | None | Pattern | Dict[str, Any]] | Callable[[], str | bool | int | None | Pattern | Dict[str, Any]]]

type: Type | None

aliases: Tuple[str, ...] | None

archivebox.core package

Subpackages

archivebox.core.migrations package

Submodules

archivebox.core.migrations.0001_initial module

class archivebox.core.migrations.0001_initial.**Migration**(*name*, *app_label*)

Bases: Migration

initial = True

dependencies = []

operations = [<CreateModel name='Snapshot', fields=[('id', <django.db.models.fields.UUIDField>), ('url', <django.db.models.fields.URLField>), ('timestamp', <django.db.models.fields.CharField>), ('title', <django.db.models.fields.CharField>), ('tags', <django.db.models.fields.CharField>), ('added', <django.db.models.fields.DateTimeField>), ('updated', <django.db.models.fields.DateTimeField>)]>]

Module contents

Submodules

archivebox.core.admin module

class archivebox.core.admin.**ArchiveResultInline**(*parent_model*, *admin_site*)

Bases: TabularInline

model

alias of ArchiveResult

property media

class archivebox.core.admin.**TagInline**(*parent_model*, *admin_site*)

Bases: TabularInline

model

alias of Snapshot_tags

property media

class archivebox.core.admin.**AutocompleteTags**

Bases: object

model

alias of Tag

search_fields = ['name']

class archivebox.core.admin.**AutocompleteTagsAdminStub**

Bases: object

name = 'admin'

class archivebox.core.admin.**SnapshotActionForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

Bases: ActionForm

base_fields = {'action': <django.forms.fields.ChoiceField object>, 'select_across': <django.forms.fields.BooleanField object>, 'tags': <django.forms.models.ModelMultipleChoiceField object>}

declared_fields = {'action': <django.forms.fields.ChoiceField object>, 'select_across': <django.forms.fields.BooleanField object>, 'tags': <django.forms.models.ModelMultipleChoiceField object>}

property media

Return all media required to render the widgets on this form.

class archivebox.core.admin.SnapshotAdmin(*model*, *admin_site*)

Bases: SearchResultsAdminMixin, ModelAdmin

list_display = ('added', 'title_str', 'files', 'size', 'url_str')

sort_fields = ('title_str', 'url_str', 'added', 'files')

readonly_fields = ('info', 'bookmarked', 'added', 'updated')

search_fields = ('id', 'url', 'timestamp', 'title', 'tags__name')

fields = ('timestamp', 'url', 'title', 'tags', 'info', 'bookmarked', 'added', 'updated')

list_filter = ('added', 'updated', 'tags', 'archiveresult__status')

ordering = ['-added']

actions = ['add_tags', 'remove_tags', 'update_titles', 'update_snapshots', 'resnapshot_snapshot', 'overwrite_snapshots', 'delete_snapshots']

autocomplete_fields = ['tags']

inlines = [<class 'archivebox.core.admin.ArchiveResultInline'>]

list_per_page = 40

action_form

alias of *SnapshotActionForm*

get_urls()

get_queryset(*request*)

Return a QuerySet of all model instances that can be edited by the admin site. This is used by changelist_view.

tag_list(*obj*)

info(*obj*)

title_str(*obj*)

files(*obj*)

size(*obj*)

url_str(*obj*)

grid_view(*request*, *extra_context=None*)

update_snapshots(*request*, *queryset*)

update_titles(*request*, *queryset*)

resnapshot_snapshot(*request*, *queryset*)

overwrite_snapshots(*request*, *queryset*)

delete_snapshots(*request*, *queryset*)

add_tags(*request*, *queryset*)

remove_tags(*request*, *queryset*)

property media

class archivebox.core.admin.TagAdmin(*model*, *admin_site*)

Bases: ModelAdmin

list_display = ('slug', 'name', 'num_snapshots', 'snapshots', 'id')

sort_fields = ('id', 'name', 'slug')

readonly_fields = ('id', 'num_snapshots', 'snapshots')

search_fields = ('id', 'name', 'slug')

fields = ('id', 'num_snapshots', 'snapshots', 'name', 'slug')

actions = ['delete_selected']

ordering = ['-id']

num_snapshots(*obj*)

snapshots(*obj*)

property media

class archivebox.core.admin.ArchiveResultAdmin(*model*, *admin_site*)

Bases: ModelAdmin

list_display = ('id', 'start_ts', 'extractor', 'snapshot_str', 'tags_str', 'cmd_str', 'status', 'output_str')

sort_fields = ('start_ts', 'extractor', 'status')

readonly_fields = ('id', 'uuid', 'snapshot_str', 'tags_str')

search_fields = ('id', 'uuid', 'snapshot__url', 'extractor', 'output', 'cmd_version', 'cmd', 'snapshot__timestamp')

fields = ('id', 'uuid', 'snapshot_str', 'tags_str', 'snapshot', 'extractor', 'status', 'start_ts', 'end_ts', 'output', 'pwd', 'cmd', 'cmd_version')

autocomplete_fields = ['snapshot']

list_filter = ('status', 'extractor', 'start_ts', 'cmd_version')

ordering = ['-start_ts']

list_per_page = 40

snapshot_str(*obj*)

tags_str(*obj*)

cmd_str(*obj*)

output_str(*obj*)

property media

class archivebox.core.admin.ArchiveBoxAdmin(*name='admin'*)

Bases: AdminSite

site_header = 'ArchiveBox'

index_title = 'Links'

site_title = 'Index'

get_urls()

add_view(*request*)

archivebox.core.apps module

class archivebox.core.apps.CoreConfig(*app_name*, *app_module*)

Bases: AppConfig

name = 'core'

default_auto_field = 'django.db.models.UUIDField'

archivebox.core.settings module

class archivebox.core.settings.NoisyRequestsFilter(*name=''*)

Bases: Filter

filter(*record*)

Determine if the specified record is to be logged.

Returns True if the record should be logged, or False otherwise. If deemed appropriate, the record may be modified in-place.

archivebox.core.tests module

archivebox.core.urls module

archivebox.core.views module

class archivebox.core.views.**HomepageView**(***kwargs*)

Bases: View

get(*request*)

class archivebox.core.views.**SnapshotView**(***kwargs*)

Bases: View

get(*request*, *path*)

class archivebox.core.views.**PublicIndexView**(***kwargs*)

Bases: ListView

template_name = 'public_index.html'

model

alias of Snapshot

paginate_by = 40

ordering = ['-added']

get_context_data(***kwargs*)

Get the context for this view.

get_queryset(***kwargs*)

Return the list of items for this view.

The return value must be an iterable and may be an instance of *QuerySet* in which case *QuerySet* specific behavior will be enabled.

get(**args*, ***kwargs*)

class archivebox.core.views.**AddView**(***kwargs*)

Bases: UserPassesTestMixin, FormView

template_name = 'add.html'

form_class

alias of AddLinkForm

get_initial()

Prefill the AddLinkForm with the 'url' GET parameter

test_func()

get_context_data(***kwargs*)

Insert the form into the context dict.

form_valid(*form*)

If the form is valid, redirect to the supplied URL.

dispatch(*request*, **args*, ***kwargs*)

class archivebox.core.views.**HealthCheckView**(***kwargs*)

Bases: View

A Django view that renders plain text "OK" for service discovery tools

get(*request*)

Handle a GET request

archivebox.core.welcome_message module

archivebox.core.wsgi module

WSGI config for archivebox project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see https://docs.djangoproject.com/en/2.1/howto/deployment/wsgi/

Module contents

archivebox.extractors package

Submodules

archivebox.extractors.archive_org module

archivebox.extractors.archive_org.**should_save_archive_dot_org**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.archive_org.**save_archive_dot_org**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

submit site to archive.org for archiving via their service, save returned archive url

archivebox.extractors.archive_org.**parse_archive_dot_org_response**(*response: bytes*) → Tuple[List[str], List[str]]

archivebox.extractors.dom module

archivebox.extractors.dom.**should_save_dom**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.dom.**save_dom**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

print HTML of site to file using chrome –dump-html

archivebox.extractors.favicon module

archivebox.extractors.favicon.**should_save_favicon**(*link: Link*, *out_dir: str | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.favicon.**save_favicon**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

download site favicon from google's favicon api

archivebox.extractors.git module

archivebox.extractors.git.**should_save_git**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.git.**save_git**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

download full site using git

archivebox.extractors.media module

archivebox.extractors.media.**should_save_media**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.media.**save_media**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 3600*) → *ArchiveResult*

Download playlists or individual video, audio, and subtitles using youtube-dl or yt-dlp

archivebox.extractors.pdf module

archivebox.extractors.pdf.**should_save_pdf**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.pdf.**save_pdf**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

print PDF of site to file using chrome –headless

archivebox.extractors.screenshot module

archivebox.extractors.screenshot.**should_save_screenshot**(*link: Link*, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.screenshot.**save_screenshot**(*link: Link*, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

take screenshot of site using chrome –headless

archivebox.extractors.title module

class archivebox.extractors.title.**TitleParser**(**args*, ***kwargs*)

Bases: HTMLParser

property title

handle_starttag(*tag*, *attrs*)

handle_data(*data*)

handle_endtag(*tag*)

archivebox.extractors.title.**get_html**(*link:* Link, *path: Path*, *timeout: int = 60*) → str

Try to find wget, singlefile and then dom files. If none is found, download the url again.

archivebox.extractors.title.**should_save_title**(*link:* Link, *out_dir: str | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.title.**extract_title_with_regex**(*html*)

archivebox.extractors.title.**save_title**(*link:* Link, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

try to guess the page's title from its content

archivebox.extractors.wget module

archivebox.extractors.wget.**should_save_wget**(*link:* Link, *out_dir: Path | None = None*, *overwrite: bool | None = False*) → bool

archivebox.extractors.wget.**save_wget**(*link:* Link, *out_dir: Path | None = None*, *timeout: int = 60*) → *ArchiveResult*

download full site using wget

archivebox.extractors.wget.**wget_output_path**(*link:* Link) → str | None

calculate the path to the wgetted .html file, since wget may adjust some paths to be different than the base_url path.

See docs on wget –adjust-extension (-E)

Module contents

archivebox.extractors.**get_default_archive_methods**() → List[tuple[str, Callable[[*Link*, Path | None, bool | None], bool], Callable[[*Link*, Path | None, int], *ArchiveResult*]]]

archivebox.extractors.**get_archive_methods_for_link**(*link:* Link) → Iterable[tuple[str, Callable[[*Link*, Path | None, bool | None], bool], Callable[[*Link*, Path | None, int], *ArchiveResult*]]]

archivebox.extractors.**ignore_methods**(*to_ignore: List[str]*) → Iterable[str]

archivebox.extractors.**archive_link**(*link:* Link, *overwrite: bool = False*, *methods: Iterable[str] | None = None*, *out_dir: Path | None = None*) → *Link*

download the DOM, PDF, and a screenshot into a folder named after the link's timestamp

archivebox.extractors.**archive_links**(*all_links: Iterable[*Link*] | QuerySet*, *overwrite: bool = False*, *methods: Iterable[str] | None = None*, *out_dir: Path | None = None*) → List[*Link*]

archivebox.index package

Submodules

archivebox.index.csv module

archivebox.index.csv.**links_to_csv**(*links: List[*Link*]*, *cols: List[str] | None = None*, *header: bool = True*, *separator: str = ','*, *ljust: int = 0*) → str

archivebox.index.csv.**to_csv**(*obj: Any*, *cols: List[str]*, *separator: str = ','*, *ljust: int = 0*) → str

archivebox.index.html module

archivebox.index.html.**parse_html_main_index**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkout*) → Iterator[str]

parse an archive index html file and return the list of urls

archivebox.index.html.**generate_index_from_links**(*links: List[*Link*]*, *with_headers: bool*)

archivebox.index.html.**main_index_template**(*links: List[*Link*]*, *template: str = 'static_index.html'*) → str

render the template for the entire main index

archivebox.index.html.**write_html_link_details**(*link:* Link, *out_dir: str | None = None*) → None

archivebox.index.html.**link_details_template**(*link:* Link) → str

archivebox.index.html.**render_django_template**(*template: str*, *context: Mapping[str, str]*) → str

render a given html template string with the given template content

archivebox.index.html.**snapshot_icons**(*snapshot*) → str

archivebox.index.json module

archivebox.index.json.**generate_json_index_from_links**(*links: List[*Link*]*, *with_headers: bool*)

archivebox.index.json.**parse_json_main_index**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkout*) → Iterator[*Link*]

parse an archive index json file and return the list of links

archivebox.index.json.**write_json_link_details**(*link:* Link, *out_dir: str | None = None*) → None

write a json file with some info about the link

archivebox.index.json.**parse_json_link_details**(*out_dir: Path | str*, *guess: bool | None = False*) → *Link* | None

load the json link index from a given directory

archivebox.index.json.**parse_json_links_details**(*out_dir: Path | str*) → Iterator[*Link*]

read through all the archive data folders and return the parsed links

class archivebox.index.json.**ExtendedEncoder**(***, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: JSONEncoder

Extended json serializer that supports serializing several model fields and objects

default(*obj*)

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

archivebox.index.json.**to_json**(*obj: ~typing.Any*, *indent: int | None = 4*, *sort_keys: bool = True*, *cls=<class 'archivebox.index.json.ExtendedEncoder'>*) → str

archivebox.index.schema module

WARNING: THIS FILE IS ALL LEGACY CODE TO BE REMOVED.

DO NOT ADD ANY NEW FEATURES TO THIS FILE, NEW CODE GOES HERE: core/models.py

exception archivebox.index.schema.**ArchiveError**(*message*, *hints=None*)

Bases: Exception

class archivebox.index.schema.**ArchiveResult**(*cmd: List[str]*, *pwd: str | None*, *cmd_version: str | None*, *output: str | Exception | NoneType*, *status: str*, *start_ts: datetime.datetime*, *end_ts: datetime.datetime*, *index_texts: List[str] | None = None*, *schema: str = 'ArchiveResult'*)

Bases: object

cmd: List[str]

pwd: str | None

cmd_version: str | None

output: str | Exception | None

status: `str`

start_ts: `datetime`

end_ts: `datetime`

index_texts: `List[str] | None = None`

schema: `str = 'ArchiveResult'`

typecheck() → None

classmethod guess_ts(*dict_info*)

classmethod from_json(*json_info*, *guess=False*)

to_dict(**keys*) → dict

to_json(*indent=4*, *sort_keys=True*) → str

to_csv(*cols: List[str] | None = None*, *separator: str = ','*, *ljust: int = 0*) → str

classmethod field_names()

property duration: `int`

class archivebox.index.schema.**Link**(*timestamp: str, url: str, title: Optional[str], tags: Optional[str], sources: List[str], history: Dict[str, List[archivebox.index.schema.ArchiveResult]] = <factory>, updated: Optional[datetime.datetime] = None, schema: str = 'Link'*)

Bases: `object`

timestamp: `str`

url: `str`

title: `str | None`

tags: `str | None`

sources: `List[str]`

history: `Dict[str, List[ArchiveResult]]`

updated: `datetime | None = None`

schema: `str = 'Link'`

overwrite(***kwargs*)

pure functional version of dict.update that returns a new instance

typecheck() → None

as_snapshot()

classmethod from_json(*json_info*, *guess=False*)

to_json(*indent=4*, *sort_keys=True*) → str

to_csv(*cols: List[str] | None = None*, *separator: str = ','*, *ljust: int = 0*) → str

snapshot_id

classmethod field_names()

property link_dir: str

property archive_path: str

property archive_size: float

property url_hash

property scheme: str

property extension: str

property domain: str

property path: str

property basename: str

property base_url: str

property bookmarked_date: str | None

property updated_date: str | None

property archive_dates: List[datetime]

property oldest_archive_date: datetime | None

property newest_archive_date: datetime | None

property num_outputs: int

property num_failures: int

property is_static: bool

property is_archived: bool

latest_outputs(*status: str = None*) → Dict[str, str | Exception | None]

get the latest output that each archive method produced for link

canonical_outputs() → Dict[str, str | None]

predict the expected output paths that should be present after archiving

archivebox.index.sql module

archivebox.index.sql.**parse_sql_main_index**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/a*
→ Iterator[*Link*]

archivebox.index.sql.**remove_from_sql_main_index**(*snapshots: QuerySet*, *atomic: bool = False*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/che*
→ None

archivebox.index.sql.**write_link_to_sql_index**(*link:* Link)

archivebox.index.sql.**write_sql_main_index**(*links: List[*Link*]*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/a*) → None

archivebox.index.sql.**write_sql_link_details**(*link:* Link, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkout*) → None

archivebox.index.sql.**list_migrations**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[Tuple[bool, str]]

archivebox.index.sql.**apply_migrations**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[str]

archivebox.index.sql.**get_admins**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[str]

Module contents

archivebox.index.**merge_links**(*a:* Link, *b:* Link) → *Link*

deterministially merge two links, favoring longer field values over shorter, and "cleaner" values over worse ones.

archivebox.index.**validate_links**(*links: Iterable[*Link*]*) → List[*Link*]

archivebox.index.**archivable_links**(*links: Iterable[*Link*]*) → Iterable[*Link*]

remove chrome://, about:// or other schemed links that cant be archived

archivebox.index.**fix_duplicate_links**(*sorted_links: Iterable[*Link*]*) → Iterable[*Link*]

ensures that all non-duplicate links have monotonically increasing timestamps

archivebox.index.**sorted_links**(*links: Iterable[*Link*]*) → Iterable[*Link*]

archivebox.index.**links_after_timestamp**(*links: Iterable[*Link*]*, *resume: float | None = None*) → Iterable[*Link*]

archivebox.index.**lowest_uniq_timestamp**(*used_timestamps: OrderedDict*, *timestamp: str*) → str

resolve duplicate timestamps by appending a decimal 1234, 1234 -> 1234.1, 1234.2

archivebox.index.**timed_index_update**(*out_path: Path*)

archivebox.index.**write_main_index**(*links: List[*Link*]*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Writes links to sqlite3 file for a given list of links

archivebox.index.**load_main_index**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*, *warn: bool = True*) → List[*Link*]

parse and load existing index with any new links from import_path merged in

archivebox.index.**load_main_index_meta**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → dict | None

archivebox.index.**parse_links_from_source**(*source_path: str*, *root_url: str | None = None*, *parser: str = 'auto'*) → Tuple[List[*Link*], List[*Link*]]

archivebox.index.**fix_duplicate_links_in_index**(*snapshots: QuerySet*, *links: Iterable[*Link*]*) → Iterable[*Link*]

Given a list of in-memory Links, dedupe and merge them with any conflicting Snapshots in the DB.

archivebox.index.**dedupe_links**(*snapshots: QuerySet*, *new_links: List[*Link*]*) → List[*Link*]

The validation of links happened at a different stage. This method will focus on actual deduplication and timestamp fixing.

archivebox.index.**write_link_details**(*link:* Link, *out_dir: str | None = None*, *skip_sql_index: bool = False*) → None

archivebox.index.**load_link_details**(*link:* Link, *out_dir: str | None = None*) → *Link*

check for an existing link archive in the given directory, and load+merge it into the given link dict

archivebox.index.**q_filter**(*snapshots: QuerySet*, *filter_patterns: List[str]*, *filter_type: str = 'exact'*) → QuerySet

archivebox.index.**search_filter**(*snapshots: QuerySet*, *filter_patterns: List[str]*, *filter_type: str = 'search'*) → QuerySet

archivebox.index.**snapshot_filter**(*snapshots: QuerySet*, *filter_patterns: List[str]*, *filter_type: str = 'exact'*) → QuerySet

archivebox.index.**get_indexed_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

indexed links without checking archive status or data directory validity

archivebox.index.**get_archived_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

indexed links that are archived with a valid data directory

archivebox.index.**get_unarchived_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev*) → Dict[str, *Link* | None]

indexed links that are unarchived with no data directory or an empty data directory

archivebox.index.**get_present_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs that actually exist in the archive/ folder

archivebox.index.**get_valid_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs with a valid index matched to the main index and archived content

archivebox.index.**get_invalid_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs that are invalid for any reason: corrupted/duplicate/orphaned/unrecognized

archivebox.index.**get_duplicate_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs that conflict with other directories that have the same link URL or timestamp

archivebox.index.**get_orphaned_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs that contain a valid index but aren't listed in the main index

archivebox.index.**get_corrupted_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

dirs that don't contain a valid index and aren't listed in the main index

archivebox.index.**get_unrecognized_folders**(*snapshots*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/a*) → Dict[str, *Link* | None]

dirs that don't contain recognizable archive data and aren't listed in the main index

archivebox.index.**is_valid**(*link:* Link) → bool

archivebox.index.**is_corrupt**(*link:* Link) → bool

archivebox.index.**is_archived**(*link:* Link) → bool

archivebox.index.**is_unarchived**(*link:* Link) → bool

archivebox.index.**fix_invalid_folder_locations**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkc*) → Tuple[List[str], List[str]]

archivebox.parsers package

Submodules

archivebox.parsers.generic_json module

archivebox.parsers.generic_json.**parse_generic_json_export**(*json_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse JSON-format bookmarks export files (produced by pinboard.in/export/, or wallabag)

archivebox.parsers.generic_json.**PARSER**(*json_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse JSON-format bookmarks export files (produced by pinboard.in/export/, or wallabag)

archivebox.parsers.generic_rss module

archivebox.parsers.generic_rss.**parse_generic_rss_export**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse RSS XML-format files into links

archivebox.parsers.generic_rss.**PARSER**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse RSS XML-format files into links

archivebox.parsers.generic_txt module

archivebox.parsers.generic_txt.**parse_generic_txt_export**(*text_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse links from a text file, ignoring other text

archivebox.parsers.generic_txt.**PARSER**(*text_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse links from a text file, ignoring other text

archivebox.parsers.medium_rss module

archivebox.parsers.medium_rss.**parse_medium_rss_export**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Medium RSS feed files into links

archivebox.parsers.medium_rss.**PARSER**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Medium RSS feed files into links

archivebox.parsers.netscape_html module

archivebox.parsers.netscape_html.**parse_netscape_html_export**(*html_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse netscape-format bookmarks export files (produced by all browsers)

archivebox.parsers.netscape_html.**PARSER**(*html_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse netscape-format bookmarks export files (produced by all browsers)

archivebox.parsers.pinboard_rss module

archivebox.parsers.pinboard_rss.**parse_pinboard_rss_export**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Pinboard RSS feed files into links

archivebox.parsers.pinboard_rss.**PARSER**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Pinboard RSS feed files into links

archivebox.parsers.pocket_html module

archivebox.parsers.pocket_html.**parse_pocket_html_export**(*html_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Pocket-format bookmarks export files (produced by getpocket.com/export/)

archivebox.parsers.pocket_html.**PARSER**(*html_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Pocket-format bookmarks export files (produced by getpocket.com/export/)

archivebox.parsers.shaarli_rss module

archivebox.parsers.shaarli_rss.**parse_shaarli_rss_export**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Shaarli-specific RSS XML-format files into links

archivebox.parsers.shaarli_rss.**PARSER**(*rss_file: IO[str]*, ***_kwargs*) → Iterable[*Link*]

Parse Shaarli-specific RSS XML-format files into links

Module contents

Everything related to parsing links from input sources.

For a list of supported services, see the README.md. For examples of supported import formats see tests/.

archivebox.parsers.**parse_links_memory**(*urls: List[str]*, *root_url: str | None = None*)

parse a list of URLS without touching the filesystem

archivebox.parsers.**parse_links**(*source_file: str*, *root_url: str | None = None*, *parser: str = 'auto'*) → Tuple[List[*Link*], str]

parse a list of URLs with their metadata from an RSS feed, bookmarks export, or text file

archivebox.parsers.**run_parser_functions**(*to_parse: IO[str]*, *timer*, *root_url: str | None = None*, *parser: str = 'auto'*) → Tuple[List[*Link*], str | None]

archivebox.parsers.**save_text_as_source**(*raw_text: str*, *filename: str = '{ts}-stdin.txt'*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → str

archivebox.parsers.**save_file_as_source**(*path: str*, *timeout: int = 60*, *filename: str = '{ts}-{basename}.txt'*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → str

download a given url's content into output/sources/domain-<timestamp>.txt

Submodules

archivebox.main module

archivebox.main.**help**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Print the ArchiveBox help message and usage

archivebox.main.**version**(*quiet: bool = False*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Print the ArchiveBox version and dependency information

archivebox.main.**run**(*subcommand: str*, *subcommand_args: List[str] | None*, *stdin: IO | None = None*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Run a given ArchiveBox subcommand with the given list of args

archivebox.main.**init**(*force: bool = False*, *quick: bool = False*, *setup: bool = False*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Initialize a new ArchiveBox collection in the current directory

archivebox.main.**status**(*out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Print out some info and statistics about the archive collection

archivebox.main.**oneshot**(*url: str*, *extractors: str = ''*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*)

Create a single URL archive folder with an index.json and index.html, and all the archive method outputs. You can run this to archive single pages without needing to create a whole collection with archivebox init.

archivebox.main.**add**(*urls: str | List[str]*, *tag: str = ''*, *depth: int = 0*, *update: bool = False*, *update_all: bool = False*, *index_only: bool = False*, *overwrite: bool = False*, *init: bool = False*, *extractors: str = ''*, *parser: str = 'auto'*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[*Link*]

Add a new URL or list of URLs to your archive

archivebox.main.**remove**(*filter_str: str | None = None*, *filter_patterns: List[str] | None = None*, *filter_type: str = 'exact'*, *snapshots: QuerySet | None = None*, *after: float | None = None*, *before: float | None = None*, *yes: bool = False*, *delete: bool = False*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[*Link*]

Remove the specified URLs from the archive

archivebox.main.**update**(*resume: float | None = None*, *only_new: bool = True*, *index_only: bool = False*, *overwrite: bool = False*, *filter_patterns_str: str | None = None*, *filter_patterns: List[str] | None = None*, *filter_type: str | None = None*, *status: str | None = None*, *after: str | None = None*, *before: str | None = None*, *extractors: str = ''*, *out_dir: Path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → List[*Link*]

Import any new links from subscriptions and retry any previously failed/skipped links

archivebox.main.**list_all**(*filter_patterns_str: str | None = None*, *filter_patterns: List[str] | None = None*, *filter_type: str = 'exact'*, *status: str | None = None*, *after: float | None = None*, *before: float | None = None*, *sort: str | None = None*, *csv: str | None = None*, *json: bool = False*, *html: bool = False*, *with_headers: bool = False*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Iterable[*Link*]

List, filter, and export information about archive entries

archivebox.main.**list_links**(*snapshots: QuerySet | None = None*, *filter_patterns: List[str] | None = None*, *filter_type: str = 'exact'*, *after: float | None = None*, *before: float | None = None*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Iterable[*Link*]

archivebox.main.**list_folders**(*links: List[*Link*]*, *status: str*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → Dict[str, *Link* | None]

archivebox.main.**setup**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Automatically install all ArchiveBox dependencies and extras

archivebox.main.**config**(*config_options_str: str | None = None*, *config_options: List[str] | None = None*, *get: bool = False*, *set: bool = False*, *reset: bool = False*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Get and set your ArchiveBox project configuration values

archivebox.main.**schedule**(*add: bool = False*, *show: bool = False*, *clear: bool = False*, *foreground: bool = False*, *run_all: bool = False*, *quiet: bool = False*, *every: str | None = None*, *depth: int = 0*, *overwrite: bool = False*, *update: bool = False*, *import_path: str | None = None*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*)

Set ArchiveBox to regularly import URLs at specific times using cron

archivebox.main.**server**(*runserver_args: List[str] | None = None*, *reload: bool = False*, *debug: bool = False*, *init: bool = False*, *quick_init: bool = False*, *createsuperuser: bool = False*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Run the ArchiveBox HTTP server

archivebox.main.**manage**(*args: List[str] | None = None*, *out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Run an ArchiveBox Django management command

archivebox.main.**shell**(*out_dir: Path = Posix-Path('/home/docs/checkouts/readthedocs.org/user_builds/archivebox/checkouts/dev')*) → None

Enter an interactive ArchiveBox Django shell

archivebox.manage module

archivebox.system module

archivebox.system.**run**(*cmd*, **args*, *input=None*, *capture_output=True*, *timeout=None*, *check=False*, *text=False*, *start_new_session=True*, ***kwargs*)

Patched of subprocess.run to kill forked child subprocesses and fix blocking io making timeout=innefective Mostly copied from https://github.com/python/cpython/blob/master/Lib/subprocess.py

archivebox.system.**atomic_write**(*path: Path | str*, *contents: dict | str | bytes*, *overwrite: bool = True*) → None

Safe atomic write to filesystem by writing to temp file + atomic rename

archivebox.system.**chmod_file**(*path: str*, *cwd: str = '.'*) → None

chmod -R <permissions> <cwd>/<path>

archivebox.system.**copy_and_overwrite**(*from_path: str | Path*, *to_path: str | Path*)

copy a given file or directory to a given path, overwriting the destination

archivebox.system.**get_dir_size**(*path: str | Path*, *recursive: bool = True*, *pattern: str | None = None*) → Tuple[int, int, int]

get the total disk size of a given directory, optionally summing up recursively and limiting to a given filter list

archivebox.system.**dedupe_cron_jobs**(*cron: CronTab*) → CronTab

class archivebox.system.**suppress_output**(*stdout=True*, *stderr=True*)

Bases: `object`

A context manager for doing a "deep suppression" of stdout and stderr in Python, i.e. will suppress all print, even if the print originates in a compiled C/Fortran sub-function.

This will not suppress raised exceptions, since exceptions are printed to stderr just before a script exits, and after the context manager has exited (at least, I think that is why it lets exceptions through).

with suppress_stdout_stderr():
rogue_function()

archivebox.util module

archivebox.util.**detect_encoding**(*rawdata*)

archivebox.util.**scheme**(*url*)

archivebox.util.**without_scheme**(*url*)

archivebox.util.**without_query**(*url*)

archivebox.util.**without_fragment**(*url*)

archivebox.util.**without_path**(*url*)

archivebox.util.**path**(*url*)

archivebox.util.**basename**(*url*)

archivebox.util.**domain**(*url*)

archivebox.util.**query**(*url*)

archivebox.util.**fragment**(*url*)

archivebox.util.**extension**(*url*)

archivebox.util.**base_url**(*url*)

archivebox.util.**without_www**(*url*)

archivebox.util.**without_trailing_slash**(*url*)

archivebox.util.**hashurl**(*url*)

archivebox.util.**urlencode**(*s*)

archivebox.util.**urldecode**(*s*)

archivebox.util.**htmlencode**(*s*)

archivebox.util.**htmldecode**(*s*)

archivebox.util.**short_ts**(*ts*)

archivebox.util.**ts_to_date_str**(*ts*)

archivebox.util.**ts_to_iso**(*ts*)

archivebox.util.**is_static_file**(*url: str*)

archivebox.util.**enforce_types**(*func*)
Enforce function arg and kwarg types at runtime using its python3 type hints

archivebox.util.**docstring**(*text: str | None*)
attach the given docstring to the decorated function

archivebox.util.**str_between**(*string: str*, *start: str*, *end: str = None*) → str
(<abc>12345</def>, <abc>, </def>) -> 12345

archivebox.util.**parse_date**(*date: Any*) → datetime | None
Parse unix timestamps, iso format, and human-readable strings

archivebox.util.**download_url**(*url: str*, *timeout: int = None*) → str
Download the contents of a remote url and return the text

archivebox.util.**get_headers**(*url: str*, *timeout: int = None*) → str
Download the contents of a remote url and return the headers

archivebox.util.**chrome_args**(***options*) → List[str]
helper to build up a chrome shell command with arguments

archivebox.util.**chrome_cleanup**()
Cleans up any state or runtime files that chrome leaves behind when killed by a timeout or other error

archivebox.util.**ansi_to_html**(*text*)
Based on: https://stackoverflow.com/questions/19212665/python-converting-ansi-color-codes-to-html

class archivebox.util.**AttributeDict**(**args*, ***kwargs*)
Bases: dict

Helper to allow accessing dict values via Example.key or Example['key']

class archivebox.util.**ExtendedEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: JSONEncoder

Extended json serializer that supports serializing several model fields and objects

default(*obj*)

Implement this method in a subclass such that it returns a serializable object for o, or calls the base implementation (to raise a TypeError).

For example, to support arbitrary iterators, you could implement default like this:

```
def default(self, o):
    try:
        iterable = iter(o)
    except TypeError:
        pass
    else:
        return list(iterable)
    # Let the base class default method raise the TypeError
    return JSONEncoder.default(self, o)
```

Module contents

1.5 Meta

1.5.1 Roadmap

Comment here to discuss the contribution roadmap:Official Roadmap Discussion.

Planned Specification

(this is not set in stone, just a rough estimate)

v0.7: Schema improvements

- move config loading logic into settings.py
- move all the extractors into "plugin" style folders that register their own config
- right now, the paths of the extractor output are scattered all over the codebase, e.g. `output.pdf` (should be moved to constants at the top of the plugin config file)
- make out_dir, link_dir, extractor_dir, naming consistent across codebase
- remove `timestamps` as primary keys in favor of hashes, UUIDs, or some other slug
- create a migration system for folder layout independent of the index (`mv` is atomic at the FS level, so we just need a `transaction.atomic(): move(oldpath, newpath); snap.data_dir = newpath; snap.save()`)
- make `Tag` a real model `ManyToMany` with Snapshots

- allow multiple Snapshots of the same site over time + CLI / UI to manage those, + migration from old style `#2020-01-01` hack to proper versioned snapshots

v0.8: Security

- Add CSRF/CSP/XSS protection to rendered archive pages
- Provide secure reverse proxy in front of archivebox server in docker-compose.yml
- Create UX flow for users to setup session cookies / auth for archiving private sites
  - cookies for wget, curl, etc low-level commands
  - localstorage, cookies, indexedb setup for chrome archiving methods

v0.9: Performance

- setup huey, break up archiving process into tasks on a queue that a worker pool executes
- setup pyppeteer2 to wrap chrome so that it's not open/closed during each extractor

v1.0: Full headless browser control

- run user-scripts / extensions in the context of the page during archiving
- community userscripts for unrolling twitter threads, reddit threads, youtube comment sections, etc.
- pywb-based headless browser session recording and warc replay
- archive proxy support
  - support sending upstream requests through an external proxy
  - support for exposing a proxy that archives all downstream traffic

...

v2.0 Federated or distributed archiving + paid hosted service offering

- ZFS / merkel tree for storing archive output subresource hashes
- DHT for assigning merkel tree hash:file shards to nodes
- tag system for tagging certain hashes with human-readable names, e.g. title, url, tags, filetype etc.
- distributed tag lookup system

Major long-term changes

- release **`pip`, `apt`, `pkg`, and `brew` packaged distributions** for installing ArchiveBox
- add an **optional web GUI** for managing sources, adding new links, and viewing the archive
- switch to django + **sqlite db with migrations system** & json/html export for managing archive schema changes and persistence
- modularize internals to allow importing individual components
- switch to sha256 of URL as unique link ID
- support **storing multiple snapshots** of pages over time
- support **custom user puppeteer scripts to run while archiving** (e.g. for expanding reddit threads, scrolling thread on twitter, etc)
- support named collections of archived content with different user access permissions
- support sharing archived assets via DHT + torrent / ipfs / ZeroNet / other sharing system

Smaller planned features

- support pushing pages to multiple 3rd party services using ArchiveNow instead of just archive.org
- body text extraction to markdown (using ~~fathom~~ readability and mercury)
- featured image / thumbnail extraction
- auto-tagging links based on important/frequent keywords in extracted text (like pocket)
- automatic article summary paragraphs from extracted text with nlp summarization library
- full-text search of extracted text with ~~elasticsearch/elasticlunr/ag~~ sonic and ripgrep
- download closed-caption subtitles from Youtube and other video sites (TODO: submit the subtitle files to the full-text search index)
- try pulling dead sites from archive.org and other sources if original is down (https://github.com/hartator/wayback-machine-downloader)
- And more in the issues list…

IMPORTANT: *Please don't work on any of these major long-term tasks without contacting me first, work is already in progress for many of these, and I may have to reject your PR if it doesn't align with the existing work!*

Past Releases

To see how this spec has been scheduled / implemented / released so far, read these pull requests:

- v0.1.x pre-git-history (~2017)
- v0.2.x (~2018/12)
- v0.3.x (~2019/03)
- v0.4.x (~2019/04)
- v0.5.x (~2020/11)

- v0.6.x (~2021/03)
- sabbatical / coding hiatus for 2022 / mostly answered tickets + merged PRs
- v0.7.x next … (planned for 2023)

1.5.2 Changelog

If you're having an issue with a breaking change, or migrating your data between versions, open an issue to get help.

`ArchiveBox` was previously named `Pocket Archive Stream` and then `Bookmark Archiver`.

`THIS PAGE HAS BEEN MOVED`: See the releases page for versioned source downloads and full changelog.

Many thanks to our 100+ contributors and everyone in the web archiving community!

- v0.4.9 released
  - `pip install archivebox` https://pypi.org/project/archivebox/
  - `docker run archivebox/archivebox` https://hub.docker.com/r/archivebox/archivebox
  - https://archivebox.readthedocs.io/en/latest/
  - https://github.com/ArchiveBox/ArchiveBox/releases
- easy migration from previous versions

```
cd path/to/your/archive/folder
archivebox init
archviebox add 'https://example.com'
archviebox add 'https://getpocket.com/users/USERNAME/feed/all' --depth=1
```

- full transition to Django Sqlite DB with migrations (making upgrades between versions much safer now)
- maintains an intuitive and helpful CLI that's backwards-compatible with all previous archivebox data versions
- uses argparse instead of hand-written CLI system: see `archivebox/cli/archivebox.py`
- new subcommands-based CLI for `archivebox` (see below)
- new Web UI with pagination, better search, filtering, permissions, and more
- 30+ assorted bugfixes, new features, and tickets closed
- for more info, see: https://github.com/ArchiveBox/ArchiveBox/releases/tag/v0.4.9

- v0.2.4 released
- better archive corruption guards (check structure invariants on every parse & save)
- remove title prefetching in favor of new FETCH_TITLE archive method
- slightly improved CLI output for parsing and remote url downloading
- re-save index after archiving completes to update titles and urls
- remove redundant derivable data from link json schema
- markdown link parsing support
- faster link parsing and better symbol handling using a new compiled URL_REGEX

- v0.2.3 released
- fixed issues with parsing titles including trailing tags
- fixed issues with titles defaulting to URLs instead of attempting to fetch
- fixed issue where bookmark timestamps from RSS would be ignored and current ts used instead
- fixed issue where ONLY_NEW would overwrite existing links in archive with only new ones
- fixed lots of issues with URL parsing by using `urllib.parse` instead of hand-written lambdas
- ignore robots.txt when using wget (sssshhh don't tell anyone)
- fix RSS parser bailing out when there's whitespace around XML tags
- fix issue with browser history export trying to run ls on wrong directory

---

- v0.2.2 released
- Shaarli RSS export support
- Fix issues with plain text link parsing including quotes, whitespace, and closing tags in URLs
- add USER_AGENT to archive.org submissions so they can track archivebox usage
- remove all icons similar to archive.org branding from archive UI
- hide some of the noisier youtubedl and wget errors
- set permissions on youtubedl media folder
- fix chrome data dir incorrect path and quoting
- better chrome binary finding
- show which parser is used when importing links, show progress when fetching titles

---

- v0.2.1 released with new logo
- ability to import plain lists of links and almost all other raw filetypes
- WARC saving support via wget
- Git repository downloading with git clone
- Media downloading with youtube-dl (video, audio, subtitles, description, playlist, etc)

- v0.2.0 released with new name
- renamed from **Bookmark Archiver** -> **ArchiveBox**

- v0.1.0 released
- support for browser history exporting added with `./bin/archivebox-export-browser-history`
- support for chrome `--dump-dom` to output full page HTML after JS executes

---

- v0.0.3 released
- support for chrome `--user-data-dir` to archive sites that need logins
- fancy individual html & json indexes for each link
- smartly append new links to existing index instead of overwriting

---

- v0.0.2 released
- proper HTML templating instead of format strings (thanks to https://github.com/bardisty!)
- refactored into separate files, wip audio & video archiving

---

- v0.0.1 released
- Index links now work without nginx url rewrites, archive can now be hosted on github pages
- added setup.sh script & docstrings & help commands
- made Chromium the default instead of Google Chrome (yay free software)
- added env-variable configuration (thanks to https://github.com/hannah98!)
- renamed from **Pocket Archive Stream** -> **Bookmark Archiver**
- added Netscape-format export support (thanks to https://github.com/ilvar!)
- added Pinboard-format export support (thanks to https://github.com/sconeyard!)
- front-page of HN, oops! apparently I have users to support now :grin:?
- added Pocket-format export support

- v0.0.0 released: created Pocket Archive Stream 2017/05/05

1.5.3 Supporting Development

Github Sponsors: https://github.com/sponsors/pirate

Patreon: https://www.patreon.com/theSquashSH

Paypal: https://paypal.me/NicholasSweeting

If you want to help keep ArchiveBox alive and flourishing, donate to support more development.

You can also hire me to do corporate installs and prioritize developing your feature requests.

https://Monadical.com also sometimes helps with ArchiveBox development.

If you have any questions or want to partner with this project, contact me at: `hello @ archivebox . io`.

1.5.4 Web Archiving Community

Join us on our new ArchiveBox community chat server: https://Zulip.ArchiveBox.io

Just getting started and want to learn more about why Web Archiving is important? Check out this article: On the Importance of Web Archiving.

The internet archiving community is surprisingly far-reaching and almost universally friendly!

Whether you want to learn which organizations are the big players in the web archiving space, want to find a specific open source tool for your web archiving need, or just want to see where archivists hang out online, this is my attempt at an index of the entire web archiving community.

- *The Master ListsCommunity-maintained indexes of web archiving tools and groups by IIPC, COPTR, ArchiveTeam, Wikipedia, & the ASA.*
- *Web Archiving SoftwareOpen source tools and projects in the internet archiving space.*
  - *Bookmarking Services*
  - *Well-Known Open Source Projects*
  - *Public Archiving Services*
  - *ArchiveBox Alternatives*
  - *Smaller Utilities*
- *Reading ListArticles, posts, and blogs relevant to ArchiveBox and web archiving in general.*
  - *Blogs*
  - *Articles*
  - *ArchiveBox-Specific Posts, Tutorials, and Guides*
  - *ArchiveBox Discussions in News & Social Media*
- *CommunitiesA collection of the most active internet archiving communities and initiatives.*
  - *Most Active Web-Archiving Communities*
  - *Other Web Archiving Communities*
  - *General Archiving Foundations, Coalitions, Initiatives, and Institutes*

The Master Lists

Indexes of archiving institutions and software maintained by other people. If there's anything archivists love doing, it's making lists.

- **COPTR Wiki of Web Archiving Tools (COPTR)**
- **Awesome Web Archiving Tools (IIPC)**
- Spreadsheet Comparison of Archiving Tools (DataTogether)
- Awesome Web Crawling Tools
- Awesome Web Scraping Tools
- ArchiveTeam's List of Software (ArchiveTeam.org)

- List of Web Archiving Initiatives (Wikipedia.org)
- Directory of Archiving Organizations (American Society of Archivists)

Web Archiving Projects

Bookmarking Services

- **Pocket Premium Bookmarking tool that provides an archiving service in their paid version, run by Mozilla**
- **Pinboard Bookmarking tool that provides archiving in a paid version, run by a single independent developer**
- **Raindrop Bookmarking tool with archiving in their paid version, run by a company est. 2011**
- Instapaper Bookmarking alternative to Pocket/Pinboard (with no archiving)
- Wallabag / Wallabag.it Self-hostable web archiving server that can import via RSS
- Shaarli Self-hostable bookmark tagging, archiving, and sharing service
- ReadWise A paid Pocket/Pinboard alternative that includes article snippet and highlight saving
- Diigo Another brookmarking/annotation service with archiving as a paid feature

---

From the Archive.org & Archive-It teams

- **Archive.org The O.G. wayback machine provided publicly by the Internet Archive (Archive.org)**
- **Archive.it commercial Wayback-Machine solution**
- **Heretrix The king of internet archiving crawlers, powers the Wayback Machine**
- **Brozzler chrome headless crawler + WARC archiver maintained by Archive.org**
- WarcProx warc proxy recording and playback utility
- WarcTools utilities for dealing with WARCs
- Grab-Site An easy preconfigured web crawler designed for backing up websites
- WPull A pure python implementation of wget with WARC saving
- More on their Github…

From the Rhizome.org/WebRecorder.io/Conifer team

- **Conifer by Rhizome.org An open-source personal archiving server that uses pywb under the hood** previously known as *Webrecorder.io*
- **Webrecorder.net Suite of open source projects and tools, led by Ilya Kreymer, to capture interactive websites and replay them at a later time as accurately as possible**
- **pywb The python wayback machine, the codebase forked off archive.org that powers webrecorder**

- warcit Create a warc file out of a folder full of assets
- WebArchivePlayer A tool for replaying web archives
- warcio fast streaming asynchronous WARC reader and writer
- More on their Github…

From the Old Dominion University: Web Science Team

- **ipwb A distributed web archiving solution using pywb with ipfs for storage**
- **archivenow tool that pushes urls into all the online archive services like Archive.is and Archive.org**
- node-warc Parse And Create Web ARChive (WARC) files with node.js
- WAIL Web archiver GUI using Heritrix and OpenWayback
- Squidwarc User-scriptable, archival crawler using Chrome
- WAIL (Electron) Electron app version of the original wail for creating and interacting with web archives
- **warcreate a Chrome extension for creating WARCs from any webpage**
- More on their Github…

From the Archives Unleashed Team

- AUT Archives Unleashed Toolkit for analyzing web archives (formerly WarcBase)
- Warclight A Rails engine for finding and searching web archives
- More on their Github…

From the IIPC team

- **OpenWayback Open source project developing core Wayback-Machine components**
- **awesome-web-archiving Large list of archiving projects and orgs**
- JWARC A Java library for reading and writing WARC files.
- More on their Github…

Other Public Archiving Services

- https://perma.cc
- https://www.pagefreezer.com
- https://www.smarsh.com
- https://www.stillio.com
- https://archive.is / https://archive.today
- https://archive.st
- https://arquivo.pt
- http://theoldnet.com
- https://timetravel.mementoweb.org/
- https://freezepage.com/
- https://webcitation.org/archive
- https://archiveofourown.org/
- https://megalodon.jp/
- https://www.webarchive.org.uk/ukwa/
- https://github.com/HelloZeroNet/ZeroNet (super cool project)
- Google, Bing, DuckDuckGo, and other search engine caches

Other ArchiveBox Alternatives

- **browsertrix-crawler / ArchiveWeb.page + ReplayWeb.page + pywb Webrecorder.io's archiving suite has the highest fidelity, and can flawlessly archive YouTube, Twitter, FB and other complex, JS-heavy SPAs**
- **SingleFile Web Extension / CLI util for Firefox and Chrome to save a web page as a single HTML file**
- **Memex by Worldbrain.io a beautiful, user-friendly browser extension that archives all history with full-text search, annotation support, and more**
- **Hypothes.is a web/pdf/ebook annotation tool that also archives content**
- **Reminiscence extremely similar to ArchiveBox, uses a Django backend + UI and provides auto-tagging and summary features with NLTK**
- **Shaarchiver very similar project that archives Firefox, Shaarli, or Delicious bookmarks and all linked media, generating a markdown/HTML index**
- **Archivy Python-based self-hosted knowledge base embedded into your filesystem**
- **Polarized a desktop application for bookmarking, annotating, and archiving articles offline**
- **Diskernet Archiving tool that uses the Chrome debugger protocol to save each page as-loaded in the browser** (aka 22120 by c0fe or i5ik)
- **Photon a fast crawler with archiving and asset extraction support**
- LinkAce A self-hosted bookmark management tool that saves snapshots to archive.org
- Trilium Personal web UI based knowledge-base with web clipping and note-taking

- Herodotus Django-based web archiving tool with a focus on collecting text-based content
- Buku Browser-independent bookmark manager CLI written in Python3 and SQLite3
- ReadableWebProxy A proxying archiver that downloads content from sites and can snapshot multiple versions of sites over time
- Perkeep "Perkeep lets you permanently keep your stuff, for life."
- Fossilo A commercial archiving solution that appears to be very similar to ArchiveBox
- Archivematica web GUI for institutional long-term archiving of web and other content
- Headless Chrome Crawler distributed web crawler built on puppeteer with screenshots
- WWWofle old proxying recorder software similar to ArchiveBox
- Erised Super simple CLI utility to bookmark and archive webpages
- Zotero collect, organize, cite, and share research (mainly for technical/scientific papers & citations)
- TiddlyWiki Non-linear bookmark and note-taking tool with archiving support
- Joplin Desktop + mobile app for knowledge-base-style info collection and notes (w/ optional plugin for archiving)
- Hunchly A paid web archiving / session recording tool design for OSINT
- Monolith CLI tool for saving complete web pages as a single HTML file
- Obelisk Go package and CLI tool for saving web page as single HTML file
- Percollate A command-line tool to turn web pages into beautiful, readable PDF, EPUB, or HTML docs.
- Shiori Simple bookmark manager + readability archiver built with Go (like a clone of Pocket)
- Munin Archiver Social media archiver for Facebook, Instagram and VKontakte accounts.
- **Wayback Archiving in style like ArchiveBox, but with a chat.**

Smaller Utilities

Random helpful utilities for web archiving, WARC creation and replay, and more…

- https://github.com/TheCakeIsNaOH/xbs-to-archivebox A utility to sync xBrowserSync bookmarks with ArchiveBox
- https://github.com/karlicoss/promnesia A browser extension that collects and collates all the URLs you visit into a hierarchical/graph structure with metadata
- https://github.com/vrtdev/save-page-state A Chrome extension for saving the state of a page in multiple formats
- https://github.com/jsvine/waybackpack command-line tool that lets you download the entire Wayback Machine archive for a given URL
- https://github.com/hartator/wayback-machine-downloader Download an entire website from the Internet Archive Wayback Machine.
- https://github.com/Lifesgood123/prevent-link-rot Replace any broken URLs in some content with Wayback machine URL equivalents
- https://en.archivarix.com download an archived page or entire site from the Wayback Machine
- https://proofofexistence.com prove that a certain file existed at a given time using the blockchain
- https://github.com/chfoo/warcat for merging, extracting, and verifying WARC files

- https://github.com/mozilla/readability tool for extracting article contents and text
- https://github.com/mholt/timeliner All your digital life on a single timeline, stored locally
- https://github.com/wkhtmltopdf/wkhtmltopdf Webkit HTML to PDF archiver/saver
- Sheetsee-Pocket project that provides a pretty auto-updating index of your Pocket links (without archiving them)
- Pocket -> IFTTT -> Dropbox Post by Christopher Su on his Pocket saving IFTTT recipe
- http://squidman.net/squidman/index.html
- https://wordpress.org/plugins/broken-link-checker/
- https://github.com/ArchiveTeam/wpull
- http://freedup.org/
- https://en.wikipedia.org/wiki/Furl
- https://preservica.com/digital-archive-software-1/active-digital-preservation For-profit company offering a digital preservation software suite
- https://github.com/karlicoss/grasp capture webpages from Firefox and Chrome into Org-mode documents
- https://github.com/dgtlmoon/changedetection.io Change detection and monitoring of web page content changes
- *And many more on the other lists…*

Reading List

A collection of blog posts and articles about internet archiving, contact me / open an issue if you want to add a link here!

Blogs Friends of ArchiveBox

- https://blog.archive.org
- https://webrecorder.net/blog
- https://netpreserveblog.wordpress.com
- https://blog.conifer.rhizome.org/
- https://ws-dl.blogspot.com
- https://siarchives.si.edu/blog
- https://parameters.ssrc.org
- https://sr.ithaka.org/publications
- https://ait.blog.archive.org
- https://brewster.kahle.org
- https://ianmilligan.ca
- https://medium.com/@giovannidamiola

Articles We Like About Internet Archiving

- https://items.ssrc.org/parameters/on-the-importance-of-web-archiving/
- https://theconversation.com/your-internet-data-is-rotting-115891
- https://www.bbc.com/future/story/20190401-why-theres-so-little-left-of-the-early-internet
- https://sr.ithaka.org/publications/the-state-of-digital-preservation-in-2018/
- https://gizmodo.com/delete-never-the-digital-hoarders-who-collect-tumblrs-1832900423
- https://siarchives.si.edu/blog/we-are-not-alone-progress-digital-preservation-community
- https://www.gwern.net/Archiving-URLs
- http://brewster.kahle.org/2015/08/11/locking-the-web-open-a-call-for-a-distributed-web-2/
- https://lwn.net/Articles/766374/
- https://en.wikipedia.org/wiki/List_of_Web_archiving_initiatives
- https://medium.com/@giovannidamiola/making-the-internet-archives-full-text-search-faster-30fb11574ea9
- https://xkcd.com/1909/
- https://samsaffron.com/archive/2012/06/07/testing-3-million-hyperlinks-lessons-learned#comment-31366
- https://www.gwern.net/docs/linkrot/2011-muflax-backup.pdf
- https://thoughtstreams.io/higgins/permalinking-vs-transience/
- http://ait.blog.archive.org/files/2014/04/archiveit_life_cycle_model.pdf
- https://blog.archive.org/2016/05/26/web-archiving-with-national-libraries/
- https://blog.archive.org/2014/10/28/building-libraries-together/
- https://ianmilligan.ca/2018/03/27/ethics-and-the-archived-web-presentation-the-ethics-of-studying-geocities/
- https://ianmilligan.ca/2018/05/22/new-article-if-these-crawls-could-talk-studying-and-documenting-web-archives-provenance/
- https://ws-dl.blogspot.com/2019/02/2019-02-08-google-is-being-shuttered.html

If any of these links are dead, you can find an archived version on https://archive.sweeting.me or https://web.archive.org.

ArchiveBox-Specific Posts, Tutorials, and Guides

Beware: many of these may be outdated, as ArchiveBox has frequent updates and continual improvement.

- "Install ArchiveBox on SaltBox.dev" https://docs.saltbox.dev/sandbox/apps/archivebox/#3-setup
- "ArchiveBox is an open-source self-hosted web archiving system for the web and the desktop" https://medevel.com/archivebox/
- "Install ArchiveBox on a One-Click Docker Application" https://www.vultr.com/docs/install-archivebox-on-a-oneclick-docker-application/
- "ArchiveBox, una solución para crear nuestro propio Archive.org en miniatura y personalizado" https://www.genbeta.com/herramientas/archivebox-solucion-para-crear-nuestro-propio-archive-org-miniatura-personalizado
- "ArchiveBoxPython" https://www.bilibili.com/s/video/BV1ib4y1X7SL

- “ - ” https://habr.com/ru/company/vdsina/blog/550180/
- “ArchiveBox, una solución para crear nuestro propio Archive.org en miniatura y personalizado” https://www.genbeta.com/herramientas/archivebox-solucion-para-crear-nuestro-propio-archive-org-miniatura-personalizado
- “Preserve the Internet With ArchiveBox” https://www.cyberpunks.com/preserve-the-internet-with-archivebox/
- “ . ArchiveBox” https://xakep.ru/2021/02/01/archivebox/
- “Internet” http://www.diglog.com/story/1045192.html
- “How to Make Your Own Internet Archive With ArchiveBox” https://nixintel.info/osint-tools/make-your-own-internet-archive-with-archive-box/
- “Mit ArchiveBox Webseiten auf der Festplatte archivieren” https://www.linux-community.de/ausgaben/linuxuser/2020/12/mit-archivebox-webseiten-auf-der-festplatte-archivieren/
- “ArchiveBoxWEB” https://zhen.bushini.de/14738.html / https://www.1fishsauce.com/?p=4206
- “: Kiwix & ArchiveBox” https://blog.csdn.net/JackLang/article/details/108328791
- “” https://www.pcpc.me/tech/self-hosted-read-later-app
- “How to install ArchiveBox to preserve websites you care about” https://blog.sleeplessbeastie.eu/2019/06/19/how-to-install-archivebox-to-preserve-websites-you-care-about/
- “How to remotely archive websites using ArchiveBox” https://blog.sleeplessbeastie.eu/2019/06/26/how-to-remotely-archive-websites-using-archivebox/
- “How to Create Your Own Private Self-Hosted Read-It-Later App” https://www.makeuseof.com/tag/self-hosted-read-later-app/
- “How to use CutyCapt inside ArchiveBox” https://blog.sleeplessbeastie.eu/2019/07/10/how-to-use-cutycapt-inside-archivebox/
- “Automate ArchiveBox with Google Spreadsheet to Backup your internet” https://manfred.life/archivebox
- “♪ConoHaArchiveBox” https://qiita.com/CloudRemix/items/691caf91efa3ef19a7ad
- “WEB-ARCHIV TEIL 8: WALLABAG UND ARCHIVEBOX” http://webermartin.net/blog/web-archiv-teil-8-wallabag-und-archivebox/
- https://metaxyntax.neocities.org/entries/7.html

ArchiveBox Discussions in News & Social Media

- **Aggregators:****ProductHunt**, **AlternativeTo**, **SaaSHub**, Logiciels, SteemHunt, Recurse Center: The Joy of Computing, Github Changelog, Dev.To Ultra List, O’Reilly 4 Short Links, JaxEnter
- **Blog Posts & Podcasts:**Korben.info, Defining Desktop Linux Podcast #296 (0:55:00), Binärgewitter Podcast #221, Schrankmonster.de, La Ferme Du Web
- **Hacker News threads and comments:**#1, #2, #3, **#4**, and many more…
- **Reddit r/DataHoarder, r/SelfHosted, etc. posts and comments**:#1, #2, #3, #4, #5 , **#6**, #7, #8, and many more…
- **Twitter:**Python Trending, PyCoder’s Weekly, Python Hub, Smashing Magazine, and many more…

Communities

Most Active Communities

- **The Internet Archive (Archive.org)** (USA)
- **International Internet Preservation Consortium (IIPC)** (International)
- **The Archive Team**, **URL Team**, **r/ArchiveTeam** (International)
- **Rhizome.org** The digital preservation group that works on Conifer by Rhizome formerly Webrecorder.io (USA)
- **Webrecorder.net** Formerly known[1] as Webrecorder.io is a project Led by Ilya Kreymer, that researches and develops web archiving tools, widely used by the community.
- **Old Dominion University: Web Science and Digital Libraries (WS-DL @ ODU)** (Virginia, USA)
- **r/DataHoarder**, **r/Archivists**, **r/DHExchange** (International)
- The Eye Non-profit working on content archival and long-term preservation (Europe)
- Digital Preservation Coalition & their Software Tool Registry (COPTR) (UK & Wales)
- Archives Unleashed Project and UAP Github (Canada)

Web Archiving Communities

Follow these technological and organizational archiving hubs for the latest archiving news.

- Canadian Web Archiving Coalition (Canada)
- Web Archives for Historical Research Group (Canada)
- Smithsonian Institution Archives: Digital Curation (Washington D.C., USA)
- National Digital Stewardship Alliance (NDSA) (USA)
- Digital Library Federation (DLF) (USA)
- Council on Library and Information Resources (CLIR) (USA)
- Digital Curation Centre (DCC) (UK)
- ArchiveMatica & their Community Wiki (International)
- Professional Development Institutes for Digital Preservation (POWRR) (USA)
- Institute of Museum and Library Services (IMLS) (USA)
- Stanford Libraries Web Archiving (USA)
- Society of American Archivists: Electronic Records (SAA) (USA)
- BitCurator Consortium (BCC) (USA)
- Ethics & Archiving the Web Conference (Rhizome/Webrecorder.io) (USA)
- Archivists Round Table of NYC (USA)

General Archiving Foundations, Coalitions, Initiatives, and Institutes

Find your local archiving group in the list and see how you can contribute!

- Community Archives and Heritage Group (UK & Ireland)
- Open Preservation Foundation (OPF) (UK & Europe)
- Software Preservation Network (International)
- ITHAKA, Portico, JSTOR, ARTSTOR, S+R (USA)
- Archives and Records Association (UK & Ireland)
- Arkivrådet AAS (Sweden)
- Asociación Española de Archiveros, Bibliotecarios, Museologos y Documentalistas (ANABAD) (Spain)
- Associação dos Arquivistas Brasileiros (AAB) (Brazil)
- Associação Portuguesa de Bibliotecários, Archivistas e Documentalistas (BAD) (Portugal)
- Association des archivistes français (AAF) (France)
- Associazione Nazionale Archivistica Italiana (ANAI) (Italy)
- Australian Society of Archivists Inc. (Australia)
- International Council on Archives (ICA)
- International Records Management Trust (IRMT)
- Irish Society for Archives (Ireland)
- Koninklijke Vereniging van Archivarissen in Nederland (Netherlands)
- State Archives Administration of the People's Republic of China (China)
- Academy of Certified Archivists
- Archivists and Librarians in the History of the Health Sciences
- Archivists for Congregations of Women Religious
- Archivists of Religious Institutions
- Association of Catholic Diocesan Archivists
- Association of Moving Image Archivists
- Council of State Archivists
- National Association of Government Archives and Records Administrators
- National Episcopal Historians and Archivists
- Archival Education and Research Institute
- Archives Leadership Institute
- Georgia Archives Institute
- Modern Archives Institute
- Western Archives Institute
- Association des archivistes du Québec
- Association of Canadian Archivists
- Canadian Council of Archives/Conseil canadien des archives

- Archives Association of British Columbia
- Archives Association of Ontario
- Archives Council of Prince Edward Island
- Archives Society of Alberta
- Association for Manitoba Archives
- Association of Newfoundland and Labrador Archives
- Council of Nova Scotia Archives
- Réseau des services d'archives du Québec
- Saskatchewan Council for Archives and Archivists

You can find more organizations and initiatives on these other lists:

- Wikipedia.org List of Web Archiving Initiatives
- SAA List of USA & Canada Based Archiving Organizations
- SAA List of International Archiving Organizations
- Digital Preservation Coalition's Member List

ArchiveBox Community Resources

ArchiveBox Chat Rooms

- Official ArchiveBox Zulip Chat Server
- Unofficial ArchiveBox Matrix chat room (old)
- Github Discussions

ArchiveBox on Social Media

- Twitter: @ArchiveBoxApp
- LinkedIn: ArchiveBox
- YouTube: @ArchiveBoxApp
- Reddit: r/ArchiveBox
- Alternative.to
- ReposHub

ArchiveBox on Package Distribution Platforms

- Python PyPI
- Docker Hub
- ArchLinux AUR
- Ubuntu Launchpad PPA

---

^ Back to Top ^

PYTHON MODULE INDEX

a

archivebox, 109
archivebox.cli, 72
archivebox.cli.archivebox_add, 70
archivebox.cli.archivebox_config, 70
archivebox.cli.archivebox_help, 71
archivebox.cli.archivebox_init, 71
archivebox.cli.archivebox_list, 71
archivebox.cli.archivebox_manage, 71
archivebox.cli.archivebox_remove, 71
archivebox.cli.archivebox_schedule, 71
archivebox.cli.archivebox_server, 72
archivebox.cli.archivebox_shell, 72
archivebox.cli.archivebox_status, 71
archivebox.cli.archivebox_update, 72
archivebox.cli.archivebox_version, 72
archivebox.config, 73
archivebox.config_stubs, 85
archivebox.core, 93
archivebox.core.admin, 88
archivebox.core.apps, 91
archivebox.core.migrations, 88
archivebox.core.migrations.0001_initial, 87
archivebox.core.settings, 91
archivebox.core.tests, 92
archivebox.core.urls, 92
archivebox.core.views, 92
archivebox.core.welcome_message, 93
archivebox.core.wsgi, 93
archivebox.extractors, 95
archivebox.extractors.archive_org, 93
archivebox.extractors.dom, 93
archivebox.extractors.favicon, 94
archivebox.extractors.git, 94
archivebox.extractors.media, 94
archivebox.extractors.pdf, 94
archivebox.extractors.screenshot, 94
archivebox.extractors.title, 95
archivebox.extractors.wget, 95
archivebox.index, 100
archivebox.index.csv, 96
archivebox.index.html, 96
archivebox.index.json, 96
archivebox.index.schema, 97
archivebox.index.sql, 99
archivebox.main, 105
archivebox.manage, 107
archivebox.parsers, 104
archivebox.parsers.generic_json, 102
archivebox.parsers.generic_rss, 103
archivebox.parsers.generic_txt, 103
archivebox.parsers.medium_rss, 103
archivebox.parsers.netscape_html, 103
archivebox.parsers.pinboard_rss, 103
archivebox.parsers.pocket_html, 104
archivebox.parsers.shaarli_rss, 104
archivebox.system, 107
archivebox.util, 107

INDEX

A

action_form (*archivebox.core.admin.SnapshotAdmin attribute*), 89
actions (*archivebox.core.admin.SnapshotAdmin attribute*), 89
actions (*archivebox.core.admin.TagAdmin attribute*), 90
add() (*in module archivebox.cli*), 72
add() (*in module archivebox.main*), 105
add_tags() (*archivebox.core.admin.SnapshotAdmin method*), 90
add_view() (*archivebox.core.admin.ArchiveBoxAdmin method*), 91
AddView (*class in archivebox.core.views*), 92
aliases (*archivebox.config_stubs.ConfigDefault attribute*), 87
ALLOWED_HOSTS (*archivebox.config_stubs.ConfigDict attribute*), 85
ansi_to_html() (*in module archivebox.util*), 108
apply_migrations() (*in module archivebox.index.sql*), 100
archivable_links() (*in module archivebox.index*), 100
archive_dates (*archivebox.index.schema.Link property*), 99
archive_link() (*in module archivebox.extractors*), 95
archive_links() (*in module archivebox.extractors*), 96
archive_path (*archivebox.index.schema.Link property*), 99
archive_size (*archivebox.index.schema.Link property*), 99
archivebox
    module, 109
archivebox.cli
    module, 72
archivebox.cli.archivebox_add
    module, 70
archivebox.cli.archivebox_config
    module, 70
archivebox.cli.archivebox_help
    module, 71
archivebox.cli.archivebox_init
    module, 71
archivebox.cli.archivebox_list
    module, 71
archivebox.cli.archivebox_manage
    module, 71
archivebox.cli.archivebox_remove
    module, 71
archivebox.cli.archivebox_schedule
    module, 71
archivebox.cli.archivebox_server
    module, 72
archivebox.cli.archivebox_shell
    module, 72
archivebox.cli.archivebox_status
    module, 71
archivebox.cli.archivebox_update
    module, 72
archivebox.cli.archivebox_version
    module, 72
archivebox.config
    module, 73
archivebox.config_stubs
    module, 85
archivebox.core
    module, 93
archivebox.core.admin
    module, 88
archivebox.core.apps
    module, 91
archivebox.core.migrations
    module, 88
archivebox.core.migrations.0001_initial
    module, 87
archivebox.core.settings
    module, 91
archivebox.core.tests
    module, 92
archivebox.core.urls
    module, 92
archivebox.core.views
    module, 92
archivebox.core.welcome_message
    module, 93

archivebox.core.wsgi
    module, 93
archivebox.extractors
    module, 95
archivebox.extractors.archive_org
    module, 93
archivebox.extractors.dom
    module, 93
archivebox.extractors.favicon
    module, 94
archivebox.extractors.git
    module, 94
archivebox.extractors.media
    module, 94
archivebox.extractors.pdf
    module, 94
archivebox.extractors.screenshot
    module, 94
archivebox.extractors.title
    module, 95
archivebox.extractors.wget
    module, 95
archivebox.index
    module, 100
archivebox.index.csv
    module, 96
archivebox.index.html
    module, 96
archivebox.index.json
    module, 96
archivebox.index.schema
    module, 97
archivebox.index.sql
    module, 99
archivebox.main
    module, 105
archivebox.manage
    module, 107
archivebox.parsers
    module, 104
archivebox.parsers.generic_json
    module, 102
archivebox.parsers.generic_rss
    module, 103
archivebox.parsers.generic_txt
    module, 103
archivebox.parsers.medium_rss
    module, 103
archivebox.parsers.netscape_html
    module, 103
archivebox.parsers.pinboard_rss
    module, 103
archivebox.parsers.pocket_html
    module, 104
archivebox.parsers.shaarli_rss
    module, 104
archivebox.system
    module, 107
archivebox.util
    module, 107
ArchiveBoxAdmin (*class in archivebox.core.admin*), 91
ArchiveError, 97
ArchiveResult (*class in archivebox.index.schema*), 97
ArchiveResultAdmin (*class in archivebox.core.admin*), 90
ArchiveResultInline (*class in archivebox.core.admin*), 88
as_snapshot() (*archivebox.index.schema.Link method*), 98
atomic_write() (*in module archivebox.system*), 107
AttributeDict (*class in archivebox.util*), 108
autocomplete_fields (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
autocomplete_fields (*archivebox.core.admin.SnapshotAdmin attribute*), 89
AutocompleteTags (*class in archivebox.core.admin*), 88
AutocompleteTagsAdminStub (*class in archivebox.core.admin*), 88

B

base_fields (*archivebox.core.admin.SnapshotActionForm attribute*), 88
base_url (*archivebox.index.schema.Link property*), 99
base_url() (*in module archivebox.util*), 108
BaseConfig (*class in archivebox.config_stubs*), 85
basename (*archivebox.index.schema.Link property*), 99
basename() (*in module archivebox.util*), 107
bin_hash() (*in module archivebox.config*), 74
bin_path() (*in module archivebox.config*), 74
bin_version() (*in module archivebox.config*), 74
BIND_ADDR (*archivebox.config_stubs.ConfigDict attribute*), 85
bookmarked_date (*archivebox.index.schema.Link property*), 99

C

canonical_outputs() (*archivebox.index.schema.Link method*), 99
check_data_folder() (*in module archivebox.config*), 79
check_dependencies() (*in module archivebox.config*), 77
check_migrations() (*in module archivebox.config*), 81
CHECK_SSL_VALIDITY (*archivebox.config_stubs.ConfigDict attribute*), 86

check_system_config() (*in module archivebox.config*), 75
chmod_file() (*in module archivebox.system*), 107
chrome_args() (*in module archivebox.util*), 108
CHROME_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
chrome_cleanup() (*in module archivebox.util*), 108
CHROME_HEADLESS (*archivebox.config_stubs.ConfigDict attribute*), 86
CHROME_SANDBOX (*archivebox.config_stubs.ConfigDict attribute*), 86
CHROME_TIMEOUT (*archivebox.config_stubs.ConfigDict attribute*), 86
CHROME_USER_AGENT (*archivebox.config_stubs.ConfigDict attribute*), 86
CHROME_USER_DATA_DIR (*archivebox.config_stubs.ConfigDict attribute*), 86
cmd (*archivebox.index.schema.ArchiveResult attribute*), 97
cmd_str() (*archivebox.core.admin.ArchiveResultAdmin method*), 91
cmd_version (*archivebox.index.schema.ArchiveResult attribute*), 97
config() (*in module archivebox.cli*), 72
config() (*in module archivebox.main*), 106
CONFIG_FILE (*archivebox.config_stubs.ConfigDict attribute*), 85
ConfigDefault (*class in archivebox.config_stubs*), 87
ConfigDict (*class in archivebox.config_stubs*), 85
COOKIES_FILE (*archivebox.config_stubs.ConfigDict attribute*), 86
copy_and_overwrite() (*in module archivebox.system*), 107
CoreConfig (*class in archivebox.core.apps*), 91
CURL_ARGS (*archivebox.config_stubs.ConfigDict attribute*), 87
CURL_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
CURL_USER_AGENT (*archivebox.config_stubs.ConfigDict attribute*), 86

D

DEBUG (*archivebox.config_stubs.ConfigDict attribute*), 85
declared_fields (*archivebox.core.admin.SnapshotActionForm attribute*), 89
dedupe_cron_jobs() (*in module archivebox.system*), 107
dedupe_links() (*in module archivebox.index*), 101
default (*archivebox.config_stubs.ConfigDefault attribute*), 87
default() (*archivebox.index.json.ExtendedEncoder method*), 97
default() (*archivebox.util.ExtendedEncoder method*), 109
default_auto_field (*archivebox.core.apps.CoreConfig attribute*), 91
delete_snapshots() (*archivebox.core.admin.SnapshotAdmin method*), 90
dependencies (*archivebox.core.migrations.0001_initial.Migration attribute*), 88
detect_encoding() (*in module archivebox.util*), 107
dispatch() (*archivebox.core.views.AddView method*), 92
docstring() (*in module archivebox.util*), 108
domain (*archivebox.index.schema.Link property*), 99
domain() (*in module archivebox.util*), 107
download_url() (*in module archivebox.util*), 108
duration (*archivebox.index.schema.ArchiveResult property*), 98

E

end_ts (*archivebox.index.schema.ArchiveResult attribute*), 98
enforce_types() (*in module archivebox.util*), 108
ExtendedEncoder (*class in archivebox.index.json*), 97
ExtendedEncoder (*class in archivebox.util*), 108
extension (*archivebox.index.schema.Link property*), 99
extension() (*in module archivebox.util*), 108
extract_title_with_regex() (*in module archivebox.extractors.title*), 95

F

field_names() (*archivebox.index.schema.ArchiveResult class method*), 98
field_names() (*archivebox.index.schema.Link class method*), 99
fields (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
fields (*archivebox.core.admin.SnapshotAdmin attribute*), 89
fields (*archivebox.core.admin.TagAdmin attribute*), 90
files() (*archivebox.core.admin.SnapshotAdmin method*), 89
filter() (*archivebox.core.settings.NoisyRequestsFilter method*), 91
find_chrome_binary() (*in module archivebox.config*), 74
find_chrome_data_dir() (*in module archivebox.config*), 74
fix_duplicate_links() (*in module archivebox.index*), 100
fix_duplicate_links_in_index() (*in module archivebox.index*), 101

fix_invalid_folder_locations() (*in module archivebox.index*), 102
FOOTER_INFO (*archivebox.config_stubs.ConfigDict attribute*), 86
form_class (*archivebox.core.views.AddView attribute*), 92
form_valid() (*archivebox.core.views.AddView method*), 92
fragment() (*in module archivebox.util*), 108
from_json() (*archivebox.index.schema.ArchiveResult class method*), 98
from_json() (*archivebox.index.schema.Link class method*), 98

G

generate_index_from_links() (*in module archivebox.index.html*), 96
generate_json_index_from_links() (*in module archivebox.index.json*), 96
get() (*archivebox.core.views.HealthCheckView method*), 93
get() (*archivebox.core.views.HomepageView method*), 92
get() (*archivebox.core.views.PublicIndexView method*), 92
get() (*archivebox.core.views.SnapshotView method*), 92
get_admins() (*in module archivebox.index.sql*), 100
get_archive_methods_for_link() (*in module archivebox.extractors*), 95
get_archived_folders() (*in module archivebox.index*), 101
get_chrome_info() (*in module archivebox.config*), 75
get_code_locations() (*in module archivebox.config*), 75
get_commit_hash() (*in module archivebox.config*), 74
get_context_data() (*archivebox.core.views.AddView method*), 92
get_context_data() (*archivebox.core.views.PublicIndexView method*), 92
get_corrupted_folders() (*in module archivebox.index*), 102
get_data_locations() (*in module archivebox.config*), 75
get_default_archive_methods() (*in module archivebox.extractors*), 95
get_dependency_info() (*in module archivebox.config*), 75
get_dir_size() (*in module archivebox.system*), 107
get_duplicate_folders() (*in module archivebox.index*), 102
get_external_locations() (*in module archivebox.config*), 75
get_headers() (*in module archivebox.util*), 108
get_html() (*in module archivebox.extractors.title*), 95
get_indexed_folders() (*in module archivebox.index*), 101
get_initial() (*archivebox.core.views.AddView method*), 92
get_invalid_folders() (*in module archivebox.index*), 101
get_orphaned_folders() (*in module archivebox.index*), 102
get_present_folders() (*in module archivebox.index*), 101
get_queryset() (*archivebox.core.admin.SnapshotAdmin method*), 89
get_queryset() (*archivebox.core.views.PublicIndexView method*), 92
get_real_name() (*in module archivebox.config*), 74
get_unarchived_folders() (*in module archivebox.index*), 101
get_unrecognized_folders() (*in module archivebox.index*), 102
get_urls() (*archivebox.core.admin.ArchiveBoxAdmin method*), 91
get_urls() (*archivebox.core.admin.SnapshotAdmin method*), 89
get_valid_folders() (*in module archivebox.index*), 101
get_version() (*in module archivebox.config*), 74
GIT_ARGS (*archivebox.config_stubs.ConfigDict attribute*), 87
GIT_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
GIT_DOMAINS (*archivebox.config_stubs.ConfigDict attribute*), 86
grid_view() (*archivebox.core.admin.SnapshotAdmin method*), 89
guess_ts() (*archivebox.index.schema.ArchiveResult class method*), 98

H

handle_data() (*archivebox.extractors.title.TitleParser method*), 95
handle_endtag() (*archivebox.extractors.title.TitleParser method*), 95
handle_starttag() (*archivebox.extractors.title.TitleParser method*), 95
hashurl() (*in module archivebox.util*), 108
HealthCheckView (*class in archivebox.core.views*), 93
help() (*in module archivebox.cli*), 72
help() (*in module archivebox.main*), 105
hint() (*in module archivebox.config*), 74

history (*archivebox.index.schema.Link attribute*), 98
HomepageView (*class in archivebox.core.views*), 92
htmldecode() (*in module archivebox.util*), 108
htmlencode() (*in module archivebox.util*), 108

I

ignore_methods() (*in module archivebox.extractors*), 95
IN_DOCKER (*archivebox.config_stubs.ConfigDict attribute*), 85
index_texts (*archivebox.index.schema.ArchiveResult attribute*), 98
index_title (*archivebox.core.admin.ArchiveBoxAdmin attribute*), 91
info() (*archivebox.core.admin.SnapshotAdmin method*), 89
init() (*in module archivebox.cli*), 72
init() (*in module archivebox.main*), 105
initial (*archivebox.core.migrations.0001_initial.Migration attribute*), 88
inlines (*archivebox.core.admin.SnapshotAdmin attribute*), 89
is_archived (*archivebox.index.schema.Link property*), 99
is_archived() (*in module archivebox.index*), 102
is_corrupt() (*in module archivebox.index*), 102
is_static (*archivebox.index.schema.Link property*), 99
is_static_file() (*in module archivebox.util*), 108
IS_TTY (*archivebox.config_stubs.ConfigDict attribute*), 85
is_unarchived() (*in module archivebox.index*), 102
is_valid() (*in module archivebox.index*), 102

L

latest_outputs() (*archivebox.index.schema.Link method*), 99
Link (*class in archivebox.index.schema*), 98
link_details_template() (*in module archivebox.index.html*), 96
link_dir (*archivebox.index.schema.Link property*), 99
links_after_timestamp() (*in module archivebox.index*), 100
links_to_csv() (*in module archivebox.index.csv*), 96
list() (*in module archivebox.cli*), 73
list_all() (*in module archivebox.main*), 106
list_display (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
list_display (*archivebox.core.admin.SnapshotAdmin attribute*), 89
list_display (*archivebox.core.admin.TagAdmin attribute*), 90
list_filter (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
list_filter (*archivebox.core.admin.SnapshotAdmin attribute*), 89
list_folders() (*in module archivebox.main*), 106
list_links() (*in module archivebox.main*), 106
list_migrations() (*in module archivebox.index.sql*), 100
list_per_page (*archivebox.core.admin.ArchiveResultAdmin attribute*), 91
list_per_page (*archivebox.core.admin.SnapshotAdmin attribute*), 89
list_subcommands() (*in module archivebox.cli*), 72
load_all_config() (*in module archivebox.config*), 75
load_config() (*in module archivebox.config*), 74
load_config_file() (*in module archivebox.config*), 74
load_config_val() (*in module archivebox.config*), 74
load_link_details() (*in module archivebox.index*), 101
load_main_index() (*in module archivebox.index*), 100
load_main_index_meta() (*in module archivebox.index*), 100
lowest_uniq_timestamp() (*in module archivebox.index*), 100

M

main() (*in module archivebox.cli.archivebox_add*), 70
main() (*in module archivebox.cli.archivebox_config*), 70
main() (*in module archivebox.cli.archivebox_help*), 71
main() (*in module archivebox.cli.archivebox_init*), 71
main() (*in module archivebox.cli.archivebox_list*), 71
main() (*in module archivebox.cli.archivebox_manage*), 71
main() (*in module archivebox.cli.archivebox_remove*), 71
main() (*in module archivebox.cli.archivebox_schedule*), 71
main() (*in module archivebox.cli.archivebox_server*), 72
main() (*in module archivebox.cli.archivebox_shell*), 72
main() (*in module archivebox.cli.archivebox_status*), 71
main() (*in module archivebox.cli.archivebox_update*), 72
main() (*in module archivebox.cli.archivebox_version*), 72
main_index_template() (*in module archivebox.index.html*), 96
manage() (*in module archivebox.cli*), 73
manage() (*in module archivebox.main*), 106
media (*archivebox.core.admin.ArchiveResultAdmin property*), 91
media (*archivebox.core.admin.ArchiveResultInline property*), 88
media (*archivebox.core.admin.SnapshotActionForm property*), 89

media (*archivebox.core.admin.SnapshotAdmin property*), 90
media (*archivebox.core.admin.TagAdmin property*), 90
media (*archivebox.core.admin.TagInline property*), 88
MEDIA_TIMEOUT (*archivebox.config_stubs.ConfigDict attribute*), 85
MERCURY_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
merge_links() (*in module archivebox.index*), 100
Migration (*class in archivebox.core.migrations.0001_initial*), 87
model (*archivebox.core.admin.ArchiveResultInline attribute*), 88
model (*archivebox.core.admin.AutocompleteTags attribute*), 88
model (*archivebox.core.admin.TagInline attribute*), 88
model (*archivebox.core.views.PublicIndexView attribute*), 92
module
    archivebox, 109
    archivebox.cli, 72
    archivebox.cli.archivebox_add, 70
    archivebox.cli.archivebox_config, 70
    archivebox.cli.archivebox_help, 71
    archivebox.cli.archivebox_init, 71
    archivebox.cli.archivebox_list, 71
    archivebox.cli.archivebox_manage, 71
    archivebox.cli.archivebox_remove, 71
    archivebox.cli.archivebox_schedule, 71
    archivebox.cli.archivebox_server, 72
    archivebox.cli.archivebox_shell, 72
    archivebox.cli.archivebox_status, 71
    archivebox.cli.archivebox_update, 72
    archivebox.cli.archivebox_version, 72
    archivebox.config, 73
    archivebox.config_stubs, 85
    archivebox.core, 93
    archivebox.core.admin, 88
    archivebox.core.apps, 91
    archivebox.core.migrations, 88
    archivebox.core.migrations.0001_initial, 87
    archivebox.core.settings, 91
    archivebox.core.tests, 92
    archivebox.core.urls, 92
    archivebox.core.views, 92
    archivebox.core.welcome_message, 93
    archivebox.core.wsgi, 93
    archivebox.extractors, 95
    archivebox.extractors.archive_org, 93
    archivebox.extractors.dom, 93
    archivebox.extractors.favicon, 94
    archivebox.extractors.git, 94
    archivebox.extractors.media, 94
    archivebox.extractors.pdf, 94
    archivebox.extractors.screenshot, 94
    archivebox.extractors.title, 95
    archivebox.extractors.wget, 95
    archivebox.index, 100
    archivebox.index.csv, 96
    archivebox.index.html, 96
    archivebox.index.json, 96
    archivebox.index.schema, 97
    archivebox.index.sql, 99
    archivebox.main, 105
    archivebox.manage, 107
    archivebox.parsers, 104
    archivebox.parsers.generic_json, 102
    archivebox.parsers.generic_rss, 103
    archivebox.parsers.generic_txt, 103
    archivebox.parsers.medium_rss, 103
    archivebox.parsers.netscape_html, 103
    archivebox.parsers.pinboard_rss, 103
    archivebox.parsers.pocket_html, 104
    archivebox.parsers.shaarli_rss, 104
    archivebox.system, 107
    archivebox.util, 107

N

name (*archivebox.core.admin.AutocompleteTagsAdminStub attribute*), 88
name (*archivebox.core.apps.CoreConfig attribute*), 91
newest_archive_date (*archivebox.index.schema.Link property*), 99
NoisyRequestsFilter (*class in archivebox.core.settings*), 91
num_failures (*archivebox.index.schema.Link property*), 99
num_outputs (*archivebox.index.schema.Link property*), 99
num_snapshots() (*archivebox.core.admin.TagAdmin method*), 90

O

oldest_archive_date (*archivebox.index.schema.Link property*), 99
oneshot() (*in module archivebox.cli*), 73
oneshot() (*in module archivebox.main*), 105
ONLY_NEW (*archivebox.config_stubs.ConfigDict attribute*), 85
operations (*archivebox.core.migrations.0001_initial.Migration attribute*), 88
ordering (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
ordering (*archivebox.core.admin.SnapshotAdmin attribute*), 89
ordering (*archivebox.core.admin.TagAdmin attribute*), 90

ordering (*archivebox.core.views.PublicIndexView attribute*), 92
output (*archivebox.index.schema.ArchiveResult attribute*), 97
OUTPUT_DIR (*archivebox.config_stubs.ConfigDict attribute*), 85
OUTPUT_PERMISSIONS (*archivebox.config_stubs.ConfigDict attribute*), 85
output_str() (*archivebox.core.admin.ArchiveResultAdmin method*), 91
overwrite() (*archivebox.index.schema.Link method*), 98
overwrite_snapshots() (*archivebox.core.admin.SnapshotAdmin method*), 90

P

PACKAGE_DIR (*archivebox.config_stubs.ConfigDict attribute*), 85
paginate_by (*archivebox.core.views.PublicIndexView attribute*), 92
parse_archive_dot_org_response() (*in module archivebox.extractors.archive_org*), 93
parse_date() (*in module archivebox.util*), 108
parse_generic_json_export() (*in module archivebox.parsers.generic_json*), 102
parse_generic_rss_export() (*in module archivebox.parsers.generic_rss*), 103
parse_generic_txt_export() (*in module archivebox.parsers.generic_txt*), 103
parse_html_main_index() (*in module archivebox.index.html*), 96
parse_json_link_details() (*in module archivebox.index.json*), 96
parse_json_links_details() (*in module archivebox.index.json*), 97
parse_json_main_index() (*in module archivebox.index.json*), 96
parse_links() (*in module archivebox.parsers*), 104
parse_links_from_source() (*in module archivebox.index*), 101
parse_links_memory() (*in module archivebox.parsers*), 104
parse_medium_rss_export() (*in module archivebox.parsers.medium_rss*), 103
parse_netscape_html_export() (*in module archivebox.parsers.netscape_html*), 103
parse_pinboard_rss_export() (*in module archivebox.parsers.pinboard_rss*), 103
parse_pocket_html_export() (*in module archivebox.parsers.pocket_html*), 104
parse_shaarli_rss_export() (*in module archivebox.parsers.shaarli_rss*), 104
parse_sql_main_index() (*in module archivebox.index.sql*), 99
PARSER() (*in module archivebox.parsers.generic_json*), 102
PARSER() (*in module archivebox.parsers.generic_rss*), 103
PARSER() (*in module archivebox.parsers.generic_txt*), 103
PARSER() (*in module archivebox.parsers.medium_rss*), 103
PARSER() (*in module archivebox.parsers.netscape_html*), 103
PARSER() (*in module archivebox.parsers.pinboard_rss*), 103
PARSER() (*in module archivebox.parsers.pocket_html*), 104
PARSER() (*in module archivebox.parsers.shaarli_rss*), 104
path (*archivebox.index.schema.Link property*), 99
path() (*in module archivebox.util*), 107
PUBLIC_INDEX (*archivebox.config_stubs.ConfigDict attribute*), 85
PUBLIC_SNAPSHOTS (*archivebox.config_stubs.ConfigDict attribute*), 85
PublicIndexView (*class in archivebox.core.views*), 92
pwd (*archivebox.index.schema.ArchiveResult attribute*), 97

Q

q_filter() (*in module archivebox.index*), 101
query() (*in module archivebox.util*), 107

R

READABILITY_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
readonly_fields (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
readonly_fields (*archivebox.core.admin.SnapshotAdmin attribute*), 89
readonly_fields (*archivebox.core.admin.TagAdmin attribute*), 90
remove() (*in module archivebox.cli*), 73
remove() (*in module archivebox.main*), 105
remove_from_sql_main_index() (*in module archivebox.index.sql*), 99
remove_tags() (*archivebox.core.admin.SnapshotAdmin method*), 90
render_django_template() (*in module archivebox.index.html*), 96
resnapshot_snapshot() (*archivebox.core.admin.SnapshotAdmin method*), 90

RESOLUTION (*archivebox.config_stubs.ConfigDict attribute*), 86
RESTRICT_FILE_NAMES (*archivebox.config_stubs.ConfigDict attribute*), 85
run() (*in module archivebox.main*), 105
run() (*in module archivebox.system*), 107
run_parser_functions() (*in module archivebox.parsers*), 104
run_subcommand() (*in module archivebox.cli*), 72

S

SAVE_ARCHIVE_DOT_ORG (*archivebox.config_stubs.ConfigDict attribute*), 86
save_archive_dot_org() (*in module archivebox.extractors.archive_org*), 93
SAVE_DOM (*archivebox.config_stubs.ConfigDict attribute*), 86
save_dom() (*in module archivebox.extractors.dom*), 93
SAVE_FAVICON (*archivebox.config_stubs.ConfigDict attribute*), 86
save_favicon() (*in module archivebox.extractors.favicon*), 94
save_file_as_source() (*in module archivebox.parsers*), 104
SAVE_GIT (*archivebox.config_stubs.ConfigDict attribute*), 86
save_git() (*in module archivebox.extractors.git*), 94
SAVE_MEDIA (*archivebox.config_stubs.ConfigDict attribute*), 86
save_media() (*in module archivebox.extractors.media*), 94
SAVE_MERCURY (*archivebox.config_stubs.ConfigDict attribute*), 86
SAVE_PDF (*archivebox.config_stubs.ConfigDict attribute*), 86
save_pdf() (*in module archivebox.extractors.pdf*), 94
SAVE_READABILITY (*archivebox.config_stubs.ConfigDict attribute*), 86
SAVE_SCREENSHOT (*archivebox.config_stubs.ConfigDict attribute*), 86
save_screenshot() (*in module archivebox.extractors.screenshot*), 94
SAVE_SINGLEFILE (*archivebox.config_stubs.ConfigDict attribute*), 86
save_text_as_source() (*in module archivebox.parsers*), 104
SAVE_TITLE (*archivebox.config_stubs.ConfigDict attribute*), 86
save_title() (*in module archivebox.extractors.title*), 95
SAVE_WARC (*archivebox.config_stubs.ConfigDict attribute*), 86
SAVE_WGET (*archivebox.config_stubs.ConfigDict attribute*), 86
save_wget() (*in module archivebox.extractors.wget*), 95
SAVE_WGET_REQUISITES (*archivebox.config_stubs.ConfigDict attribute*), 86
schedule() (*in module archivebox.cli*), 73
schedule() (*in module archivebox.main*), 106
schema (*archivebox.index.schema.ArchiveResult attribute*), 98
schema (*archivebox.index.schema.Link attribute*), 98
scheme (*archivebox.index.schema.Link property*), 99
scheme() (*in module archivebox.util*), 107
search_fields (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
search_fields (*archivebox.core.admin.AutocompleteTags attribute*), 88
search_fields (*archivebox.core.admin.SnapshotAdmin attribute*), 89
search_fields (*archivebox.core.admin.TagAdmin attribute*), 90
search_filter() (*in module archivebox.index*), 101
SECRET_KEY (*archivebox.config_stubs.ConfigDict attribute*), 85
server() (*in module archivebox.cli*), 73
server() (*in module archivebox.main*), 106
setup() (*in module archivebox.cli*), 72
setup() (*in module archivebox.main*), 106
setup_django() (*in module archivebox.config*), 83
shell() (*in module archivebox.cli*), 73
shell() (*in module archivebox.main*), 106
short_ts() (*in module archivebox.util*), 108
should_save_archive_dot_org() (*in module archivebox.extractors.archive_org*), 93
should_save_dom() (*in module archivebox.extractors.dom*), 93
should_save_favicon() (*in module archivebox.extractors.favicon*), 94
should_save_git() (*in module archivebox.extractors.git*), 94
should_save_media() (*in module archivebox.extractors.media*), 94
should_save_pdf() (*in module archivebox.extractors.pdf*), 94
should_save_screenshot() (*in module archivebox.extractors.screenshot*), 94
should_save_title() (*in module archivebox.extractors.title*), 95
should_save_wget() (*in module archivebox.extractors.wget*), 95
SHOW_PROGRESS (*archivebox.config_stubs.ConfigDict attribute*), 85
SINGLEFILE_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
site_header (*archivebox.core.admin.ArchiveBoxAdmin*

attribute), 91
site_title (*archivebox.core.admin.ArchiveBoxAdmin attribute*), 91
size() (*archivebox.core.admin.SnapshotAdmin method*), 89
snapshot_filter() (*in module archivebox.index*), 101
snapshot_icons() (*in module archivebox.index.html*), 96
snapshot_id (*archivebox.index.schema.Link attribute*), 98
snapshot_str() (*archivebox.core.admin.ArchiveResultAdmin method*), 91
SnapshotActionForm (*class in archivebox.core.admin*), 88
SnapshotAdmin (*class in archivebox.core.admin*), 89
snapshots() (*archivebox.core.admin.TagAdmin method*), 90
SnapshotView (*class in archivebox.core.views*), 92
sort_fields (*archivebox.core.admin.ArchiveResultAdmin attribute*), 90
sort_fields (*archivebox.core.admin.SnapshotAdmin attribute*), 89
sort_fields (*archivebox.core.admin.TagAdmin attribute*), 90
sorted_links() (*in module archivebox.index*), 100
sources (*archivebox.index.schema.Link attribute*), 98
start_ts (*archivebox.index.schema.ArchiveResult attribute*), 98
status (*archivebox.index.schema.ArchiveResult attribute*), 97
status() (*in module archivebox.cli*), 73
status() (*in module archivebox.main*), 105
stderr() (*in module archivebox.config*), 74
stdout() (*in module archivebox.config*), 74
str_between() (*in module archivebox.util*), 108
suppress_output (*class in archivebox.system*), 107

T

tag_list() (*archivebox.core.admin.SnapshotAdmin method*), 89
TAG_SEPARATOR_PATTERN (*archivebox.config_stubs.ConfigDict attribute*), 87
TagAdmin (*class in archivebox.core.admin*), 90
TagInline (*class in archivebox.core.admin*), 88
tags (*archivebox.index.schema.Link attribute*), 98
tags_str() (*archivebox.core.admin.ArchiveResultAdmin method*), 91
template_name (*archivebox.core.views.AddView attribute*), 92
template_name (*archivebox.core.views.PublicIndexView attribute*), 92
TERM_WIDTH() (*in module archivebox.config*), 83
test_func() (*archivebox.core.views.AddView method*), 92
timed_index_update() (*in module archivebox.index*), 100
TIMEOUT (*archivebox.config_stubs.ConfigDict attribute*), 85
timestamp (*archivebox.index.schema.Link attribute*), 98
title (*archivebox.extractors.title.TitleParser property*), 95
title (*archivebox.index.schema.Link attribute*), 98
title_str() (*archivebox.core.admin.SnapshotAdmin method*), 89
TitleParser (*class in archivebox.extractors.title*), 95
to_csv() (*archivebox.index.schema.ArchiveResult method*), 98
to_csv() (*archivebox.index.schema.Link method*), 98
to_csv() (*in module archivebox.index.csv*), 96
to_dict() (*archivebox.index.schema.ArchiveResult method*), 98
to_json() (*archivebox.index.schema.ArchiveResult method*), 98
to_json() (*archivebox.index.schema.Link method*), 98
to_json() (*in module archivebox.index.json*), 97
ts_to_date_str() (*in module archivebox.util*), 108
ts_to_iso() (*in module archivebox.util*), 108
type (*archivebox.config_stubs.ConfigDefault attribute*), 87
typecheck() (*archivebox.index.schema.ArchiveResult method*), 98
typecheck() (*archivebox.index.schema.Link method*), 98

U

update() (*in module archivebox.cli*), 73
update() (*in module archivebox.main*), 105
update_snapshots() (*archivebox.core.admin.SnapshotAdmin method*), 90
update_titles() (*archivebox.core.admin.SnapshotAdmin method*), 90
updated (*archivebox.index.schema.Link attribute*), 98
updated_date (*archivebox.index.schema.Link property*), 99
url (*archivebox.index.schema.Link attribute*), 98
URL_DENYLIST (*archivebox.config_stubs.ConfigDict attribute*), 85
url_hash (*archivebox.index.schema.Link property*), 99
url_str() (*archivebox.core.admin.SnapshotAdmin method*), 89
urldecode() (*in module archivebox.util*), 108
urlencode() (*in module archivebox.util*), 108

USE_CHROME (*archivebox.config_stubs.ConfigDict attribute*), 87
USE_COLOR (*archivebox.config_stubs.ConfigDict attribute*), 85
USE_CURL (*archivebox.config_stubs.ConfigDict attribute*), 86
USE_GIT (*archivebox.config_stubs.ConfigDict attribute*), 87
USE_MERCURY (*archivebox.config_stubs.ConfigDict attribute*), 86
USE_READABILITY (*archivebox.config_stubs.ConfigDict attribute*), 86
USE_SINGLEFILE (*archivebox.config_stubs.ConfigDict attribute*), 86
USE_WGET (*archivebox.config_stubs.ConfigDict attribute*), 86
USE_YOUTUBEDL (*archivebox.config_stubs.ConfigDict attribute*), 87

V

validate_links() (*in module archivebox.index*), 100
version() (*in module archivebox.cli*), 72
version() (*in module archivebox.main*), 105

W

WGET_ARGS (*archivebox.config_stubs.ConfigDict attribute*), 87
WGET_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87
wget_output_path() (*in module archivebox.extractors.wget*), 95
wget_supports_compression() (*in module archivebox.config*), 74
WGET_USER_AGENT (*archivebox.config_stubs.ConfigDict attribute*), 86
without_fragment() (*in module archivebox.util*), 107
without_path() (*in module archivebox.util*), 107
without_query() (*in module archivebox.util*), 107
without_scheme() (*in module archivebox.util*), 107
without_trailing_slash() (*in module archivebox.util*), 108
without_www() (*in module archivebox.util*), 108
write_config_file() (*in module archivebox.config*), 74
write_html_link_details() (*in module archivebox.index.html*), 96
write_json_link_details() (*in module archivebox.index.json*), 96
write_link_details() (*in module archivebox.index*), 101
write_link_to_sql_index() (*in module archivebox.index.sql*), 99
write_main_index() (*in module archivebox.index*), 100
write_sql_link_details() (*in module archivebox.index.sql*), 100
write_sql_main_index() (*in module archivebox.index.sql*), 100

Y

YOUTUBEDL_ARGS (*archivebox.config_stubs.ConfigDict attribute*), 87
YOUTUBEDL_BINARY (*archivebox.config_stubs.ConfigDict attribute*), 87