wkhtmltopdf

All downloads are currently hosted via GitHub releases, so you can browse for a specific download or use the links below.

Do not use wkhtmltopdf with any untrusted HTML – be sure to sanitize any user-supplied HTML/JS, otherwise it can lead to complete takeover of the server it is running on! Please read the project status for the gory details.

Stable

The current stable series is **0.12.6**, which was released on June 11, 2020 – see changes since 0.12.5.

OS/Distribution	Supported on	Architectures				
Windows	Installer (Vista or later)	64-bit	32-bit			
	7z Archive (XP/2003 or later)	64-bit	32-bit			
macOS	Installer (10.7 or later)	64-bit				
Debian	11 (**bullseye**)	amd64	i386	arm64	ppc64el	raspberrypi
	10 (**buster**)	amd64	i386	arm64	ppc64el	raspberrypi
	9 (**stretch**)	amd64	i386	arm64		raspberrypi
Ubuntu	22.04 (**jammy**)	amd64		arm64	ppc64el	
	20.04 (**focal**)	amd64		arm64	ppc64el	
	18.04 (**bionic**)	amd64	i386	arm64	ppc64el	
	16.04 (**xenial**)	amd64	i386	arm64		
AlmaLinux	9	x86_64		aarch64		
	8	x86_64		aarch64	ppc64le	
CentOS	7	x86_64	i686	aarch64	ppc64le	
	6	x86_64	i686			
Amazon Linux	2 (package)	x86_64		aarch64		
	2 (lambda zip)	x86_64				
openSUSE Leap	15	x86_64		aarch64	ppc64le	
Arch Linux	20200705	x86_64				

All of the above packages were produced automatically and were built on the latest OS/distribution patch release at the time of the release.

Archive

Please note that bug reports **will not be accepted** against the following, which are considered obsolete. It is recommended

to use the latest stable release instead, and report an issue if there is a regression from a previous release.

Date	Release
2018-06-11	0.12.5
2019-04-30	0.12.1.4 (linux-only)
2016-11-22	0.12.4
2016-03-02	0.12.3.2 (windows-only)
2016-01-30	0.12.3.1 (windows-only)
2016-01-20	0.12.3
2015-07-12	0.12.2.4 (windows-only)
2015-06-20	0.12.2.3 (windows-only)
2015-04-06	0.12.2.2 (windows-only)
2015-01-19	0.12.2.1
2015-01-09	0.12.2
2014-06-26	0.12.1
2014-02-06	0.12.0

If you need versions older than `0.12.0`, you can look at the obsolete downloads.

FAQ

Why do you have static builds with patched Qt?

Good question. Some features require you to use a patched Qt, because those aren't yet upstream – please read the project status for a longer explanation.

Most Linux distributions (*quite understandably*) would prefer that this project upstreamed the patches, and choose to compile without those features. This leads to quite different behavior – you get a later web engine, but behavior can vary from distribution to distribution.

Why are there no "generic" Linux builds (*which were provided earlier*)?

Although the builds are static, it is very important to understand what it means in the context of Qt – on which wkhtmltopdf is built. A static build means that *only* Qt is linked in this manner – the remaining system packages still need to be installed. Over a period of time, major areas of divergence between distributions were found by trial and error:

- **different library versions**: not every distribution provides the same versions. This was especially the case for `libpng` and `libjpeg`, with a lot of distributions choosing between the 1.2, 1.5 and 1.6 series for the former and multiple versions of `libjpeg` and/or its fork `libjpeg-turbo`. While this could be addressed easily by linking them statically (and was actually done so for previous releases) – it broke down when it came to the next point.
- **different OpenSSL versions**: due to OpenSSL having a bad track record then (*it's better now*), distributions started aggressively upgrading their OpenSSL version and disabling unused parts of the library. This led to a situation where there was effectively zero backward compatibility and things started breaking randomly – see #3001 for a very long read of the problems faced. This was the direct motivation to create a separate packaging repository.
- **incompatible libc**: not every distribution has the same glibc version. If you compile with a later version, it won't work on a distribution which uses an older version. This was worked around earlier by using CentOS 6 (which had an old enough glibc version). But due to the rise of Docker, the `alpine` image became very popular. This doesn't use glibc at all, but the musl libc. So the generic binaries never really worked on Alpine.

While Python has also tried to do this using manylinux – it doesn't always work well (e.g. `alpine` is *not* recommended with binary wheels if you google for it), and requires you to statically link everything. This may work for them, but wkhtmltopdf also depends on the runtime configuration on actual fonts installed (i.e. `fontconfig` and `freetype2`). It's not possible to abstract everything out and test/fix everything for every OS/distribution with the limited resources this project has – it makes more sense to make distribution-specific versions which are almost guaranteed to work, as they use the specific versions that the distribution has packaged.

I don't see an appropriate download for my platform!

If the distribution you are using is listed:

- but not the specific patch release – try it, as it's very likely to work regardless.
- the major release isn't listed – we only support LTS versions, so try a LTS version older than your release.
- cannot install package – you can always extract it (google for `extract from <format>`), but you'll need to have the dependencies installed.

Head over to the packaging repository and start a discussion if your platform isn't listed.

How do I use it with FaaS setups?

You'll need to extract the distribution-specific package, bundle it with necessary libraries, configuration and/or fonts and then upload it. See this StackOverflow question for Google Cloud Functions. PRs are welcome to expand this section, if you have more information about this – this is not a setup that the maintainer uses

How do I use it in AWS Lambda?

All files required for lambda layer are packed in one zip archive (Amazon Linux 2 / lambda zip). You may test it locally by unpacking the archive into the `layer` directory and running next commands:

```
$ docker run --rm -it -v$PWD/layer:/opt amazonlinux:2
bash-4.2# LD_LIBRARY_PATH=/opt/lib FONTCONFIG_PATH=/opt/fonts /opt/bin/wkhtmltopdf
https://google.com/ /opt/google.pdf
```

After that, you may find a pdf file generated from the google home page in your `layer` directory.

To use `wkhtmltox` in your lambda function you may put the content of the archive together with your lambda function or create a layer. Don't forget to provide an environment variable for `fontconfig` (`FONTCONFIG_PATH=/opt/fonts`).

In case you use Serverless framework you may add the next lines to your `serverless.yml` file:

```
layers:
  wkhtmltoxLayer:
    name: wkhtmltox
    description: wkhtmltox binaries for pdf/image generation
    package:
      artifact: wkhtmltox-x.xx.xxx.amazonlinux2_lambda.zip

functions:
    PdfGenerator:
        handler: generatePdf.handler
        layers:
            - { Ref: WkhtmltoxLayerLambdaLayer }
        environment:
            FONTCONFIG_PATH: /opt/fonts
```

Symantec reports a virus `WS.Reputation.1` for the Windows builds

This is a false positive reported because Symantec has not seen this file before – see this clarification for details.
