Pandoc a universal document converter

Installing pandoc

The simplest way to get the latest pandoc release is to use the installer.

Download the latest installer

For alternative ways to install pandoc, see below under the heading for your operating system.

Windows

There is a package installer at pandoc's download page. This will install pandoc, replacing older versions, and update your path to include the directory where pandoc's binaries are installed.

If you prefer not to use the msi installer, we also provide a zip file that contains pandoc's binaries and documentation. Simply unzip this file and move the binaries to a directory of your choice.

Alternatively, you can install pandoc using Chocolatey:

```
choco install pandoc
```

Chocolatey can also install other software that integrates with Pandoc. For example, to install `rsvg-convert` (from librsvg, covering formats without SVG support), Python (to use Pandoc filters), and MiKTeX (to typeset PDFs with LaTeX):

```
choco install rsvg-convert python miktex
```

Or, you can install pandoc using winget:

```
winget install --source winget --exact --id
JohnMacFarlane.Pandoc
```

Using multiple installation methods can result in two separate installations of pandoc; it is recommended to properly uninstall pandoc before switching to an alternative installation method.

By default, Pandoc creates PDFs using LaTeX. We recommend installing it via MiKTeX. With the option `--pdf-engine`, you however can specify other programs for this task.

macOS

There is a package installer at pandoc's download page. If you later want to uninstall the package, you can do so by downloading this script and running it with `perl uninstall-pandoc.pl`.

Alternatively, you can install pandoc using Homebrew:

```
brew install pandoc
```

Homebrew can also install other software that integrates with Pandoc. For example, to install librsvg (its `rsvg-convert` covers formats without SVG support), Python (to use Pandoc filters), and BasicTeX (to typeset PDFs with LaTeX):

```
brew install librsvg python homebrew/cask/basictex
```

Note: On unsupported versions of macOS (more than three releases old), Homebrew installs from source, which takes additional time and disk space for the `ghc` compiler and dependent Haskell libraries.

We also provide a zip file containing the binaries and man pages, for those who prefer not to use the installer. Simply unzip the file and move the binaries and man pages to whatever directory you like.

By default, Pandoc creates PDFs using LaTeX. Because a full MacTeX installation uses four gigabytes of disk space, we recommend BasicTeX or TinyTeX and using the `tlmgr` tool to

install additional packages as needed. If you receive errors warning of fonts not found:

```
tlmgr install collection-fontsrecommended
```

With the option `--pdf-engine`, you however can specify other programs for this task.

Linux

Check whether the pandoc version in your package manager is not outdated. Pandoc is in the Debian, Ubuntu, Slackware, Arch, Fedora, NiXOS, openSUSE, gentoo and Void repositories.

To get the latest release, we provide a binary package for amd64 architecture on the download page.

The executable is statically linked and has no dynamic dependencies or dependencies on external data files. Note: because of the static linking, the pandoc binary from this package cannot use lua filters that require external lua modules written in C.

Both a tarball and a deb installer are provided. To install the deb:

```
sudo dpkg -i $DEB
```

where `$DEB` is the path to the downloaded deb. This will install the `pandoc` executable and man page.

If you use an RPM-based distro, you may be able to install the deb from our download page using `alien`.

On any distro, you may install from the tarball into `$DEST` (say, `/usr/local/` or `$HOME/.local`) by doing

```
tar xvzf $TGZ --strip-components 1 -C $DEST
```

where `$TGZ` is the path to the downloaded zipped tarball. For Pandoc versions before 2.0, which don't provide a tarball, try instead

```
ar p $DEB data.tar.gz | tar xvz --strip-components 2 -C
$DEST
```

You can also install from source, using the instructions below under Compiling from source. Note that most distros have the Haskell platform in their package repositories. For example, on Debian/Ubuntu, you can install it with `apt-get install haskell-platform`.

By default, Pandoc creates PDFs using LaTeX. We recommend installing TeX Live via your package manager. (On Debian/Ubuntu, `apt-get install texlive`.) With the option `--pdf-engine`, you however can specify other programs for this task.

Chrome OS

On Chrome OS, pandoc can be installed using the chromebrew package manager with the command:

```
crew install pandoc
```

This will automatically build and configure pandoc for the specific device you are using.

BSD

Pandoc is in the NetBSD and FreeBSD ports repositories.

Docker

The official Docker images for pandoc can be found at https://github.com/pandoc/dockerfiles and at dockerhub.

The pandoc/core image contains `pandoc`.

The pandoc/latex image also contains the minimal LaTeX installation needed to produce PDFs using pandoc.

To run pandoc using Docker, converting `README.md` to `README.pdf`:

```
docker run --rm --volume "`pwd`:/data" --user `id -u`:`id
-g` pandoc/latex README.md -o README.pdf
```

GitHub Actions

Pandoc can be run through GitHub Actions. For some examples, see https://github.com/pandoc/pandoc-action-example.

GitLab CI/CD

Pandoc can be run through GitLab CI/CD. For some examples, see https://gitlab.com/pandoc/pandoc-ci-example.

Compiling from source

If for some reason a binary package is not available for your platform, or if you want to hack on pandoc or use a non-released version, you can install from source.

Getting the pandoc source code

Source tarballs can be found at https://hackage.haskell.org/package/pandoc. For example, to fetch the source for version 1.17.0.3:

```
wget https://hackage.haskell.org/package/pandoc-
1.17.0.3/pandoc-1.17.0.3.tar.gz
tar xvzf pandoc-1.17.0.3.tar.gz
cd pandoc-1.17.0.3
```

Or you can fetch the development code by cloning the repository:

```
git clone https://github.com/jgm/pandoc
cd pandoc
```

Note: there may be times when the development code is broken or depends on other libraries which must be installed separately. Unless you really know what you're doing, install the last released version.

Quick stack method

The easiest way to build pandoc from source is to use stack:

1. Install stack. Note that Pandoc requires stack >= 1.7.0.

2. ```
   stack setup
   stack install pandoc-cli
   ```
   ```

`stack setup` will automatically download the ghc compiler if you don't have it. `stack install` will install the `pandoc` executable into `~/.local/bin`, which you should add to your `PATH`. This process will take a while, and will consume a considerable amount of disk space.

Quick cabal method

1. Install ghcup. This will give you `ghc` and `cabal`.

2. Update your package database:

   ```
   cabal update
   ```

3. Use `cabal` to install pandoc and its dependencies:

   ```
   cabal install pandoc-cli
   ```

   This procedure will install the released version of pandoc, which will be downloaded automatically from HackageDB. The `pandoc` executable will be placed in `$HOME/.cabal/bin` on linux/unix/macOS and in `%APPDATA%\cabal\bin` on Windows. Make sure this directory is in your path.

   If you want to install a modified or development version of pandoc instead, switch to the source directory before running the above command – cabal will use the local code for all projects mentioned in the `cabal.project`.

4. You should now be able to run `pandoc`:

   ```
   pandoc --help
   ```

5. Cabal does not install the `pandoc.1` man page, but you can copy it from the `man/` directory of the source code to `/usr/local/share/man/man1/` or wherever man pages go on your system.

Custom cabal method

This is a step-by-step procedure that offers maximal control over the build and installation. Most users should use the quick install, but this information may be of use to packagers. For more details,

see the Cabal User's Guide. These instructions assume that the pandoc source directory is your working directory. You will need cabal version 2.0 or higher.

1. Install dependencies: in addition to the Haskell platform, you will need a number of additional libraries. You can install them all with

   ```
   cabal update
   cabal install --only-dependencies
   ```

2. Configure:

   ```
   cabal configure --prefix=DIR --bindir=DIR --
   libdir=DIR \
     --datadir=DIR --libsubdir=DIR --datasubdir=DIR --
   docdir=DIR \
     --htmldir=DIR --program-prefix=PREFIX --program-
   suffix=SUFFIX \
     --mandir=DIR --flags=FLAGSPEC --enable-tests
   ```

   All of the options have sensible defaults that can be overridden as needed.

   `FLAGSPEC` is a list of Cabal configuration flags, optionally preceded by a `-` (to force the flag to `false`), and separated by spaces. `pandoc`'s flags include:

   - `embed_data_files`: embed all data files into the binary (default no). This is helpful if you want to create a relocatable binary.

   `pandoc-cli`'s flags include:

   - `lua`: compile in support for Lua filters and custom writers.

   - `server`: compile in support for running in HTTP server mode when the executable is renamed (or symlinked as) `pandoc-server`.

3. Build:

   ```
   cabal build
   cabal test
   ```

4. Build API documentation:

```
cabal haddock --html-location=URL --hyperlink-source
```

Creating a relocatable binary

It is possible to compile pandoc such that the data files pandoc uses are embedded in the binary. The resulting binary can be run from any directory and is completely self-contained. With cabal, add `-fembed_data_files` to the `cabal configure` or `cabal install` commands.

With stack, use `--flag pandoc:embed_data_files`.

Running tests

Pandoc comes with an automated test suite. To run with cabal, `cabal test`; to run with stack, `stack test`.

To run particular tests (pattern-matching on their names), use the `-p` option:

```
cabal test --test-options='-p markdown'
```

Or with stack:

```
stack test --test-arguments='-p markdown'
```

It is often helpful to add `-j4` (run tests in parallel) and `--hide-successes` (don't clutter output with successes) to the test arguments as well.

If you add a new feature to pandoc, please add tests as well, following the pattern of the existing tests. The test suite code is in `test/test-pandoc.hs`. If you are adding a new reader or writer, it is probably easiest to add some data files to the `test` directory, and modify `test/Tests/Old.hs`. Otherwise, it is better to modify the module under the `test/Tests` hierarchy corresponding to the pandoc module you are changing.

Running benchmarks

To build and run the benchmarks:

```
cabal configure --enable-benchmarks && cabal build
cabal bench
```

or with stack:

```
stack bench
```

To use a smaller sample size so the benchmarks run faster:

```
cabal bench --benchmark-options='-s 20'
```

To run just the markdown benchmarks:

```
cabal bench --benchmark-options='markdown'
```