pandoc Dockerfiles

This repo contains a collection of Dockerfiles to build various pandoc container images.

Contents

Available Images

Docker images hosted here have the following variants:

- minimal: kept as small as possible. See the pandoc/minimal repository.
- core: suitable for common conversion tasks; includes additional libraries and programs. See the pandoc/core repository.
- latex: builds on top of the core image, and provides a basic LaTeX installation in addition. This includes all packages that `pandoc` *might* use, and any libraries needed by these packages. See the pandoc/latex repository.
- extra: extends the latex image with a curated selection of templates, filters, fonts, etc. See the pandoc/extra repository.

All images are based on the `alpine` stack. The pandoc/minimal, pandoc/latex and pandoc/extra images are also available with an `ubuntu` stack.

Usage

> **Note**: this section describes how to use the docker images. Please refer to the `pandoc` manual for usage information about `pandoc`.

Docker images are pre-provisioned computing environments, similar to virtual machines, but smaller and cleverer. You can use these images to convert document wherever you can run docker images, without having to worry about pandoc or its dependencies. The images bring along everything they need to get the job done.

Basic Usage

1. Install Docker if you don't have it already.

2. Start up Docker. Usually you will have an application called "Docker" on your computer with a rudimentary graphical user interface (GUI). You can also run this command in the command-line interface (CLI):

   ```
   open -a Docker
   ```

3. Open a shell and navigate to wherever the files are that you want to convert.

   ```
   cd path/to/source/dir
   ```

   You can always run `pwd` to check whether you're in the right place.

4. Run docker by entering the below commands in your favorite shell.

   Let's say you have a `README.md` in your working directory that you'd like to convert to HTML.

   ```
   docker run --rm --volume "`pwd`:/data" --user `id -u`:`id -g` pandoc/latex:2.6 README.m
   ```

   The `--volume` flag maps some directory on *your machine* (lefthand side of the colons) to some directory *in the container* (righthand side), so that you have your source files available for pandoc to convert. `pwd` is quoted to protect against spaces in filenames.

   Ownership of the output file is determined by the user executing pandoc *in the container*. This will generally be a user different from the local user. It is hence a good idea to specify for docker the user and group IDs to use via the `--user` flag.

   `pandoc/latex:2.6` declares the image that you're going to run. It's always a good idea to hardcode the version, lest future releases break your code.

   It may look weird to you that you can just add `README.md` at the end of this line, but that's just because the `pandoc/latex:2.6` will simply prepend `pandoc` in front of anything you write after `pandoc/latex:2.6` (this is known as the `ENTRYPOINT` field of the Dockerfile). So what you're really running here is `pandoc README.md`, which is a valid pandoc command.

   If you don't have the current docker image on your computer yet, the downloading and unpacking is going to take a while. It'll be (much) faster the next time. You don't have to worry about where/how Docker keeps these images.

Pandoc Scripts

Pandoc commands have a way of getting pretty long, and so typing them into the command line can get a little unwieldy. To get a better handle of long pandoc commands, you can store them in a script file, a simple text file with an `*.sh` extension such as

```
#!/bin/sh
pandoc README.md
```

The first line, known as the *shebang* tells the container that the following commands are to be executed as shell commands. In our case, we really don't use a lot of shell magic, we just call pandoc in the second line (though you can get fancier, if you like). Notice that the `#!/bin/sh` will *not* get you a full bash shell, but only the more basic ash shell that comes with Alpine linux on which the pandoc containers are based. This won't matter for most uses, but if you want to write writing more complicated scripts you may want to refer to the `ash` manual.

Once you have stored this script, you must make it executable by running the following command on it (this may apply only to UNIX-type systems):

```
chmod +x script.sh
```

You only have to do this once for each script file.

You can then run the completed script file in a pandoc docker container like so:

```
docker run --rm --volume "`pwd`:/data" --entrypoint "/data/script.sh" pandoc/latex:2.6
```

Notice that the above `script.sh` *did* specify `pandoc`, and you can't just omit it as in the simpler command above. This is because the `--entrypoint` flag *overrides* the `ENTRYPOINT` field in the docker file (`pandoc`, in our case), so you must include the command.

GitHub Actions

GitHub Actions is an Infrastructure as a Service (IaaS) from GitHub that allows you to automatically run code on GitHub's servers on every push (or a bunch of other GitHub events).

Such continuous integration and delivery (CI/CD) may be useful for many pandoc users. Perhaps, you're using pandoc convert some markdown source document into HTML and deploy the results to a webserver. If the source document is under version control (such as git), you might want pandoc to convert and deploy *on every commit*. That is what CI/CD does.

To use pandoc on GitHub Actions, you can leverage the docker images of this project.

To learn more how to use the docker pandoc images in your GitHub Actions workflow, see these examples.

Building custom images

The official images are bare-bones, providing everything required to use pandoc and Lua filters, but not much more. Often, one will want to have additional software available. This is best achieved by building custom Docker images.

For example, one may want to use advanced spellchecking as demonstrated in the [spellcheck] in the Lua filters collection. This requires the *aspell* package as well as language-specific packages. A good solution would be to define a new Dockerfile and to use `pandoc/core` as the base package:

```
FROM pandoc/core:latest
RUN apk --no-cache add aspell aspell-en aspell-fr
```

Create a new image by running `docker build --tag=pandoc-with-aspell .` in the directory containing the Dockerfile. Now you can use `pandoc-with-aspell` instead of `pandoc/core` to get access to spellchecking in your image.

See Docker documentation for more details, for example part 2 of the Get Started guide.

spellcheck

License

Code in this repository is licensed under the GNU General Public License, version 2.0 or later.