# How We Automated Our Ebook Builds With Pandoc and KindleGen



*Published on* 28 November 2013 *by*

Kent Bye

Puppet Labs is all about automation, so when we published our new continuous delivery ebook we wanted to create a workflow that could convert a Markdown file into fully featured EPUB and MobiPocket ebooks with as little manual intervention as possible.

The solution we eventually settled on used Pandoc to convert a Markdown page into an EPUB file, and Amazon's KindleGen command-line program to generate a MobiPocket file.

There are just a few simple metadata files you'll need to manually edit, but in the end Pandoc does a lot of the heavy lifting to automatically generate internal navigation files and a table of contents based upon the structure of your HTML/Markdown document.

I'll start with an overview of the workflow, show what the final commands look like, and then talk about some of the files that you'll have to set up in order to add all the proper

## Tags

DevOps (243)
Cloud (125)
PuppetConf (118)
Puppet Enterprise (113)
Modules (87)
Continuous delivery (66)
IT automation (54)

metadata. Finally, I'll talk about some helpful tools you can use to validate and test your ebook files.

# Why Creating EPUB and MobiPocket Files Is Hard

The prospect of creating MobiPocket or EPUB versions of your ebook can be overwhelming. Check out Amazon's Kindle Publishing Guidelines or read through this epic ebook creation tutorial from BB ebooks, which provides a comprehensive list of all the things you need to do.

The basic workflow they suggest is that you need to convert your content to HTML. Sounds simple enough, but then you have to do a bunch of additional things including:

- Adding anchor tags to the HTML for the chapter markers and subchapter headers
- Adding tags within your markup to indicate page breaks
- Handcrafting an internal navigation file named `toc.ncx`
- Creating a very similar HTML version of the table of contents
- Creating a cover image & title page
- Creating a `content.opf` XML file that defines all sorts of additional publishing metadata, the manifest of chapters, the spine ncx navigation, and the HTML table of contents guide file.

That's a lot of grunt work and manual steps that computers should be able to help out with. Fortunately, it turns out that there's a much better and easier way to create an ebook.

First off, manually adding HTML tags to content is a pain. That's why John Gruber created Markdown: so writers could use a lightweight syntax that would convey meaning without adding opening and closing HTML tags, and without needing to see the fully-rendered presentation of the formatting. Starting from Markdown was one of our requirements.


Windows (50)

Learning Puppet (46)

AWS (42)

VMware (42)

Automation (37)

Security (37)

Puppet Camp (34)

Docker (32)

Openstack (32)

Configuration management (31)

Containers (29)

Open source (24)

PuppetConf 2016 (21)

I then started crafting some HTML from Markdown that would be converted to a Kindle-ready MobiPocket file with Amazon's KindleGen program. However, the problem with starting with creating a MobiPocket file is that there's a ton of additional manual steps listed above that had no clear path toward automation.

So I wasn't able to find a satisfying tool to be able to start with creating a MobiPocket file. The consensus seems to be that it's better with an EPUB version, and then use that to generate your MobiPocket file with KindleGen.

There are a lot of automated tools out there for converting Markdown to EPUB, but the biggest issue issue I ran into is that many either completely ignore the process of generating an HTML table of contents and ncx navigation file, or the process of adding these and other metadata was too manual and unwieldy.

This is where Pandoc steps in to save the day.

# Pandoc

Pandoc is emerging as the Swiss army knife of document conversions. It's really taking off in the academic and scientific community since it takes lightweight Markdown (or a dozen other) inputs and outputs to LaTeX, HTML, EPUB, PDF and about 30 other formats.

This is about the simplest pandoc command you can run.

```
pandoc -o my-ebook.epub my-ebook.md
```

That command converts a Markdown file to an EPUB file. As you might guess, Pandoc looks at the extensions to figure out what type of conversion you want to do.

You can also concatenate input files by just adding a space and the file name. This command combines three Markdown files into one EPUB file.

```
pandoc -o my-ebook.epub chapter01.md chapter02.md chapter03.md
```

The best thing about Pandoc is that it uses `H1` tags to determine the break points for chapters. This means you don't have to manually split your content into individual HTML chapter files. You can also designate H2 or H3 tags as the break points with the `--epub-chapter-level` option. It also automatically generates the `toc.ncx` navigation files and HTML table of contents and adds all of the appropriate anchor tags so you don't have to manually craft these files or add any page breaks. You get all of these navigation features for free based upon the inherent HTML structure of your document.

If you want to have second- or third-level navigation within these navigation files, you can pass in the table of contents depth, an it'll look to your `H2` and `H3` tags and automatically add anchor tags to your HTML and add it to the table of contents and navigation.

Pandoc also has a very lightweight way to add all of the additional metatdata, and even add a customized CSS file if you'd like. You can read Pandoc's comprehensive README file on GitHub to learn all the various options.

In the end, Pandoc proved to be the Holy Grail of ebook generation, so let's install Pandoc and KindleGen and take a look at the commands I ended up using.

# Installing Pandoc & KindleGen

Pandoc has a lot of dependencies, so you should try to install it using a package management system or as a prebuilt binary rather than trying to compile it from source.

You can download a Mac ready DMG from here or look at other installation options here.

You can download Amazon's KindleGen tarball from here,

unpack it, and place the KindleGen command somewhere where it's available in your shell's path. I placed it in `/usr/local/bin`.

Now let's take a look at the final commands and unpack what's happening.

# Pandoc & KindleGen Commands for Ebook Conversion

John MacFarlane -- author of Pandoc -- wrote up a simple blog post for how to generate an ebook. This was a great beginner's guide, but I needed to add some additional flags and a bit more information in order to create fully functional EPUB and MobiPocket files.

Here are the commands that I ran in order to generate an EPUB and MobiPocket file from Markdown.

First get an EPUB version with Pandoc:

```
pandoc -o my-ebook.epub title.txt my-ebook.md --epub-cover-image=cover.jpg --epub-metadata=metadata.xml --toc --toc-depth=2 --epub-stylesheet=stylesheet.css
```

Then you can feed that into KindleGen to generate a MobiPocket file:

```
kindlegen my-ebook.epub
```

That's it! Done! You should now have both a `my-ebook.epub` & `my-ebook.mobi` file to start testing.

Let's unpack what's happening with each argument we used:

- `-o my-ebook.epub`
  - Tells pandoc to generate an EPUB file
- `title.txt`
  - Creates a title page and some title and author metadata. You can optionally append this information to the beginning of your first input document.
- `my-ebook.md`
  - The actual ebook content in Markdown format. It contains multiple `H1` tags, which will get split into different chapters. You can concatenate more files here if need to add chapters. You can also pass HTML as input or add URLs here.
- `--epub-cover-image=cover.jpg`
  - Creates a cover image file and embeds the proper metadata
- `--epub-metadata=metadata.xml`
  - You can define additional Dublin core tags about the file, including published date, author, publisher, rights, and language. Some of this is pulled from the title page, such as title, author & published date.
- `--toc`
  - This tells Pandoc to add an HTML table of contents to the beginning of the EPUB file. The machine navigation and toc.ncx file are generated automatically, but I found that the HTML version is not always included.
- `--toc-depth=2`
  - Tells Pandoc to look at the `H2` tags and add then as secondary navigation links within both the navigation file and the HTML index. By default, it will look at a depth of 3.
- `--epub-stylesheet=stylesheet.css`
  - You can include this option if you want to make any CSS changes or tweaks

# Metadata Files for Pandoc

Let's look at what's in a couple of these metadata files that we passed into Pandoc.

The `title.txt` file is just two lines containing the book title and the author. This generates the title page as well as some additional metadata:

```
% My Ebook Title
% John Doe
```

The `metadata.xml` file adds in additional metadata to the file according to the Dublin Core Metadata Element Set

```
<dc:title>My Ebook Title</dc:title>
<dc:language>en-US</dc:language>
<dc:creator opf:file-as="Doe, John" o
pf:role="aut">John Doe</dc:creator>
<dc:publisher>John Doe Publishing</dc
:publisher>
<dc:date opf:event="publication">2013
-11-07</dc:date>
<dc:rights>Copyright ©2013 by John Do
e</dc:rights>
```

Finally, you can make additional design tweaks by altering the default EPUB CSS stylesheet. What I did was create a simple prototype EPUB document with `pandoc -o my-ebook.epub my-ebook.md`.

To look at the guts of an EPUB file, you can change the `.epub` extension to `.zip`. An EPUB file is technically just a compressed folder containing a number of HTML files, images, and a number of other XML manifest and metadata files.

I ended up needing to use the B1FreeArchiver to extract the zip file, since the default Mac Archive Utility got into a recursive loop of unpacking it to a `*.zip.cpgz` and then back into a `*.zip` file. Other alternatives to solving this are here.

You'll see a stylesheet.css in the unzipped folder that you can copy back into the base directory where you're executing the conversion. Feel free to add any CSS styling changes here

and be sure to set it with the `--epub-stylesheet` flag.

So after you've generated some initial EPUB and MobiPocket files, you're going to want to test them and iterate on formatting and design.

# A Sample Markdown File

If you want a fun sample Markdown file to convert into an EPUB and MobiPocket format, then download Pandoc's README & user guide to a file named README.md.

You can run the following command to generate an EPUB file.

```
pandoc -o readme.epub README.md
```

You'll notice that the ten `H1` headers were converted to chapters and that the default ROC depth is three. Also note that the title page is generated from the first three lines instead of requiring a `title.txt` file.

# Previewing Your EPUB & MobiPocket Files

Essential tools for previewing your ebook files include:

- Adobe's free Digital Editions EPUB reader
- Amazon's Kindle Previewer, which allows you to preview what the MobiPocket file looks like all of the variants of their e-ink devices, Kindle Fires and Kindle for iOS readers.

Note that these are just emulators: It's good to take your new files for a spin on the actual devices, since emulators don't always render the ebook true to how it would appear on the emulated device.

A good tool I used to confirm that all of the metadata was

properly added was Calibre. Calibre is an ebook conversion program with a GUI and some great features to confirm that the metadata you entered into `metadata.xml` was saved to the file properly.

# Validating & Debugging your EPUB & MobiPocket files

It's also good to run your EPUB file through a validator to be sure you don't have any broken links or other warnings or errors. You can upload your EPUB file to the International Digital Publishing Forum website, and it'll run the EpubCheck validator on it.

If you'd prefer to run it locally, then you can download EpubCheck from their releases and run the following command after unzipping it:

```
java -jar ~/Downloads/epubcheck-3.0.1
/epubcheck-3.0.1.jar my-ebook.epub
```

If you're curious about what the source HTML looks like on an EPUB, then you can simply change the `*.EPUB` extension to `*.zip` and unarchive the folder as mentioned above.

You can also decompile your MobiPocket file, then you can use this Mobi Unpack (v047) program as described by Liz Castro here.

You can navigate to the Mobi_Unpack_v047 folder from the terminal, and then run

```
python Mobi_Unpack_v047.pyw
```

Select your MobiPocket input file, and an appropriate output directory, and then hit "Start." It'll unpack your MobiPocket file into a mobi7 folder with the source files for e-ink readers

and an mobi8 folder with the KF8 files used for the Kindle Fire.

If you're getting some validation errors, then BB Ebooks has an excellent troubleshooting section in their how-to write up.

# Learning Pandoc's Markup

There are many different flavors and implementations of Markdown syntax. Pandoc uses its own special flavor of Markdown.

The Pandoc author wrote a neat online tool called Babelmark 2 that allows you to compare and contrast how different markup syntaxes will render into HTML.

For example, I wanted to find which Markdown variant provided the easiest table syntax. I ended up settling on this pipe table syntax.

For more details on the nuances of the Pandox syntax, then take a look at the "Pandoc's markdown" section on pandoc's README.

You can also test out conversion snippets on John MacFarlane's Try pandoc! page.

I enjoyed using the Mou markdown editor in order to get bootstrapped on Markdown because it has a nice live preview. However, it doesn't use Pandoc's Markdown syntax, and can get slow if the file starts to get too big.

There is a nice browser-based, markdown live editor that is compliant with Pandoc's syntax called Markx. They have a hosted version at http://markx.herokuapp.com that is helpful for getting a little bit more interactive experience for how the Markdown will render to HTML.

There was also an alpha-quality app called Kalam that uses node-webkit to create a live preview of Pandoc syntax. I found it to be a bit buggy, and limited as an interactive editor, but helpful as another Pandoc live preview option.

# Images, Design & CSS Media Queries

Pandoc and KindleGen are powerful tools that get you 80-90 percent of the way there in creating professional looking ebooks. The last 10-20 percent is a lot of iteration on formatting, styling and special CSS rules, and even customized fonts with the `--epub-embed-font` option.

One issue that I ran into is making sure that images were the best size on all platforms. What may have looked okay on an e-ink reader looked way too small on the Kindle Fire. One way that Amazon suggests dealing with this is to use media queries.

Chapter 8 of the Amazon Kindle Publishing Guide talks about how to target the Kindle Fire KF8 or older e-ink MobiPocket files.

> For KF8 CSS styles, use the media query `@media amzn-kf8`. This is only applied for the KF8 format.
>
> For Mobi CSS styles, use the media query `@media amzn-mobi`. This is only applied for the Mobi format.

There are also some sample files listed on the sidebar that you can download with different Kindle-specific formatting tips. The KF8Sample is particularly instructive for all of the formatting options that are available in the new KF8 format.

We're excited about this first iteration of an ebook generation workflow. If you'd like to see how our conversions came out, then be sure to download our free Continuous Delivery ebook

## Share via:

## Posted in: Tips & How To

---

Yves

4 years ago

Wow, what a great article! I have been trying to settle on which markup solution to use for generating latex, epub and html from the same source. There are a few options, but I also require footnotes and other little things that I found mutlimarkdown or pandoc were best suited for. Installation of these is always the most difficult part on Linux. I will give Pandoc a try to see if it suits my needs. Do you have a favorite editor (preferable cross platform)?

Reply

---

## Kent Bye

4 years ago

Reply

---

## yves

4 years ago

Thanks a lot for your reply Kent. I have been playing around with Pandoc for the last hour. Seems to do all that I need to do. It will so much faster to write in Pandoc and export to Latex than writing latex myself.

I will definitely check out those tools.

Reply

---

## Tara Roys

4 years ago

I can't tell you how much time this blog post has saved me. Thank you!

I will add an addendum: When using the pandoc command exactly as listed,

pandoc -o my-ebook.epub title.txt my-ebook.md --epub-cover-image=cover.jpg --epub-metadata=metadata.xml --toc --toc-depth=2 --epub-stylesheet=stylesheet.css

I got the following error when I tried to convert the resulting epub into a mobi with kindlegen.

Error(opfparser):E20006: There are more than one title defined in OPF metadata. But none of them is refined with "title-type" as "main" title. Refer

http://idpf.org/epub/30/spec/epub30-publications.html#sec-opf-dctitle for more info.

I solved it by deleting the title.txt from the command. My reasoning was that for some reason pandoc was reading the title from both title.txt and metadata.xml, and in the conversion process something was clashing.

I don't know exactly what title.txt does, so I don't know what removing it does, but the resulting kindle ebook looks fine to me, and when I uploaded it to the Amazon Web Store as a test, it said it passed their validation standards, so I guess the title.txt is not really necessary.

So my question is: what exactly does the title.txt do for you, and what am I losing by getting rid of it?

Cheers!
Tara Roys

Reply

---

## Kent Bye

4 years ago

The title.txt was put in there in order to have a title page, but I believe in the latest version of Pandoc, the author removed some of those redundancies. When I had written John MacFarlane, he said, "You'll be interested to know that I've made some improvements to the EPUB writer for the next release (which should be soon). You can now specify all EPUB metadata in a YAML block in the source document itself."
That latest version has since come out, and I haven't verified what's changed.
I couldn't get the YAML block to work properly, and you can read more info about that here: http://johnmacfarlane.net/pandoc/README.html#epub-metadata

And I'd defer you to reading through the Pandoc README file at http://johnmacfarlane.net/pandoc/README.html for the latest, up to date information.

Reply

---

## Daniel Li

4 years ago

Hi, Sara,
Nice to hear that you successfully converted a .epub book with the --epub-metadata=metadata.xml option. I was trying to do the same but I got the error, and I am wondering if my metadata.xml file is not correct. Could you please post your full metadata.xml file? That could help me and others with the same problem.
Thanks
Daniel

Reply

---

## Rob Levin

4 years ago

Pandoc install now at:
https://github.com/jgm/pandoc/releases/

Reply

---

## Bill Maddock

3 years ago

Much thanks for the insights in your article.

Regarding converting to epub, I have found several awkward problems(quite annoying) using it.

* Why, oh why is the default for toc generation numbered toc headings ? In a normal book or ebook you rarely, if ever, number the chapter links in your toc . Stupid.

* In regard to the silly and clunky way you have to put in line breaks in your text in pandoc -- how difficult can it be just to convert an ordinary text line break into an html line break? If html editors can do it then why can't pandoc markdown do it? Using double spaces to denote line breaks is not useful because you cannot SEE that it is there. I got so sick of this I now just put an html break on a new line like this:

Some text in a paragraph.
<br/>
Now you can SEE that you have a line break.

* I'm a fiction novel writer and that means that my first paragragh at the beginning of all my chapters is not indented whereas all paragraghs after the first paragraph should have an indent. This is a normal standard in fiction. How do you do this easily and quickly in pandoc markdown? I also want to be able to do this easily and quickly without having to learn perl, ruby, php or whatever. And if I have to do it by html in pandoc then that means I would have to code about 700 paragraghs separately in html in my book by hand. Not easy and not helpful.

I can see why pandoc markdown would be a wonderful utility for academia -- perhaps because they only ever write thesis papers in block text only with no indents. But as a useful utility for novel writer like me, pandoc does have some major inadequacies.

Reply

---

## Liam C-F

3 years ago

Hi Bill, I don't know if you're still struggling with this, but I can answer some of your questions.

I can't speak for the TOC, since I don't bother with it at all (I think they're ugly, and for fiction they're kind of unnecessary... maybe if I was doing a short story collection). It's probably easily fixed with a line in the CSS file, though of course the issue is in figuring out exactly what that magical line is...

Line breaks working the way they do is a design choice for markdown, and it's that way because a lot of computer nerds like to have their text editors automatically place line breaks at a certain line length -- so, once the line gets up to (say) 70 characters, there will be a line break inserted automatically. I think it's a pretty ugly cludge in these days of reliable word-wrap, but there it is -- and, of course, you wouldn't want *those* line breaks being translated into output formats like epub!

However, pandoc offers a more visible way of placing linebreaks in the output format: you can put a backslash (`\`)at the end of the line.

I feel your pain with regards to paragraph indentation (though, really, that's a general epub/html problem, rather than pandoc's fault specifically). Here's a link to the CSS file I use for my epubs (http://pastebin.com/NVcbnVeE), which solves this problem. The main article describes how to use a custom CSS file with pandoc. The relevant lines are:

```
p { margin-bottom: 0; }
p + p { text-indent: 1.3em; margin-top: 0; }
hr { visibility: hidden; }
```

I use <hr>s to signify scene breaks within chapters, and render the horizontal lines invisible with that bit of CSS. The p+p there signifies that those rules are only applied to paragraphs that are directly after another paragraph -- so paragraphs after a heading or <hr> are *not* indented.

I hope that helps some!

Reply

---

Roberts — about a year ago

Breaks before H1. Thanks Liam, this is very helpful and I will be using that stylesheet. In general I find that pandoc works very well. I cannot figure out yet how one could make level 1 headings to _not_ have a page break before them. I am not sure if the breaks are within epub, or if they are there because the readers display that way. What I know is that the epub file is split in chapters at H1 level headings. What I would like is to be able to produce epub that would have level 1 headings without page breaks before them.

Reply

## Add new comment

**Your name** *

**Email** *

*The content of this field is kept private and will not be shown*

*publicly.*

**Comment** *

B I | Format | Source

B I | Format | Source

About text formats

**Text format**

Basic HTML

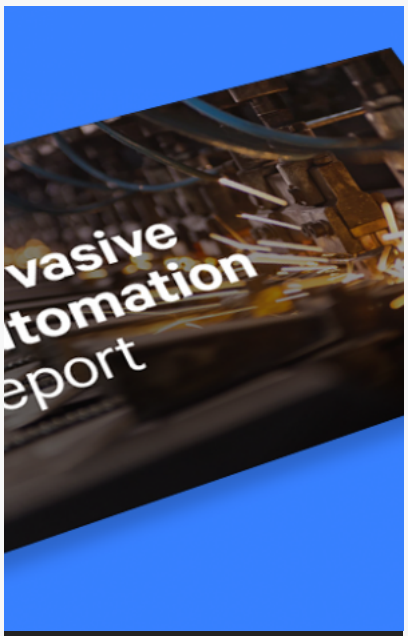
- Allowed HTML tags: <a href hreflang>   <cite> <blockquote cite> <code> <ul type> <ol start type> <li> <dl> <dt> <dd> <h2 id> <h3 id> <h4 id> <h5 id> <h6 id>
- Lines and paragraphs break automatically.
- Web page addresses and email addresses turn into links automatically.

Save

Preview

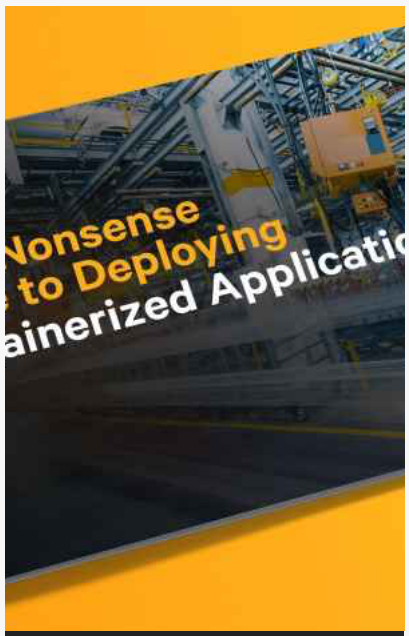
# Related content

See more



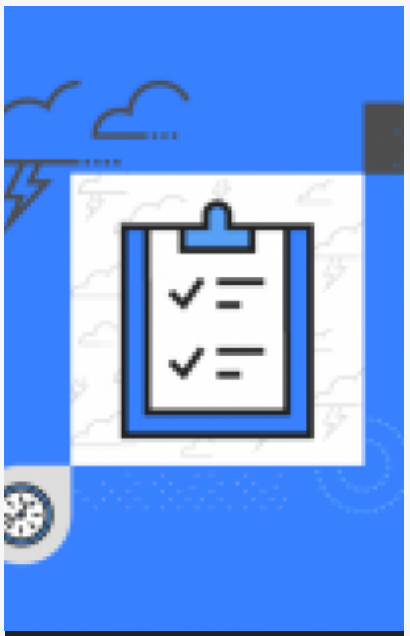

White paper

Pervasive Automation




Ebook

A No-Nonsense



Blog post

Combining PowerShell


Blog post

Puppetize Live update: