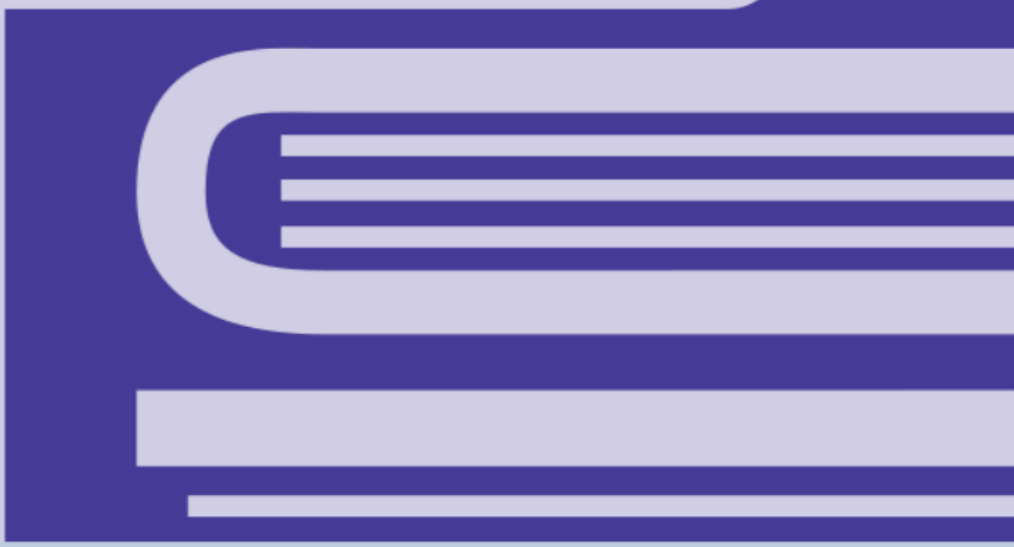

User's Guide

# BOOKSQUISHER

A multiformat book creation system based on Asciidoctor and build scripts from the book Pro Git, 2nd Edition

BY MICHAEL S. MARMOR

# Booksquisher

## *User's Guide*

Michael S. Marmor

Version 0, 2019-02-06

# Table of Contents

# Chapter 1. Start Here

# Booksquisher

## 1.1. Quick Overview

If you are looking for a well-designed system for writing books in one master format and allowing the computer to generate great looking, ready-to-publish versions in EPUB3, AZW3/MOBI (Kindle Format 8), PDF, and HTML5, then Booksquisher is for you. Don't "roll your own" book building system. Use the same open source system that super-smart people have already battle-tested in the real world.

The Booksquisher Docker image was originally created to provide a ready-to-run toolchain for building the open source book *Pro Git, 2nd Edition*, written by Scott Chacon and Ben Straub. You can still use it for making the *Pro Git 2* book. That said, you can *also* use the Docker image and associated Example Book Templates to create **your own** books based on the same fantastic technology stack and build approach as *Pro Git 2*.

The Booksquisher Example Book Templates are simplifications of the same markup language, book structure, and build system used by the *Pro Git 2* book. (Except they are only a few pages long instead of 500+ pages, so they are way easier to figure out when you are getting started!)

The documentation in this User's Guide explains step-by-step how to use the tools in the Booksquisher Docker image to generate finished books from the Booksquisher Example Book Templates, as well as the *Pro Git 2* book source code.

Important Links:

- Booksquisher Docker image on Docker Hub
- Booksquisher Example Book Templates on GitHub
- Pro Git 2 Book Source on GitHub

## 1.2. Introduction

> Booksquisher is a containerized, multi-format book creation system based on Asciidoctor and the build scripts used to generate the open source book *Pro Git, 2nd Edition*. Booksquisher consists of a Docker image containing all the tools, Example Book Templates, and this User's Guide. You can use Booksquisher anywhere that you can run Docker, which includes computers running Windows, Mac, and Linux operating systems.

The Booksquisher Docker image on Docker Hub contains the toolchain and core dependencies

required for building the English language version of the book *Pro Git, Second Edition* by Scott Chacon and Ben Straub. The AsciiDoc source for the *Pro Git 2* book as well as the build scripts are located on GitHub in the progit2 project.

The progit2 project uses an excellent build system for generating high-quality EPUB3, MOBI (Kindle Format 8, also called AZW3), PDF, and HTML5 books from a single source. The book content is written in AsciiDoc. The tools in the Booksqisher Docker image convert this AsciiDoc into beautiful books in multiple formats. The authors clearly learned a lot from creating the first edition of the *Pro Git* book, which used a different markdown language and toolchain (based on pandoc). We can all benefit from their experience by studying their AsciiDoc source and using their build system as a template. *Pro Git 2* is a high-profile open source book, and given its subject, it receives a lot of attention and collaborative input from seriously-smart developers. A lot of capable eyes have reviewed the book source and build system.

The Booksquisher Docker image and Example Book Templates give you an easy way to use the **structure** behind the *Pro Git 2* AsciiDoc source and build system as a model for creating books of your own. Booksquisher provides the tools, examples, and documentation you need to get started and focus on writing rather than the technical mechanics of generating formatted books.

## 1.3. Background

When I first started working with the *Pro Git 2* build system, I had a bear of a time getting all the tools and dependencies set up correctly. Once I figured out all the details and dependencies, I created a Docker image to containerize all the requirements into a package I could reuse easily on any platform.

*A technical note you can safely ignore:*

I later discovered the asciidoctor/docker-asciidoctor Docker image which contains most, but not all the needed tools. The Booksquisher Docker image on Docker Hub is now based on the asciidoctor/docker-asciidoctor image as a parent.

I created the Booksquisher Docker image as a convenience for my own use. I hope you find it useful as well. The Docker image itself consists of minor additions to the work of others. What you are reading now is the documentation that I wish I had as I banged my head on the keyboard in frustration trying to get the *Pro Git 2* book to build without errors and then to render correctly. In the process of figuring this out, I've gained a bit of knowledge that I'd like to pass on in these notes. I want technically capable writers to be able to use Booksquisher as a "black box" book creation system without needing to dive into the specifics of the underlying toolchain. Ideally, you should not need to know that the underlying system is Alpine Linux, or what version of Ruby is running in the container.

A huge percentage of books can be created by simply modifying the Booksquisher example book templates. More sophisticated markup syntax can be learned by reading the AsciiDoc Writers Guide on the Asciidoctor website. If you want to dive deeper, check out the Asciidoctor User Manual. If you procrastinate (like I do) by geeking out on open source tech, spend your time on mastering Asciidoctor-specific markup rather than figuring out underlying build dependencies! (This entire project was fueled by procrastination energy; I know of what I speak.)

## 1.4. Versions of This User's Guide

The Booksquisher User's Guide is available for download in several formats.

You are currently reading version 0 of the User's Guide (updated 2019-02-06). There are other formats of this Guide that you can download. These other formats are generated from the same source and should be more or less identical in content.

Visit https://www.booksquisher.com to read the User's Guide online or download other versions of this guide in EPUB3, Kindle KF8/MOBI/AZW3, and PDF formats.

## 1.5. TL;DR: Why You Need the Booksquisher Docker Image

You need the Booksquisher Docker image so that you can get up and running with a battle-tested book creation process without spending days figuring out how to get the tools to work. You still need to be a rock-star-ninja computer user, since you will be creating your book in AsciiDoc and using the command line. The Booksquisher Docker image gives you a machine-independent toolchain configuration and environment that is known to work. That will save you time, I promise.

# Chapter 2. Getting Started

## 2.1. Command Overview

For those that already know the tools and just want to see the commands, here they are. If this makes no sense to you, don't worry, all of this will be explained in greater detail in the next section of the User's Guide. Once you are familiar with how Booksquisher works, you can come back to this section to remember the most important commands.

```
$ git clone --depth 1 --origin source https://github.com/mmarmor/booksquisher-example-
book-templates.git new-book-project
$ docker pull marmor/booksquisher
$ docker run --name booksquish -it -v /<path-to>/new-book-project/:/documents/
marmor/booksquisher
[booksquisher]# cd /documents/book-template-1-simple
[booksquisher]# bundle install
[booksquisher]# bundle exec rake book:build
```

The generated books will be located in `new-book-project/book-template-1-simple/generated-books` on your local computer.

The Booksquisher Templates (but not the *Pro Git 2* book) allow you to quickly remove the build artifacts and all the generated books with a *clean* command:

```
[booksquisher]# bundle exec rake book:clean
```

You can combine cleaning *and* building into a single command. This is the command you will use the most as you write your book and incrementally run a build to see how it looks:

```
[booksquisher]# bundle exec rake book:clean book:build
```

Exit and shut down the Docker container when you are done with Booksquisher for the day:

```
[booksquisher]# exit
```

The Docker container is still on your system, but no longer running:

```
$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS     NAMES
84eb14767b6e        booksquisher        "/bin/bash"         13 seconds ago      Exited
(0) 9 seconds ago            booksquish
```

Re-start and attach to your container again. The same directory mapping you created before will still be ready to use:

```
$ docker start booksquish
$ docker attach booksquish
[booksquisher]#
```

Don't want to use this *container* any more? Remove it like this:

```
$ docker container rm booksquish
```

The `marmor/booksquisher` Docker *image* is still on your system (even if you remove all the *containers*), so you can quickly make a new *container* from the image with the `docker run` command we used when we got started above:

```
$ docker images
REPOSITORY                      TAG                 IMAGE ID            CREATED
SIZE
marmor/booksquisher             latest              72ebe621b148        16 hours ago
531MB
$ docker run --name booksquish -it -v /<path-to>/booksquisher-example-book-
templates/:/documents/ marmor/booksquisher
```

Get the latest version of the Booksquisher Docker image on Docker Hub:

```
$ docker pull marmor/booksquisher
```

## 2.2. Prerequisites

To use the Booksquisher Docker image, you'll need Docker installed.

- Windows
- OS X
- Linux

You should also have Git installed to get the Booksquisher example book templates (or the `progit2` source) on your local machine.

- Git

*A technical note about Git that you can safely ignore:*

Note that Git is also included within the toolchain inside the Booksquisher Docker container. This is potentially confusing, so let me explain: The “internal” Git is used to pull the document version number which is included in the generated books. For this reason (and many others), it makes sense to use Git on your local machine when you work on your own book projects. The “internal” Git in the container is also used in the `progit2` build script to fetch the most recent development version of `asciidoctor-epub3`, one of the critical tools for this build process—and still in alpha—from GitHub. The Docker image already contains an alpha-release version of `asciidoctor-epub3` which you can use when building your own books, but the developers of the `progit2` build script want us using the latest *development version* at the time you run `bundle install` (see below). All other tools needed to build the book are included in the Booksquisher Docker image.

## What You Don’t Need

The wonderful thing about Docker is that you don’t need to have Asciidoctor, Ruby, or other book-building software installed on your local computer (or the associated interlocking maze of dependencies and version requirements). The Booksquisher Docker image contains all the necessary tools to take the book source code as input and generate high-quality formatted books in EPUB3, Kindle Format 8 (AZW3/MOBI), PDF, and HTML5 format as output. The Docker image has all the tools to do this for the Booksquisher Example Book Templates, the *Pro Git 2* book, the *User’s Guide* you are reading now, or for creating your own books.

# Chapter 3. Step-by-Step Instructions for Building the Booksquisher Example Book Templates

## 3.1. Getting the Files

Use Git to clone the Booksquisher Example Book Templates on GitHub to your computer. Your local machine will need to have Git installed to do this:

```
$ git clone --depth 1 https://github.com/mmarmor/booksquisher-example-book-
templates.git
```

Get the Booksquisher Docker image on Docker Hub with `docker pull`:

```
$ docker pull marmor/booksquisher
```

## 3.2. Running and Checking the Container

Run the Booksquisher Docker image to create a container on your local machine. Also, map the `booksquisher-example-book-templates` book source directory on your local machine to the `/documents` directory *inside* the container. See below for specific Windows and Linux examples:

```
$ docker run -it -v <path to booksquisher-example-book-templates
directory>:/documents/ marmor/booksquisher
```

For clarity, here are specific examples of the `docker run` command above for both Windows and Linux:

Specific example for Docker for Desktop on Windows (running in Windows PowerShell):

```
PS C:\Users\marmor> docker run -it -v C:\Users\marmor\Desktop\booksquisher-example-
book-templates:/documents/ marmor/booksquisher
```

Specific example for Docker on Linux:

```
$ docker run -it -v /home/marmor/booksquisher-example-book-templates/:/documents/
marmor/booksquisher
```

Regardless of what operating system you have on your own computer, your terminal should display the Booksquisher welcome and the container's Linux prompt:

```
** Hi! Welcome to Booksquisher!
** Please see the Booksquisher User's Guide at www.booksquisher.com
[booksquisher]#
```

Booksquisher runs Linux *inside the container* and that is where we will run the next commands. All the commands below are to be run inside the Docker container Linux terminal at the `[booksquisher]#` prompt, not on your local computer.

Check that we are in the directory `/documents` within the container:

```
[booksquisher]# pwd
/documents
```

Check that the `booksquisher-example-book-templates` source code is visible when you list the directory with `ls`.

```
[booksquisher]# ls
README.md
book-template-1-simple
book-template-2-more-complex
```

It is helpful to understand what you see when you list the `/documents` directory as we did above. This directory is **both** on your local computer *and* inside the Docker container. You see it in the Docker container because we mapped it from your computer into the Docker container with the `docker run` command above. This directory mapping allows the tools in the Docker container to process the book source files on your local computer. Books generated by tools in the container will be written to your local computer through the mapping.

Said in another way, Booksquisher is a pre-configured container for book-building tools such as Asciidoctor, allowing you to process book source code on your computer without actually installing the required software and prerequisites on your operating system. This should begin to make more sense as we generate formatted book output files and view them on your local computer.

Change directory into the first example book template `book-template-1-simple` and then list the source files for that example book. Here we will use the `tree` command instead of `ls` so you can see the structure of the example template:

```
[booksquisher]# cd book-template-1-simple

[booksquisher]# pwd
/documents/book-template-1-simple

[booksquisher]# tree
.
├──── Gemfile
├──── README.md
├──── Rakefile
├──── book
│     ├──── 02-name
│     │     └──── sections
│     │          └──── example-section-to-be-included.asc
│     ├──── cover.png
│     └──── preface.asc
├──── ch01-name.asc
├──── ch02-name.asc
├──── generated-output
├──── images
│     ├──── booksquisher-logo.svg
│     └──── example-image.png
├──── my-book-name.asc
├──── research
│     ├──── character-sketches
│     │     └──── character-sketch-template.txt
│     ├──── ideas
│     │     └──── ideas-go-here.txt
│     └──── setting-sketches
│          └──── setting-sketch-template.txt
└──── sample-output
      ├──── images
      │     ├──── booksquisher-logo.svg
      │     └──── example-image.png
      ├──── my-book-name.epub
      ├──── my-book-name.html
      ├──── my-book-name.mobi
      └──── my-book-name.pdf
```

## 3.3. Building the Book

Now we are ready to run the build commands for the `book-template-1-simple` example template. This is a directory in the Booksquisher Example Book Templates on GitHub project you cloned earlier using Git. Pause for a second and think about what we are about to do:

- We are going to run scripts which are part of the Example Book Template project that is sitting on disk on your local computer, but are mapped into the Docker container as `/documents/book-template-1-simple/`.

- The scripts are going to be run from Linux within the Booksquisher Docker container, which has all the software needed to act on the scripts.
- The output of the scripts will be formatted book files, which will be placed into the `/documents/book-template-1-simple/generated-output/` directory in the container, which is the **same** as `booksquisher-example-book-templates/book-template-1-simple/generated-output/` on your local computer.

Clear as mud, right? This description makes it sound complicated, but in practice it is not. Here is the point: the tools in the container are going to process the book source files and then put finished book files on your local computer in the `generated-output` folder.

The Booksquisher Docker container contains all the prerequisites for building the book into all the output formats. (Or at least that was true at the time I created the Booksquisher Docker image!) When you run `bundle install` — which we are about to do below — the script checks for any missing or out-of-date Ruby Gems, downloads them and installs them. So if there *are* new requirements, the script should take care of that for you. This process should work without failing because the Booksquisher container already has the required build software for the Gems to compile natively.

*The Next Command Could Take a Few Minutes to Run!*

Depending on your network connection speed, the first time you run `bundle install` may *take as long as 5-minutes*! **You do not get a lot of visual feedback as this is running, so don't panic if it seems stalled—just wait a bit**. I know I said above that Booksquisher contains all the prerequisites for building the book, but there is one important exception. Some build scripts, including the one for `book-template-1-simple` (as well as *Pro Git 2*) fetch the *current development version* of `asciidoctor-epub3` directly from GitHub—as well as the latest Ruby Gem metadata from `rubygems.org`. The `bundle install` process also checks for updated versions of many of the Ruby Gems needed to build the book. Luckily, we only need to run `bundle install` **one time** to get set up for building this book template. (If you are interested, you can study the `Gemfile` in the `book-template-1-simple` directory to see what Gems `bundle install` is verifying for this specific book template. Review the `Rakefile` to see how the build process works.)

Be sure you are in the `/documents/book-template-1-simple` directory in the Docker container, then run `bundle install`.

Note that you should expect to see a warning message that says *'Don't run Bundler as root*'. The Booksquisher container only has a single user, which is root. You can safely ignore this warning message.

```
[booksquisher]# pwd
/documents/book-template-1-simple

[booksquisher]# bundle install
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and installing
your bundle as root will break this application
for all non-root users on this machine.
Fetching https://github.com/asciidoctor/asciidoctor-epub3
Fetching gem metadata from https://rubygems.org/..........
Resolving dependencies...
Using rake 12.3.2
Using Ascii85 1.0.3
Using public_suffix 3.0.3
Using addressable 2.6.0
Using afm 0.2.2
Using asciidoctor 1.5.6.1
Using mini_portile2 2.1.0
Using nokogiri 1.6.8.1
Using rubyzip 1.2.2
Using gepub 0.6.9.2
Using thread_safe 0.3.6
Using asciidoctor-epub3 1.5.0.alpha.9.dev from
https://github.com/asciidoctor/asciidoctor-epub3 (at master@d42f444)
Using pdf-core 0.7.0
Using ttfunk 1.5.1
Using prawn 2.2.2
Using prawn-icon 1.3.0
Using css_parser 1.6.0
Using prawn-svg 0.27.1
Using prawn-table 0.2.2
Using hashery 2.1.2
Using ruby-rc4 0.1.5
Using pdf-reader 2.2.0
Using prawn-templates 0.1.1
Using safe_yaml 1.0.4
Using polyglot 0.3.5
Using treetop 1.5.3
Using asciidoctor-pdf 1.5.0.alpha.16
Using awesome_print 1.8.0
Using bundler 1.17.3
Using coderay 1.1.2
Using epubcheck 3.0.1
Using json 2.1.0
Using kindlegen 3.0.3
Using multi_json 1.13.1
Using pygments.rb 1.2.1
Bundle complete! 11 Gemfile dependencies, 35 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

Once `bundle install` is run once successfully for *this book template* you should not need to run it

again. When you are ready to try out the template `book-template-2-more-complex` you will need to run `bundle install` again in that directory. Each Booksquisher book (as well as the *Pro Git 2* book) has its own build script (`Rakefile` and `Gemfile`), so you need to run `bundle install` for each book project you work on.

You might notice that a `Gemfile.lock` file is generated when you run `bundle install`. If you delete `Gemfile.lock` you will need to run `bundle install` again.

Assuming all went well with `bundle install`, you can now build all versions of the book (PDF, EPUB3, MOBI/KF8/AZW3, and HTML5) with this single command:

```
[booksquisher]# bundle exec rake book:build
```

You should expect to see something like this:

```
[booksquisher]# bundle exec rake book:build

Converting to HTML...
cp -r images generated-output/images/
 -- HTML output at my-book-name.html
Converting to EPub...
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
 -- Epub output at my-book-name.epub
Converting to Mobi (kf8)...
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
 -- Mobi output at my-book-name.mobi
Converting to PDF... (this one takes a while)
 -- PDF output at my-book-name.pdf
Done. Look for generated books in the generated-output directory.
```

You can safely ignore the `::Fixnum is deprecated` warnings. This is a known issue, and later releases of the tools within Booksquisher should eliminate this warning message. The output files are not impacted.

Look for the newly created PDF, EPUB3, MOBI, and HTML5 book files in the `generated-output` folder which is also on your local machine in the `booksquisher-example-book-templates/book-template-1-simple/generated-output` directory.

You should be able to open `my-book-name.pdf` with Adobe Acrobat or other PDF software on your local computer. Before you re-generate the files again with `bundle exec rake book:build` you should make sure you close your PDF reader, so that it does not prevent new versions of the file from being written.

Note that `my-book-name.mobi8` and `my-book-name-kf8.epub` are *intermediate* files created in the process of creating the `my-book-name.mobi` file. You don't need the intermediate files, they only exist as stepping stones to the final `.mobi` file.

*Understanding MOBI and KF8/AZW3 formats*

Even though our generated Kindle file has a `.mobi` extension, it is really a Kindle Format 8 (KF8) file internally. KF8 replaced the older MOBI format. The new Kindle Format 8 includes *both* the old-style MOBI database and a newer KF8 database in the same file for backwards compatibility using older Amazon Kindle readers. The KF8 format can have either a `.mobi` or `.azw3` extension.

The Booksquisher example book templates are modifications of the original *Pro Git 2* build scripts. The Booksquisher version is designed to put the generated books into a `generated-output` directory as opposed to the top level of the book project. The Booksquisher version also adds a *clean* feature.

To delete **all** of the generated book files and intermediate build artifacts you can run:

```
[booksquisher]# bundle exec rake book:clean
```

This should remove everything from the `generated-output` directory except for the hidden `.gitignore` file.

You can combine the process of removing old book versions *and* generating new ones into a single command like this:

```
[booksquisher]# bundle exec rake book:clean book:build
```

As you work on your new book on your local computer, you can keep the Booksquisher console open, and use the up-arrow key at the `[booksquisher]#` prompt to bring up the last command `bundle exec rake book:clean book:build` and simply press enter to clean up *and* build all versions of the book. This is the command and workflow I recommend as you write. (In fact, I'm going to run `bundle exec rake book:clean book:build` right now to clean and build the Booksquisher User's Guide!)

## 3.4. You are Doing Great!

If you have made it this far, congratulations! Your next mission—should you choose to accept it—is to modify the example templates so that they are useful as a starting point for creating your own books. We will cover that process in the next section Modifying the Booksquisher Example Book Templates.

As I mentioned before, a huge percentage of books can be created by simply modifying the Booksquisher Example Book Templates. More sophisticated markup syntax can be learned by reading the AsciiDoc Writers Guide on the Asciidoctor website. If you want to dive deeper, check out the Asciidoctor User Manual.

Now that you know how to build the Booksquisher Example Book Templates on GitHub, you can easily build the *Pro Git, Second Edition* book. *Pro Git 2* is a 500+ page monster compared to the example templates, but the concepts are the same. If you want to see what you can accomplish with AsciiDoc and this build system, build the progit2 project and study the book source. The section

Step-by-Step Instructions for Building the Book Pro Git 2 walks you through the process, but with a little less explanation than we have given here for the Booksquisher Example Book Templates. You can also try building the Booksquisher User's Guide on GitHub — the book you are reading now.

# Chapter 4. Modifying the Booksquisher Example Book Templates

## 4.1. Make a Plan

It is helpful to have a plan for your book before you start modifying a book template for your use. Even if you do not have book content created yet, you will still need to define the structure of your book, at least partially, before you can work with it in Booksquisher. If you are like most people, you will need to change the structure of your book many times as your ideas evolve, so don't let the planning process paralyze you.

Here is one plan of attack:

1. Define the name of your book ("My Fabulous Novel").
2. Define the primary filename of your book ("my-fabulous-novel"). All the various final-formatted versions will use this filename, for example, `my-fabulous-novel.epub`, `my-fabulous-novel.pdf`, `my-fabulous-novel.html`, etc.
3. Decide on the names of the first few chapters and the main sections of each chapter. ("Chapter 1: The Dirty Harmonica" has two sections: "Kansas" and "BBQ Sauce").

## 4.2. A Few Tips Before we Start Editing

- Build the book template before you start modifying it and make sure that it builds properly and without any errors. Examine the generated files and see if you like the formatting and output. Make a few small changes to the template and then build the book again. If the book does not build the way that you are expecting, try to figure out why (and fix it!) before making any other modifications to the template. Working in small iterative steps is the best way to get started.
- Use a text editor that you are already comfortable using. Don't try to learn a new text editor *and* Booksquisher at the same time. Which text editor is best is a topic where people have strong personal opinions. Because of that, I have moved the discussion about text editors to the Helpful Tips chapter in the section Text Editors. Here I'll limit myself to one sentence: If you don't already have a favorite text editor I suggest you use Visual Studio Code, which is free, easy to learn, runs on all major platforms, and has great support—via extensions—for AsciiDoc, Git, and code spell checking.
- Rather than immediately removing the existing content of the templates consider commenting them out until you have a working replacement. So rather than just deleting or editing `include::ch01-name.asc[]` comment it out with `//`:

  ```
  // include::ch01-name.asc[]
  ```

  Put your version right below the commented out version so you can compare the two.

  If you want to comment out large sections of the template file, you can use block comments:

```
////
This multi-line block will not
appear in the rendered output.
////
```

- You can put sub-sections of a chapter into multiple files to make large books easier to manage. There is an example of this in the `book-template-1-simple` Booksquisher Template in the file `ch02-name.asc`. Look for the code that looks like:

  ```
  include::book/02-name/sections/example-section-to-be-included.asc[]
  ```

  Organize sections in the book directory in clearly named chapter directories and files. In the example above it is easy to see that this is a section of chapter two. This description sounds complicated, but it is not. Examine the structure of the example templates to see how sub-sections of chapters are included. If you want to see a very complex example, take a look at the Pro Git 2 Book Source on GitHub.

- When designing a book cover image you might want to try Canva, which is web-based graphic design software. The free-tier of Canva will allow you to create a beautiful looking book cover with minimal effort. Start by creating a custom sized image with a width of 1050 px and a height of 1600 px (16:9 resolution). Once you have started a new blank canvas of this size (which is typical for Kindle cover images), search Canva for "Book Cover." When you select a cover template, it will be resized to fit your 1050 px x 1600 px image. (Note that most of Canva's Book Cover templates are 1410 px x 2250 px, which is the size of a typical paper-back book.) When you are done designing your image download it in PNG format. You can rename it to `cover.png` and use it instead of the default image in the `book` sub-directory of your Booksquisher Template.
- You can learn more about Working with Git, GitHub, and Book Templates in the Helpful Tips chapter. Specifically, that section shows you how to remove Git tracking from a template and create your own repository on your local computer and on GitHub.

## 4.3. Edit the Booksquisher Template

Now that we have made a plan for our book and have some basic structural ideas in mind, let's modify the `book-template-1-simple` Booksquisher Template to reflect our decisions.

To change the name of the book, the author, and to define which chapter files to include:

1. Rename the primary "spine" book file from `my-book-name.asc` to the name you want for your primary book file (for example, `my-fabulous-novel.asc`).
2. Open your new book file (`my-fabulous-novel.asc`) with the text editor of your choice, and change the name at the top of the file to the proper name of your book. In this case "My Book Name: A Booksquisher Template" becomes "My Fabulous Novel."
3. Adjust the number of equal signs below the name to match the exact length of the new title.
4. Change the author name from "A. Nonymous" to your name.

5. Change the include directive for the first chapter from `include::ch01-name.asc[]` to the file name you want to use for your first chapter (for example, `include::ch01-dirty-harmonica.asc[]`). Repeat this for each chapter that you have already decided.
6. Notice that the book cover image is set up with the line `:front-cover-image: image:book/cover.png[width=1050,height=1600]`. You can find the `cover.png` image in the `book` sub-directory. If you use a book cover image with other dimensions you should put the appropriate dimensions here. Note that 1050 px x 1600 px (16:9 resolution) is a typical Amazon Kindle cover size.

Notice all the other settings in your primary book file (`my-fabulous-novel.asc` or whatever you have named the main file for your book). There is a **lot** you can do here, which you can learn about in the AsciiDoc Writers Guide.

Now we will edit the `Rakefile`, which defines the build process. We need to change the name of the book to make it consistent with your new primary book file (for example, "my-book-name" gets changed to "my-fabulous-novel"). The line in the `Rakefile` will look like this:

```
book_name = 'my-fabulous-novel'
```

Next, we need a text file for each chapter you defined. The name of the chapter file needs to match the include directive that you defined in the primary book file. For example, you can rename `ch01-name.asc` to `ch01-dirty-harmonica.asc`. Since this Booksquisher template only has a few chapters, you will need to create new files, or copy existing chapter files and rename them as needed.

You can model the content of each chapter file after the template. When in doubt about syntax look at the AsciiDoc Writers Guide and then, if you want to go deeper, the Asciidoctor User Manual.

# Chapter 5. Step-by-Step Instructions for Building the Book Pro Git 2

The instructions below assume you have already completed the chapter Step-by-Step Instructions for Building the Booksquisher Example Book Templates, and have a high-level understanding of the process. All the steps for building the book *Pro Git 2* are given here, but with a bit less explanation.

## 5.1. Getting the progit2 Files

Use Git to clone the Pro Git 2 Book Source on GitHub to your computer. Since we are interested in the book as a *template* we use `--depth 1` to avoid downloading the editing history for the project. We also give the project a new name `new-book-project` so that it can serve as a template for our own book project.

```
$ git clone --depth 1 --origin source https://github.com/progit/progit2.git new-book-
project
```

If you have not done so already, run the `docker pull` command below to download the Booksquisher Docker image on Docker Hub. If you have already installed the Booksquisher Docker image, this command will check to see that it is the latest version, and will download the latest if a newer version is available.

```
$ docker pull marmor/booksquisher
```

## 5.2. Running and Checking the Container

Run the docker image to create a container on your local machine. Also, map the book source directory on your local machine to the `/documents` directory inside the container.

```
$ docker run -it -v <path to progit2 source>:/documents/ marmor/booksquisher
```

Specific example for Docker for Windows (running in Windows PowerShell):

```
PS C:\Users\marmor> docker run -it -v C:\Users\marmor\Desktop\new-book-
project:/documents/ marmor/booksquisher
```

Specific example for Docker on Linux:

```
$ docker run -it -v /home/marmor/new-book-project/:/documents/ marmor/booksquisher
```

Booksquisher runs Linux *inside the container* and that is where we will run the next commands. All

the commands below are to be run inside the Docker container Linux terminal, not on your local machine. Regardless of what operating system you have on your own computer, your terminal should now display the container's Linux prompt (`[booksquisher]#`).

Check that we are in `/documents` and the `progit2` source is visible when you list the directory with `ls`.

```
[booksquisher]# pwd

/documents
```

When you run `ls` in `/documents` you should see a listing of the `progit2` source that should look similar to this:

```
[booksquisher]# ls

A-git-in-other-environments.asc
B-embedding-git-in-your-applications.asc
C-git-commands.asc
CONTRIBUTING.md
Gemfile
LICENSE.asc
Pro.ico
README.asc
Rakefile
TRANSLATING.md
TRANSLATION_NOTES.asc
atlas.json
book
callouts
ch01-getting-started.asc
ch02-git-basics-chapter.asc
ch03-git-branching.asc
ch04-git-on-the-server.asc
ch05-distributed-git.asc
ch06-github.asc
ch07-git-tools.asc
ch08-customizing-git.asc
ch09-git-and-other-systems.asc
ch10-git-internals.asc
diagram-source
images
index.asc
progit.asc
script
status.json
theme
```

## 5.3. Building the Book

Now we are ready to run the build commands for the `progit2` source. A quick review of how this works: We are going to run scripts which are part of the `progit2` source that is on disk on your local computer, but is also mapped into the Docker container as `/documents`. The scripts are going to be run from Linux within the Booksquisher Docker container, which has all the software needed to act on the scripts. The output of the scripts will be formatted book files, which will be placed into the root level of the `/documents` directory in the container, which is the same as root-level of the `progit2` Git repository on your local computer. Said in a much simpler way, the tools in the container are going to put finished book files on your local computer.

*The Next Command Could Take a Few Minutes to Run!*

Depending on your network connection speed, the first time you run `bundle install` may *take as long as 5-minutes* since the build script for `progit2` fetches the current **development** version of `asciidoctor-epub3` from GitHub as well as Ruby Gem metadata from `rubygems.org`. **You do not get a lot of visual feedback as this is running, so don't panic when it seems stalled—just wait a bit**. The `bundle install` process also checks for updated versions of many of the Ruby Gems needed to build the book.

The Booksquisher Docker container contains all the prerequisites for building the *Pro Git 2* book into all the output formats. (Or at least that was true at the time I created the Docker image!) In any event, when you run `bundle install` the script checks for any missing or out of date Ruby Gems, downloads them and installs them. So if there are new requirements, the script should take care of that for you. This process should work without failing because the Booksquisher Docker image already has the required build software for the Gems to compile natively. Luckily, we only need to run `bundle install` one time to get set up for building the `progit2` source.

Be sure you are in the `/documents` directory in the Docker container, then run `bundle install`:

```
[booksquisher]# bundle install
```

Note that you should expect to see a warning message that says *'Don't run Bundler as root*'. The Booksquisher container only has a single user, which is root. You can safely ignore this warning message.

You should expect to see output that looks similar to this:

```
[booksquisher]# bundle install

Don't run Bundler as root. Bundler can ask for sudo if it is needed, and installing
your bundle as root will break this application for all non-root users on this
machine.
Fetching https://github.com/asciidoctor/asciidoctor-epub3
Fetching gem metadata from https://rubygems.org/..........
Resolving dependencies...
Using rake 12.3.2
Using Ascii85 1.0.3
Using public_suffix 3.0.3
Using addressable 2.6.0
Using afm 0.2.2
Using asciidoctor 1.5.6.1
Using mini_portile2 2.1.0
Using nokogiri 1.6.8.1
Using rubyzip 1.2.2
Using gepub 0.6.9.2
Using thread_safe 0.3.6
Using asciidoctor-epub3 1.5.0.alpha.9.dev from
https://github.com/asciidoctor/asciidoctor-epub3 (at master@d42f444)
Using pdf-core 0.7.0
Using ttfunk 1.5.1
Using prawn 2.2.2
Using prawn-icon 1.3.0
Using css_parser 1.6.0
Using prawn-svg 0.27.1
Using prawn-table 0.2.2
Using hashery 2.1.2
Using ruby-rc4 0.1.5
Using pdf-reader 2.2.0
Using prawn-templates 0.1.1
Using safe_yaml 1.0.4
Using polyglot 0.3.5
Using treetop 1.5.3
Using asciidoctor-pdf 1.5.0.alpha.16
Using awesome_print 1.8.0
Using bundler 1.17.3
Using coderay 1.1.2
Using epubcheck 3.0.1
Using json 2.1.0
Using kindlegen 3.0.3
Using multi_json 1.13.1
Using pygments.rb 1.2.1
Bundle complete! 11 Gemfile dependencies, 35 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

Once `bundle install` is run once successfully you should not need to run it again for this book project.

Assuming all went well with bundle install, you can build all versions of the book (PDF, EPUB3, MOBI, and HTML5) with this single command:

```
[booksquisher]# bundle exec rake book:build
```

*Pro Git, 2nd Edition* is a 500+ page book. Be aware that the creation of the PDF file may take a few minutes to run. The other output file formats are created quite quickly.

You should expect to see something similar to this when you run `bundle exec rake book:build`:

```
[booksquisher]# bundle exec rake book:build

Generating contributors list
Converting to HTML...
 -- HTML output at progit.html
Converting to EPub...
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
 -- Epub output at progit.epub
Converting to Mobi (kf8)...
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
/usr/lib/ruby/gems/2.4.0/gems/nokogiri-1.6.8.1/lib/nokogiri/xml/document.rb:44:
warning: constant ::Fixnum is deprecated
 -- Mobi output at progit.mobi
Converting to PDF... (this one takes a while)
 -- PDF output at progit.pdf
[booksquisher]#
```

You can safely ignore the `::Fixnum is deprecated` warnings. This is a known issue, and later releases of the tools within Booksquisher should eliminate this warning message. The output files are not impacted.

Look for the newly created PDF, EPUB3, MOBI (Kindle Format 8), and HTML5 book files on your local computer at the top level of the `progit2` directory created by Git (we called this directory `new-book-project` when we cloned the project with Git). The output files can also be seen in the `/documents` folder of the Booksquisher container, which is the same directory mapped to your local computer.

## 5.4. Build System Differences: Pro Git 2 vs. Booksquisher Templates

There are a few key differences between the build system in the Booksquisher Example Book

Templates on GitHub and the Pro Git 2 Book Source on GitHub that you should be aware of:

1. In the Booksquisher Example Book Templates the generated books are placed into the `generated-output` directory. In *Pro Git 2* the generated books are placed into the *top level directory* of the project.
2. The Booksquisher templates have a *clean* command that you can use as part of the build process. The *Pro Git 2* build process does not have this command. Here is how the *clean* command works in the Booksquisher Example Book Templates on GitHub:

   To delete **all** of the generated book files and intermediate build artifacts you can run:

   ```
   [booksquisher]# bundle exec rake book:clean
   ```

   This should remove everything from the `generated-output` directory except for the hidden `.gitignore` file.

   You can combine the process of removing old book versions *and* generating new ones into a single command like this:

   ```
   [booksquisher]# bundle exec rake book:clean book:build
   ```

# Chapter 6. Helpful Tips

## 6.1. Working with Git, GitHub, and Book Templates

If you plan to modify a Booksquisher Example Book Template and create your book based on that template, you don't really care about the version history of the template files. You care about *your* modifications and *your* new book, not the history of the template itself. This section will explain how to *remove* Git tracking from a template project and how to set up *your own* Git repository on your computer and GitHub. Essentially, we are breaking the connection with the original repository where you downloaded the template and creating a new repository for your new book.

When you download the Booksquisher Example Book Templates on GitHub using `git clone ...` there will be a hidden `.git` directory in the root of the project. This `.git` directory is where Git keeps all the project version tracking information. If you delete this hidden `.git` directory all tracking information will be lost. That sounds like a bad thing, but in this case, it is just what we want.

Here is an example using the Booksquisher Example Book Templates on GitHub.

```
$ git clone --depth 1 --origin source https://github.com/mmarmor/booksquisher-example-
book-templates.git my-fabulous-novel
$ cd my-fabulous-novel
$ rm -rf .git
```

This is slightly confusing because the Booksquisher Example Book Templates on GitHub project repository contains multiple example books, each in its own directory. Once you remove the hidden `.git` directory you can take any one of these example books and set it up as the basis for your new book.

Let's use the source code from *Pro Git, 2nd Edition* as an example template to demonstrate this process step-by-step. Here are the commands to download the project and remove the hidden `.git` directory:

```
$ git clone --depth 1 --origin source https://github.com/progit/progit2.git new-book-
project
$ cd new-book-project
$ rm -rf .git
```

This might look different in your operating system (for example `rm -rf` is a Linux-platform command). You could also remove the hidden `.git` directory from within the Booksquisher container with `rm -rf .git` once you have run Booksquisher and mounted the directory.

Note that we are using `--depth 1` in our Git clone command. If we are going to delete the version history, there is no point in downloading anything beyond the current version. For a large book with a complex editing history in Git, like *Pro Git, 2nd Edition*, using `--depth 1` will dramatically

reduce the size of the downloaded project. Also, note that `new-book-project` can be any name you want to give your new book project that will be based on the example template.

### Creating your Own Git Repository

Let's say that you want to create your *own* Git repository for the book template (now the source for your book and no longer just a model) and push it to your own GitHub account. Once you have followed the instructions above to remove the hidden `.git` directory you can follow these steps:

Go to GitHub in your web browser, make sure you are logged in with your account, and create a new repository. In our example, we will use the name `new-book-project`. GitHub will give you the URL for the project when you finish creating it. In my case, it might look like `https://github.com/mmarmor/new-book-project.git`.

Now go back to your computer, and using the command line change into the root of your project directory. The commands below will create a new repository on your local machine, add all the template files, commit the template files, set the URL of your GitHub project, and then push the data to GitHub.

```
$ git init
$ git add *
$ git commit -m "first commit"
$ git remote add origin https://github.com/mmarmor/new-book-project.git
$ git push -u origin master
```

Once you do this, you can check your project on GitHub in your web browser, and you should see your files there. Your book project is now tracked in Git, and you can use a typical Git and GitHub workflow as you work with your book source code. If you need help learning Git, may I suggest the book Pro Git, Second Edition ☺. Now that you have the source code for the book, you can follow the instructions in the chapter Step-by-Step Instructions for Building the Book Pro Git 2 to build it.

To change the book version number you need to tag your repository like this:

```
$ git tag -a v0.1-alpha -m "halloween alpha release"
```

## 6.2. Text Editors

Booksquisher uses a tool called Asciidoctor to convert AsciiDoc syntax into finished books in various formats. AsciiDoc is just plain text with added syntax to help provide layout hints for Asciidoctor to interpret as it creates your final books. Since the files we need to create and edit are just plain text, you can use *any* text editor that you want. That said, some text editors provide more support, such as syntax highlighting, which is helpful when working with AsciiDoc files.

Talking about text editors with advanced computer users is a lot like talking about politics. People get passionate—and vocal—about their choices, likes, and dislikes. I'll go further—for many technical people their choice of text editor defines them as a member of a community and activates a "tribal identity" that goes to the heart of who they are and how they see themselves in the world.

To someone outside of the software development community, this may seem farfetched and hard to believe, but I firmly believe that this is true. I mention this because no matter what I say about this topic there will be those who disagree—often with a vituperative outburst.

I've been a GNU Emacs user for a long time. There is a good statistical chance that I've been using GNU Emacs longer than you have been alive. I've personally taught several dozen people how to use it, and I've published "how-to" articles about Emacs that have been read by thousands of people (see my GNU Emacs 101 piece on Medium). Given that much "skin in the game," you may be surprised that I am *not* recommending that you run out and learn Emacs to use Booksquisher and AsciiDoc!

As I mentioned back in the chapter Modifying the Booksquisher Example Book Templates in the section A Few Tips Before we Start Editing I am recommending that you use a text editor that you are **already comfortable using**.

Don't try to learn a new text editor *and* AsciiDoc syntax at the same time. It will slow you down and make you feel clumsy and overwhelmed.

As I said earlier, if you don't already have a favorite text editor that you are comfortable using, I suggest you use Visual Studio Code, which is free, easy to learn, runs on all major platforms, and has excellent support—via extensions—for AsciiDoc, Git, and code spell checking.

To be clear, Visual Studio Code is not just a "low end" option for beginners. When busy junior developers ask me what text editor they should learn in 2019, I'm now pointing them to Visual Studio Code *instead of Emacs* since I find they become more productive more quickly when compared to the powerful old-school editors. If they are looking for a long term learning investment and *do not need to be productive immediately*, I suggest diving into Emacs an often point them to GNU Emacs 101. I'm still a *much* faster editor in GNU Emacs than I am in Visual Studio Code. And some things in Code drive me bonkers (likely because of my Emacs muscle memory, and because I have become accustomed to small automation scripts I have created for Emacs over the years). All that said—if you are just getting started in the software development world (or AsciiDoc world!) Visual Studio Code is a strong choice for a lot of languages and use cases.

For a second opinion (that differs from mine), you might read the recommendations in the AsciiDoc Writers Guide.

## 6.3. Learning Docker

Docker can be complicated. Luckily, Booksquisher uses only the most basic Docker commands and concepts. If you have never used Docker before, you can learn enough to use Booksquisher effectively in *less than half-an-hour*. Really.

In my opinion, Docker has developed a reputation for being complex because it has lots of advanced features intended to manage extraordinarily-complicated software applications. For our purposes, you can completely ignore subjects like Kubernetes, Docker Swarm, Docker Compose and a bunch of other stuff that exist primarily for large organizations to use Docker to host large-scale container-based applications.

As exciting as they may be, we don't care about "containerized microservice architectures" or

"orchestration." In fact, we don't care about any of the "Enterprise" features of Docker. For our needs, we need to be able to run a single Docker container on our personal computer and interact with the container's command line interface. That's it. No fancy networking. No need to run or manage lots of containers at the same time. We need to be able to run and maintain one container on our own computer. You don't even need to learn to create Docker images since the Booksquisher image already exists in Docker Hub and can be downloaded with a single `docker pull` command. Once you learn the basics and get Docker installed it simplifies a bunch of software dependency issues that used to be complicated—and that is why Booksquisher uses it.

That said, we do need to understand a few concepts, like the difference between an image and a container. We also need a handful of commands to be able to download and run Docker images, and to stop, start, attach and remove Docker containers.

## Essential Docker Commands

Here is a list of the essential Docker commands that I think you will need to use Booksquisher effectively:

```
docker pull ...
docker images
docker run ...
docker container ls --all
docker container rm ...
docker container stop ...
docker container start ...
docker container attach ...
```

## Resources for Learning Docker

I wish I could say that the best place to learn about Docker is from the Docker Documentation website. It is true that the Docker documentation is excellent and there is a tutorial that is great if you are a software developer. But if you are not a developer, the documentation and tutorial are likely to overwhelm you with details that are not important for learning the basics you need to run and manage one Docker container like Booksquisher. The folks at Docker assume your job is to create and manage "swarms" of interrelated Docker containers in a complex environment—and that is not our problem at all.

There are links on the Docker Documentation website that will help you get and install Docker. The version of Docker that we care about is the free *Docker Community Edition*. Links in the Docker Documentation bring you to the Get started with Docker tutorial, which is excellent if you are a developer and overwhelming if you are not. 80% of what you need to know about Docker is on the first page of the Get started with Docker tutorial: *Get Started, Part 1: Orientation and setup*. By page two of the tutorial they have you creating containers and deploying a complex web-based application that assumes you are a software developer. This second page *does* show examples of starting, stopping and removing containers, but there is lots of unnecessary complexity for our purpose.

I have one more suggestion which may not be helpful for you at all, but it is worth mentioning. I

live in the United States, and my local public library provides free access to the Lynda.com online course website if you have a library card. I used Lynda.com as a resource when I first got started with Docker and found the courses there to be quite helpful.

You may discover equally good content online. If you do, please send me an email or a GitHub pull request so that we can update this User's Guide with links to specific tutorials and learning resources.

## A Note for Windows Users who *Also* use VMware or VirtualBox

Let me start by saying that Docker Desktop for Windows works great on Windows 10. I'm using it right now, and it does just what I need. That said, there is something important you need to know:

You cannot run VMware or VirtualBox **AND** Docker Desktop for Windows at the same time!

That is a real bummer for people like me who depend on running full virtual machines for some projects, but also want to be able to run Docker on Windows. There is a workaround, and I'm using it successfully, but it is not optimal.

The issue is that Docker Desktop for Windows uses Microsoft Hyper-V (and *ONLY* Microsoft Hyper-V) as a hypervisor to host the underlying Linux system on which Docker depends. Microsoft Hyper-V cannot run at the same time as other hypervisors. As a result, if you install Docker Desktop for Windows, you will find that your VMware or VirtualBox installation will simply stop working.

*Technical Stuff: Only attempt this if you understand what you are doing!*

The way around it (at least as of February 2019 when I write this) is to set your computer up to dual-boot. On my system, I have set up the default boot to enable Hyper-V, which also allows Docker Desktop for Windows to run. I have another boot option called "No Hyper-V" that will enable VMware to function (but not Docker!) Credit for this hack goes to a blog post from Scott Hanselman. Here is his recipe to make this work:

Note the first command gives you a GUID (in the example below this is `{ff-23-113-824e-5c5144ea}`) which you copy and then use with the second command:

```
C:\>bcdedit /copy {current} /d "No Hyper-V"
The entry was successfully copied to {ff-23-113-824e-5c5144ea}.

C:\>bcdedit /set {ff-23-113-824e-5c5144ea} hypervisorlaunchtype off
The operation completed successfully.
```

The commands above set the boot option for Docker (with Hyper-V) to be the default. If you plan to use VMware or VirtualBox more than Docker, you may need to figure out how to reverse the logic.

On Scott's blog post he is using Windows 8 and had limited time to choose which option to boot. On Windows 10 you have a friendly boot menu and a reasonable wait time to make a choice.

I hope the folks at Docker can figure out how to remove this limitation and allow us to pick the

hypervisor of our choice.

## 6.4. Calling The `asciidoctor` Commands Directly

The authors of the progit2 project build system want you to build the *Pro Git 2* book by calling `bundle exec rake book:build`. They want you to use `bundle` to make sure that the exact required versions of toolchain programs, as they have defined them, are used. This helps keep book builds repeatable and cuts down on rendering related issues. This is important for large projects with lots of contributors. The authors want the generated book to look correct and consistent regardless of who builds it. You may want that too and can adapt their build process for your book project. In fact, I strongly recommend that you use the build process that is used in the Booksquisher Example Book Templates on GitHub, which is based on what you will find in the progit2 project.

That said, there are other ways that you can invoke the tools in the Booksquisher Docker container. In a less formal setting, you might want to use the underlying commands that build individual output formats. An alternative to calling the `bundle` command is to call the `asciidoctor` family of commands directly.

The commands below use the `asciidoctor`, `asciidoctor-pdf` and `asciidoctor-epub` command-line programs in the Booksquisher Docker container.

To build a book with HTML5 formatted output, use the command:

```
[booksquisher]# asciidoctor mybook.asc
```

To learn more about the `asciidoctor` command, see the Asciidoctor website.

To build a book with PDF formatted output, use the command:

```
[booksquisher]# asciidoctor-pdf mybook.asc
```

To learn more about the options for `asciidoctor-pdf` see the Asciidoctor PDF website.

To build a book with EPUB3 formatted output, use the command:

```
[booksquisher]# asciidoctor-epub3 mybook.asc
```

To build a book with Kindle (KF8/MOBI/AZW3) formatted output, use the command:

```
[booksquisher]# asciidoctor-epub3 -a ebook-format=kf8 mybook.asc
```

To learn more about the options for `asciidoctor-epub3` for generating both EPUB3 and Kindle files see the Asciidoctor EPUB3 website.

# Chapter 7. Wrapping Up

## 7.1. Useful Links

**Booksquisher Documentation Links**

- The online version of this User's Guide can be found at https://www.booksquisher.com
- Download the Booksquisher User's Guide in EPUB3 format
- Download the Booksquisher User's Guide in Kindle AZW3 format
- Download the Booksquisher User's Guide in PDF format

**Booksquisher Links**

- Booksquisher Docker image on Docker Hub
- Booksquisher Example Book Templates on GitHub

**Pro Git, 2nd Edition Book Links**

- Pro Git, Second Edition
- Pro Git 2 Book Source on GitHub

**Asciidoctor Links**

- Asciidoctor
- AsciiDoc Writers Guide
- Asciidoctor User Manual
- Asciidoctor EPUB3
- Asciidoctor PDF
- asciidoctor/docker-asciidoctor Docker image on which Booksquisher is based

**Docker Links**

- Get started with Docker
- Docker Documentation
- Docker for Windows
- Docker for OS X
- Docker for Linux

**Other Links Mentioned in the User's Guide**

- Git
- Canva
- Lynda.com
- Visual Studio Code
- GNU Emacs

- GNU Emacs 101

## 7.2. Acknowledgments and Credits

**Acknowledgments**

- Scott Chacon, Ben Straub the authors of Pro Git, Second Edition, and all the contributors to the progit2 project.
- Guillaume Scheibel and Damien Duportal who are the maintainers of asciidoctor/docker-asciidoctor on which the Booksquisher Docker image is based
- Stuart Rackham, the creator of AsciiDoc and the many individuals in the AsciiDoc community
- Nick Hengeveld, Ryan Waldron, Sarah White, Dan Allen and the many individuals in the Asciidoctor community
- The authors and community supporting DocBook
- Leslie Lamport and the LaTeX project community, on which a great deal of this technology is directly and indirectly based

**Credits**

- The design sub-elements used to create the Booksquisher Logo on GitHub are Books Stack From Top View SVG Vector and Construction Clamp SVG Vector, both licensed under Creative Commons BY 4.0
- The favicon design used in the web version was created by Dave Gandy and licensed under Creative Commons BY 3.0
- This document was composed in AsciiDoc and generated with Asciidoctor, Asciidoctor EPUB3, and Asciidoctor PDF from the Booksquisher Docker image on Docker Hub

## 7.3. About the Author

The Booksquisher User's Guide, Docker image, Example Book Templates, and logo are created by Michael S. Marmor.

- Personal Website
- LinkedIn Profile
- GitHub Profile

I work with companies to build offshore software development teams around the world. If you are considering nearshore or offshore development in Asia, Latin America, or Eastern Europe, I can make connections and coach you through the process. (Consulting PDF)

Since 1999 I have spent about half of my time overseas on various software and IT projects. I'm American—based in Charlotte, North Carolina—but I have lived and worked in six Indian cities and the Himalayan country of Bhutan. (As I write this I am currently in Goa, India enjoying the wonderful February weather!)

I am a serious yoga practitioner, an amateur bluegrass mandolin player, a ham radio operator, a learning photographer, and an aspiring writer. My email address is my last name at gmail.com—please feel free to reach out to me.