Texmlbus, a build system to convert documents to XML and other formats

Heinrich Stamerjohanns

Abstract

Here I present an automatic open source build system that supports the conversion process of a collection of documents written in LaTeX or other TeX formats. With Texmlbus [4], the TeX to XML BUild System, documents can not only be converted to PDF, but also to other output formats — such as markup languages like HTML. In particular, conversion to XML, HTML and MathML is supported via LaTeXML. Texmlbus can schedule jobs among several workers (possibly on different hosts), extract and analyze the outcome of the conversion process of each document and store results in its own database. Result documents as well as statistics about the results of the build process can be easily retrieved using a web browser.

1 Introduction

LaTeX is the preferred document source for many scientists and authors who publish results that include mathematical formulas. In addition to print-oriented formats (PDF), there is also a need to export such documents into other formats, such as XML-based documents that more easily support additional services like search and navigation, as well as integrating the document or parts of it in web pages.

Automatic conversion of such collections of TeX documents can be a time-intensive task. The conversion result needs to be checked for errors and the visual appearance needs to be verified for each document. A build system that collects the result and status of conversions offers easily clickable links to result documents and log files, and collects and

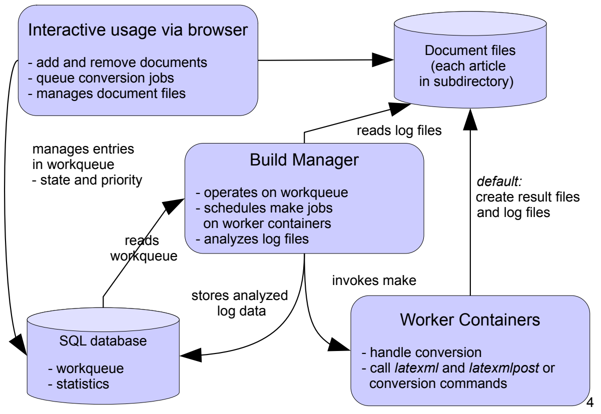


Figure 1: Components of Texmlbus

show statistics can provide helpful support in order to manage such conversion tasks. Texmlbus is such a system; its main components are shown in Figure 1.

Using LaTeXML [3] as the default conversion processor, this system is based on the arXMLiv build system [5, 6], which I had written as a member of the arXMLiv group at Jacobs University Bremen. It was used to convert large collections of scientific publications of the Cornell e-Print archive arχiv to XML and Content-MathML. That build system had not only been useful for converting documents, but also by generating statistics for conversion errors and warnings; thus our group was able to generate feedback to the LaTeXML developers on where to focus improvements.

While the conversion of arχiv documents is now being done with CorTex [2] the original arXMLiv build system has now become Texmlbus. Texmlbus still uses LaTeXML as the main conversion processor, but can easily be extended to support other conversion processors and formats as well. It still can be used for conversions of thousands of documents, but now development is focused in particular on

- easy installation on any platform
- simple (interactive) use of system
- possibility for targets other than XHTML
- using the same targets with different systems

To make the system more easily installable and runnable on any platform, docker containers have been chosen. To simplify usage, interactive components have been added to the system. Documents can now be added, queued and removed via a web browser. It is also now possible to organize document collections in self-defined sets. This makes it easier to navigate through and work with many documents.

The new build system also supports targets other than XHTML. With only a few lines of code, it is possible to add additional targets such as specific XML-dialects or other validators to the system.

Also, so-called *stages* have been added to the system. A stage is defined as a combination of a target (e.g., XHTML) and a specific docker image that implements the conversion. It is therefore possible to have several stages that generate the same target, but with different versions (e.g., TeX Live 2020 and TeX Live 2021).

2 System

In order to ease deployment on any platform, the system runs inside several docker containers, which share a common file system containing the documents. There is a docker container for the build manager (the web server backend which hosts the application), a container for a relational database,

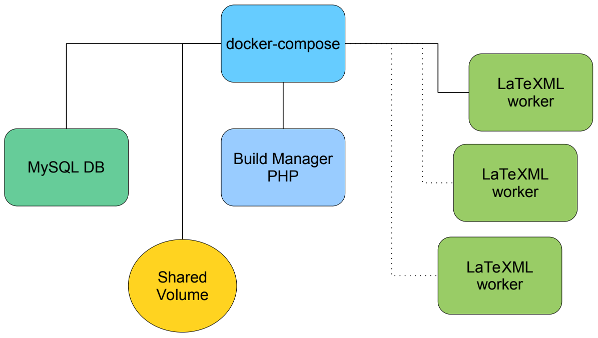
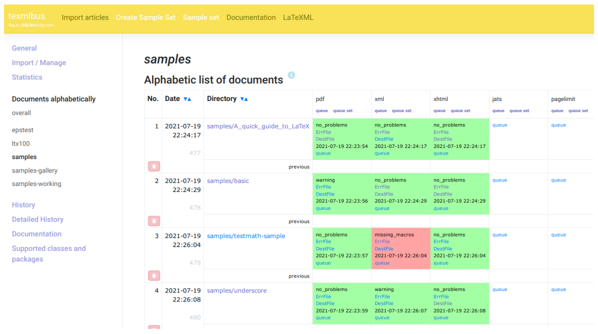


Figure 2: Docker setup using `docker-compose`



samples

Alphabetic list of documents

No.	Date ▼▲	Directory ▼▲	pdf queue queue set	xml queue queue set	xhtml queue queue set	jats queue queue set	pagelimit queue queue set
1	2021-07-19 22:24:17 477	samples/A_quick_guide_to_LaTeX	no_problems ErrFile DestFile 2021-07-19 22:23:54 queue	no_problems ErrFile DestFile 2021-07-19 22:24:17 queue	no_problems ErrFile DestFile 2021-07-19 22:24:17 queue	queue	queue
		previous					
2	2021-07-19 22:24:29 478	samples/basic	warning ErrFile DestFile 2021-07-19 22:23:56 queue	no_problems ErrFile DestFile 2021-07-19 22:24:29 queue	no_problems ErrFile DestFile 2021-07-19 22:24:29 queue	queue	queue
		previous					
3	2021-07-19 22:26:04 479	samples/testmath-sample	no_problems ErrFile DestFile 2021-07-19 22:23:57 queue	missing_macros ErrFile DestFile 2021-07-19 22:26:04 queue	no_problems ErrFile DestFile 2021-07-19 22:26:04 queue	queue	queue
		previous					
4	2021-07-19 22:26:08 480	samples/underscore	no_problems ErrFile DestFile 2021-07-19 22:23:59 queue	warning ErrFile DestFile 2021-07-19 22:26:07 queue	no_problems ErrFile DestFile 2021-07-19 22:26:08 queue	queue	queue

Figure 3: Texmlbus document overview

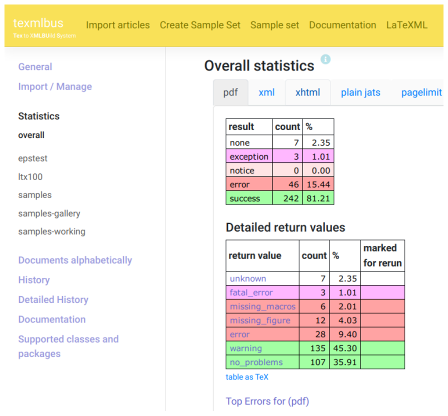
which stores information about the documents and result statistics, and one or more worker containers that actually run the conversion. The workers also run a minimal web server, so jobs can be invoked via API requests over HTTP. An overview of the docker setup is shown in Figure 2.

Documents can be either copied to the file system and then scanned for import or uploaded as zip files via a web browser. It is also possible to directly import and update files from Overleaf via a git-bridge.

Then, documents (or whole sets) can be added to a work queue interactively. The build manager operates on the work queue and schedules jobs on distributed worker containers. It keeps an internal list of available hosts (each container is a host) and distributes conversion jobs among these hosts. The conversion jobs on the worker containers are invoked via an API request over HTTP. A worker container typically runs a make process to invoke `latexml` and `latexmlpost` for the conversion to XHTML.

After the conversion has finished on the worker container, the build manager is notified and then collects data by analyzing log files and stores the result in the database. The results are then shown in an overview.

The build manager can, for example, collect the names of missing macros that are not yet supported



Overall statistics

result	count	%
none	7	2.35
exception	3	1.01
notice	0	0.00
error	46	15.44
success	242	81.21

Detailed return values

return value	count	%	marked for rerun
unknown	7	2.35	
fatal_error	3	1.01	
missing_macros	6	2.01	
missing_figure	12	4.03	
error	28	9.40	
warning	135	45.30	
no_problems	107	35.91	

Figure 4: Texmlbus statistics page

by LaTeXML or give an overview of files for which the conversion failed.

The document overview (see Figure 3) lists documents in collections (sets) and shows the conversion status for each stage. On that page the TeX source, error logs for each stage, and the result documents are reachable with one click. The backend can also create cumulative statistics of the result log data and creates lists such as top fatal errors or most missing macros on-the-fly.

To convert TeX documents to XHTML, LaTeXML needs so-called binding-files (with the suffix `.ltxml`). By using the list of missing macros one can easily determine which macros need more support in order to improve the conversion results.

It is also possible to easily extend the system and create additional stages that convert to other formats or use another TeX environment. As a stage is defined as a combination of target and container, one can easily create a container that uses a different TeX Live distribution. Therefore one can easily compare the results of different distributions and gather statistics about conversion results (Figure 4).

Furthermore, the system is not limited to conversions to other formats. Last year, W. Duivesteijn wrote about "How to cheat the page limit" [1] for conference papers and identified typical TeX commands that are used to circumvent page limits. With a few lines of code I have created a pagelimit stage that tries to find these typical commands in a document and categorizes the document accordingly.

3 Conclusion

The Texmlbus build system allows for easily converting LaTeX documents to XHTML and MathML or other destination formats, and gathers statistics about the conversion results. Since binding files' coverage of LaTeXML is still not complete, this system helps to identify missing style file support for document collections and therefore gives feedback on where to focus development efforts.

Other converters can also be used for conversions. Docker containers that provide such converters need to be extended to provide some additional HTTP support so they can be invoked via an HTTP API. It should be simpler to directly plug in such containers, so additional stages can be added even more easily.

4 Acknowledgements

I am grateful to Overleaf for its support of the project.

Please email me with any problems or questions.

References

[1] W. Duivesteijn, S. Hess, X. Du. How to cheat the page limit. *WIREs Data Mining and Knowledge Discovery* 10, Feb. 2020. `10.1002/widm.1361`

[2] D. Ginev. `CorTeX`: A general purpose processing framework for corpora of scientific documents. `https://github.com/dginev/CorTeX`

[3] B. Miller, D. Ginev. `LaTeXML`: A LaTeX to XML converter. `https://dlmf.nist.gov/LaTeXML/`

[4] H. Stamerjohanns. `texmlbus`: A build system to convert documents to XML and other formats. `https://github.com/stamer/texmlbus`

[5] H. Stamerjohanns, M. Kohlhase. Transforming the arχiv to XML. In *9th International Conference, AISC 2008 15th Symposium, Calculemus 2008 7th International Conference, MKM 2008 Birmingham, UK, July 28–August 1, 2008*, S. Autexier, J. Campbell, et al., eds., Intelligent Computer Mathematics, pp. 574–582. Springer Verlag, 2008.

[6] H. Stamerjohanns, M. Kohlhase, et al. Transforming large collections of scientific publications to XML. *Mathematics in Computer Science* 3(3):299–307, 2010. `https://kwarc.info/kohlhase/papers/mcs10.pdf`

◇ Heinrich Stamerjohanns
Oldenburg, Germany
`heinrich.stamerjohanns (at) gmail.com`
`https://github.com/stamer/texmlbus`