`docshots`: LaTeX Package that Renders TeX Samples Next to Their PDF Snapshots*

Yegor Bugayenko
yegor256@gmail.com

2026-07-15, 0.5.0

The TeX processor must be invoked with the `--shell-escape` option, and the following must be installed on the host system: `pdflatex`, Perl, Ghostscript, and pdfcrop. The `--shell-escape` option is required because the package launches these external programs from within the TeX run in order to compile and crop each snippet. The package operates on Windows as well as on Unix, provided these tools are on the `PATH`.

1 Introduction

When demonstrating to the readers of a document how certain TeX commands are to be employed, the most cogent approach is to display precisely how the entire document will be rendered as a PDF, by means of a subprocess (such as `pdflatex`) that performs the rendering. To the author's best knowledge, no existing package addresses this need; hence the present, modest package was devised.

For instance, the following code:

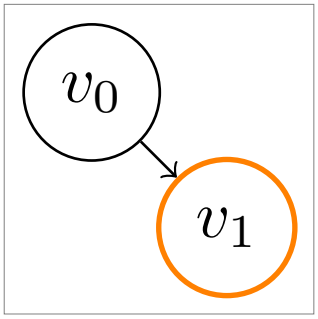
```
\begin{docshot}
\documentclass{article}
\usepackage{xcolor}
\pagestyle{empty}
\begin{document}
  Hello, {\color{orange}\LaTeX}!
\end{document}
\end{docshot}
```

is rendered thus:


```
\documentclass{article}
\usepackage{xcolor}
\pagestyle{empty}
\begin{document}
  Hello, {\color{orange}\LaTeX}!
\end{document}
```

*The sources are in GitHub at yegor256/docshots

A more elaborate example follows:



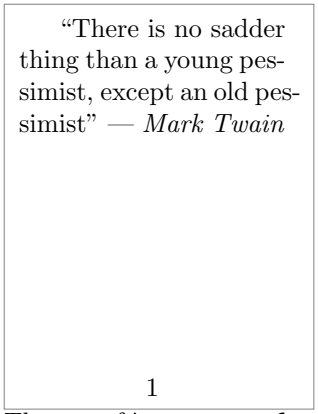

```
\documentclass{article}
\usepackage{tikz}
\pagestyle{empty}
\begin{document}
\begin{tikzpicture}
\node [circle,draw] (v0) {$v_0$};
\node [circle,draw=orange,thick,
  below right of=v0] (v1) {$v_1$};
\draw [->] (v0) -- (v1);
\end{tikzpicture}
\end{document}
```

The body of every `docshot` environment must be a complete, self-contained LaTeX document: it begins with `\documentclass`, loads whatever packages it requires, and wraps its content in `\begin{document}` and `\end{document}`. The body is written verbatim to a separate `.tex` file and compiled on its own, exactly as the two examples above demonstrate.

The figure on the left is produced by a subprocess that executes `pdflatex` on the `.tex` content extracted from the source file. Once the TeX sources have been successfully processed, pdfcrop is invoked to trim the document.

`pdflatex` is run with the `-interaction=errorstopmode` and `-halt-on-error` options. Consequently, TeX processing halts at the first error encountered. The TeX log should be consulted to ascertain the cause of any failure; the `log` option, described below, copies the full output of `pdflatex` into that log.

Whenever text rather than a single figure is to be rendered, the geometry package is recommended for adjusting the page dimensions:




```
\documentclass{article}
\usepackage[paperwidth=2in,
  paperheight=2.3in]{geometry}
\begin{document}
``There is no sadder thing than
a young pessimist, except an old
pessimist'' --- \emph{Mark Twain}
\end{document}
```

The use of `\pagestyle{empty}`, as in the preceding docshots, is likewise possible.

2 Package Options

All options listed below are supplied to the package in the usual way, within the optional argument of `\usepackage[...]{docshots}`.

`pdflatex` The default command-line tool for converting `.tex` into `.pdf` is `pdflatex`. This

may, however, be altered by means of the `pdflatex` package option, for example:

```
\documentclass{article}
\usepackage[pdflatex=/usr/local/bin/pdflatex]{docshots}
\begin{document}
\begin{docshot}
Hello, world!
\end{docshot}
\end{document}
```

`gs` The default location of Ghostscript is simply `gs` on Unix and `gswin64c` on Windows. This may be altered by means of the `gs` package option, for example:

```
\usepackage[gs=/usr/bin/ghostscript]{docshots}
```

`pdfcrop` The default location of `pdfcrop` is simply `pdfcrop`. This may be altered by means of the `pdfcrop` package option, for example:

```
\usepackage[pdfcrop=/bin/pdfcrop]{docshots}
```

`margin` When the rendered PDF is cropped, a margin is preserved around the content. The default value may be modified by the `margin` package option:

```
\usepackage[margin=10]{docshots}
```

`nocrop` Cropping may be disabled by means of the `nocrop` option:

```
\usepackage[nocrop]{docshots}
```

`hspace` The horizontal distance between the image and its verbatim TEX source may be configured by means of the `hspace` package option:

```
\usepackage[hspace=1em]{docshots}
```

`left` `right` The default width of the image may be altered by the `left` option, while the width of the verbatim TEX source may be adjusted by the `right` option:

```
\usepackage[left=2in,right=.5\linewidth]{docshots}
```

`dtx` Whenever this package is employed within `.dtx` documentation, the `dtx` package option must be added. Under this option, all leading comment symbols (the `%` characters) are removed from the start of each line:

```
\usepackage[dtx]{docshots}
```

`tmpdir` The default location for temporary files is `_docshots`. This may be altered by means of the `tmpdir` option:

```
\usepackage[tmpdir=/tmp/foo]{docshots}
```

`runs` By default, `pdflatex` is invoked but once per docshot. This number may be modified by means of the `runs` package option, which proves serviceable when BibTEX processing is required, for example:

```
\usepackage[runs=3]{docshots}
```

`small` `tiny` The latitude available for formatting verbatim texts is limited; nevertheless, the font may be reduced somewhat by means of the `small` package option, or made considerably smaller still by the `tiny` option:

```
\usepackage[small]{docshots}
```

`log` By means of the `log` option, the package writes all available logs into the main TEX log. By default, this is not done, and the output of `pdflatex` compilation, for instance, remains hidden. The option is employed thus:

```
\usepackage[log]{docshots}
```

`inputminted` By default, the verbatim text is rendered by means of the `\VerbatimInput` command. This may be substituted with `\inputminted` from the minted package, for example:

```
\usepackage{minted}
\setminted[java]{frame=lines,framesep=2mm}
\usepackage[inputminted=java]{docshots}
```

`lstinputlisting` By default, the verbatim text is rendered by means of the `\VerbatimInput` command. This may be substituted with `\lstinputlisting` from the listings package, for example:

```
\usepackage{listings}
\lstset{basicstyle=\small}
\usepackage[lstinputlisting]{docshots}
```

3 Fine-tuning Options

`\docshotOptions` By default, the verbatim text is rendered by means of the `\VerbatimInput` command with no options. Additional options may be supplied via the `\docshotOptions` command:

```
\docshotOptions{firstline=4}
\begin{docshot}
...
\end{docshot}
```

The options are cleared upon the first rendering of a docshot.

4 Prerequisites

`\docshotPrerequisite` Whenever certain files are required to be present alongside the `.tex` snippet during rendering by `pdflatex`, the `\docshotPrerequisite` command may be employed, accepting a single mandatory argument. The argument denotes the name of a file to be copied from the current directory to the temporary directory in which all snippets are rendered. The command may appear either in the body of the document or in the preamble—the placement is immaterial, provided that it precedes the docshot which requires the prerequisite. For example:

```
\documentclass{article}
\usepackage{docshots}
\docshotPrerequisite{duck.jpg}
\begin{document}
\begin{docshot}
  \documentclass{article}
  \usepackage{graphicx}
  \pagestyle{empty}
```

```
  \begin{document}
    A favoured photograph of a duck:
    \includegraphics[width=2in]{duck.jpg}
  \end{document}
\end{docshot}
\end{document}
```

`\docshotAfter` Should some action be required after each `pdflatex` run of a docshot, the `\docshotAfter` command may be placed immediately before the `docshot` environment. For example, when a bibliography file is to remain available to every snippet and biber is to be invoked on each in order to process citations:

```
\documentclass{article}
\usepackage{docshots}
\docshotPrerequisite{main.bib}
\begin{document}
\docshotAfter{biber $2}
\begin{docshot}
  \documentclass{acmart}
  \usepackage[natbib=true]{biblatex}
  \addbibresource{main.bib}
  \pagestyle{empty}
  \begin{document}
    The book of \citet{knuth1984} is essential reading.
    \printbibliography
  \end{document}
\end{docshot}
\end{document}
```

The script supplied as the first argument of `\docshotAfter` receives three arguments at runtime:

`$1` the cycle of `pdflatex` processing (1, 2, ...),

`$2` the hash of the snippet,

`$3` the name of the `.tex` file.

`$3` is, in essence, `$2` with an appended `.tex` suffix. `\docshotAfter` applies solely to the first `docshot` environment that follows it; `\docshotAfter` must therefore be specified before every `docshot` in which such post-processing is desired.

5 Implementation

First, we include iexec in order to execute shell commands, such as `pdflatex` and `pdfcrop`:

```
1 \RequirePackage{iexec}
```

Then, we include fancyvrb in order to render verbatim texts:

```
2 \RequirePackage{fancyvrb}
```

Then, we include xcolor in order to make borders gray:

```
3 \RequirePackage{xcolor}
```

Then, we include graphicx in order to be able to use the `\includegraphics` command:

```
4 \RequirePackage{graphicx}
```

Then, we include tikz in order to be able to position elements:

```
\RequirePackage{tikz}
```

Then, we process package options with the help of pgfopts: We also detect Windows, in order to pick the right default for Ghostscript, the right file separator for shell commands (a backslash, since `cmd.exe` rejects forward slashes in redirection targets), and an expanding variant of the string-replacement command for later use in

```
\RequirePackage{pgfopts}
\RequirePackage{ifluatex}
\RequirePackage{ifxetex}
\RequirePackage{expl3}
\newif\ifdocshots@windows
\ExplSyntaxOn
\sys_if_platform_windows:TF{\docshots@windowstrue}{\docshots@windowsfalse}
\cs_generate_variant:Nn \str_replace_all:Nnn { Nne }
\newcommand\docshots@aftercmd[3]{%
  \str_set:Nx \l_tmpa_str { \docshots@after }%
  \str_replace_all:Nne \l_tmpa_str { $1 } { #1 }%
  \str_replace_all:Nne \l_tmpa_str { $2 } { #2 }%
  \str_replace_all:Nne \l_tmpa_str { $3 } { #3 }%
  \cs_set:Npx \docshots@aftersh { \str_use:N \l_tmpa_str }%
}
\ExplSyntaxOff
\ifdocshots@windows
  \def\docshots@gsdefault{gswin64c}%
  \edef\docshots@sep{\@backslashchar}%
\else
  \def\docshots@gsdefault{gs}%
  \def\docshots@sep{/}%
\fi
```

On Unix we render inside the temporary directory. On Windows we cannot, since `cmd.exe` keeps a `cd` in effect after the parentheses close, which would send iexec's exit-code file to the wrong place; instead we stay put and point `pdflatex` at the directory with `-output-directory`. The `pdflatex` arguments keep forward slashes, since TEX reads a backslash as an escape character:

```
\ifdocshots@windows
  \def\docshots@render{"\docshots@pdflatex"
    -interaction=errorstopmode -halt-on-error -shell-escape
    -output-directory="\docshots@tmpdir/\jobname"
    "\docshots@tmpdir/\jobname/\hash.tex"}%
\else
  \def\docshots@render{cd "\docshots@tmpdir/\jobname" &&
    "\docshots@pdflatex"
    -interaction=errorstopmode -halt-on-error -shell-escape
    \hash.tex}%
\fi
\def\docshots@log{}
\pgfkeys{
  /docshots/.cd,
  dtx/.store in=\docshots@dtx,
  log/.code=\def\docshots@log{log},
  nocrop/.code=\def\docshots@nocrop{},
```

```
  lstinputlisting/.store in=\docshots@lstinputlisting,
  inputminted/.store in=\docshots@inputminted,
  tmpdir/.store in=\docshots@tmpdir,
  tmpdir/.default=_docshots\ifxetex-xe\else\ifluatex-lua\fi\fi,
  small/.store in=\docshots@small,
  tiny/.store in=\docshots@tiny,
  runs/.store in=\docshots@runs,
  runs/.default=1,
  pdflatex/.store in=\docshots@pdflatex,
  pdflatex/.default=pdflatex,
  gs/.store in=\docshots@gs,
  gs/.default=\docshots@gsdefault,
  pdfcrop/.store in=\docshots@pdfcrop,
  pdfcrop/.default=pdfcrop,
  margin/.store in=\docshots@margin,
  margin/.default=5,
  hspace/.store in=\docshots@hspace,
  hspace/.default=.05\linewidth,
  left/.store in=\docshots@left,
  left/.default=.3\linewidth,
  right/.store in=\docshots@right,
  right/.default=.55\linewidth,
  tmpdir,pdflatex,gs,pdfcrop,margin,hspace,left,right,runs
}
\ProcessPgfOptions{/docshots}
```

Then, we print the version of `pdflatex` to the TEX log:

```
\iexec[\docshots@log,quiet]{"\docshots@pdflatex" --version}%
```

Then, we print the version of pdfcrop to the TEX log:

```
\ifdefined\docshots@nocrop\else%
  \iexec[\docshots@log,quiet]{"\docshots@pdfcrop" --version}%
\fi%
```

Then, we print the version of `ghostscript` to the TEX log, ignoring a failure, since Ghostscript is not strictly required for rendering:

```
\iexec[\docshots@log,quiet,ignore]{"\docshots@gs" --version}%
```

Then, we make a directory where all temporary files will be kept. We use Perl instead of `mkdir`, so that the code works on Windows too:

```
\iexec[null]{perl -MFile::Path=make_path -e "make_path(shift)"
  "\docshots@tmpdir/\jobname"}%
```

`\docshots@mdfive` Then, we define a command for MD5 hash calculating of a file, with the help of pdftexcmds:

```
\RequirePackage{pdftexcmds}
\makeatletter
\newcommand\docshots@mdfive[1]{\pdf@filemdfivesum{#1}}
\makeatother
```

`docshot` Then, we define `docshot` environment:

```
\makeatletter
\newenvironment{docshot}
{\VerbatimEnvironment\begin{VerbatimOut}
  {\docshots@tmpdir/\jobname/docshots-temp.tex}}
{\end{VerbatimOut}%
```

If we are in `dtx` mode, leading percent characters must be removed:

```
\ifdefined\docshots@dtx%
  \iexec[null]{perl -i -0777pe "s/(\\n|^)\\x{25}\\s?/\\1/g"
    \docshots@tmpdir/\jobname/docshots-temp.tex}%
\fi%
```

We calculate MD5 hashsum of the file content:

```
\def\hash{\docshots@mdfive{\docshots@tmpdir/\jobname/docshots-temp.tex}}%
```

If the PDF with the required name already exists, we ignore this step. Otherwise, we copy `docshots-temp.tex` into a new file and run `pdflatex`:

```
\IfFileExists{\docshots@tmpdir/\jobname/\hash.pdf}
  {\message{docshots: Won't render,
    the PDF '\docshots@tmpdir/\jobname/\hash.pdf' already exists^^J}}
  {\iexec[\docshots@log,quiet]{perl -MFile::Copy -e "copy(shift,shift) or die q(cannot copy
    "\docshots@tmpdir/\jobname/docshots-temp.tex"
    "\docshots@tmpdir/\jobname/\hash.tex"}%
  \message{docshots: rendering at line no. \the\inputlineno^^J}%
  \foreach \n in {1,...,\docshots@runs}{%
    \iexec[\docshots@log,
      stdout=\docshots@tmpdir\docshots@sep\jobname\docshots@sep\hash.stdout,
      exit=\docshots@tmpdir\docshots@sep\jobname\docshots@sep\hash.ret,
      quiet,trace]{\docshots@render}%
    \ifnum\n=1\else%
      \message{docshots: pdflatex run no.\n^^J}%
    \fi%
```

If a post-processing command was registered by we substitute its `$1`, `$2`, and `$3` placeholders and execute it. There is no shell script and no `chmod`, so the mechanism works on Windows too:

```
    \ifdefined\docshots@after%
      \begingroup%
        \docshots@aftercmd{\n}{\hash}{\hash.tex}%
        \iexec[\docshots@log,quiet]{cd "\docshots@tmpdir\docshots@sep\jobname" &&
          \docshots@aftersh}%
      \endgroup%
    \fi}}%
\global\let\docshots@after\@undefined%
```

If a cropped version of the PDF with the required name already exists, we ignore this step. Otherwise, we ask `pdfcrop` to crop the PDF:

```
\IfFileExists{\docshots@tmpdir/\jobname/\hash.crop.pdf}
  {\message{docshots: Won't pdfcrop,
    the PDF '\docshots@tmpdir/\jobname/\hash.crop.pdf'
    already exists^^J}}
  {\ifdefined\docshots@nocrop
    \iexec[\docshots@log,quiet]{perl -MFile::Copy -e "copy(shift,shift) or die q(cannot cop
      "\docshots@tmpdir/\jobname/\hash.pdf"
      "\docshots@tmpdir/\jobname/\hash.crop.pdf"}%
    \else%
    \iexec[\docshots@log,quiet]{"\docshots@pdfcrop"
      --gscmd "\docshots@gs"
      --margins 0
      "\docshots@tmpdir/\jobname/\hash.pdf"
      "\docshots@tmpdir/\jobname/\hash.crop.pdf"}%
```

```
      \fi}%
  \message{docshots: the PDF is ready from line no. \the\inputlineno^^J}%
```

Finally, we render the two-column content:

```
  \begingroup%
  \par%
  \tikz[baseline=(a.north)]
    \node (a) [draw=gray,inner sep=\docshots@margin,
      line width=.2pt]
    {\includegraphics[width=\docshots@left]
      {\docshots@tmpdir/\jobname/\hash.crop.pdf}};%
  \hspace{\docshots@hspace}%
  \begin{minipage}[t]{\docshots@right}%
    \vspace{0pt}%
    \ifdefined\docshots@lstinputlisting%
      \ifdefined\docshots@opts%
        \expandafter\lstset\expandafter{\docshots@opts}%
      \fi%
      \lstinputlisting{\docshots@tmpdir/\jobname/\hash.tex}%
    \else\ifdefined\docshots@inputminted%
      \expandafter\inputminted\expandafter[\docshots@opts]
        {\docshots@inputminted}
        {\docshots@tmpdir/\jobname/\hash.tex}%
    \else%
      \fvset{numbers=left,numbersep=3pt}%
      \fvset{frame=leftline,framerule=.2pt,rulecolor=\color{gray}}%
      \fvset{samepage=true}%
      \fvset{commandchars=none}%
      \fvset{baselinestretch=1}%
      \ifdefined\docshots@small%
        \fvset{fontsize=\small}%
      \fi%
      \ifdefined\docshots@tiny%
        \fvset{fontsize=\scriptsize}%
      \fi%
      \ifdefined\docshots@opts%
        \expandafter\fvset\expandafter{\docshots@opts}%
      \fi%
      \VerbatimInput{\docshots@tmpdir/\jobname/\hash.tex}%
    \fi\fi%
    \vspace{0pt}%
  \end{minipage}%
  \par%
  \endgroup%
  \docshotOptions{}%
}
\makeatother
```

`\docshotPrerequisite` Then, we define `\docshotPrerequisite` command:

```
\makeatletter
\newcommand\docshotPrerequisite[1]{
  \iexec[\docshots@log,quiet]{perl -MFile::Copy -e "copy(shift,shift) or die q(cannot copy)"
    "#1" "\docshots@tmpdir/\jobname/#1"}%
  \message{docshots: File '#1' copied to
```

```
    '\docshots@tmpdir/\jobname/#1'^^J}%
}
\makeatother
```

`\docshotAfter` Then, we define `\docshotAfter` command. Rather than writing a shell script, we merely remember the command as a detokenized string; the `docshot` environment substitutes its placeholders and runs it after each `pdflatex`:

```
\makeatletter
\newcommand\docshotAfter[1]{
  \xdef\docshots@after{\detokenize{#1}}%
  \message{docshots: post-processing command '\docshots@after' registered^^J}%
}
\makeatother
```

`\docshotOptions` Finally, we define `\docshotOptions` command:

```
\makeatletter
\gdef\docshots@opts{}
\newcommand\docshotOptions[1]{%
  \gdef\docshots@opts{#1}%
}
\makeatother

\endinput
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.