Continuous Integration and \TeX\ with Org-mode

by Rohit Goswami, MInstP

TeX in the cloud

August 6, 2021
Introduction

Find me here: https://rgoswami.me

Who?

- Rohit Goswami MInstP
  - Doctoral Researcher in the Jonsson Group, University of Iceland, Faculty of Physical Sciences and Science Institute
TeX is the lingua franca of academic communities

Collaborations with TeX revolve around proprietary systems
  - Overleaf

Or collaborators require some expertise with TeX
Mitigation Mechanisms

Everyone wants TeX output without writing TeX

- pandoc, orgmode promise TeX without the pain
- Cloud build machines are cheap to deploy now

Goals

- A nonexpert TeX workflow which requires no proprietary tools
  - Transparent git and CI setup
- Expert friendly in terms of templating
\documentclass{article}
\author{John Doe}
\title{Astounding Discoveries}
\begin{document}
\maketitle
\section{TeX}
Hello World
\end{document}

- Not bad
  - Fairly comprehensive
- Quickly gets out of hand

Trivial for all examples which fit on slides
Splitting Code

- **.cls** files: Loaded with `\documentclass` and `\usepackage`
- **.sty** files: Style files or packages (including beamer themes)
- **.rc** files: Control files for build systems (`.latexmkrc` or `Makefile`)

- What CTAN handles typically
  - Popularly managed by `texlive` distributions
  - Abstracts TeX and LaTeX (styling) away from document writing
    - Great for collaboration
Straying Away

Orgmode

#+author: John Doe
#+title: Astounding Discoveries
* TeX

Hello World

Pandoc Markdown

# TeX
Hello World

pandoc -s thing.md -o thing.tex
	--metadata title="Astounding Discoveries" author="John Doe"

Appears more readable and easier to write however...
Polluted Outputs

```
winsize -l {base,orgOne,pandocOne}.tex
```

8 base.tex
15 orgOne.tex
63 pandocOne.tex
86 total

- Generated files involve template substitution
Pandoc Substitution

- Top down approach
- Fixed locations in a template (e.g. zenYoda)
  - Variables expanded into TeX
- YAML metadata

```
$for(header-includes)$
$header-includes$
$endfor$
```

```
header-includes:
  - \numberwithin{figure}{section}
  - \numberwithin{equation}{section}
```
Not strictly true (preset variables)
Conceptual Differences

- **org** exporter options assume only one output
  - Allows arbitrary **emacs-lisp** evaluations
  - Sharing configurations can be clunky
- **pandoc** shares configuration system for multiple outputs
  - Sane defaults, good templating options
  - Easy to share templates
Continuous Integration

- No one likes switching computers to test
  - MacOS, Windows (WSL often), Many Linux distributions
- There are far too many options nowadays
  - Wercker, Travis CI, Shippable, GitLab CI, Github Actions
- Mostly transient docker or nix based systems
  - Setup can be annoying without nix

\TeX Gains

- Single reproducible source of truth for TeX
  - The CI machine configuration
Teaching CI about TeX

- Relying on build machine OS texlive is fragile
  - texliveonfly can get packages “on the fly”

Basic TeXLive Profile

```
selected_scheme scheme-basic
TEXDIR /tmp/texlive
TEXMFCONFIG ~/.texlive/texmf-config
TEXMFHOME ~/texmf
TEXMFLocal /tmp/texlive/texmf-local
TEXMFSYSCONFIG /tmp/texlive/texmf-config
TEXMFSYSVAR /tmp/texlive/texmf-var
TEXMFVAR ~/.texlive/texmf-var
option_doc 0
option_src 0
```
export PATH=/tmp/texlive/bin/x86_64-linux:$PATH
if ! command -v texlua > /dev/null; then
    wget http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
    tar -xzf install-tl-unx.tar.gz
    cd install-tl-20*
    ./install-tl --profile=$1
    cd ..
fi
tlmgr install luatex scheme-small \\
    biber \\
    beamer \\
    xetex \\
    pdflatex \\
    latexmk \\
    etoolbox \\
    minted \\
    teXliveonfly
tlmgr option --autobackup 0
tlmgr update --self --all --no-auto-install
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2.3.4
    - name: Install package
      run: |
        sudo apt-get install -y python-pygments emacs
    - name: Setup LaTeX
      run: |
        export PATH=/tmp/texlive/bin/x86_64-linux:$PATH
        export PATH=$HOME/texmf/bin:$PATH
        scripts/getTexLive.sh $(pwd)/scripts/texlive.profile
Running functions

Setting variables

```
(require 'ox-extra) ;; :ignoreheading:ignore:
(ox-extras-activate '(ignore-headlines))
/org-babel-tangle/
(ox-extras-activate 'ignore-headlines)
(setq org-latex-pdf-process (list "latexmk -shell-escape -f -pdfxe %f"))
(setq org-latex-listings 'minted)
(setq org-latex-minted-options
  '(("bgcolor" "white") ("breaklines" "true") ("linenos" "true") ("style" "tango")))
(add-hook 'after-save-hook '(lambda () (org-beamer-export-to-latex) t)) ;; Export
```
Org Syntax for TeX

Source blocks

```
#+begin_src <lang> :exports <code/none/results> :eval <never> +#end_src
```

Direct TeX export

```
#+begin_export <lang> #+end_export
```

Org Syntax for TeX

- Source blocks :: ~#+begin_src <lang> :exports
- Direct TeX export :: ~#+begin_export <lang>

```cpp
#include <stdio.h>
int main()
{
    return 1;
}
#+end_src```

Org and Packages

- Effectively generates `.cls` and `.sty` files

---

**Beamer Theme**: ignoreheading:ignore:

:PROPERTIES:
:BEAMER_env: ignoreheading
:VISIBILITY: folded
:END:

```latex
\begin{document}
\usepackage{tikz}
\usetikzlibrary{calc}
\usepackage{mathrsfs}
\usepackage{fontspec}
\defaultfontfeatures{Ligatures=TeX}
\newif\ifbeamer@pixelitem
\beamer@pixelitemtrue
\DeclareOptionBeamer{nopixelitem}{\beamer@pixelitemfalse}
\ProcessOptionsBeamer

% define colours
% taken from pickton on Adobe Kuler:
% https://kuler.adobe.com/Some-Kind-Of-Execushares-color-theme-3837185/
\definecolor{ExecusharesRed}{RGB}{230,37,52}
\definecolor{ExecusharesBlack}{RGB}{43,40,40}
\definecolor{ExecusharesBlue}{RGB}{22,190,207}
\definecolor{ExecusharesWhite}{RGB}{255,255,243}
\definecolor{ExecusharesGrey}{RGB}{107,110,108}
```

---
In body TeX can be directly written in export blocks

#+LATEX_HEADER: can be used to add to document headers

```latex
#+LATEX_COMPILER: xelatex
#+LATEX_HEADER: \PassOptionsToPackage{unicode=true}{hyperref}
#+LATEX_HEADER: \PassOptionsToPackage{hyphens}{url}
#+LATEX_HEADER: \PassOptionsToPackage{dvipsnames,svgnames*,x11names*,table}{xcolor}
#+LATEX_HEADER: \usepackage{amssymb,amsmath}
#+LATEX_HEADER: \usepackage{mathtools}
#+LATEX_HEADER: \usepackage{physics}
#+LATEX_HEADER: \usepackage{hyperref}
#+LATEX_HEADER: % Make use of float-package and set default placement for figures to H
#+LATEX_HEADER: \usepackage{float}
#+LATEX_HEADER: \floatplacement{figure}{H}
```
Generating Classes

- `#+LATEX_CLASS: myclass` is populated from `org-latex-classes`
  - So we need to add to it before use
- Or use it as part of the single file setup

```lisp
(append-to-list
  'org-latex-classes
  '(
    "documentclass[a4paper, sfsidenotes, openany, justified]{tufte-book}"
    ("\part{%s}" . "\part*{%s}"
     "\chapter{%s}" . "\chapter*{%s}"
     "\section{%s}" . "\section*{%s}"
     "utf8" . "utf8x"
     "\subsection{%s}" . "\subsection*{%s}")))
```
Replacing Jupyter

```python
#BEGIN_SRC python :results output file :exports both
import numpy as np
import matplotlib.pyplot as plt

xs = np.linspace(0,1,myN+1)
ts = xs
X, Y = np.meshgrid(xs,ts)
fig = plt.figure(figsize=(12,10))
ax = fig.gca(projection='3d')
surf=ax.plot_surface(X, Y, T, cmap=cm.coolwarm)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
surf.colorbar(surf, shrink=0.55, aspect=8)
ax.view_init(elev=15, azim=120)
plt.xlabel('Time')
plt.ylabel('Space')
plt.title("Crank Nicholson Heat Equation Solver, Q2")
plt.savefig('images/plotp2.png', dpi = 300)
plt.close()
print('images/plotp2.png')
#END_SRC
```
Much nicer (and more native) than Jupyter

Part a

Given the IBCs:

\[
\begin{align*}
    u(x, 0) &= 2 \cosh x & \text{for } 0 \leq x \leq 1 \\
    u(0, t) &= 2e^{2t} & \text{for } 0 \leq t \leq 1 \\
    u(1, t) &= (e^2 + 1)e^{2t-1} & \text{for } 0 \leq t \leq 1
\end{align*}
\]

Recall that the exact solution for \( u_t = 2u_{xx} \) for \( 0 \leq x \leq 1, 0 \leq t \leq 1 \); is \( u(x, t) = e^{2t+x} + e^{2t-x} \).

We can use the same generic function defined earlier, and simply need to write in the appropriate conditions.

```python
def drichp2a(x):
    return (2*np.cosh(x))

def lef2a(t):
    return (2*np.exp(2*t))

def righ2a(t):
    return (np.exp(2+1)*(np.exp(2*t-1)))
```
(require 'package)
(setq package-check-signature nil)
(add-to-list 'package-archives '("melpa" . "https://melpa.org/packages/")) t)
(package-initialize)
(unless package-archive-contents (package-refresh-contents))
(package-install 'use-package)
(package-install 'org)
(dolist (package '(use-package))
  (unless (package-installed-p package)
    (package-install package)))
(use-package org-ref
  :ensure t)
(require 'ox-latex)
;; Define an interactive function for easy testing
(defun org-beamer-export-to-pdf-directory (files)
  "Export all FILES to latex."
  (interactive "Export org files to tex")
;; Export all org files given on the command line
(org-beamer-export-to-pdf-directory argv)
More completely, see this script

With this action

```yaml
- name: Generate TeX
  run: emacs -q -nl --script scripts/org2tex.el src/filename.org

- name: Build pdf
  run: |

export PATH=/tmp/texlive/bin/x86_64-linux:$PATH
export PATH=$HOME/texmf/bin:$PATH
cd src/
texliveonfly -c latexmk -a ”-pdfxe -shell-escape -f” wgtqc.tex
```
Omitted Topics

Caching  CI rebuilds can be sped up with caching mechanisms
Emacs-Lisp  Too much and too irrelevant for TeX in general
Advanced Concepts  CI configurations and custom \texttt{emacs} setups; a lot more detail here
Jupyter and Org  Orgmode can be used as a fully fledged multi-language plain text Jupyter replacement for data science
Going beyond single files with :noweb yes

- Uses named blocks for clarity #+NAME: orgConf
- Named blocks are not tangled

1. (eval-after-load 'ox (require 'ox-koma-letter))
2. (with-eval-after-load 'ox-latex
3. <<tex_process>>
4. <<common_pkgs>>
5. <<tufte_book>>
6. <<koma_art>>
7. )
Conclusions

- orgmode provides a viable alternative syntax for writing TeX
  - Can be used on public clouds without knowing emacs
- TeX is here to stay
- Abstracting complexity away from users is good
  - Public cloud usage spares installation issues
    - Enables git workflows
- Alternative syntaxes provide more natural usage for novices
  - orgmode facilitates native execution
Thank you