Autobuild Python and R Lessons

Release beta

Sep 03, 2019

Contents:

Below you will find documentation for a CI Pipeline that we built at earth Lab. This build automagically takes *Rmarkdown* and *Jupyter notebook* files, runs them and then creates an output markdown file to be used in GitHub pages.

The build is in operation but needs considerable cleanup. All contributions are welcome.

CHAPTER 1

About Auto-Building Jupyter Notebooks and RMarkdown Files: Our CI Pipeline

When we originally created earthdatascience.org, we never imagined that it would balloon into a resource being used by thousands of people around the world! Ok so maybe we did dream that would happen, but we never knew it would happen so quickly. As our lessons became more popular and our user base grew (we are now up to >56,000 unique users a month) we realized that maintaining this content was a challenge. We wanted an easier way to update lessons.

Previously we had been updating this manually. This was fine when only one or two of our team was contributing. It became increasingly challenging when we wanted different people to help us update content.

We built this pipeline to make it easier for anyone to contribute content to earthdatascience.org. Through this pipeline, anyone who has access to our private repo, can submit a new lesson to our website. This repo could easily be public if you don't need to hide homework answers! Rather than worrying about building the lessons, the user can submit a pr with a `Rmarkdown` or `Python` notebook. That file gets built, and images produced by the file are created and moved to our website repo. If the build fails, the user can identify why it failed and then submit a change.

The build also includes a cron job that is set to run weekly. This is an ongoing test that the lessons build. We hope that other people find this build useful. It needs a lot of cleanup. As such, all contributions are welcome!

CHAPTER 2

Autobuild Pipeline for Earthdatascience.org - Overview

The building Jupyter NOtebooks and RMarkdown files for the www.earthdatascience.org learning portal build has several high level steps.

1. Original lessons are stored in the `earth-analytics-lessons` repo. This is where you will find `.ipynb`, `.rmd` and image files that are manually added to lessons. When a lesson is updated or an image is added to the repo, it triggers a build that produces a markdown file for the lesson and moves any related images.
2. Once lessons are built, they are moved to the `eds-lessons-website` directory. This is where the final versions of the .md files and associated images (both manually added and auto-build created) live.
3. The last step is temporary. Because the live website currently lives on `earthlab.github.io`, we push files committed to the master branch of the `eds-lessons-website` repo to the live `earthlab.github.io` website. A smaller CI build is in place to process those commits. Most people will not need step 3. And it's very likely that you might not need step 2 either.

Our build has extra layers because we host homework assignment answers in the build. Those answers can not be public!

2.1 Directories

2.1.1 Earth Analytics Lessons (`earth-analytics-lessons`)

This is where the source lesson files (referred to throughout as "notebooks") for the website posts (courses, workshops, tutorials, blog) live. These include:

- R posts: RMarkdown notebooks (`.rmd`)
- Python posts: Jupyter notebooks (`.ipynb`)
- Images associated with posts (in `images` directory). IMPORTANT: images in this directory that are manually added are stored here. images associated with building a notebook. are created in the CI build and pushed so they never exist in this repo.

This repo also contains the YAML file for CircleCI builds and scripts for generating posts and logging changes/errors when updates to post content are made.

2.1.2 Earth Data Science Website (`eds-lessons-website`)

This is where the posts output by `earth-analytics-lessons` in `.md` format are stored. They are pushed to a branch on the remote with the same name as the one in `eds-lessons-website/earth-analytics-lessons` containing changes made to website content.

2.1.3 Earth Analytics (`earth-analytics`)

The earth-analytics directory is the data directory that is required for most earth analytics course builds. For that reason, lessons for the Earth Analytics courses are rendered from this directory (this is the working directory for the builds). It stores data required for each lesson, simulating the course folder an Earth Analytics student would have on a personal machine. The **earth-analytics** directory is created *only* while notebooks are rendered in the CircleCI build.

2.2 CircleCI Pipeline

2.2.1 Workflows

One-time Builds (`build`)

This workflow is run whenever changes to post content are pushed to `earth-analytics-lessons`.

1. Check conda environment
    - Lists all packages (R, Python, and Linux) installed in the `conda` environment and package versions.
2. Get latest lesson commit
    - Working directory: `~/earth-analytics-lessons`
    - Fetches any changes from the remote earth-analytics-lessons repo using `git diff`
    - Stores these changes in a temporary log file (`changed_files.txt`) using `scripts/parse_commit.py`. This is read to generate posts whenever changes are pushed to individual notebooks.
    - Storing information this way in temp log files allows us to read it in for use in later build steps.
3. Get current branch
    - Working directory `~/earth-analytics-lessons`
    - Grabs the name of the branch that triggered the CircleCI build and stores it in a temporary log file (`current_branch.txt`).
4. Fetch most recent commit message
    - Working directory `~/earth-analytics-lessons`
    - Grabs the message from the commit that triggered the CircleCI build and stores it in a temporary log file (`commit_msg_latest.txt`).
5. Display modified Jupyter notebooks (.ipynb files)
    - Working directory: `earth-analytics-lessons`

- Reads in and displays any changed Python notebooks (`.ipynb`) from `changed_notebooks.txt`.

6. Display modified `.Rmd` files
   - Working directory: `~/earth-analytics-lessons`
   - From `changed_notebooks.txt`, reads in and displays any changed RMarkdown notebooks (`.rmd`)
7. Display modified images
   - Any images that are manually added to the website will be displayed during this step.
   - This step will normally *do nothing*, since images are in most cases generated DURING the notebook rebuilds. Images for notebooks should not be pushed directly to `~/earth-analytics-lessons`, except in special cases when updating objects meant to remain permanently on the build, i.e. headers/footers for blog posts and static images for lessons.
8. Clone eds-lessons-website
   - Working directory: `root`
   - This step clones the `eds-lessons-website` repo locally
   - If the `earth-analytics-lessons` branch that triggered the CircleCI build is something other than `master`, this step will `git checkout` a new branch for `eds-lessons-website` that matches the one for `earth-analytics-lessons`.
9. Delete removed/renamed/moved images from website
   - In this step, any images that are manually removed, renamed, or moved in a commit to `~/earth-analytics-lessons` are also removed in the matching directory on `~/eds-lessons-website/images/...`.
   - This should *only* be done to images that are added to the build manually (i.e., for blog posts). Images that are generated by notebooks are handled separately (see `Delete removed/renamed/moved posts from website`).
10. Delete removed/renamed/moved posts from website
    - In this step, the post (`.md`) named for any notebooks (`.rmd`, `.ipynb`) that were manually removed, renamed, or moved in a commit to `~/earth-analytics-lessons` is removed from the matching directory on `~/eds-lessons-website/_posts/...`.
    - Any images associated with the post are also deleted.
      - These images are stored in a subdirectory of the directory where the source `.rmd` or `.ipynb` notebook lives.
      - When the notebook is rebuilt, these images are pushed over to the matching directory on `~/eds-lessons-website/images/...`.
      - During this step, the subdirectory on `~/eds-lessons-website/images/...` is deleted along with the post.
        * *(There is no need to remove the images in this way on `~/earth-analytics-lessons` because they are never permanent. They are only ever generated as intermediate outputs when the notebook builds, then immediately pushed over to `~/eds-lessons-website`.)*
11. Move manually added images to website
    - Working directory: `~/earth-analytics-lessons`
    - If images were added to the website (see #6), this step reads in and displays any changed images present in `changed_image_files.txt`.

1. Checks whether a matching directory containing the image already exists on `eds-lessons-website` and if not makes it.
2. Copies the image to the matching directory on `eds-lessons-website`.

This ensures that the manually-added images will also get pushed to `eds-lessons-website` in sync with their addition to `earth-analytics-lessons`.

12. Execute Jupyter notebooks and export to md
    - Working directory: `earth-analytics-lessons`
    - Executes all changed Python notebooks (`.ipynb`, read in from `changed_notebooks.txt`) using the script `scripts/generate_posts.py`, which renders each notebook to a markdown post.
    - Notebooks that had problems rendering are recored in the temporary log file `nb_errors.txt`
13. Knit `.Rmd` to `.md`
    - Working directory: `earth-analytics-lessons`
    - Executes all changed R notebooks (`.rmd`, read in from `changed_notebooks.txt`) using the script `scripts/knit_lessons.R`, which renders each notebook to a markdown post using `knitr`.
    - Notebooks that had problems rendering are recored in the temporary log file `nb_errors.txt`
14. Generate `.gitignore` to bypass posts with errors
    - Working directory `eds-lessons-website`
    - If there were any problem notebooks (either R or Python) recorded in `nb_errors.txt`, this step makes a temporary `.gitignore` for the push to the remote `eds-lessons-website` repo
      - Any notebooks that could not render properly to markdown will be skipped over in the push (while those that were successful *are* pushed).
    - A reference to the `.gitignore` itself must also be added to the `.gitignore` so that it remains temporary and is also not pushed to the remote (which would cause the problem posts to be ignored with successive commits, even after they are fixed!)
15. Trigger eds-lessons-website push deployment
    - Working directory: `eds-lessons-website`
    - This step will only push a commit when one or more notebooks was successfully rendered to a post. (In other words, if all notebooks failed to render successfully, no changes will be pushed).
    - Changes will also (in rare cases) be pushed if updates were made to images that need to permanently live in `earth-analytics-lessons` (i.e. blog post footers).
    - If there are changes to push, the commit message from `earth-analytics-lessons` is fetched and passed for the changes that are staged.
    - Then the changes are pushed to the branch checked out for `eds-lessons-website` at the beginning of the build.

    a. Display problem notebooks and trigger build fail as needed
      - This step has to be nested in the deploy step because it is the last part of the job that will run.
      - We need to ensure that any posts that were rendered successfully get pushed to `eds-lessons-website`, so we can't trigger the build fail any sooner.

Weekly Build (`weekly`)

This workflow builds ALL posts automatically, according to a `cron` schedule. It *does not* push any content to `eds-lessons-website`, rather it only tests generating the posts and flags the ones with errors.

1. Check conda environment
   - Lists all packages (R, Python, and Linux) installed in the `conda` environment and package versions.
   - This is identical to the step run at the start of the `build` job.
2. Clone eds-lessons-website
   - Working directory: `root`
   - *See xx*.
   - The key difference is that this *only* clones `master` because the changes are never pushed. The `eds-lessons-website` directory just needs to be in place so that the posts are written to the correct locations as they are rendered.
3. Execute all rmd files with knitr
   - Working directory: `earth-analytics-lessons`
   - This step uses `scripts/knit_lessons.R` to test rendering *all* `.rmd` files to `.md` posts using `knitr`.
   - See here for help with passing flags (i.e., "rebuild all") to `scripts/knit_lessons.R`
4. Execute all notebooks with papermill
   - Working directory: `earth-analytics-lessons`
   - This step uses `scripts/generate_posts.py` to test rendering *all* `.ipynb` files to `.md` posts.
   - See here for help with passing flags (i.e., "rebuild all") to `scripts/generate_posts.py`
5. Display problem notebooks and trigger build fail as needed
   - See `15. Trigger eds-lessons-website push deployment`.
   - This *is not* nested in the deploy stage for the weekly build. Since nothing gets pushed, the output here just shows which notebooks are currently breaking.

2.3 Scripts and Utilities (`scripts`)

Scripts and utilities for rendering notebooks, fetching data, etc. are located in `~/earth-analytics-lessons/scripts`.

2.3.1 CircleCI Build

`parse_commit.py`

Parse commit and separate out into different file types. Grabs the latest commit and separates out committed file via file type. It outputs three temporary log files during the build:

- Changed Jupyter notebooks (`.ipynb`) files in `changed_notebooks.txt`.
- Changed RMarkdown notebooks (`.rmd`) in `changed_rmd_files.txt`.
- Changed images (`.jpg`/`.jpeg`/`.gif`/`.png`) in `changed_image_files.txt`.

parse_deleted.py

Parse deleted files and separate out into different file types. This is currently kept separate from `parse_commit.py` because it handles the file types somewhat differently. We could think about merging these scripts down the road.

- The key difference is that notebooks (`.ipynb` and `.rmd`) are written to the same logfile here, rather than separate ones. This is because we

ignore-dirs.txt

Notebooks in any directories listed in this text file (one directory per line) *will not* be rebuilt. This is useful when rebuilding many directories at once (i.e., a full course's worth of lessons, or the Weekly job).

2.3.2 Python Lessons

generate_posts.py

Build .ipynb file to .md file.

- **Flags**
  - ***none*** In the default case, if no flags are given, `knit_lessons.R` will just knit whatever files are passed after it is called with `Rscript`.

    Usage

    ```
    python scripts/generate_posts.py courses/earth-analytics-python/02-intro-to-
    ↪lidar-and-raster/2018-02-05-intro-lidar-raster-landing-page.ipynb
    ```

  - **all**

    Rebuilds every `.ipynb` file from **all** directories not ignored.

    Usage

    ```
    python scripts/generate_posts.py all
    ```

  - **dir** *dir1 dir2* **...**

    Rebuilds a list of directories given after the `dir` flag.

    Usage

    ```
    python scripts/generate_posts.py dir courses/earth-analytics-python/02-intro-
    ↪to-lidar-and-raster
    ```

2.3.3 R Lessons

knit_lessons.R

Build/knit .Rmd file to .md file

- **Flags**

- ***none*** In the default case, if no flags are given, `knit_lessons.R` will just knit whatever files are passed after it is called with `Rscript`.

  Usage

  ```
  Rscript scripts/knit_lessons.R courses/earth-analytics-r/02-time-series-data/
  ↪2017-01-01-week-02-time-series-data.Rmd
  ```

- **`all`**

  Rebuilds every `.rmd` file from **all** directories not ignored.

  Usage

  ```
  Rscript scripts/knit_lessons.R all
  ```

- **`dir`** *`dir1 dir2 ...`*

  Rebuilds a list of directories given after the `dir` flag.

  Usage

  ```
  Rscript scripts/knit_lessons.R dir courses/earth-analytics-r/02-time-series-
  ↪data
  ```

`twitter_auth.R`

This script is used to read Twitter credentials into the R knit environment to authenticate the Twitter API for Week 13 R lessons. It is called from a few of the Week 13 lessons but *never* needs to be called directly from the build.

This script calls an .RDS file that is encrypted in the earth-analytics-lessons repo. It reads the RDS file in and unencrypts it. We are using the earthdatascience twitter handle for the app attached with all twitter lessons.

> leah has the encryption code cyphr is used for the encryption

2.3.4 Downloading Data

`get_data_rmd.R`

Download data and restructure paths for .Rmd files. This is an external script that downloads the data needed for the .Rmd files and restructures the paths so that they are referenced correctly in the script and things run. This relies heavily on a lookup table for managing data downloads.

`url_codes_by_week.csv`

A lookup table is used by `knit_lessons.R` to manage data downloads for each R lesson. It consists of several columns:

- **`week`**

  The topic names for each week of the EDS R course. These are analogous to the topic names provided for each lesson in the python course. Right now, the names simply match the subdirectories in `~/eds-lessons-dev/courses/earth-analytics-r`.

- **`dirname`**

  The names of the subdirectories in `~/earth-analytics` that stores each week's data. If *left blank*, the target file will be stored in the root folder of `~/earth-analytics` rather than in a subdirectory.

- **url**

  The URLs for each file download.

- **file**

  The target filename. If *left blank*, the downloaded file is simply named with the value in the `dirname` column. This is the default behavior, since most of the data downloads are `.zip` files that contain the bulk of files needed to build a week's lessons and serve as the main directory for that week's data once extracted.

  However, for rows where the `file` column contains a value, the downloaded file is named for that value instead. This is primarily used for supplemental data downloads that are extracted or moved into a week's directory after the main data download has taken place.

- **ext**

  The target file extension. If this is a `.zip` file, it is extracted to the directory specified by `dirname`. Other file types are downloaded to the root folder, then moved to the target directory specified by `dirname`.

Special considerations

- **Download order matters!**

  This file assumes that the main lesson data will always be downloaded/extracted first so that the subdirectory represented by `dirname` exists before any further file downloads take place. As such, the first row for a week of lessons should always represent the main lesson zip file, with any successive rows representing supplemental files (see Weeks 8, 13).

- **Multiple lesson data downloads per week**

  In some cases (i.e., for the Week 7 lessons), a previous week's data might be needed in addition to the files specific to that week. This is handled by:

  - Repeating the week ID in the `week` column (i.e., `07-multispectral-remote-sensing`).
  - Providing separate `dirname` values to store the files that need to be read in by the lessons. For example, since Week 7 also requires the Week 3 data, `dirname` is set to `week-03` to handle the Week 3 download and `week-07` for the Week 7 download.

- **Nested files**

  Sometimes supplementary files must be nested in the subdirectory for a given week (i.e., the Week 8 lesson rebuilds).

  In this case, the `dirname` value can simply be modified to represent the relative path to the files. For example, the supplementary files for Week 8 are extracted to the nested folder `week-08/landsat`.

2.3.5 Deprecated

`get_all_notebooks.py`

- This script had been used to fetch the names of all `.ipynb` and `.rmd` files using `glob`, but now this is handled by `generate_posts.py`.

2.3.6 Misc/Unsorted

- **`rmarkdown_to_ntbk.py`** Converts an RMardown file to a Jupyter notebook.

  [name=Joe] This doesn't appear to be used anywhere now, should it be deprecated?

- `jupyter_templates`

[name=Joe] Looks like these are used by `knit_lessons.R` to convert RMarkdown files to Jekyll-flavored markdown. However, should confirm.

- `landsat-modis-data`
  - ???
- `test-remove-files`
  - ???

2.4 Rebuilding Notebooks

2.4.1 Debugging

Local Debugging

Notebooks may be debugged on your local machine using `RStudio` (R lessons) or `JupyterLab` (Python lessons).

`RStudio` setup for debugging for R notebooks

1. Fetch the data needed to test the lesson locally can be fetched via CLI:

```
R --silent -e "source('scripts/get_data_rmd.R'); get_data(path = 'path_to_file')"
```

where `'path_to_file'` is the name of an `.rmd` file in the target directory. For example, if we wanted to fetch the data for local testing of the `03-lidar-raster-data` notebooks, the name of the landing page `courses/earth-analytics-r/03-lidar-raster-data/2017-01-01-week-03.Rmd` could be substituted in.

1. Transfer the `earth-analytics` folder containing the lesson data to the approrpriate directory. The data will by default download to `~/earth-analytics`. To make it accessible by the `.rmd` file locally, use

```
cp -R ~/earth-analytics ../earth-analytics
```

to mimic the directory structure of the CircleCI build.

1. Open a new RStudio session and set the project working directory to wherever `earth-analytics-lessons` is stored on your machine.
2. Adjust the global environment options to start RStudio sessions in the root directory of `earth-analytics-lessons`.

   a. Navigate to `Tools -> Global Options -> RMarkdown`

   b. In the `Evaluate chunks in directory` menu, select `Project` so that RStudio opens by default in the root folder of `earth-analytics-lessons`, NOT the directory containing the notebook (the default behavior).

3. Add a ***temporary*** code chunk (without comments) to change the root directory to `~/earth-analytics` which contains the `data` folder:

```
# ```{r}
knitr::opts_knit$set(root.dir = file.path(dirname(getwd()), 'earth-analytics'))
#```
```

> [name=Joe] YMMV here, since there are known issues with directory management when working with RMarkdown in RStudio. When I tested this locally (RStudio v1.1.463 for OSX), I could not get it to work, but in other instances this did the trick.

Note that these instructions are specific to R lessons. There are a couple of key differences for blog posts:

1. Data *does not* need to be downloaded into an `earth-analytics` directory.
2. The root directory does not need to be changed. It should suffice to set the global RMarkdown options so `Evaluate chunks in directory:Project` as above, since blog posts knit in `~/earth-analytics-lessons` rather than `~/earth-analytics` (see `scripts/knit_lessons.R`).

Debugging Python notebooks with `JupyterLab` / `Jupyter Notebook`

- Jupyter notebooks (`.ipynb`) can be run as-is without additional setup, since they are set up to fetch lesson data using `earthpy`.
- If there are any notebooks with missing data, the landing page can be run first to fetch it.

  > [name=Joe] Leah, I know this was the case previously when I was testing rebuilds of the notebooks (i.e., I remember Week 2 `lidar-intro/2018-02-05-lidar03-chm-dtm-dsm.ipynb` used to have this problem). However, it looks like the data downloads are now in place throughout the notebooks? If this is the case, just delete this bullet.

2.5 Rebuilding specific course directories

IMPORTANT: Be careful when rerunning an entire directory. This works well as a one-time effort when a dir needs to be rebuilt. However because the website repo will already have commits in it on this branch, it could lead to git issues in the build.

*Also be sure to run a force build AFTER the website repo has been cloned. Otherwise things will fail because the scripts write to that website repo and create directories. if they write BEFORE it's cloned then it's not a git repo and can't clone. *

Ideally, a force build of a directory should be run on a fresh branch to avoid merge conflicts.

> Both scripts support the `all` flag which will build all lessons.

2.5.1 R Lessons

Call `scripts/knit_lessons.R` with the first argument `dir`, which is a flag to rebuild a directory, followed by the relative paths to any directories you wish to rebuild.

Running via command line (SSH)

Single directory

```
Rscript scripts/knit_lessons.R dir courses/earth-analytics-r/08-multispectral-remote-
↪sensing-fire
```

Multiple directories

```
Rscript scripts/knit_lessons.R dir courses/earth-analytics-r/08-multispectral-remote-
↪sensing-fire courses/earth-analytics-r/13-programmatic-data-access
```

Running in CircleCI Pipeline

Insert a temporary build step anywhere *after* `eds-lessons-website` is cloned and *before* `Generate .gitignore to bypass posts with errors` that contains `Rscript scripts/knit_lessons.R dir...`, for example:

```
- run:
    name: Test directory rebuild
    working_directory: ~/earth-analytics-lessons
    command: |

      Rscript scripts/knit_lessons.R dir courses/earth-analytics-r/08-multispectral-
↪remote-sensing-fire courses/earth-analytics-r/13-programmatic-data-access
```

2.5.2 Python lessons

Rebuilds of Python (`.ipynb`) lessons and posts work in much the same way as the R lessons but rely on the script `scripts/generate_posts.py` rather than `scripts/knit_lessons.R`.

Running via command line (SSH)

Single directory

```
python scripts/generate_posts.py dir courses/earth-analytics-python/02-intro-to-lidar-
↪and-raster
```

Multiple directories

```
python scripts/generate_posts.py dir courses/earth-analytics-python/02-intro-to-lidar-
↪and-raster courses/earth-analytics-python/03-intro-to-python-and-time-series-data
```

Running in CircleCI Pipeline

```
- run:
    name: Test directory rebuild
    working_directory: ~/eds-lessons-dev
    command: |

     python scripts/generate_posts.py dir courses/earth-analytics-python/02-intro-to-
↪lidar-and-raster courses/earth-analytics-python/03-intro-to-python-and-time-series-
↪data
```

2.6 Build Secrets

Two kinds of build secrets exist:

1. Secret **variables** used in the CircleCI build are stored under `Settings/earthlab/earth-analytics-lessons`.
2. Encrypted **files** that store build secrets live in the `earth-analytics-lessons` repo in the `keys` directory.

2.6.1 Github API tokens

- Variables:
  - `EDS_LESSONS_GITHUB_TOKEN_EL`
    - This is the GitHub API token that enables parsing commits and pushing/pulling changes from the CircleCI build.
    - Instructions to create the token can be found here.

      [name=Joe] Verify that this is correct! I didn't generate the initial set, but this looks like what is needed.
  - `EDS_LESSONS_GITHUB_TOKEN` (deprecated)
    - This is the OLD token and can be deleted.

2.6.2 Twitter keys

Twitter keys are used in the R and Python Earth Analytics lessons on programmatic data access to illustrate accessing the Twitter API. There is a google doc that stores the login credentials for our Twitter app if they ever get lost.

R Lessons

For R Lessons, the Twitter credentials are stored in an encrypted file on the build (in `keys`).

- Variables:
  - `rtweet_pass`
    - This is the passphrase used to decrypt the `.rds` file containing the full credentials. It is used by `scripts/twitter_auth.R`.
    - Leah has a hard copy of the passphrase if needed, as well as a copy of the script to generate the encrypted `.rds` file locally.
- Files:
  - `keys/ea_twitter_secret.rds`
    - This is the encrypted `.rds` file that stores the Twitter API credentials.

Python Lessons

Twitter credentials are stored as build environment files on the Python side, as they are called directly from lessons to authenticate `tweepy`. This is accomplished with `os.env`, which fetches system environment variables.

- Variables:

  - `tw_access_token`
    * The Twitter API access token.
  - `tw_access_secret`
    * The secret Twitter API access token.
  - `tw_consumer_key`
    * The Twitter API consumer key.
  - `tw_consumer_secret`
    * The Twitter API consumer secret token.

Qualtrics keys

Qualtrics keys are used to authenticate the Qualtrics API for blog posts. Authentication is carried out using the `qtoolkit` R package developed by Earth Lab. If these ever get lost, Leah has a local copy that can be shared securely with KeyBase.

- Variables:
  - `qualtrics_id`
    * The Qualtrics user ID.
  - `qualtrics_token`
    * The Qualtrics API token.

2.7 Building Blog Posts

2.7.1 Relative Paths For Images in Rmd

All paths for images in Rmd file need to be relative to the Rmd file as follows: `images/blog/`

rather than the previously used: `../images/blog`

This means that the image directory is a subdirectory of the primary working directory for the blog posts (e.g. `root_dir` in build env).

Each blog post also needs a subdirectory under `images/blog/` to store generated plot images; for example: `images/blog/2019-03-18-four-must-have-skills-in-earth-data-science/`

2.7.2 Plot Footer Generation

The plot footer is generated in the build from three images in `images/blog/`:

1. `earth-lab-logo-white.png`: latest Earth Lab logo with white font
2. `plot-footer-twitter.png`: latest Earth Lab Twitter handle in all white
3. `black-banner.png`: solid black banner for background of footer

The plot footer generation is completed in four steps in the Rmd file:

1. footer is generated with magick package using the three files listed above:

```
earthlab_orig <- image_read(path = "images/blog/earth-lab-logo-white.png") %>%
   image_scale("x80")
twitter_orig <- image_read(path = "images/blog/plot-footer-twitter.png") %>%
   image_scale("x70")
black_banner <- image_read(path = "images/blog/black-banner.png")
earthlab_logo <- image_composite(image_scale(black_banner, "1000x100"), earthlab_orig,
↪ offset = "+30+10")
twitter_logo <- image_composite(image_scale(black_banner, "1000x100"), twitter_orig,␣
↪offset = "+540+15")
logo <- image_append(image_scale(c(earthlab_logo, twitter_logo)), stack = FALSE)
```

1. code executed to generate and save a plot image without the footer:

```
ggsave("images/blog/2019-03-18-four-must-have-skills-in-earth-data-science/earth-data-
↪science-skills-in-demand-automated-workflows.png", plot_auto, height = 15, width =␣
↪15, units = "in", device = "png")
```

1. plot image without footer is read as variable:

```
plot <- image_read("images/blog/2019-03-18-four-must-have-skills-in-earth-data-
↪science/earth-data-science-skills-in-demand-automated-workflows.png") %>%
  image_border("white", "15x15")
```

1. footer is appended to the plot image without footer; the appended image is then written out to replace the original plot image file without the footer

```
final_plot <- image_append(image_scale(c(plot, logo), "1000"), stack = TRUE)
image_write(final_plot, "images/blog/2019-03-18-four-must-have-skills-in-earth-data-
↪science/earth-data-science-skills-in-demand-automated-workflows.png")
```

2.8 Cliff Notes – force rebuild specifical directories

Rebuild just the tutorials

```
      - run:
          name: Force rebuild all workshops in this pr
          working_directory: ~/earth-analytics-lessons
          command: |
            Rscript scripts/knit_lessons.R dir workshops/
            python scripts/generate_posts.py dir workshops/
```

CHAPTER 3

Remote Debugging & SSH with CircleCI

Remote debugging refers to debugging performed on the CircleCI build.

3.1 Interactive debugging via SSH

Debugging can be performed interactively via CLI, using SSH to acess a build after it is run. This can be handy for checking the content of intermediate files generated within the build, whether files updated as expected, etc.

You will need a local RSA key to access the build via SSH. See Authenticating SSH for instuctions on setting up an RSA key on your machine.

On the status page for your CircleCI job, select the drop-down menu next to the button at the upper-right that says `Rebuild` and select `Rerun job with SSH`. The job will rerun *without deploying changes* and when finished will instead list a remote address where the job can be accessed. In the terminal, enter the command

```
ssh -p XXXXX xx.xxx.xxx.xx
```

using the values given at the end of the CircleCI build after `-p`. These represent the unique ID and address of the remote CircleCI job, respectively. Answer `yes` to the prompt about the remote RSA key.

Once you are finished debugging, leave the job with the command `exit`. The job will remain active for another 10 minutes, then time out. **Note** that you can SSH into a job more than once, but it *will not* have the same ID and address as last time.

3.2 Debugging an R Lesson knit job

It is not possible to interactively debug an `.rmd` as it is knitting. However, variables in the notebook may be viewed from the knit progress report once the build completes by inserting this code chunk (uncommented):

```
# ```{r message = F}
# message(print(...))
# ```
```

where `...` is some variable or value you are interested in viewing from the knit job. The nested `print` function should coerce whatever data type the target variable/value is to a string that can be output by `message`.

3.3 Authenticating SSH

CircleCI provides these instructions for authenticating SSH with your account.

You will need to set up a local RSA key before being able to enter a build. This is device-specific, so if you are working on multiple machines, you will need to run through these steps several times.

1. Checking whether a local SSH key exists:

   https://help.github.com/en/articles/error-permission-denied-publickey

2. Generating a new local SSH key if one is needed:

   https://help.github.com/en/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

3. Adding the new SSH key to GitHub account:

   https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account

   Note that you will need to copy the SSH key you generated *from the terminal* to your clipboard. There are different ways to do this based on your OS.

4. Finally, verify your connection to GitHub:

   ```
   ssh -T git@github.com
   ```

   Enter `yes` if prompted. Then you should see a message like:

   ```
   Hi jvtcl! You've successfully authenticated...
   ```

CHAPTER 4

Future To Do List

There are many items that are still lingering that need to be cleaned up in this build

eventually these will become issues but for now, the list below is a start at items we never got around to cleaning up.

- Cleanup the knit lessons script
- Twitter lesson timeout issue
- Might be unneeded scripts in the repo –
  - rmd –> notebook script
  - cleanup deprecated folder
- Add docstrings to functions

CHAPTER 5

Indices and tables

- genindex
- modindex
- search