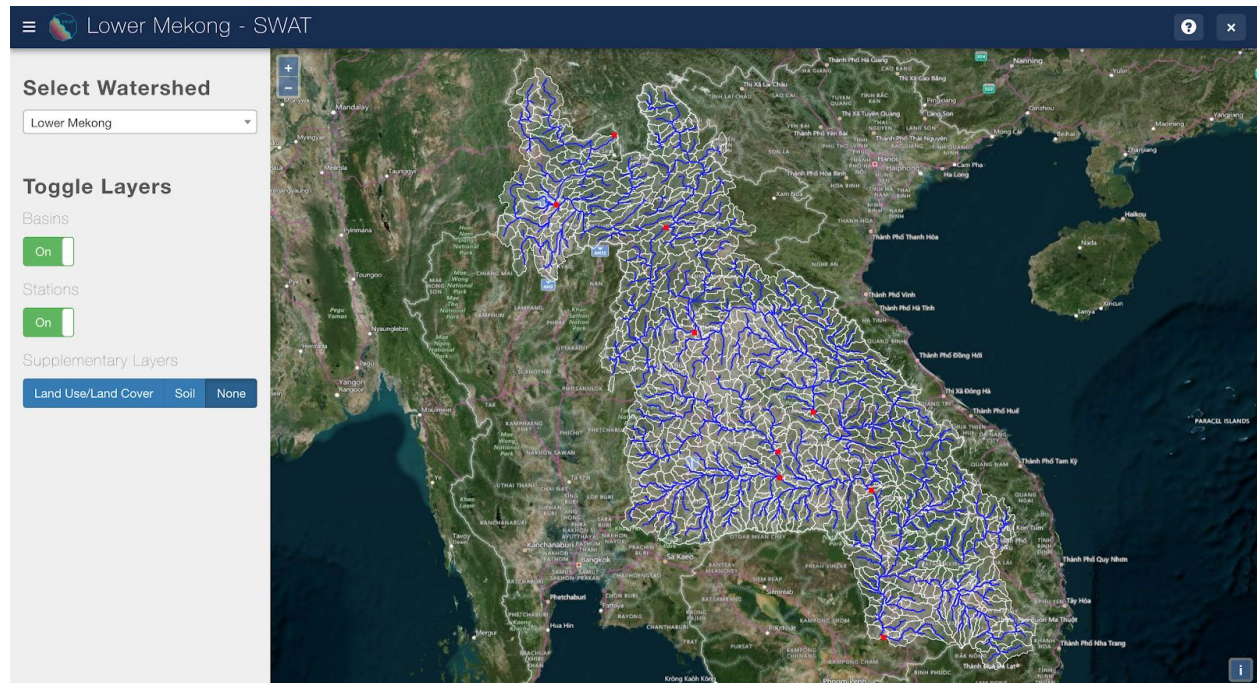


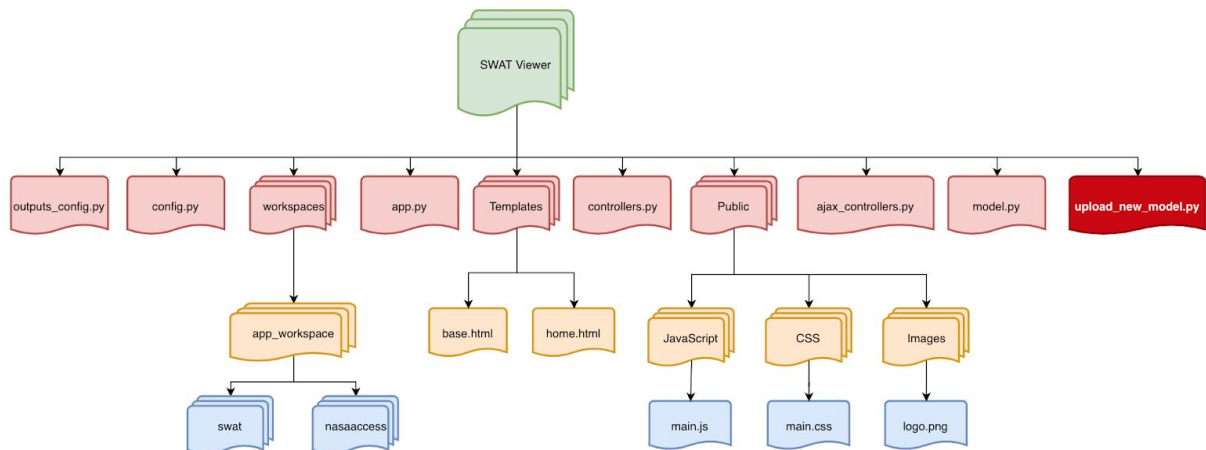
SWAT Data Viewer - Upload New SWAT Model



Overview

The SWAT Data Viewer was developed to act as a virtual and open access data store for stakeholders and decision-makers to view and download SWAT model inputs and outputs for their area of interest. The app features various spatial and temporal data visualization interfaces (mainly in the form of maps and time series plots). Unlike many of the SWAT related web applications, this application is completely modular meaning that it can be duplicated, customized and served from any server running the Tethys platform and can display the data from any valid SWAT model.

The modular capabilities of this application are facilitated through the use of web database (PostgreSQL) and map (Geoserver) services. Before the app can visualize and share the data from a SWAT model, the raw temporal (time-series) and spatial (watershed features and land information layers) need to be uploaded to the PostgreSQL database and Geoserver, respectively. Thus, as part of the app code package (shown below), a python script for uploading the data is provided.



This tutorial will explain the process of preparing and uploading a new SWAT model to the app for data visualization.

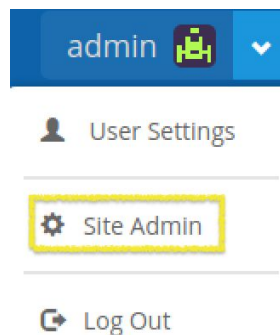
Instructions

1 Check whether SWAT Data Viewer app is installed properly

- If you have not already done it, install the SWAT Data Viewer app by following the instructions found in the “SWAT Data Viewer - Installation Guide”

2 Make sure SWAT Data Viewer app is linked to a PostgreSQL database

- In your web browser, navigate to the main page of the tethys portal that the SWAT Data Viewer app is installed on.
- Login as the administrator
- Open the Site Admin Settings



- Scroll down to the “TETHYS APPS” section and select “Installed Apps”



- Select the SWAT Data Viewer (or the custom name you gave it) from the list of installed apps
- Scroll down to the “PERSISTENT STORE DATABASE SETTINGS” and make sure the `swat_db` is linked to the Tethys portals PostgreSQL service

PERSISTENT STORE DATABASE SETTINGS						
NAME	DESCRIPTION	SPATIAL	INITIALIZED	PERSISTENT STORE SERVICE	REQUIRED	ERRORS
swat_db	Primary database for SWAT Online app.			local		

- Note: If the database was created separate from the app, the “INITIALIZED” column will be instead of . This will not cause any issues with the app however.

3 Check PostgreSQL database and table structure

When the app is installed, the PostgreSQL database and all of its tables should automatically be created. If unsure, you can check to see the different tables in the database using the PSQL feature that is built into the tethys installation.

- a. Open an SQL interface for the app's database using the following command in the hosting server's terminal

```
psql -d <database_name> -h <hostname> -p <port> -U  
<username>
```

- b. To list all the tables in the database:

```
\dt
```

- c. To list the columns in a table:

```
\d+ <table_name>
```

The following sections will outline the various tables in the swat_db PostgreSQL database that is created during the SWAT Data Viewer installation process and then used by the app to visualize and process data.

watershed

Field	Description
id	ID number that will be used throughout the other tables to ensure the app knows what data belongs to which watershed
name	Name of the watershed/SWAT model This will be the same name used for all files related to the given watershed/model

output_rch

Field	Description
watershed_id	ID number corresponding the the id field in the "watershed" table

year_month_day	The date that the value in the 'val' field represents
reach_id	ID of the stream reach within the watershed Should match the 'subbasin' field in the stream shapefile
var_name	Variable name that the value in the 'val' field represents
val	Data value for a given reach_id, date, and variable combination

output_sub

Field	Description
watershed_id	ID number corresponding the the id field in the "watershed" table
year_month_day	The date that the value in the 'val' field represents
sub_id	ID of the subbasin within the watershed Should match the 'subbasin' field in the Subbasin shapefile
var_name	Variable name that the value in the 'val' field represents
val	Data value for a given subbasin_id, date, and variable combination

watershed_info

Field	Description
watershed_id	ID number corresponding the the id field in the "watershed" table
rch_start	First date that rch data is available for a given watershed (obtained from output_rch table)
rch_end	Last date that rch data is available for a given watershed (obtained from output_rch table)
rch_vars	Available rch variables for a given watershed (obtained from output_rch table)
sub_start	First date that sub data is available for a given watershed (obtained from output_sub table)
sub_end	Last date that sub data is available for a given watershed (obtained from output_sub table)
sub_vars	Available sub variables for a given watershed (obtained from output_sub table)
lulc	Was there a lulc map provided for the given watershed? ('Yes'/'No')

soil	Was there a soil map provided for the given watershed? ('Yes'/'No')
stations	Was there a gauge stations shapefile provided for the given watershed? ('Yes'/'No')
rch	Was there an output.rch file provided for the given watershed? ('Yes'/'No')
sub	Was there an output.sub file provided for the given watershed? ('Yes'/'No')
nasaaccess	Was there a DEM file provided for the given watershed to use for nasaaccess processing? ('Yes'/'No')

lulc

Field	Description
watershed_id	ID number corresponding the the id field in the "watershed" table
value	Unique value in the lulc TIFF file that corresponds to a land use/land cover type
lulc	Four character classification of the lulc type (i.e. RICE, WATR)
lulc_class	lulc class name (i.e. forest)
lulc_subclass	Lulc subclass name (i.e. 'evergreen' and 'deciduous' would be subclasses of the 'forest' class)
class_color	Color that the app will use to display a given lulc class' coverage percentage in a pie chart
subclass_color	Color that the app will use to display a given lulc subclass' coverage percentage in a pie chart and that will correspond to the display of the lulc map (This color on the pie chart should match the color on the map)

soil

Field	Description
watershed_id	ID number corresponding the the id field in the "watershed" table
value	Unique value in the soil TIFF file that corresponds to a land use/land cover type
soil_class	Soil class name
class_color	Color that the app will use to display a given lulc class' coverage percentage in a pie chart and that will correspond to the display of the soil map (This color on the pie chart should match the color on the map)

stream_connect

Field	Description
watershed_id	ID number corresponding the the id field in the “watershed” table
stream_id	ID of a stream in the stream shapefile
to_node	ID of the stream downstream of the given stream_id

4 Prepare SWAT inputs/outputs for upload

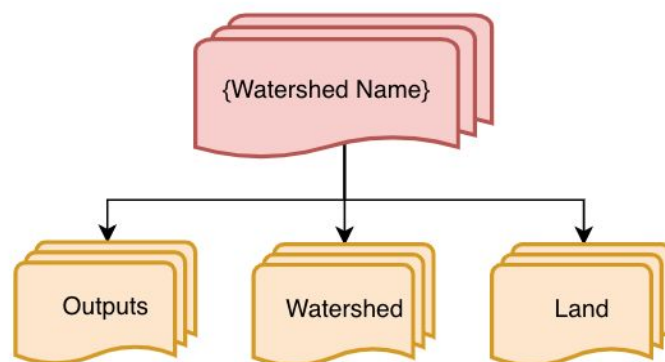
For the SWAT Data Viewer application to function fully, there needs to be at least one SWAT model available for the app to view. All of the data/files that the app uses will be located in one of four locations: (1) File on the server, (2) PostgreSQL database, (3) Geoserver, and (4) the app’s workspace (app_workspace folder in the app package). The sections below describes the various files/data that the app is able to process, where they needs to be located, and whether the file is required for the app to run.

Data file structure

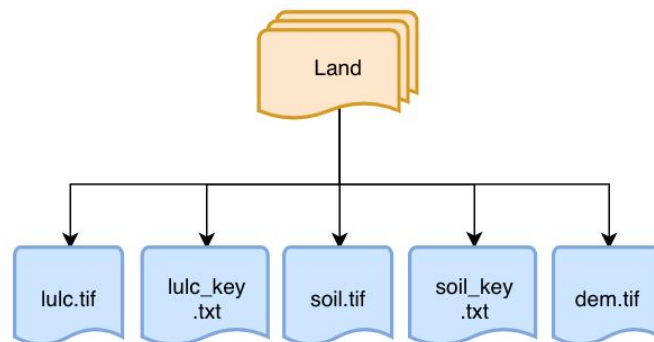
The data will need to be organized in the following file structure for the upload_new_model.py functions to work properly.

Top Level

- Create a new folder that will contain all of the data and name it the same name that you want to show in the app
- Create 3 subfolders in the <watershed_name> folder and name them “Land”, “Outputs”, and “Watershed”



Land folder



Data/File	Description	Locations	Req'd?
Land use/Land cover (lulc) map with lookup table	TIFF file containing lulc information over the watershed Comma delimited .txt file containing lulc classes (i.e agriculture), subclasses (i.e. rice crops), and hexadecimal color codes for each value in the lulc TIFF file	Geoserver PostgreSQL File on server	No
Soil map with lookup table	TIFF file containing soil information over the watershed Comma delimited .txt file containing soil type names and a hexadecimal color code for each value in the soil TIFF file	Geoserver PostgreSQL File on server	No
DEM	TIFF file containing elevation data over the watershed to be used when users want to run nasaaccess functions from the SWAT data viewer	File on server PostgreSQL	No

lulc_key.txt

lulc_key.txt is a comma-delimited lookup table that the app will use to compute and display coverage statistics for the SWAT model's land use/land cover map. The information in this file will be uploaded to the PostgreSQL database.

Value	Landuse	lulc_class	lulc_subclass	class_color	subclass_color
Integer code used in lulc.tif pixels	4-char class code used by SWAT	lulc class name (i.e. forest)	lulc subclass name (i.e. deciduous forest)	Hex color code for lulc_class	Hex color code for lulc_subclass


```

Value, Landuse, lulc_class, lulc_subclass, class_color, subclass_color
10, WATR, water, water, #0055ff, #0055ff
15, SWRN, barren, barren - rock outcrops, #91522f, #91522f
16, URBN, urban, urban, #ff0000, #ff0000
21, RICE, agriculture, rice - 1 crop/year, #ff7070, #ffadad
22, RISU, agriculture, rice - 2 crops/year, #ff7070, #ff54f0
23, SUGC, agriculture, mixed annual crops - other than rice, #ff7070, #ffe100
24, RNGE, agriculture, shifting cultivation - cleared before 2010-herbaceous cover, #ff7070, #ffae00
25, HAY, agriculture, shifting cultivation - cleared in 2010, #ff7070, #912f08
26, HAY, agriculture, shifting cultivation - partially cleared in 2010, #ff7070, #593200
31, RNGB, forest, deciduous shrubland broadleaved, #1a8e16, #ed6600
32, FRSD, forest, deciduous broadleaved, #1a8e16, #a0d89e
33, FRST, forest, deciduous/evergreen, #1a8e16, #269924
34, FRSE, forest, evergreen broadleaved, #1a8e16, #013d00
35, FRST, forest, evergreen/deciduous broadleaved, #1a8e16, #026801
36, RNGB, forest, bamboo scrub/forest mostly evergreen, #1a8e16, #99ff00
41, RNGE, grassland, grassland - sparse vegetation, #969696, #969696
42, FRST, forest, industrial forest plantation, #1a8e16, #8300db
43, WETF, wetland, shrubland/herbaceous riparian areas, #0092ed, #0092ed
127, NONE, NoData, NoData, #000000, #000000

```

Note: Be sure to include any NoData values in the list.

soil_key.txt

soil_key.txt is a comma-delimited lookup table that the app will use to compute and display coverage statistics for the SWAT model's soil map. The information in this file will be uploaded to the PostgreSQL database.

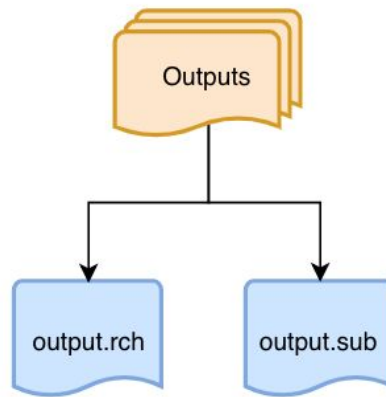
Value	Name	Color
Integer code used in soil.tif pixels	Soil type name	Hex color code for the given soil type

```

Value, Name, color
4260, AfAfBfAoQ1A, #fff500
4261, AfAfApGdAg, #ff3d0c
4264, AgLgAoSgGdAp, #5788f2
4265, AgAgLgBfAfWdGd, #ff7200
4267, AoNdAfLBf, #671dbc
4284, AoAoLBdAh, #23a274
4325, GeGmJeGh, #ff8e8e
4383, ILcBk, #c5ff7a
4393, JeGeGm, #f4d83f
4408, LgLoGeJeAg, #79dbff
4414, NhNhAoGAp, #eb984e
4452, AfAfAoAg, #0c65ff
4464, AoNdAfVcAg, #b23399
4486, FaLFaf, #af9a2f
4491, FoPhGAf, #aa2e4b
4494, FrFoVNdL, #6f48a5
4499, GdJeGmGh, #4fb7cc
4502, GeJeGm, #cc894b
4544, NdBdVcLE, #772d3b
4587, VpVcLGF87, #e2b7bf
4588, VpVcLGF88, #d5d800
11375, Cnd, #424c68

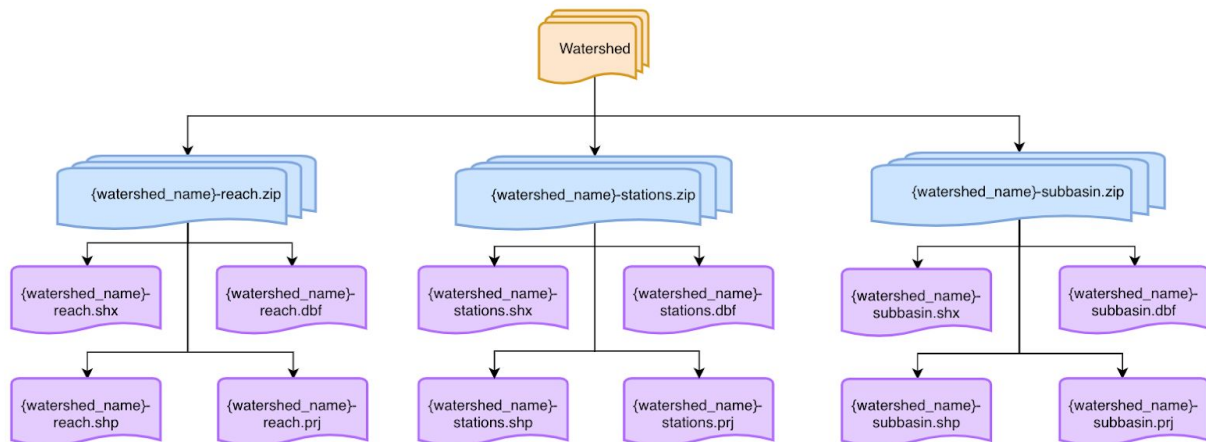
```

Outputs folder



Data/File	Description	Locations	Req'd?
output.rch and output.sub	Daily SWAT model output files with IDs that correspond to the "subbasin" fields in the stream and subbasin shapefiles	PostgreSQL	At least one of them

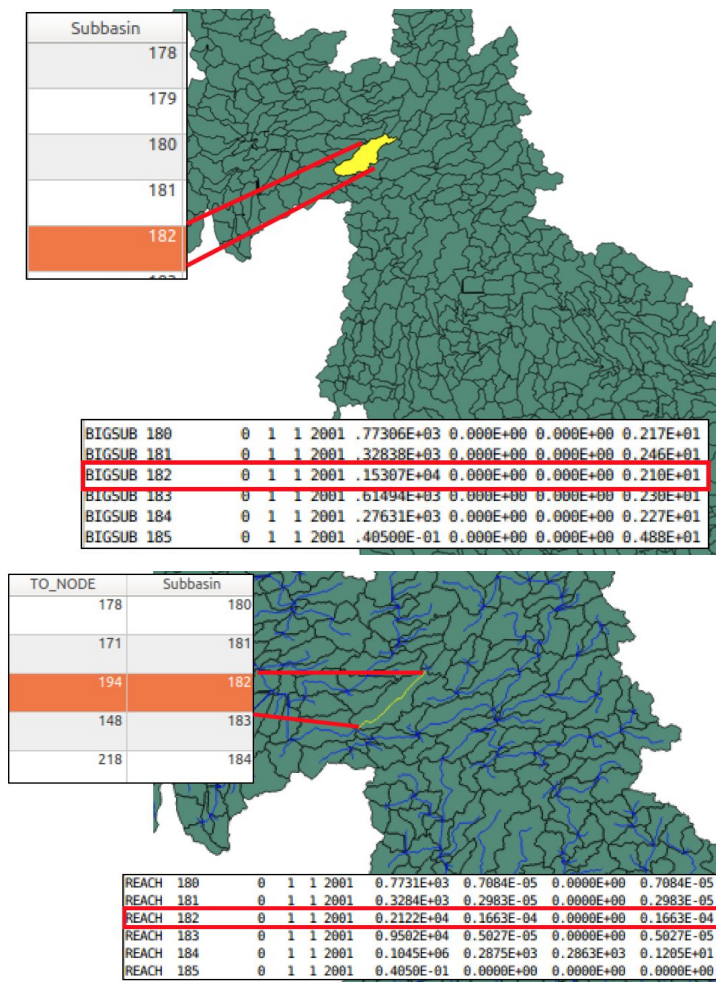
Watershed folder



Data/File	Description	Locations	Req'd?
Stream Shapefile (.zip file containing .shp, .shx, .dbf, and .prj files)	Polyline shapefile containing stream reach information. Required attribute fields: "subbasin", "TO_NODE"	Geoserver PostgreSQL	Yes

Subbasin Shapefile (.zip file containing .shp, .shx, .dbf, and .prj files)	Polygon shapefile containing subbasin information Required attribute fields: "subbasin"	Geoserver	Yes
Gauge stations Shapefile (.zip file containing .shp, .shx, .dbf, and .prj files)	Point shapefile containing gauge station information	Geoserver PostgreSQL	No

- i. The attribute table for the both the subbasin and reach shapefiles need to have a field called "subbasin" that has a value that correspond with each other.
- ii. The attribute table for the reach shapefile needs to have a field called "TO_NODE" that identifies the subbasin id of the next downstream reach



5 Copy data over to the app's hosting server/computer

If you prepared the data elsewhere, copy the entire data folder to the app's hosting server/computer.

Some of the data (the Land folder) will need to remain saved on that machine even after being uploaded to the geoserver/database and will need to be located in the “swat_data” folder. It’s recommended to put the data directly in that folder from the beginning to avoid confusion.

5 Update input arguments in upload_new_model.py

The upload_new_model.py file contains 9 separate functions that are called based on data availability. The first function called (check_available_files()) opens the data folder, makes sure all required data is present and then lists all of the available data in the folder. The other functions will only be called if the data file it corresponds to is in the data folder.

This logic is executed through the following python code.

```
if new_watershed(db, watershed_name) == 0:
    available_files = check_available_files(watershed_name, data_path)
    if available_files != 1:
        upload_swat_outputs(db, os.path.join(data_path, 'Outputs'), watershed_name, sub_vars, rch_vars)
        upload_shapefiles(geoserver, os.path.join(data_path, 'Watershed'))
        upload_stream_connect(db, os.path.join(data_path, 'Watershed'), watershed_name)
        upload_tiff_files(geoserver, os.path.join(data_path, 'Land'), watershed_name)
        if 'lulc_key.txt' in available_files['Land']:
            upload_lulc_key(db, os.path.join(data_path, 'Land'), watershed_name)
        if 'soil_key.txt' in available_files['Land']:
            upload_soil_key(db, os.path.join(data_path, 'Land'), watershed_name)
        watershed_info(watershed_name, available_files, sub_vars, rch_vars)
    print('SUCCESS: Upload Complete!')
```

Before executing the upload_new_model.py script, there are a number of input parameters that need to be changed in the file.

- a. Change watershed_name to the match the name of the data folder (the name you want the app to display for the watershed model)
 - i. **This name should have no spaces and should be all lowercase.** Use ‘_’ instead of spaces. The app will convert this to Title case with spaces when displaying it.
- b. Specify the file path to the data folder (to the top level folder)
 - i. If the folder is structured correctly according to step 4, the functions will know how to access all the necessary files
- c. Specify the output.sub variables that you want to upload to the database
 - i. The sub_column_list contains all of the variable codes in the output.sub file
 - ii. Be sure that the variables you list here match the codes in sub_column_list
 - iii. If you’re not uploading any output.sub data, you can leave this blank (i.e. sub_vars = “)
- d. Specify the output.rch variables that you want to upload to the database
 - i. The rch_column_list contains all of the variable codes in the output.rch file

- ii. Be sure that the variables you list here match the codes in `rch_column_list`
 - iii. If you're not uploading any output.rch data, you can leave this blank (i.e. `rch_vars = ""`)
- e. Make sure the db object contains the correct information for connecting to the app's PostgreSQL database
- f. Make sure the geoserver object contains the correct information for connecting to the app's geoserver

6 Run `upload_new_model.py`

- a. Open the terminal for the server/computer containing the SWAT Data Viewer app package and data folder
- b. Navigate to the app package folder containing the `upload_new_model.py` script
- c. Activate the 'tethys' conda environment
- d. Run the script

```
python upload_new_model.py
```

7 Delete data files from server

If space is limited on your server/computer, you can delete most of the data files after the script has completed.

- a. The only files that need to remain on the server are:
 - i. `lulc.tif`
 - ii. `soil.tif`
 - iii. `dem.tif`
- b. If you haven't done so already, move the data folder to the `swat_data` folder so that the app can find the TIFF files for further processing

