To setup automated emails, 3 components are required:

1. Django-celery
   - An asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well
2. Redis
   - Acts as a celery "broker": passes messages between a Django project and Celery workers
   - it is a "data structures server"
   - i.e. not a plain-key value store
   - Instead of storing string key – string value pairs, it can store other data types as values (not just strings, but other data structures: lists, sets, hashes, etc)
3. Supervisor
   - Supervisor is a client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems

- **If Redis is not installed system-wide on a given computer, you must install it:**
  - (OSX): `brew install redis`
    - Note this requires use of homebrew
  - (Ubuntu): `sudo apt-get install redis-server`
- Each requires a python package that is already included in `sails_requirements.txt`
  - must use `pip install` while SaILS virtual environment is active
- Configuration instructions for Redis and Supervisor are provided for OSX and Ubuntu:

<u>**Redis**</u>

1. **OSX**
   - To run in background (port 6379), automatically upon startup of server:
     - `ln -sfv /usr/local/opt/redis/*.plist ~/Library/LaunchAgents`
     - Can also run it in the foreground using:
       - `redis-server`
   - Note the config file is located at:
     - `user/local/etc/redis.conf`
   - Check that redis is running via its command line interface:
     - `redis-cli`
     - `PING`
       - should return `PONG`
     - `CTRL-C` (to close command line interface)
2. **Ubuntu**
   - To run in background (port 6379), automatically upon startup of server:
     - `sudo update-rc.d redis-server default`
     - Can also run it in the foreground using:
       - `redis-server`
   - Note the config file is located at:
     - ?
   - Check that redis is running via its command line interface:
     - `redis-cli`
     - `PING`
       - should return `PONG`
     - `CTRL-C` (to close command line interface)

<u>**Supervisor**</u>

1. **OSX & Ubuntu**
   - Create a supervisor directory:
     - `mkdir /etc/supervisor`
   - Create a sub-supervisor directory:
     - `mkdir /etc/supervisor/conf.d`
   - Create a directory for log output
     - `mkdir /var/log/supervisor`
   - Create a supervisor config file (copy existing one provided with SaILS):
     - **(OSX):** `cp /path/to/sails/project/sails_new/ils/supervisor/supervisor_osx.conf /etc/supervisor/supervisord.conf`

        - **(Ubuntu):** `cp /path/to/sails/project/sails_new/ils/supervisor/supervisor_ubuntu.conf /etc/supervisor/supervisord.conf`
    - Create config files for the celery scheduler and worker (copy existing ones provided with SaILS)
        - `cp /path/to/sails/project/sails_new/ils/supervisor/sails_celery.conf /etc/supervisor/conf.d/`
        - `cp /path/to/sails/project/sails_new/ils/supervisor/sails_celerybeat.conf /etc/supervisor/conf.d/`
    - Make the following changes in both: `/etc/supervisor/conf.d/sails_celery.conf` **AND** `/etc/supervisor/conf.d/sails_celerybeat.conf`:
        - `environment:`
            - If using **Ubuntu**, use `"ils.dev_settings"` instead of `ils.dev_settings`
            - If creating a production version, use `ils.prod_settings` instead of `ils.dev_settings`
        - `command:`
            - Update the absolute path to the virtual environment used for SaILS
        - `directory:`
            - Provide the absolute path to the directory containing `manage.py` of SaILS
    - Create the log files:
        - `touch /var/log/supervisor/sails_worker.log`
        - `touch /var/log/supervisor/sails_beat.log`
2. **To test, can run in the foreground:**
    - activate your virtual environment for SaILS
    - start supervisor:
        - `supervisord -c ~/etc/supervisor/supervisord.conf`
    - Register the celery `.conf` files:
        - `supervisorctl -c ~/etc/supervisor/supervisord.conf reread`
        - `supervisorctl -c ~/etc/supervisor/supervisord.conf update`
    - To actually start/stop/check status of the celery worker and scheduler:
        - `supervisorctl -c ~/etc/supervisor/supervisord.conf [start|stop|status] [sailscelery|sailscelerybeat]`
3. **OSX: Run in background, automatically upon startup of server**
    - Create the launch Daemon file (copy existing file provided with SaILS)
        - `cp /path/to/sails/project/sails_new/ils/supervisor/com.agendaless.supervisord.plist /Library/LaunchDaemons/`
    - In `/Library/LaunchDaemons/com.agendaless.supervisord.plist` replace the following line with the path to your SaILS virtual environment:
        - `<string>/path/to/sails_nsir/venv/bin/supervisord</string>`
    - Add it to launch scripts:
        - `launchctl load /Library/LaunchDaemons/com.agendaless.supervisord.plist`
    - Once server has restarted, you can use the commands provided in (2) to check its status
        - **Note:** you must use `sudo` with these commands when supervisor was launched in the background
4. **Ubuntu: Run in background, automatically upon startup of server**
    - Create the launch Daemon file (copy existing file provided with SaILS)
        - `cp /path/to/sails/project/sails_new/ils/supervisor/supervisor /etc/init.d/`
    - Add it to launch scripts:
        - `sudo update-rc.d supervisor default`
    - Once server has restarted, you can use the commands provided in (2) to check its status
        - **Note:** you must use `sudo` with these commands when supervisor was launched in the background