MORE Deployment Guide

This guide applis for both SERF and ReportWriter for most of the content, just repeat the procedure to deploy both websites

1. Deploy On Heroku

Step-by-step setup for your Heroku app could be found in following two pages:
Heroku Python App Tutorial
Heroku Django Config

Environment Setting:
In Heroku Settings, add following Config Vars:

- EMAIL_HOST (smtp.gmail.com)
- EMAIL_HOST_USER
- EMAIL_PASSWORD
- EMAIL_PORT (587)
- RECAPTCHA*PRIVATE*KEY
- RECAPTCHA*PUBLIC*KEY
- IDENTIFIER (for reportwriter application, each instance of reportwriter owns a unique identifier)

Create 2 apps, one for SERF(Enhanced-CWE), the other for ReportWriter.
SERF code is on heroku-serf branch, ReportWriter code is on heroku-reportwriter branch. (two branch only differ in Procfile, both contains all codes, better to split into 2 repositories later)

After run `git push [remote-repo-in-heroku] [local-branch]:master`, you need also run `heroku run bash --remote [remote-repo-in-heroku]` and run migrate/createsuperuser/collectstatics commands as Django application usually requires. See How to deploy different application for different branch if you are not clear how to do so.

You should be able to see the website running now, to connect SERF with ReportWriter, jump to section 3

2. Deploy on Own Server

First Switch to master branch if you would like to deploy on your own server

Database

Make sure database is correctly setup in the machine first. You can use either PostgreSQL or SQLite.

Steps to setup PostgreSQL is shown here:

- Follow instructions at PostgreSQL setup on Ubuntu to install PostgreSQL
- Create User and Database for our website

```
$ sudo -i -u postgres
$ psql
postgres=# CREATE USER [user] WITH PASSWORD '[pass]';
postgres=# CREATE DATABASE serf;
postgres=# CREATE DATABASE reportwriter;
postgres=# GRANT ALL PRIVILEGES ON DATABASE serf to user;
postgres=# GRANT ALL PRIVILEGES ON DATABASE reportwriter to user;
postgres=# \q
```

- Modify setting in Django App to connect to DB

  In *Deploy/config.ini*, modify DATABASE module and set user/pass to the user and password you added for database, let SERF connect to serf DB, ReportWriter connect to reportwriter DB.

Python

MORE website framework uses Django 1.8 (some features used are no longer supported after Django 1.9, unfortunately), and we assume Python 2.7 (Python3 not supported)

- Python Virtual Environment Setup: see Python VirtualEnv Guide how to install virtualenv and virtualenvwrapper to create the Python env for this website
- Dependencies: switch to Deploy directory and execute

  `pip install -r requirements.txt`

  In Ubuntu system, psycopg2 might not be correctly installed by pip, if so, use apt-get install python-psycopg2 to resolve this issue

  In django-simple-history library, there is a bug need to fix (already fixed in github latest version, but not reflected in stable release version), to manually fix this, go to [python-site-package-path]/simple_history/models.py and change
  `from django.utils.translation import string_concat, ugettext as _` *into*

```
from django.utils.translation import string_concat, ugettext_lazy as _
```

Django Setup

Before we can run the website, there are some more setting changes:

- In *Deploy/config.ini*, change Email Settings module to an email account you own

  This email account is used to send confirmation emails to users, so just use a temporary email if you are still developing

- In *Deploy/config.ini*, change RECAPTCHA module, use your google account to register RECAPTCHA service and get public/private key pair and replace the keys here

- You may want to change DEBUG to true temporarily during development to find out errors, but **REMEMBER to change it back to false before deployment**

Now your website should be able to run now

```
$ python manage.py migrate
$ python manage.py createsuperuser # create administrator
$ python manage.py runserver 0.0.0.0:8000
```

You can browse the website in your browser and login using administrator credential now

Web Server Setup

If you need to deploy your website instead of just development, it is necessary to setup a web server (Apache or Nginx)

To setup Nginx, reference:

- Setup Nginx and uwsgi in your machine

- Switch to Deploy repository. Open EnhancedCWE_nginx.conf and ReportWriter_nginx.conf, change directory prefix to your own directory

- Link the conf file to Nginx conf file

  ```
  $ sudo ln -s [Path to Deploy]/EnhancedCWE_nginx.conf /etc/nginx/sites-enabled/
  $ sudo ln -s [Path to Deploy]/ReportWriter_nginx.conf /etc/nginx/sites-enabled/
  # Reload Nginx
  ```

- Create the socket file used to reverse proxy from Nginx to uwsgi.
- Modify repository prefix in ini files and run uwsgi daemon process.

```
$ uwsgi --ini EnhancedCWE_uwsgi.ini
$ uwsgi --ini ReportWriter_uwsgi.ini
```

Now the web server shall be up and everything running in background. You can safely leave the machine running and demonstrate the website to others.

3. Setup Token for Website

To setup REST API token to enable communication from ReportWriter to SERF:

- In SERF dashboard, open Authtoken setting, and assign a token for one of the user you want to use to represent reportwriter's permission
- In ReportWriter dashboard, open REST API setting, enter the token you get in previous step, and API base URL is [SERF base URL]/api/v1/
- Check that when you add report in ReportWriter, you can see CWE search result retrieved from SERF

To setup REST API token to enable communication from SERF to ReportWriter:

- In ReportWriter dashboard, open Authtoken setting, and assign a token for one of the user you want to use to represent serf's permission
- In SERF dashboard, open REST API setting, add a new one, enter the token you get in previous step, the name is the unique identifier you assigned to your ReportWriter before, and API base URL is [ReportWriter base URL]/api/v1/
- Check that after you promote MUO to SERF and you add advice for that MUO, you can see the advice is synchronized back to original ReportWriter report.