Pew EFH SysAdmin Guide

Need-To-Know information for managing your Essential Fish Habitat Site and Server.
If you'd like further documentation on the project, please see the other documents here:
https://github.com/Ecotrust/PEW-EFH/tree/master/docs/Handoff

Login/Accounts

As the Systems Administrator, you will need access to the following systems. Credentials, urls, and keyfiles for these should all have been included in the share folder.

The Root Amazon Web Services dashboard

This dashboard is used for the following:

- Starting/Stopping the server
- Checking on backup status, or creating new snapshots
- Managing the Firewall
- Managing the billing
- Setting IP address (AWS Elastic IP)

You can of course do a whole lot more with this dashboard such as load balancing, monitoring, creating user accounts (IAM Roles), spinning up new servers, resizing the server, and many many other things, but the above list covers what services are currently being used.

SSH Server Access

The server runs Ubuntu 18.04 LTS. The interface is command-line only.
Access is via ssh keys only - no passwords.
Review the provided login materials for the private SSH key and username.
Review documentation for your own system to use SSH.

Code Repository

The code for the web application is all open source and is available from Github here: https://github.com/Ecotrust/PEW-EFH

Instructions for installing the tool and its requirements are covered in a separate Installation doc, which should also be available in the File Share, but can always be found in its latest version here: https://github.com/Ecotrust/PEW-EFH/wiki/installation

Stack

- Operating System
    - Ubuntu 18.04 LTS “Bionic Beaver”
- Webserver
    - NGINX
    - uWSGI

- Email
    - Postfix
- Database
    - PostgreSQL 10
    - PostGIS 2.5 (Spatial DB extension)
- Backend Framework
    - Django 1.7 (Python)
- Frontend Frameworks
    - Knockout.js
    - Twitter Boostrap
- Webmaps
    - Openlayers 2
- Map Tile Service
    - Tilestache
    - Mapserver

Python Virtual Environment

Since the tool runs on Python, it uses a Virtual Environment to organize and contain a large number of dependency libraries. These are minor and not listed above in the stack as they are often irrelevant to system performance, but are listed here: https://github.com/Ecotrust/PEW-EFH/blob/master/requirements.txt

Running some commands will require that you 'activate' this virtual environment, which enables and enforces the use of the python libraries at the specified version for this tool. Activating your python environment is straightforward, just run the 'source' command pointed at the 'activate' binary found at /usr/local/apps/PEW-EFH/pew-env/bin/activate:

```
source /usr/local/apps/PEW-EFH/pew-env/bin/activate
```

You will know that the environment is activated by seeing the virtual environment's name in parentheses at the beginning of your prompt:

```
(pew-env) username@servername:~$
```

If you ever need to exit from the virtual environment, you can use the command 'deactivate' by itself:

```
deactivate
```

Creating Superuser Accounts

As installed, the server has a superuser account for the web tool, which the tool administrator has access to. If at any time you need to create a new superuser for the tool (to log in and chance passwords, permissions, etc…), Django provides a command-line shortcut for that. Simply:

- SSH into the server
- Activate your virtual environment (see 'Python Virtual Environment' section)
- Run the django 'create superuser' command (https://docs.djangoproject.com/en/1.7/intro/tutorial02/#creating-an-admin-user):

```
python /usr/local/apps/PEW-EFH/mp/manage.py createsuperuser
```

Then just follow the prompts to create the username and password, which can then be used to log into the tool via the website.

Tool Administration

The tool has a built-in administration console that can be found here:
http://pewmarineplanner.ecotrust.org/admin/
Documentation for using this is intended for the site administrator, but can be found here:
https://github.com/Ecotrust/PEW-EFH/blob/master/docs/Handoff/Pew%20EFH%20Site%20Administration.pdf

Patching

Security

Security patches are applied automatically by the operating system as they come out. See the documentation on Automatic Updates and Unattended Upgrades here:
https://help.ubuntu.com/lts/serverguide/automatic-updates.html.en

The server will reboot when a patch requires a reboot to be installed. The reboot isn't immediate, but is configured to wait for a specific time (so you don't reboot during peak usage). The configuration file is here: `/etc/apt/apt.conf.d/50unattended-upgrades` . It is currently set to 10:00 - since the server time is in UTC, this will be 2 or 3 AM, Pacific Time.

If you ever need to apply patches by hand, you can use the built-in package manager:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Tool Code

Should the tool's code ever receive any updates, those will be available via git (the repository tool). I recommend the following commands to apply updates:

```
cd /usr/local/apps/PEW-EFH
git pull
dj migrate
dj install_media
sudo service uwsgi restart
```

The above commands do the following:

- Change directory into a recognized git-controlled directory
- Fetch and merge any code changes from the source repository
- Perform any database changes ('migrations') that may be required to support the code updates
- Collect all static files into a single place to be served by the webserver
- Restart the python app server to recognize and serve any updated files

'`dj`' is a custom alias that is short for running `python /usr/local/apps/PEW-EFH/mp/manage.py` from within the 'pew-env' virtual environment.

Backups

Once a month a cron job will run creating a snapshot of the AWS Instance Volume and deleting any snapshots older than 90 days.

You can edit or review the cronjob with

```
sudo crontab -e
```

You can review the bash script that runs once a month here:

/usr/local/apps/PEW-EFH/scripts/auto_snapshot.sh

Monitoring

Hardware

This server has Munin installed: http://munin-monitoring.org/

You can see the output here:
http://pewmarineplanner.ecotrust.org/munin/localdomain/localhost.localdomain/index.html

Uptime

The site's uptime can be monitored in any way you like, but is not done automatically. I recommend UptimeRobot.com for a free and simple tool. AWS offers such services in the dashboard as well.

Network

Network Services

This server uses NGINX as the webserver. You can control NGINX like so:

- `sudo service nginx start`
- `sudo service nginx status`
- `sudo service nginx restart`
- `sudo service nginx stop`

uWSGI is used to serve the Python program in a way that NGINX can serve it as a website:

- `sudo service uwsgi status`
- `sudo service uwsgi start`
- `sudo service uwsgi restart`
- `sudo service uwsgi stop`

Postfix, the email server, can be managed in the same way.

Firewall

Firewall configuration is managed via the AWS dashboard.

Site Configuration Files

- NGINX:
  - /etc/nginx/sites-enabled/efh
- uWSGI
  - /etc/uwsgi/apps-available/efh.ini

Webmapping

Two services related to providing maps for websites to ingest are also running on this server:

- Tilestache

    - http://tilestache.org/
    - This service serves up “map tiles” (raster images of maps) from locally stored shapefiles (map data).
    - It also is responsible for serving up “UTF Grids” - invisible files used to enable ‘vector-like’ interaction with raster images
        - More info here: https://blog.mapbox.com/how-interactivity-works-with-utfgrid-3b7d437f9ca9
- Mapserver
    - https://mapserver.org/
    - This service is used as a WMS proxy: any WMS server that does not provide web map tiles in the format needed by our tool can be queried by MapServer, which will then on-the-fly reconfigure the input to be compatible.
    - This runs as a service on its own:
        - `sudo service mapserv status`
        - `sudo service mapserv start`
        - `sudo service mapserv restart`
        - `sudo service mapserv stop`

A third external service, MapBox (http://mapbox.com) is also used to serve and query some of the layers

Debugging

Logfiles

System: /var/log/syslog
NGINX:

- Access: /var/log/access.log
- Errors: /var/log/error.log
- Site Access: /var/log/efh.access.log
- Site Errors: /var/log/efh.error.log

uWSGI logs:

- EFH MarinePlanner: /var/log/uwsgi/app/efh.log
- Tilestache: /var/log/uwsgi/app/tilestache.log

Site Is Not Loading

- Is the server running?
    - Check the AWS Dashboard -> EC2 -> Instances
        - Use the GUI to start the server if not
- Is the Webserver running?
    - `sudo service nginx status`

        - Use `sudo service nginx start` if not
- Is the python project server running?
    - `sudo service uwsgi status`
        - Use `sudo service uwsgi start` if not

Maps are not loading

There are MANY sources for maps, so it's best to work with the site administrator to sort out the following:

- Basemap or Overlay?
    - Basemap: If other layers load, the service for the basemap is most likely slow or down. Not much can be done about this - try using a basemap from a different source in the meantime
    - Overlay - see following questions
- User-provided Data layer?
    - If the layer is a user's drawing, a scenario proposal, results from grid filtering, or a user-imported layer, then the problem has to do with reading the data from the database and serving it to the front end.
        - If the data is complex:
            - If the user imported a very large or complex layer (more than 10Mb) then the data may just be too complex to be handled within the browser's memory. It is recommended that this file be deleted and a smaller/simpler one put in its place
        - If no user layers are showing:
            - Check that the database is running:
                - `sudo service postgresql status`
            - Like other services we've seen in this doc, the following commands work as well:
                - `sudo service postgresql stop`
                - `sudo service postgresql start`
                - `sudo service postgresql restart`
- Local or Remote URL?
    - Have the site administrator find the layer that is not loading and get the 'URL' field from the record.
        - Layers can be browsed, searched, and ordered here: http://pewmarineplanner.ecotrust.org/admin/data_manager/layer/
        - If the layer's url starts with either "/" or "http://pewmarineplanner.ecotrust.org" it is served by this server:
            - If the URL starts with '/tiles/' then Tilestache is serving them. It should work so long as NGINX is running
            - If the URL start with '/mapserver/' then it is using the MapServer proxy:

        - Test that MapServer is working:
            - sudo service mapserv status
            - http://pewmarineplanner.ecotrust.org/mapserver/
                - If this returns the message “No query information to decode. QUERY_STRING is set, but empty” then mapserver is running fine.
        - Test that the requested service is working:
            - If the URL contains ‘&CONN=’ then grab the url between ‘&CONN=’ and the next ‘&’ and put it in a browser to see if it is responsive.
                - If not, then the service is unavailable and cannot be helped
                - If so, then it is likely the admin settings are wrong, and the site administrator will have to debug their settings
    - If the layer begins with an external url (“http://….” or “https://…”)
        - If the URL begins “https://api.mapbox.com...”
            - Most likely this is a layer configured in MapBox studio by this project team for this project.
                - Check the url in a browser to see if the service is alive
                - Check the mapbox account to see that the url, MapBox Access Token, and MapBox Tileset ID (if given) are correct.
                - Setting these appropriately is covered in the Site Administrator Guide.
        - ELSE: Put the URL in a browser and see if the service on the other end is responsive.
            - If not, then it is down and nothing can be done on this end
            - If so, then the site administrator needs to review and fix their settings.
- No maps are loading:
    - Check if the user has javascript blocked - javascript must be enabled for the tool to work
    - Check that the user still has an internet connection
    - Check that NGINX is running
    - Check that the tool has collected all of the static files to be served by NGINX
        - Do this by running `dj install_media`, typing ‘yes’ when prompted