Installing Pew EFH MarinePlanner

Server Details:

- OS: Ubuntu Bionic Beaver 18.04LTS
- RAM: 4GB recommended
- CPUs: 2+ recommended
- Disk: 25+ Gb recommended

For latest “live” view of this document, please check out:
https://github.com/Ecotrust/PEW-EFH/wiki/installation

Installation Process:

Update the server software

```
sudo apt-get update
sudo apt-get upgrade
```

Install Git, Python, and Virtualenv

```
sudo apt-get install git vim python2.7-dev python-pip python-virtualenv -y
```

Install PostgreSQL 10

```
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
sudo apt install postgresql postgresql-contrib postgresql-10-postgis-2.5
postgresql-server-dev-10 --force-yes -y
```

Install Pillow Dependencies (Image support)

```
sudo apt-get install libtiff5-dev libjpeg8-dev zlib1g-dev libfreetype6-dev liblcms2-dev libwebp-dev tcl8.6-dev tk8.6-dev python-tk -y
```

A Few Additional Dependencies…

```
sudo apt-get install libcurl4-openssl-dev libssl-dev -y
```

Check out the code

```
sudo mkdir /usr/local/apps/
cd /usr/local/apps/
git clone https://github.com/Ecotrust/PEW-EFH.git
```

Install ElementTree:

```
cd /tmp
wget http://effbot.org/media/downloads/elementtree-1.2.6-20050316.tar.gz
tar -xvzf elementtree-1.2.6-20050316.tar.gz
cd /tmp/elementtree-1.2.6-20050316
python setup.py install
```

Geos Version Fix

Hack to support old GeoDjango on new servers (thanks to nachopro):

```
pip install "geos==0.2.1" vim /usr/local/apps/PEW-EFH/pew-env/lib/python2.7/site-packages/django/contrib/gis/geos/libgeos.py
```

- Find the line
    - `ver = geos_version().decode()`
- Edit it to read
    - `ver = geos_version().decode().split(' ')[0]`

Create and Set Local Settings

- Create /usr/local/apps/PEW-EFH/mp/settings_local.py
    - `cp /usr/local/apps/PEW-EFH/mp/settings_local.py.template /usr/local/apps/PEW-EFH/mp/settings_local.py`
- Update the settings for your new installation
    - SECRET_KEY = ‘something new’
    - DATABASES
        - 'NAME': 'efh'
        - 'USER': 'efhdbuser'
            - create this in following step
        - 'HOST': 'localhost'
        - 'PASSWORD': 'efhdbpassword'
            - create this in following step
    - MEDIA_ROOT = '/usr/local/apps/PEW-EFH/mediaroot'
    - MEDIA_URL = '/media/'
    - SOCKET_URL = ''
    - SOCIAL_AUTH_GOOGLE_PLUS_KEY = ''
    - SOCIAL_AUTH_GOOGLE_PLUS_SECRET = ''
    - ADMIN_MEDIA_PATH = '/usr/local/apps/PEW-EFH/pew-env/lib/python2.7/site-packages/django/contrib/admin/static/admin/'

Update Provisioning Script

- edit /usr/local/apps/PEW-EFH/scripts/provision.sh
    - PROJECT_NAME="PEW-EFH"
    - USER="postgres"
    - PROJECT_DIR=/usr/local/apps/PEW-EFH/mp
    - VIRTUALENV_DIR=/usr/local/apps/PEW-EFH/pew-env
    - APP_DB_NAME="efh"
    - check `CREATE DATABASE` and `\connect` values (should be `$APP_DB_NAME`)
    - any remaining 'su - vagrant' instances must be replaced with 'bash'

Create Database User and Password

- `sudo su postgres`

- `createuser -P efhdbuser`
    - from settings above, same with the password that you are prompted for
- `exit`

Add db user to pg_hba.conf:

- For PostgreSQL 10:
    - `sudo vim /etc/postgresql/10/main/pg_hba.conf`
- Else:
    - `sudo vim /etc/postgresql/{YOUR_VERSION}/main/pg_hba.conf`
- add lines:
    - `host efhdbuser efh 127.0.0.1/32 trust`
    - `host efhdbuser efh ::1/128 trust`
- save and close
- restart postgres: `sudo service postgresql restart`

Run The Provisioning Script

```
sudo su root
/usr/local/apps/PEW-efh/scripts/provision.sh
exit
```

Data Preparation

```
sudo chown -R ubuntu /usr/local/apps/PEW-EFH/mp/logs/
```

- Where **ubuntu** is the username you are using during the install.

```
dj syncdb

dj migrate
```

Loading Data Into Database

If you have a DB dump of an existing server do the following (assuming postgres is db user):

```
sudo su postgres
dropdb efh
```

```
psql
```

Then into the SQL prompt, follow along with the provision script (assuming 'efh' for db name):

```
CREATE DATABASE efh;
\connect efh
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;

insert into spatial_ref_sys (srid, auth_name, auth_srid, srtext, proj4text)
values (99996, 'EPSG', 99996, 'Marco Albers', '+proj=aea +lat_1=37.25 +lat_2=40.25
+lat_0=36 +lon_0=-72 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs');

CREATE OPERATOR CLASS gist_geometry_ops
FOR TYPE geometry USING GIST AS
STORAGE box2df,
OPERATOR        1       <<  ,
OPERATOR        2       &<  ,
OPERATOR        3       &&  ,
OPERATOR        4       &>  ,
OPERATOR        5       >>  ,
OPERATOR        6       ~=  ,
OPERATOR        7       ~   ,
OPERATOR        8       @   ,
OPERATOR        9       &<| ,
OPERATOR        10      <<| ,
OPERATOR        11      |>> ,
OPERATOR        12      |&> ,

OPERATOR        13      <-> FOR ORDER BY pg_catalog.float_ops,
OPERATOR        14      <#> FOR ORDER BY pg_catalog.float_ops,
FUNCTION        8       geometry_gist_distance_2d (internal, geometry, int4),

FUNCTION        1       geometry_gist_consistent_2d (internal, geometry, int4),
FUNCTION        2       geometry_gist_union_2d (bytea, internal),
FUNCTION        3       geometry_gist_compress_2d (internal),
```

```
FUNCTION        4        geometry_gist_decompress_2d (internal),
FUNCTION        5        geometry_gist_penalty_2d (internal, internal, internal),
FUNCTION        6        geometry_gist_picksplit_2d (internal, internal),
FUNCTION        7        geometry_gist_same_2d (geom1 geometry, geom2 geometry,
internal);
\q
```

go back to your main user (ubuntu most likely) and load the DB, followed by some quick setup on it

```
psql -U postgres -d efh -f /path/to/your/db_dump.sql
python /usr/local/apps/PEW-EFH/mp/manage.py syncdb
python /usr/local/apps/PEW-EFH/mp/manage.py migrate
python /usr/local/apps/PEW-EFH/mp/manage.py enable_sharing --all
```

Base data and Scenario Cell Planning Units (for filtering, scenario.models.GridCell)

If you are starting your database from scratch (you did not just do the load DB from SQL step) the you'll want to prime your database with the following:

```
dj loaddata /usr/local/apps/PEW-EFH/mp/fixtures/layer-data.json

dj loaddata /usr/local/apps/PEW-EFH/mp/fixtures/project-settings.json

dj loaddata /usr/local/apps/PEW-EFH/mp/fixtures/scenario_species.json

dj loaddata /usr/local/apps/PEW-EFH/mp/fixtures/scenarios_pu.json
```

Detailed Aggregate Planning Unit Layer Import (drawing.models.GridCell)

Get the SQL file created in the first steps of this process and put it somewhere handy.

Following along, use these variations on those steps to load the data:

`psql -U postgres -d efh -f /usr/local/apps/PEW-EFH/data/PU_EFH_Metrics.sql` <-replace file and path with your own

```
dj install_media
```

use `djrun` to test that things are working as expected

Additional Installation Considerations/Configuration:

Swap Space

- Most servers run SSDs now - swap is bad for SSD lifespans, and is not recommended. If you are on a spinny disk or absolutely need the help with Memory, go ahead and enable swap space now:

```
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo cp /etc/fstab /etc/fstab.bak
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab/swapfile none swap sw 0 0
```

This gives you 1G swap space (line 1) and then enforces that the swapfile is assigned at boot so you don't have to do this every time.

EMAIL

Email is used by the server primarily for helping registered users remember their usernames and passwords. It is also used for sending ‘feedback’ from users.

```
sudo apt-get install postfix
sudo apt-get install postgrey
```

edit `/usr/local/apps/PEW-EFH/mp/settings_local.py`:

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'localhost'
EMAIL_PORT = 25
EMAIL_HOST_USER = ''
```

```
EMAIL_HOST_PASSWORD = ''
EMAIL_USE_TLS = False
DEFAULT_FROM_EMAIL = 'Pew Marine Planner <noreply@pewmarineplanner.ecotrust.org>'
```

- Edit the HOST_USER and HOST_PASSWORD as necessary.
- Be sure that the domain for the DEFAULT_FROM_EMAIL matches that given when installing postfix.
    - If not (or you don't recall) you can edit `/etc/postfix/main.cf` and set your `myhostname` field to be the same domain.
- SPF: You may wish to update your DNS records to manage for SPF

Monitoring

```
sudo apt-get install munin munin-node
```

Tilestache

In an effort to support as many found layers as possible, early efforts to run a WMS locally were made before MapBox Studio offered flexible and affordable (free) plans. The process was arduous, including updating/reprojecting the shapefile, cobbling together several tools (old MapBox tools like TileMill and TileStream) to generate XML style files, and then modifying those files by hand to run on Tilestache wherever we put them on the server.

While this process should not be continued, many legacy layers (43) exist on the production server. Moving them to MapBox would take time and require restyling all of them inside of MapBox Studio, and would risk being enough data that MapBox would no longer be a free option.

Documentation on the installation process for Pew EFH servers can be found here: https://github.com/Ecotrust/PEW-EFH/wiki/Serving-UTF-Grid-Tiles

Documentation for the TileStache project can be found here: http://tilestache.org/

Installing on NGINX/uWSGI

For reference go here

From inside your virualenv:

- `sudo apt-get install libpcre3 libpcre3-dev`
- `apt-get install uwsgi uwsgi-plugin-python`
- `pip install uwsgi`
- test uwsgi install with `uwsgi --http :8000 --wsgi-file /usr/local/apps/PEW-EFH/mp/wsgi.py`
- `sudo apt-get install nginx`
- `sudo cp /usr/local/apps/PEW-EFH/deploy/efh.conf /etc/nginx/sites-available/efh`
- `sudo rm /etc/nginx/sites-enabled/default`
- `sudo ln -s /etc/nginx/sites-available/efh /etc/nginx/sites-enabled/efh`
- `sudo service nginx start`
- `sudo service nginx restart`
- copy relevant content from deploy:

  ```
  sudo cp /usr/local/apps/PEW-EFH/deploy/efh_uwsgi.ini etc/uwsgi/apps-available/efh.ini
  sudo ln -s /etc/uwsgi/apps-available/efh.ini /etc/uwsgi/apps-enabled/efh.ini
  sudo cp /usr/local/apps/PEW-EFH/deploy/rc.local.template /etc/rc.local
  ```

- `sudo chmod 777 /usr/local/apps/PEW-EFH/mp/logs/main.log` <- TODO: make this not necessary
- `sudo chmod 664 /usr/local/apps/PEW-EFH/mp/logs/main_debug.log`
- `sudo chgrp www-data /usr/local/apps/PEW-EFH/mp/logs/main_debug.log`
- `sudo reboot 0`
- In a few minutes, test your URL in a browser to see that everything came up as expected

Automatic (Unattended) Security Updates

From the document Using the "unattended-upgrades" package

Install the unattended-upgrades package if it isn't already installed (sudo apt-get install unattended-upgrades).

To enable it, do:

```
sudo dpkg-reconfigure --priority=low unattended-upgrades
```

(it's an interactive dialog) which will create `/etc/apt/apt.conf.d/20auto-upgrades` with the following contents:

```
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
```

To have the server automatically reboot when necessary to install security updates:

1. install the package `update-notifier-common`

```
sudo apt-get install update-notifier-common
```

1. edit the file `/etc/apt/apt.conf.d/50unattended-upgrades` near the bottom you will find the line

```
//Unattended-Upgrade::Automatic-Reboot "false";
```

uncomment it and set value to true:

```
Unattended-Upgrade::Automatic-Reboot "true";
```

To tell the server what time is most safe to reboot (when needed), uncomment the line

```
//Unattended-Upgrade::Automatic-Reboot-Time "02:00";
```

And set the time to your desired restart time.

Read the source document for more details.

Automatic AWS Volume Snapshots

Install AWS Cli:

- Be sure to:
  - run as root (`su root`)
  - not be in a virtual environment (`deactivate`)

```
apt-get install awscli
pip install awscli --upgrade --user
```

Next you will need to collect the following and add them to the appropriate lines near the top of /usr/local/apps/PEW-EFH/scripts/efh_prod_snapshot.sh:

- your AWS instance's region
- your AWS_ACCESS_KEY_ID
- your AWS_SECRET_ACCESS_KEY
- your instance's volume's volume ID

NOTE: You may wish to copy efh_prod_snapshot.sh to a new script to be edited locally so that you don't have to deal with merge conflicts. If you do so, be sure to keep the new script name and permissions in mind while performing the following steps.

Add those, then add the script as a cron job:

```
sudo crontab -e
```

add the job with the following line (yes, it has to be bash):

```
0 9 1 * * /bin/bash /usr/local/apps/PEW-EFH/scripts/efh_prod_snapshot.sh
```

Note - this was installed to run for a Pacific Coast audience on a server running UTC. 9AM UTC is the 'wee hours' for Pacific Time (1 or 2 AM, daylight savings time dependent). Note that you want this to happen at a different time than you set the reboot for automatic updates setting (preferably 1 or more hours before)

MapServer Installation

OGC-Compliant WMS servers may not always serve layers in the projection needed for web-mapping. Since the mapping tool under MarinePlanner (OpenLayers2) does no support on-the-fly reprojection of raster layers, we need to make sure that we can call a tool to do the work for us. MapServer can do this, if configured correctly.

If you didn't add the ubuntugis-unstable repository when installing the tool above, do it now:

```
sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
```

Then install mapserver and mapcache:

```
sudo apt-get install -y cgi-mapserver mapserver-bin mapcache-cgi mapcache-tools
```

Then do the following steps (scripted borrowing heavily from ZackadDev here:

```
sudo apt-get install spawn-fcgi
sudo cp /usr/local/apps/PEW-EFH/deploy/mapserv.init /etc/init.d/mapserv
sudo cp /usr/local/apps/PEW-EFH/deploy/mapserv.service
/etc/systemd/system/mapserv.service
sudo chmod +x /etc/init.d/mapserv
sudo systemctl enable mapserv
```

make sure you have the following in your NGINX conf in your 'server' block:

```
add_header Access-Control-Allow-Origin *;

location /mapserver/ {
        fastcgi_pass 127.0.0.1:9999;
        include fastcgi_params;
        fastcgi_param SCRIPT_NAME /usr/bin/mapserv$fastcgi_script_name;
    }
```

Add the generic mapfile:

```
sudo mkdir /mapfiles
sudo cp /usr/local/apps/PEW-EFH/deploy/generic.map /mapfiles/generic.map
```

Then reboot, or run the following to test your results:

```
/etc/init.d/mapserv start
sudo service nginx reload
```

If you see the following error when opening this, then all went well:

```
No query information to decode. QUERY_STRING is set, but empty.
```

Using a different domain name

This will be tricky. Here is a list of places that I know for sure you'll need to update the domain name (this likely is not everywhere it needs to change):

- /etc/postfix/main.cf
- /etc/nginx/sites-available/efh
- http://pewmarineplanner.ecotrust.org/admin/sites/site/1/
- Any SSL certs setup that have been added
- line 13 in /mapfiles/generic.map