UCSRB S2F SysAdmin Guide

Need-To-Know information for managing your Snow2Flow Site and Server.

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 16.04 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/ucsrb

Instructions for installing the tool and its requirements are covered in a separate Installation doc here: https://github.com/Ecotrust/ucsrb/blob/master/README.md

Stack

- Operating System
    - Ubuntu 16.04 LTS

- Webserver
    - NGINX
    - uWSGI
- Email
    - Postfix
- Database
    - PostgreSQL 9.5
    - PostGIS 2.4 (Spatial DB extension)
- Backend Framework
    - Django 1.11 (Python)
- Frontend Frameworks
    - Knockout.js
    - Twitter Boostrap
- Webmaps
    - Openlayers 4

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.

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/marineplanner-core/env/bin/activate:

```
source /usr/local/apps/marineplanner-core/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:

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

If you wish to see what python libraries are installed in a virtual environment, after you can run the following command from within the activated environment:

```
pip freeze | less
```

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/1.11/intro/tutorial02/#creating-an-admin-user):

```
python /usr/local/apps/marineplanner-core/marineplanner/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://s2f.ucsrb.org/admin/ In-depth coverage of the abilities in this site are beyond the scope of this document. Please feel free to read more about it here: https://github.com/Ecotrust/ucsrb/blob/master/docs/Handoff/Snow2Flow%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 08:00 - since the server time is in UTC, this will be 11 PM or 12 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/marineplanner-core/apps/ucsrb
git pull
dj migrate
dj collectstatic
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/marineplanner-core/marineplanner/manage.py` from within the '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/marineplanner-core/apps/ucsrb/scripts/auto_snapshot.sh

Monitoring

Hardware

This server has Munin installed: http://munin-monitoring.org/
You can see the output here:
https://s2f.ucsrb.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/marineplanner
- uWSGI

    - /usr/local/apps/marineplanner-core/deployment/marineplanner.ini

Debugging

Logfiles

System: /var/log/syslog
NGINX:

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

uWSGI logs:

- /var/log/uwsgi/app/marineplanner.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