SES Dashboard

SES Dashboard Installation Files

This document is based on the files in the encrypted archive `ses-dashboard-install-files.tar.gz.enc`.

Download this file to the destination server with the following command:

```
wget http://sesdashboard.asdsoftwaredesign.com/ses-dashboard-install-files.tar.gz.enc
```

Decrypt and unpack the installation files with the following commands:

```
openssl aes-256-cbc -d -salt -in "ses-dashboard-install-files.tar.gz.enc" -out "ses-dashboard-install-files.tar.gz" -pass pass:rd5CHNn3hD6MV6Pw
tar -zxvf ses-dashboard-install-files.tar.gz
tar -zxvf ses-dashboard.tar.gz
sudo chown -R www-data:www-data ses-dashboard
tar -zxvf ses-dashboard-docker.tar.gz
```

There will be two newly created directories called `ses-dashboard` and `ses-dashboard-docker`. The location of these directories are indicated in the rest of the documentation as `<ses-dashboard>` and `<ses-dashboard-docker>`. **You must replace commands in this documentation that contain `<ses-dashboard>` and `<ses-dashboard-docker>` to the location of these unpacked folders!**

ses-dashboard

The `ses-dashboard` directory contains all of the files for the web service. Any editing of these files will result in immediate update and change of the web service or added web application files.

ses-dashboard-docker

The `ses-dashboard-docker` directory contains helper scripts as well as the Dockerfiles used to build the Docker system images.

Installation

SES Dashboard is built into a Docker image for ease of installation and backup. There is no complex configuration needed as this has already been performed during development.

The installation instructions have been tested on Ubuntu 14.04 LTS. Different installation instructions for other Linux distributions can be found on the Docker web page.

Command line blocks can usually just be copied and pasted into a terminal window.

Docker Installation

Install the latest stable version of Docker with the following commands on the command prompt:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950
F966E92D8576A8BA88D21E9
sudo sh -c "echo deb https://get.docker.io/ubuntu docker main > /etc/apt/sources.list.d/
docker.list"
sudo apt-get update
sudo apt-get install inotify-tools lxc-docker mysql-client
```

Configure Docker with some more options by editing the /etc/default/docker configuration file:

```
sudo sed -i '/DOCKER_OPTS=/c\DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4 -r=false"' /etc/de
fault/docker
```

Configure the Ubuntu firewall, called ufw, so that data sent to ports on the firewall can be forwarded to Docker:

```
sudo sed -i '/DEFAULT_FORWARD_POLICY="DROP"/c\DEFAULT_FORWARD_POLICY="ACCEPT"' /etc/defa
```

```
ult/ufw
sudo service ufw restart
```

Enable the Docker containers that run the web server and MySQL server, called ses_app and ses_mysql respectively, to run after reboot:

```
sudo cp <ses-dashboard-docker>/docker/docker_containers.conf /etc/init
```

Docker should now be installed.

SES Dashboard Installation

SES Dashboard comprises of two Docker system images called `ses_app` and `ses_mysql`. The system images are imported by running the following commands where the encrypted archive `ses-dashboard-install-files.tar.gz.enc` was previously unpacked:

```
cat ses-app.tar.gz | sudo docker import - ses_app
cat ses-mysql.tar.gz | sudo docker import - ses_mysql
```

Docker system image import can be confirmed by running the following command:

```
sudo docker images
```

This command should show the images `ses_app` and `ses_mysql`.

SES Dashboard can now be started.

Starting SES Dashboard

Scripts have been provided to start and stop the SES Dashboard Docker images. These scripts are located in the `ses-dashboard-docker` archive.

Docker containers must be started. During this process a Docker image is placed into a Docker container and the default command in the Docker image is executed. **This command starts both Docker containers and should only need to be executed one time on a production system.** Once the docker containers have been started, they will be restarted upon a reboot of the system.

Use the following commands to start the SES Dashboard Docker images. **You must specify the absolute path of `ses-dashboard` when starting the Docker images!** This is used by Docker to find PHP files on the hard disk.

Important: During this process the server is rebooted and the `ses_app` Docker image is stopped, removed, and started again. This is done to ensure that the Docker containers will be restarted correctly when the system is rebooted. **Do not ignore this step.**

```
cd <ses-dashboard-docker>
# <ses-dashboard> must be replaced with the full absolute path of the ses-dashboard dire
ctory, or the command will not work.
sudo ./run_prod_all.sh <ses-dashboard>
# reboot the server
sudo reboot
# once the system has booted, run the following commands
cd <ses-dashboard-docker>
sudo ./app/stop.sh
sudo ./app/remove.sh
# <ses-dashboard> must be replaced with the full absolute path of the ses-dashboard dire
ctory, or the command will not work.
sudo ./app/run_prod.sh <ses-dashboard>
```

Test that rebooting the server is working properly by rebooting the server again and visiting this URL in a browser:

```
http://<hostname or IP address>/energystat/locations
```

A set of brackets ([]) should appear.

Stopping the Docker Containers

The Docker containers can also be stopped.

Important: This will remove all data from the MySQL database. The data will have to be imported or restored from backup.

```
cd <ses-dashboard-docker>
sudo ./stop_all.sh
```

Starting the Docker Containers

After a the Docker containers are stopped, they can be restarted with the following commands:

```
cd <ses-dashboard-docker>
```

```
sudo ./start_all.sh
```

Removing the Docker Containers

After a Docker container is stopped, it can be removed forever.

Important: This will remove all data from the MySQL database. The data will have to be imported or restored from backup.

```
cd <ses-dashboard-docker>
sudo ./remove_all.sh
```

Starting SES Dashboard for Development

SES Dashboard was developed on Ubuntu 14.04 Desktop using PhpStorm. It is advised that the same environment be used in further development of SES Dashboard. It is possible to run Ubuntu 14.04 on a Windows or Macintosh operating using system using VirtualBox or Parallels respectively.

SES Dashboard can also be started in a way that can be used for development. In this mode the Docker container directory `/var/www` is mapped to a directory on the host computer. A user can edit the files in the mapped directory and the changes will be present inside the Docker container.

It is required that you edit the file `<ses-dashboard-docker>/app/run_dev.sh` and replace the path to the development directory to where the development files reside on the development system. SES Dashboard was developed in the directory `/home/dehaul/prj/ses/ses-dashboard`. Replace this path with the directory where you have extracted the `ses-dashboard` archive.

Start the Docker containers in development mode:

```
cd <ses-dashboard-docker>
sudo ./run_dev_all.sh
```

The web application files in the `ses-dashboard` archive must have the proper file permissions so that they can be accessed by the web server inside the running `ses_app` Docker container. This can be accomplished by accessing the shared volumes on the `ses_app` development container in another Docker container instance.

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
```

```
chown -R www-data:www-data /var/www
```

The user of the host computer must also have permissions to edit the files in the mapped directory. This can be accomplished by adding the editing user to the `www-data` group by using GUI system settings or editing the `/etc/group` file located on the host computer.

Database Operations

Important: The `ses_app` and `ses_mysql` Docker containers must have been started before any database operations can take place.

SES Dashboard Database Backup

The MySQL database can be backed up and restored by using the regular `mysql` command on the host computer. MySQL Client tools must first be installed.

```
sudo apt-get install mysql-client
```

The user names and passwords of the MySQL users are located inside the Docker container in a file:

```
cd <ses-dashboard-docker>/mysql
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
cat /root/mysql_passwords.txt
# output from command:
ses_admin : <password here>
ses : <password here>
ses_testing : <password here>
```

The `ses_admin` user can then be used to backup the MySQL database:

```
<ses-dashboard-docker>/mysql/mysql_backup.sh ses_admin <ses_admin_password> ses
# a file with the name similar to ses_20140715_070650.tar will be created
```

The backed up database can be restored at a later time.

```
<ses-dashboard-docker>/mysql/mysql_restore.sh ses_admin <ses_admin_password> ses_2014071
5_070650.tar
# the data will be restored, overwriting the current data in the database
```

Database backups should be performed on a regular schedule using a cron job. See Ubuntu or other Linux distribution documentation about cron jobs for more information how to schedule a cron job on the host system.

Adding a New Location

Each location is a separate table inside of the database. Adding a new location requires creating a Laravel migration script, setting up the database by editing a PHP file, and importing the CSV data into the table.

Important: The migrations for the following tables have already been created:

- mc_courthouse
- mc_fleet_maintenance
- mc_hr_dept
- mc_workforce

It is only required that you execute the migration scripts in the section Executing Migrations.

Creating a Migration

A migration is an semi-generated PHP file that contains the instructions for creating and destroying a table.

Create a migration with the following commands:

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
cd /var/www/
./create_migration.sh <new_location_table_name>
# a new migration will will be located in app/database/migrations
exit
```

The newly created migration file must be edited. The migration file will be located in the `app/database/migrations` directory. Open the file and look for a code block that looks like the following:

```
Schema::create('<new_location_table_name>', function(Blueprint $table)
{
  $table->increments('id');
  $table->timestamps();
});
```

Change this code block to look the following:

```
Schema::create('<new_location_table_name>', function(Blueprint $table)
{
  $table->engine = 'InnoDB';

  $table->increments('id');
  $table->dateTime('timeUTC')->unique();
});
```

Executing Migrations

Run the following set of commands to execute the migrations:

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
cd /var/www/
php artisan migrate
# You will be warned that the application is in production. Say yes.
exit
```

Importing CSV Data

CSV data is imported with a custom created Laravel command. The table specified with `<new_location_table_name>` must have been created using the migration steps.

Important: The CSV files must be accessible to the Docker container. You must put the importable CSV files into the `<ses-dashboard>` directory so that they will be visible to the Docker container!

The following command will import all .csv files in the specified directory:

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
cd /var/www/
php artisan energy-stat:csv-import -d <data_directory> <new_location_table_name>
```

It is also possible to import a single file at a time with the following command:

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
```

```
cd /var/www/
php artisan energy-stat:csv-import -f <data_file> <new_location_table_name>
```

A cron job can be set to run a command to import a directory of CSV files by using the following commands:

```
docker run -i -t --volumes-from ses_app --link ses_mysql:mysql --name "ses_app_cron" ses
_app php /var/www/artisan energy-stat:csv-import -d <data_directory> <new_location_table
_name>
docker rm ses_app_cron
```

The energystat Web Service

Imported energy and temperature statistics are retrieved through the `energtstat` web service. It is possible to test the `energystat` web service with the `curl` command. Windows curl binaries can be downloaded from http://www.confusedbycode.com/curl/.

Locations Operator

The energy stat web service provides the current locations through the `locations` operator.

```
<URL_to_SES_Dashboard>/energystat/locations
```

The `locations` operator can be tested with `curl`:

```
curl http://<URL_to_SES_Dashboard>/energystat/locations
# Output
["mc_courthouse","mc_fleet_maintenance","mc_hr_dept","mc_workforce"]
```

Fields Operator

Fields for each location can differ from other locations. Fields can be retrieved for a location using the `fields` operator. A specific location is appended to the URL. The location must be one of the locations returned from the `locations` operator.

```
<URL_to_SES_Dashboard>/energystat/fields/<location>
```

The `fields` operator can be tested with `curl`:

```
curl http://<URL_to_SES_Dashboard>/energystat/fields/mc_courthouse
# Output
["1st_Floor_Occupied_Ave_Degrees_F","1st_Floor_Occupied_Degrees_F","1st_Floor_Occupied_M
ax_Degrees_F","1st_Floor_Occupied_Min_Degrees_F","1st_Floor_Unoccupied_Ave_Degrees_F","1
st_Floor_Unoccupied_Degrees_F","1st_Floor_Unoccupied_Max_Degrees_F","1st_Floor_Unoccupie
d_Min_Degrees_F","Accumulated_Apparent_Energy_Phase_A_Export_Vah",
... truncated for length...
"Total_Reactive_Power_Present_Demand_kVAR","Total_Real_Power_Max_Demand_Export_kW","Tota
l_Real_Power_Max_Demand_Import_kW","Total_Real_Power_Present_Demand_kW","Voltage_LL_3p_A
ve_Volts","Voltage_LN_3p_Ave_Volts","Voltage_Phase_AB_Volts","Voltage_Phase_AC_Volts","V
oltage_Phase_AN_Volts","Voltage_Phase_BC_Volts","Voltage_Phase_BN_Volts","Voltage_Phase_
CN_Volts"]
```

If an invalid location is specified, the fields operator will return:

```
"Invalid location specified."
```

```
curl http://<URL_to_SES_Dashboard>/energystat/fields/invalid_location
# Output
"Invalid location specified."
```

Read Operator

Data for a location is retrieved using the `read` operator. The location must be one of the locations returned from the `locations` operator.

```
<URL_to_SES_Dashboard>/energystat/read/<location>
```

Request

A HTTP POST request must be made to the operator with a JSON description of the data that is to be returned from the service. The JSON description must include:

- **timeFrom** - string representing the data start time in MySQL datetime format. Data returned is inclusive of this date.
- **timeTo** - string representing the data end time in MySQL datetime format. Data returned is inclusive of this date.
- **fields –** array of strings of the fields to be returned.

Important: `timeTo` is exclusive. All data in the database will be returned between the dates specified, excluding the `timeTo` date. For example to retrieve the data on March 11, 2014, specify March 11, 2014 in `timeFrom` and March 12, 2014 in `timeTo`.

Important: the `timeUTC` field is always returned in the response object and does not have to be specified.

An example JSON request to the `read` operator:

```
{
  "timeFrom": "2014-03-11",
  "timeTo": "2014-03-18",
  "fields": [
    "1st_Floor_Occupied_Degrees_F",
    "Basement_office_Degrees_F",
    "Supply_boiler_Degrees_F",
    "Total_Net_Instantaneous_Real_Power_kW"
  ]
}
```

Response

A JSON object containing the requested data is returned from the read operator. It includes:

- **errors** - array of strings indicating errors with the request.
- **data** - nested JSON object containing the fields that were requested and successfully returned.

An example JSON object returned by the description of the data to be returned from the `read` operator:

```
{
  "errors":    ["Requested field Not_A_Field not found."],
  "data":{
    "timeUTC":["2014-03-11 17:30:00","2014-03-11 17:45:00",
      ... truncated for length...
    ],
    "Total_Net_Instantaneous_Real_Power_kW":[142.44,132.53,146.55,
      ... truncated for length...
    ],
  }
}
```

The `read` operator can be tested with `curl`:

```
curl -X POST -H "Content-Type: application/json" http://localhost/energystat/read/mc_cou
```

```
rthouse -d '{"timeFrom": "2014-03-11", "timeT18", "fields": ["1st_Floor_Occupied_Degrees
_F", "Basement_office_Degrees_F", "Supply_boiler_Degrees_F", "Total_Net_Instantaneous_Re
al_Power_kW"]}'
# Output
{"errors":[],"data":{"timeUTC":["2014-03-11 17:30:00","2014-03-11 17:45:00","2014-03-11
18:00:00","2014-03-11 18:15:00","2014-03-11 18:30:00","2014-03-11 18:45:00","2014-03-11
19:00:00","2014-03-11 19:15:00","2014-03-11 19:30:00","2014-03-11 19:45:00","2014-03-11"
# ... truncated for length...
}
```

PHPUnit Tests

PHPUnit tests are included in the `<ses_dashboard>/app/tests` directory. Tests can be run with the included `phpunit` command.

```
cd <ses-dashboard-docker>/app
sudo ./run_shell.sh
# another shell will pop up... this is a shell in the Docker container
cd /var/www/
./phpunit
# PHPUnit tests will be executed and their results will be shown
```

PHP Code Documentation

Documentation is provided for all code created or modified for Sensible Energy Solutions LLC. The documentation is located in the `<ses_dashboard>/doc` directory. Open the `<ses_dashboard>/doc/index.html` file with a web browser to view the documentation.

Sample Web Application

A sample web application using the AngularJS Framework is provided in the SES Dashboard project. This sample web application comprises of the following:

Files:

- `<ses_dashboard>/app/views/ses_dashboard.php`

Directories

- `<ses_dashboard>/angular-seed`

- `<ses_dashboard>/public/bower_components`
- `<ses_dashboard>/public/js`
- `<ses_dashboard>/public/partials`

Please refer to the AngularJS documentation to improve the sample web application.

Any other JavaScript web application framework may be used with the SES Dashboard Web service.

Open Source Software

SES Dashboard employs the following open source projects in accordance to their licensing agreements:

- AngularJS
- Chart.js
- Composer
- Docker
- Laravel PHP Framework
- MySQL Database
- PHP
- PHPUnit
- Twitter Bootstrap
- Ubuntu Linux