Steps for installing and configuring a rPi for FlightLine:

Introduction

FlightLine is a small web app that is intended to be a lightweight alternative to NotauScore - the sofware that supports NotauMatic devices used for scoring F3A and Pattern contests.

It is specifically created to allow for the results to be downloaded by Score! (the application used for scoring IMAC competitions).

FlightLine's aim is to (eventually) also support other devices such as Peter Vogel's iOS based Electronic Scribe app as well as ScorePad written by Dan Carroll.

Prepare SD Card

Write the Raspberian image to the card. You can get it here: https://www.raspberrypi.org/downloads/raspbian/

When the image is written and mounted, the boot Fat32 partition should be seen. To enable ssh, create an empty file called ssh in the root folder of this partition. On a mac:

```
<Laika:danny> 07:48 ~ : touch /Volumes/boot/ssh
<Laika:danny> 07:48 ~ :
```

You can now put the SD Card into your rPi hardware and reboot.

When it is available, log into the rPi via ssh. If you cannot find the IP address, it's displayed on the console via HDMI at the end of the boot process.

Change the default password

```
pi@raspberrypi:~ $ passwd
Changing password for pi.
Current password:
New password:
Retype new password:
passwd: password updated successfully
pi@raspberrypi:~ $
```

Create local admin user

In this case I used username 'danny' and user description (full name) 'Dan Carroll'

```
pi@raspberrypi:~ $ sudo su -

    Wi-Fi is currently blocked by rfkill.
    Use raspi-config to set the country before use.

root@raspberrypi:~# USERNAME=danny
root@raspberrypi:~# USERDESC="Dan Carroll"
root@raspberrypi:~# useradd -c "${USERDESC}" -s /bin/bash -U -m -G sudo
"${USERNAME}"
root@raspberrypi:~# passwd "${USERNAME}"
New password:
Retype new password:
passwd: password updated successfully
root@raspberrypi:~#
```

Notice that there is an error about the wifi. That is because we have not configured it yet. We'll do that later on.

If you have a ssh keypair you can add the public key to the ~/.ssh/authorized_keys file.

The directory .ssh should be created with mode 750 and the file should be 600. If you don't have a key or don't know, then you can skip this part.

Get the latest dist/firmware

```
root@raspberrypi:~# apt-get update
<SNIP!>
root@raspberrypi:~# apt-get upgrade
<SNIP!>
```

Install Docker CE

```
root@raspberrypi:~# curl -sSL https://get.docker.com | sh
```

```
# Executing docker install script, commit:
442e66405c304fa92af8aadaa1d9b31bf4b0ad94
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-
https ca-certificates curl >/dev/null
+ sh -c curl -fsSL "https://download.docker.com/linux/raspbian/gpg" | apt-key
add -qq - >/dev/null
Warning: apt-key output should not be parsed (stdout is not a terminal)
+ sh -c echo "deb [arch=armhf] https://download.docker.com/linux/raspbian
buster stable" > /etc/apt/sources.list.d/docker.list
+ sh -c apt-get update -qq >/dev/null
+ [ -n  ]
+ sh -c apt-get install -y -qq --no-install-recommends docker-ce >/dev/null
+ sh -c docker version
Client: Docker Engine - Community
 Version:           19.03.7
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        7141c19
 Built:             Wed Mar  4 01:55:10 2020
 OS/Arch:           linux/arm
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.7
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       7141c19
  Built:            Wed Mar  4 01:49:01 2020
  OS/Arch:          linux/arm
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683

If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!
```

```
WARNING: Adding a user to the "docker" group will grant the ability to run
     containers which can be used to obtain root privileges on the
     docker host.
     Refer to https://docs.docker.com/engine/security/security/#docker-daemon-
attack-surface
     for more information.
root@raspberrypi:~# sudo usermod -aG docker pi
root@raspberrypi:~# sudo usermod -aG docker danny
```

Install command line utilities

Some utilities are necessary to get things working. We want this as minimal as possible so that the upgrade list is the smallest impact.

```
root@raspberrypi:~# apt-get install -y git sqlite3 docker-compose composer
Reading package lists... Done
Building dependency tree
Reading state information... Done
.
.
<SNIP! - Lots of text removed>
.
.
Processing triggers for man-db (2.8.5-2) ...
root@raspberrypi:~#
```

Install Flightline

There are 3 containers needed for flightline. Together they make up the flightline 'service'. Currently the DB connection is via sqlite to a file. This seems to be working out OK, but if it becomes a problem, then creating a DB instance wont be difficult.

This service resides in /data/flightline and is created by grabbing the git repository for 'Flightline'.

The 3 containers are:

- **Proxy** A nginx instance that is bound to ports 80 and 443 on the host and reverse-proxies the requests to the backend webserver. This allows for fine-grained control of the web traffic and logging of the requests of the client (before any API translation).
- **Web** The nginx instance that handles the web requests for static files and hands off php requests to the PHP container via php-fpm.
- **Php** The php container is only for processing php files. Separating this container allows for easy upgrading of php.

```
root@raspberrypi:~# mkdir -p /data/volumes
root@raspberrypi:~# cd /data/
root@raspberrypi:/data# git clone
https://git.dannysplace.net/scm/score/flightline.git
```

```
Cloning into 'flightline'...
remote: Counting objects: 269, done.
remote: Compressing objects: 100% (249/249), done.
remote: Total 269 (delta 76), reused 0 (delta 0)
Receiving objects: 100% (269/269), 48.15 KiB | 666.00 KiB/s, done.
Resolving deltas: 100% (76/76), done.

root@raspberrypi:/data# git clone https://git.dannysplace.net/scm/score/score-
flightline-node.git
Cloning into 'score-flightline-node'...
remote: Counting objects: 5844, done.
remote: Compressing objects: 100% (5730/5730), done.
remote: Total 5844 (delta 2724), reused 182 (delta 80)
Receiving objects: 100% (5844/5844), 9.40 MiB | 1.30 MiB/s, done.
Resolving deltas: 100% (2724/2724), done.
Checking out files: 100% (6500/6500), done.
root@raspberrypi:/data# ls -asl
total 28
4 drwxr-xr-x  7 root root 4096 Mar  8 08:17 .
4 drwxr-xr-x 23 root root 4096 Mar  8 05:22 ..
4 drwxr-xr-x  4 root root 4096 Mar  8 08:17 flightline
4 drwxr-xr-x 10 root root 4096 Mar  8 08:17 score-flightline-node
4 drwxr-xr-x  2 root root 4096 Mar  8 08:14 volumes
```

Now link the html dir of the web server, back to the score-flightline-node repo dir.

```
root@raspberrypi:/data# cd /data/volumes/
root@raspberrypi:/data/volumes# ln -s ../score-flightline-node html
root@raspberrypi:/data/volumes# ls -asl
total 8
4 drwxr-xr-x 2 root root 4096 Mar  8 08:21 .
4 drwxr-xr-x 7 root root 4096 Mar  8 08:17 ..
0 lrwxrwxrwx 1 root root   24 Mar  8 08:21 html -> ../score-flightline-node
```

There are some components of the web-app that are easily updated via composer. At the start I built these into the repo directly to save complexity, but that does not seem to be the internet way. Especially if other users might want to contribute in the future. I will also decouple datatables/bootstrap/jquery for this same reason.

The first time you run the composer command it will download the docker container. Then it will check composer.json and install the extra components in /vendor

```
root@raspberrypi:/data/volumes# cd /data/score-flightline-node/
root@raspberrypi:/data/score-flightline-node# composer install
Do not run Composer as root/super user! See https://getcomposer.org/root for
details
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
  - Installing psr/log (1.1.2): Downloading (100%)
  - Installing katzgrau/klogger (dev-master de2d3ab): Cloning de2d3ab677 from
cache
Writing lock file
Generating autoload files
root@raspberrypi:/data/score-flightline-node#
```

Note.

Now the proxy, web and php containers are essentially ready. However there is no database yet. Evetually this will be part of a web process, but for now, lets create it manually.

```
root@raspberrypi:/data/score-flightline-node# sqlite3 db/flightline.db <
include/dbCreate_v2.sql
root@raspberrypi:/data/score-flightline-node#
```

Finally, before we can start the containers we must make sure that they can write to the DB and to the log dir. It will also need to write to the api directory to create a secret token.

```
root@raspberrypi:/data/score-flightline-node# chown -R www-data:www-data log db
root@raspberrypi:/data/score-flightline-node# chgrp www-data api/[0-9]*
root@raspberrypi:/data/score-flightline-node# chmod 775 api/[0-9]*
root@raspberrypi:/data/score-flightline-node#
```

Finally, start the service!

```
root@raspberrypi:/data/score-flightline-node# cd /data/flightline/flightline/
root@raspberrypi:/data/flightline/flightline# docker-compose up
Creating network "flightline_default" with the default driver
Creating volume "flightline_web_confd" with local driver
Creating volume "flightline_html" with local driver
Creating volume "flightline_proxy_confd" with local driver
Creating volume "flightline_proxy_certs" with local driver
Creating flightline_proxy_1 ... done
Creating flightline_php_1   ... done
Creating flightline_web_1   ... done
Attaching to flightline_php_1, flightline_proxy_1, flightline_web_1
php_1    | [08-Mar-2020 09:06:14] NOTICE: fpm is running, pid 1
php_1    | [08-Mar-2020 09:06:14] NOTICE: ready to handle connections
```

At this point, the containers look like their are running. But the container is running inthe foreground and we need to run it in the background.

If you hit ctrl-c then the container will stop.

```
^CGracefully stopping... (press Ctrl+C again to force)
Stopping flightline_web_1   ... done
Stopping flightline_php_1   ... done
Stopping flightline_proxy_1 ... done
```

Now if we stop the whole service we can restart it in the background. The -d option means detach..

```
root@raspberrypi:/data/flightline/flightline# docker-compose down
Removing flightline_web_1   ... done
Removing flightline_php_1   ... done
Removing flightline_proxy_1 ... done
Removing network flightline_default

root@raspberrypi:/data/flightline/flightline# docker-compose up -d
Creating network "flightline_default" with the default driver
Creating flightline_proxy_1 ... done
Creating flightline_php_1   ... done
Creating flightline_web_1   ... done
root@raspberrypi:/data/flightline/flightline#
```

We can read the logs with the logs command:

```
root@raspberrypi:/data/flightline/flightline# docker-compose logs
Attaching to flightline_web_1, flightline_php_1, flightline_proxy_1
php_1    | [08-Mar-2020 11:25:24] NOTICE: fpm is running, pid 1
php_1    | [08-Mar-2020 11:25:24] NOTICE: ready to handle connections
root@raspberrypi:/data/flightline/flightline#
```

Set the timezone

```
ToDo
```

Configure WiFi

```
ToDo
```