Rough Deployment Notes

1. Set up a decent firewall on the production server, only allowing port 80 and 443 in for this application. Do not forget that your ssh server port also needs to be open if you are managing this server remotely through ssh.
    - Another piece to note is that docker uses iptables to set up its internal network isolation system. If you are using nftables as your interface to netfilter, then make sure you use the uppercase names for the ip filter tables. These:(INPUT, FORWARD and, OUTPUT).
    - Here is an example of an nftables ip filter for ipv4 allowing port 22, 443, and 80 TCP traffic.

```
table ip filter {
        chain INPUT {
                type filter hook input priority 0;
                # Drop invalid packets
                ct state invalid drop;
                # Allow from loopback
                iif lo accept;
                # Established/related connections
                ct state established,related accept;
                # Allow ssh
                tcp dport 22 accept;
                # Allow https traffic in on 443
                tcp dport 443 accept;
                # Allow http traffic in on 80 (This should be
redirected to https)
                tcp dport 80 accept;
                # Drop everything else
                counter drop;
        }

        chain FORWARD {
```

```
            type filter hook forward priority 0; policy accept;
            counter packets 0 bytes 0 drop
        }

        chain OUTPUT {
            type filter hook output priority 0; policy accept;
        }
    }
```

2. Install docker.
    - On Debian systems the package is called **`docker.io`**
3. Make sure you have git installed, and change directories to **`/usr/local`** on the server.

   Use git clone to pull down the NPLBAM application to this directory. If the resultant folder is named anything other than **`NPLBAM`** then rename it.
4. At this point you will need the **`secrets.tar.gz`** archive. Place this archive in the **`NPLBAM`** folder.
5. Extract this folder and change directories into the **`secrets`** directory created in the **`NPLBAM`** folder from the extraction.
6. Run the **`./place_secrets.sh`** script within that directory.
7. Change back to the **`NPLBAM`** folder and, execute **`./create_network.sh`**
8. Change directories to the **`NPLBAM/docker/nplbam_postgres`** directory, and execute the **`build.sh`** script while in that directory.
9. Do the same thing in the **`NPLBAM/docker/web_cont`** directory.
10. Now, change directories into the **`NPLBAM/service_files`** directory, and run the following command.
    - (As root) **`cp ./* /etc/systemd/system/`**
11. Now, we start the services using systemd. Execute the following commands **in order.**
    - **`systemctl enable nplbam_postgres && systemctl start nplbam_postgres`**
    - **`systemctl enable nplbam_flask && systemctl start nplbam_flask`**
    - **`systemctl enable nplbam_nginx && systemctl start nplbam_nginx`**

12. The base **NPLBAM** application is now running on the server, and will be automatically restarted whenever the server starts up.
13. Creating the first root admin account for the application (there is no default)
    - In the **`NPLBAM`** folder, execute the **`./build_venv.sh`** script (this requires python3 and the venv module installed on the machine.)
    - Execute the following, **in order**. It will result in a long string starting with b (this is your password hash) printed to your terminal. Copy everything **including the quotes**, except for the b to your clipboard.
      - **`source nplbamNV/bin/activate`**
      - **`python`**
      - **`import nacl.pwhash`**
      - **`print(nacl.pwhash.str(b'YOUR PASSWORD HERE'))`**
      - **`exit()`**
    - run **`./connect_psql.sh`** and you will be at a psql prompt for the postgres database. In that prompt execute the following commands **in order.**
      - **`\c nplbam`**
      - **`insert into "Users" (username, password, "userLVL") values ('YOUR USER NAME HERE', WHAT YOU COPIED TO YOUR CLIPBOARD HERE, 0);`**
      - **`exit`**
14. The application should now be deployed and running on this server, and you should be able to access it on port 80.
15. Please note that this deployment is **currently insecure**. For the deployment to be secure the nginx configuration file must be changed significantly, a TLS certificate must either be purchased from a CA, or CertBot needs to be set up to get auto-renewing certificates for free from Lets Encrypt; and these certificates then deployed into the NGINX container volume where nginx can see them.

    Setting up CertBot also requires modifications to the nginx configuration files so that Lets Encrypt can look for a specific file in the .well-known folder on the webserver and verify that you are the domain that you say you are.