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). As well as have the iptables -> nftables compatibility layer installed.

   - 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 back to the **NPLBAM** directory, and execute **./start_postgres.sh && ./stop_postgres.sh** to initialize the database with the root password from when **secrets** was deployed.

11. Change directories into **NPLBAM/nplbam** and execute **mkdir files** then execute **chown 9984 files AS ROOT** to create the directory for storing uploaded files, and set the required file permissions. File uploads will not work properly without this step.

12. Now, change directories into the **NPLBAM/service_files** directory, and run the following command.

    ○ (As root) **cp ./* /etc/systemd/system/**

13. 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`

14. The base **NPLBAM** application is now running on the server, and will be automatically restarted whenever the server starts up.

15. 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`

16. The application should now be deployed and running on this server, and you should be able to access it on port 80.

17. Now, to secure this deployment, we need to acquire our TLS certificates from LetsEncrypt. To do this, we need to change the nginx configuration file over to the **nginx_TLS_SETUP.conf** file, by using `mv nginx.conf nginx_clear.conf` and then `mv nginx_TLS_SETUP.conf nginx.conf` in the nginx directory.

18. In the **nginx** directory, within the file now called '**nginx.conf**', change <DOMAIN GOES HERE> to the domain (or subdomain) name we are going to be requesting a TLS cert for.

19. Execute `systemctl stop nplbam_postgres` and then execute `systemctl start nplbam_nginx` to restart the application.

20. In the main **NPLBAM** directory, execute `./obtain_certbot_cert.sh`

21. Follow through the prompts that Certbot will guide you through. For the first question the answer is 2 (for place files in webroot) and when it asks you for the webroot path, the answer is **/webroot**

22. Now we need to generate our Diffie-Hellman parameters. To do so...

    ○ Change directories to **NPLBAM/nginx/tls/live**

    ○ Execute `openssl dhparam -out dhparam-4096.pem 4096`

    ○ This can take some time to execute.

23. Now we need to set up our production **nginx.conf** configuration file. To do so, open **nginx_TLS_LIVE.conf** in the text editor of your choice.

24. Replace **every** instance of **\<DOMAIN GOES HERE\>** with our domain (or subdomain) name in the file.

25. Rename the current **nginx.conf** back to **nginx_TLS_SETUP.conf**

26. Rename **nginx_TLS_LIVE.conf** to **nginx.conf**

27. Restart the whole docker cluster by executing `systemctl stop nplbam_postgres` and then execute `systemctl start nplbam_nginx`

28. You should now have https access to the application.

29. There is one final step to enable automatic renewal of the TLS certificates.

30. Execute `systemctl enable nplbam_certbot_renew.timer` and then execute `systemctl start nplbam_certbot_renew.timer`