Vehicle Maintenance

- Lubelogger
  - Install Lubelogger

Lubelogger

Install Lubelogger

https://www.youtube.com/embed/_eAsSytpwIw

Introduction to Lubelogger

A vehicle is a long term investment. Sometimes you ask yourself is it time to reinvest? A new vehicle can sometimes be the smarter move, but data is so very important in large decisions like this.

Lubelogger is an amazing, free and open source, self hosted, awesome way to make sure you have all the data. not only for when it's time to buy a new car, but also for when you're wanting to sell your existing car. The ability to show a prospective buyer your maintenance and upgrade log is powerful. It could mean the difference of thousands in a sale.

Data should be a driving force when you are considering a new car as well. Yes, a certain repair may cost quite a bit, but is the overall life going to be extended enough that it's a net win in the end? If you are making all the right maintenance moves, it could be worth the repair.

Finally, from experience, I know that having documentation of proper, timely care can even pay off when your vehicle is "out of warranty". I had an engine just decide it was going to drop all its oil and die on a long trip (350 miles). I had to have the vehicle towed to the dealer, and have the engine replaced. They wanted to charge me for the engine, but when I showed them my maintenance records and their own recall on that engine, their tune changed.

Setup a Server

In most self hosted applications these days, you'll need a machine to act as a server. This can be a machine in your home / business (such as an old laptop or desktop, a Raspberry PI or Single Board Computer (SBC), or even the computer you're reading this article from), a VM / Container hosted on one of your machines, or a VPS (Virtual Private Server) hosted by companies like RackNerd, Digital Ocean, Linode, Vultr, and so many more. Regardless of which option you choose, you'll want to do a few things to get the server setup properly.

Install updates to our Server

Ubuntu / Debian

```
sudo apt update && sudo apt upgrade -y
```

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf update -y
```

Add a non-root / sudo user on the server

Generally, when you setup a new server, the VPS (Virtual Private Server) service sets up a default "root" user for you. It's considered unsafe to do everything as "root", so let's setup a non-root user who has super user (sudo) privileges.

`adduser <username>` You'll be prompted to enter and confirm a password for this user. You'll also be asked for some user information like Name, etc, but this is not required information. At the end, confirm the entries, and you'll have your new user.

Next, we need to add the user to the super user group.

Ubuntu / Debian

```
usermod -aG sudo <username>
```

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Install Docker and Docker Compose

Fortunately, there is a single line command that will install both Docker and Docker Compose for us on most Linux based distributions.

We need the 'curl' utility to get this to work, so if you don't have it, you'll want to install it first with

Ubuntu / Debian

```
sudo apt install curl -y
```

RedHat / CentOS / Fedora / Alma / Rocky

```
sudo dnf install curl -y
```

Next, we'll run the command to install Docker and Docker Compose:

```
curl https://get.docker.com | sh
```

You may be prompted to enter your super user password, so be ready for it. Once you do, the install should proceed.

Once complete. we want to add our user to the 'docker' group so we can do `docker` and `docker compose` commands without having to type in `sudo` each time.

`sudo usermod -aG docker <username>`

Now we'll logout / exit the session, and log right back in so the updated group will take effect.

Install LubeLogger

First, let's make our directory structure to keep our application and server organized.

`mkdir -p docker/lubelogger`

Now let's move into our new folder:

`cd docker/lubelogger`

Here we'll create three files:

`touch compose.yaml`

`touch .env`

`touch init.sql`

The compose.yaml

Now, let's add the proper configuration content to these files:

`nano compose.yaml`

Copy the following code block into that new file:

```
services:
  app:
    image: ghcr.io/hargata/lubelogger:latest
    restart: unless-stopped
    volumes:
      - ./data:/App/data
      - ./keys:/root/.aspnet/DataProtection-Keys
    ports:
      - <8080>:8080
    env_file:
      - .env
```

```
  postgres:
    image: postgres:14
    restart: unless-stopped
    environment:
      POSTGRES_USER: "lubelogger"
      POSTGRES_PASSWORD: <"a-long-strong-password-here">
      POSTGRES_DB: "lubelogger"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
      - ./data/postgres:/var/lib/postgresql/data
      - /etc/localtime:/etc/localtime:ro
```

Once pasted, you may want / need to change a couple of values in here. I have surrounded those values with less than "<" and greater than ">" signs. For the port number, this is a port mapping from your host system, to the docker virtual machine. The left side is the host port (the one we can change), and the right side is the docker port (which you should never change unless you really know what you are doing). If port 8080 is already open and available on your host system, then you don't need to change this port value, just remove the "<" and ">" from around the port number.

Next, you 100%, absolutely, must change the postgres password. Again remove the "<" and ">: from around the quoted string of characters. Use a long set of random numbers, upper, and lower case letters (around 32 characters long if possible).

Now use CTRL + O to save your changes, press Enter to confirm, and exit the nano editor with CTRL + X.

The .env

Next, let's get our .env file setup. This is an environment variable file used to set the configurations of the application. Before you go copy and pasting everything in the larger block below, just know that you can use the Administrative Settings area in the web application GUI to set all of the configurations.

If, however, you prefer, you can use environment variables to do this setup. Regardless of your choice, I do suggest you at least create the .env as our compose.yaml instructs the application to look for that file at startup.

If you've chosen to use the .env file, you can get the variables for your needs by using the application authors configuration tool. You can find it at https://lubelogger.com/configure/. Choose 'Docker Installation' by clicking that button, then begin filling all of the values you want to set via the .env file. Once done going through the configurator, it will provide the contents for your .env that you can simply copy and paste.

Once completed, use CTRL + O to save your changes, press Enter to confirm, and exit the nano editor with CTRL + X.

The 'init.sql'

The final piece we need is the init.sql file. This is a file run the very first time to make sure the database, tables, and some starting values exist in order for the application to run properly. If you are coming to this tutorial months or years after it's posting in September of 2025, then I highly recommend you go to the github project, and grab the latest init.sql text to make sure you are getting an up to date version. You can find the project here https://github.com/hargata/lubelog.

Here, for your convenience, however, is the current version of the file contents:

```
DO $$
BEGIN
    IF NOT EXISTS (SELECT schema_name FROM information_schema.schemata WHERE schema_name =
'app') THEN
        CREATE SCHEMA app;
    END IF;
END $$;
```

Again, use CTRL + O to save your changes, press Enter to confirm, and exit the nano editor with CTRL + X.

Pull Down our Docker Image and Start the Application

Now we are ready to pull our docker image, and start up the application.

`docker compose up -d && docker compose logs -f`

This single line command will pull down the images needed, start the application, and begin showing the log output of the application as it starts. You are just looking for any errors that really stand out int he logs as they fly by. If all went well, you shouldn't see any.

You can stop the logs after about 10 seconds with CTRL + C.

Now, head to your favorite modern web browser, navigate to the IP address of your host system, (if it's the machine you are actually working from, you can use `localhost`) and then a colon, with the port you set in the compose.yaml file earlier. In my case I went to

http://192.168.10.152:8080

You'll be logged right onto the site, as we haven't enabled authentication yet (unless you setup OAuth via the environment file).

Regardless, head to the upper right corner, click the icon, and select Settings.

In Settings, you may want to set the dark theme, as well as a few of the other toggles. But if you want to setup authentication to your site, you need to click the 'Enable Authentication' option near the bottom of all those toggles.

You'll immediately be asked to enter a username and password for the admin user of the site. Remember what you type, and make it a strong password.

Congratulations! You now have a really great tool you can use for adding, tracking, and keeping all the data on your vehicle(s) maintenance and care.