Self Hosted Dashboards

- Dashy - Widgets
  - Adding Widgets to the Dashy Dashboard
- Heimdall
  - Install Heimdall, a beaiful shortcut and informational dashboard
- Dashy
  - Dashy - Powerful, Informative, Configurable Self-Hosting Dashboard
- Homepage
  - Install Homepage
- Labdash
  - Install and Configure Labdash
- Homarr
  - Install Homarr Dashbaord

Dashy - Widgets

Adding Widgets to the Dashy Dashboard

https://www.youtube.com/embed/dyur-NDngBc

I showed you how to setup and use a really nice homepage / dashboard for all of your self hosted services a while back called Dashy. I've stuck with it since then, and have just been super happy with it. It has active development, and tons of new features since then. One of those features is the ability to add all kinds of other widgets to the dashy interface.

Many of you have asked me how to add those widgets, so here we go.

Depending on what you're trying to view / see in your dashboard, you may need to install another bit of software on your server to provide the information for the dashboard to show. In my case, I'm going to walk you quickly through installing Glances, a really cool application that can provide you all kinds of data about your system. You can view it directly in the terminal / cli, or it can be presented as a web page (which is what we'll be using today).

I have a video on Glances and NetData that I'll link here as well for you to check out.

What You'll Need

- Dashy Installed and Ready to go - I have a video on it here, if you want to go through that first and get it installed.
- Glances running as a web server and background service (we'll go through that next)
- About 15 minutes of your time.

Installing Glances

You need to have python3 and pip3 installed. So,d epending on your distro, you may need to use a different package manager for this, than I do.

I'm using Ubuntu 20.04, so I'll be using the apt package manger. If you're running Debian / Ubuntu based distros, the commands should work for you as is.

If you're using Fedora, RedHat, or Centos (Alma, Rocky), you'll probably want to use Yum, RPM, or DNF. For Open Suse, you'll want to use Zypper, and for Arch, Pacman or PacAUR I'm guessing.

Install Python3

Open a terminal (CLI) window, and do the following: First make sure you have the lates package updates.

```
sudo apt update
```

Enter your super user password if prompted. If you're running Debian as root, you won't need the `sudo` part, just leave it off.

Next, we'll install python3 with:

```
sudo apt install python3 -y
```

After that completes, we'll install pip3 with:

```
sudo apt install python3-pip
```

Now that those are installed, we need to install Glances and Bottle (which will allow Glances to run as a web server).

Install Glances and Bottle

Use the following pip3 commands to install Glances and Bottle:

```
pip3 install bottle
```

If you get an error, try it with `sudo` like:

```
sudo pip3 install bottle
```

Now, do the same, but for Glances:

```
pip3 install glances
```

and again, if you needed sudo for Bottle, you'll need it for Glances, so do:

```
sudo pip3 install glances
```

Once those are finished installing, you can test that glances works by running the command:

```
glances
```

in your terminal. You should see a page full of information about your system show up.

You can stop glacnes with the CTRL + C key combination.

Next, you can make sure Bottle is working and run glances in web-server mode with the command:

`glances -w`

You'll see some output on the terminal, and should have something like `Glances Web User Interface started on http://0.0.0.0:61208/` on the screen.

Now, open a browser and go to

`http://localhost:61208`

or use the ip address of the machine you are working on:

`http://192.168.1.x:61208`

of course, using the correct private IP of the machine.

You should see a nice view of Glances, almos exactly as it looked in the terminal.

Now you can stop that process in the terminal with CTRL + C, an dwe need to turn that into a service that will run automatically, even after we reboot the machine.

Create the Glances Service

We'll be adding a new file to `/etc/systemd/system/` called `glancesweb.service`

So in a terminal window do the following:

`sudo nano /etc/systemd/system/glancesweb.service`

This will open a text editor in your terminal, and it should be empty.

Use the following code to start off:

```
[Unit]
Description = Glances in Web Server Mode
After = network.target

[Service]
ExecStart = /usr/local/bin/glances  -w  -t  5

[Install]
WantedBy = multi-user.target
```

If you are running as root, this should work, but we need to make sure glances is running from the path we expect, which is currently `/usr/local/bin/`. To find this out, save the file with CTRL + O, then press Enter to confirm, then use CTRL + X to exit the nano editor.

In the terminal, do the command:

```
which glances
```

You should get output like:

```
/usr/local/bin/glances
```

but, if you aren't running as root during the install you may get something like:

```
/home/<your usre>/.local/bin/glances
```

Whatever you get, highlight it, right click, select copy, and then we'll open the nano editor back up with:

```
sudo nano /etc/systemd/sysetm/glancesweb.service
```

On that line starting with `ExecStart`, make sure to remove the path (if it's different from what you got with the `which glances` command, and replace it with what you copied.

In my case, I would make it look like

```
[Service]
ExecStart = /home/<your usre>/.local/bin/glances -w -t 5
```

Now, because it's running from my home directory, I need to add one more line just below this one. If you are running from the `/usr/local/bin` directory, you **do not** need this line.

```
User = <your user>
```

so for me, the file looks like:

```
[Service]
ExecStart = /home/brian/.local/bin/glances -w -t 5
User = brian
```

Yours should have your username of course.

Now, save the file with CTRL + O, then press Enter to confirm, and use CTRL + X to exit.

Next, we need to start and enable our service.

We do the following commands to do this:

`sudo systemctl start glances.service`

`sudo systemctl enable glances.service`

As long as you don't get any errors after each of those, you can check the status with

`sudo systemctl status glances`

You should see a row in the output near the top that shows `active`. If you see `failed`, you need to recheck the `glancesweb.service` file, and make sure you have everything correct.

Now that it's running, you can again go to the ip address of:

`http://<local ip>:61208`

and view your glances in the web browser. As long as it shows up, we are ready to move forward with getting some widgets in Dashy.

Adding Widgets to Dashy

At the time of writing, Dashy widgets can only be added from the configuration file, and not through the UI / GUI editor. I believe it's being worked on, but the config file isn't hard to modify, so let's jump into it.

If you're running Dashy in the way I showed in my video previously, you'll want to get on the server it runs on, and navigate to the foldeer where the configuration file is located. For me it's in a folder in my home directory called `docker/dashy/public`.

So I do

`cd ~/docker/dashy/public`

Now if you do

`ls`

you should see a file called `conf.yml`. This is the configuration file you want.

First, let's copy conf.yml to a new backup version, just in case we mess something up, it's easy to bring it back to the way it is right now.

`cp conf.yml conf.bak.yml`

Now, let's modify our `conf.yml` file to add a widget.

`nano conf.yml`

You may need to use `sudo`. If you see a red bar at the bottom of your nano editor, you need to exit with CTRL + X, and re-open it with `sudo`.

`sudo nano conf.yml`

Now, move down through the file until you see the first section called `sections`.

Just below that line, create a new line. Keep in mind that yaml or `.yml` is very space specific. So mind your spacing.

Let's add a new Widget section for our server. My server's name is "Aria".

```
sections:
  - name: Aria Info
```

Next, we'll add a `widgets` indicator, and our first widget. We'll add the glances cpu usage widget.

```
sections:
  - name: Aria Info
    widgets:
      - type: gl-current-cpu
        options:
          hostname: http:192.168.10.209:61208
```

At this point, you can save with CTRL + O, and then go to your browser and open your Dashy dashboard, to see your new widget and ensure it works. You need to, of course, replace the IP in the example above with the IP address of your server that you installed glances on. You may have to refresh Dashy if you're already running it, and you may have to tell firefox to release the cache then refresh (firefox is great, but it really hates to refresh and show new sutff).

Let's add another widget. Continuing in the nano editor from where we are. we'll add:

```
sections:
  - name: Aria Info
    widgets:
      - type: gl-current-cpu
        options:
          hostname: http:192.168.10.209:61208
      - type: gl-current-mem
        options:
          hostname: http:192.168.10.209:61208
```

You can again save, and take a look at your Dashy dashboard to see the new widget. You can now just go crazy adding widgets to Dashy using this same method.

Let's say you want to add two servers data. It's now hard. We just create another section for our next server like so:

```
sections:
  - name: Aria Info
    widgets:
      - type: gl-current-cpu
        options:
          hostname: http:192.168.10.209:61208
      - type: gl-current-mem
        options:
          hostname: http:192.168.10.209:61208
  - name: Liratta Info
    widgets:
      - type: gl-current-cpu
        options:
          hostname: http:192.168.10.152:61208
      - type: gl-current-mem
        options:
          hostname: http:192.168.10.152:61208
```

Notice the different name and ip address in our second section. Also, understand that the glances / bottle install and service setup, needs to be done on each server / machine you want this information from.

Now you can turn Dashy into an incrdible tool for all kinds of great information.

Support me on Patreon

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/bePatron?u=234177

Heimdall

Install Heimdall, a beaiful shortcut and informational dashboard

https://www.youtube.com/embed/qFqUXN0jxMQ

I've shown you the Homer Dash in the past, and it is a termendousely great dashboard. The only downside (if you can call it that) is having to change a configuration file in order to add new cards, modify existing cards, or remove cards.

That's where Heimdall really stands out to me. It has a very nice Graphical User Interface, and beyond getting it installed and ready to go in a docker-compose (you can use the Portainer GUI to install Heimdall BTW) you work completely in the web browser for setup and configuration. It's really a great experience from the perspecitve of making a simple, easy to use dashboard for the average person. The other really great feature is the ability to have multiple users and separated dashboards for each user when desired.

Today we'll install Heimdall using Docker-CE and Docker-Compose. After that, check out the video above for the quick GUI overview.

What you'll need

- A server or machine you want to run Heimdall from.
- Docker-CE and Docker-compose installed and ready (alternatively you can use Portainer if you have it already).
- About 30 minutes of your time (less if you just follow this guide and then jump into the UI on your own).

Installation

If you don't already have Docker and Docker-Compose installed, you'll want to get those setup first.

I have a script out on GitHub that will install Docker-CE, Docker-Compose, NGinX Proxy Manager, and Portainer-CE (all optional) for you.

Just open a terminal and run the following command to pull down the script to your local machine:

`wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh`

This will download a script called "install_docker_nproxyman.sh" to your current directory.

Change the permissions on the file to allow it to run with:

`chmod +x install_docker_nproxyman.sh`

and then run the script with:

`./install_docker_nproxyman.sh`

You'll be prompted to identify your OS/Distro. If you run an OS based on one of the options, simply select that option.

Next, you'll be asked if you want to install Docker, Docker-CE, NGinX Proxy Manager, and / or Portainer-CE.

Feel free to install them all, or just Docker and Docker-Compose. that's completely up to you.

Installing Heimdall

Now, we want to create a directory for Heimdall.

`mkdir heimdall`

and then move into that directory:

`cd heimdall`

Next we need to create a file calleed "docker-compose.yml":

`nano docker-compose.yml`

Now we need to paste the following into the file we've just opened.

```
version: "2.1"
services:
  heimdall:
    image: lscr.io/linuxserver/heimdall
    container_name: heimdall
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=America/Chicago
```

```
    volumes:
      - /home/<your-user>/heimdall/config:/config
    ports:
      - 8080:80
    restart: unless-stopped
```

You'll want to change a couple of things in the file you just pasted.

1. change the TZ (timezone) if needed, to be the correct timezone for your location.
2. change the PUID and PGID to be your user's group and user IDs. You can find them in the terminal by typing the command `id`.
3. On the left side of the colon ":" in the volume section, make sure to set the path to where you have created the "heimdall" folder above.
4. On the left side of the colon ":" in the ports section, make sure to set a port that is not in use on your host. If 8080 is free, then just use it.

Now, save the file with CTRL + O, then Enter to confirm, and exit the nano editor with CTRL + X.

Now, you just need to run the docker-compose command to bring up your dashboard.

`docker-compose up -d`

Give it time to download the image, and start the container. When you see "done" in the terminal, give it another 10 seconds or so, and then in your web browser of choice, go to the ip address of the host machine, and the port you set above.

In my case I went to:

`https://192.168.10.26:8280`

From there, you can go through the settings, setup new users, setup your preferred default search provider, and setup application shortcuts.

Enjoy!

Dashy

Dashy - Powerful, Informative, Configurable Self-Hosting Dashboard

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

Along the same lines as Homer, Heimdall, and Monitorr; I wanted to continue our look into personal web dashboards. I've posted about those few, and talked about others as well.

Sometimes the question comes up, and it's a valid one, "Why would I use this over the bookmarks tooldbar in (Chrome, firfox, etc)?"

The thing these dashboards give you, is a much lovelier UI, but also many of them provide some extra information. Heimdall and Homer can be setup to provide details about Pi-hole, qBittorrent, Sonarr, etc. Dashy gives you up / down indications at a glance, and as it's an active project, will hopefully get some more status features moving forward.

The one thing I really like about Dashy is it's various methods for configuration. You can configure it through the terminal directly in the configuration file, or you can go through the User Interface with a nicely laid out configuration tool, an interactive (WYSIWYG) configuration editor, and / or just update the .yml right there in the browser window.

For me, a person who updates their dashboard semi-regularly, it's a nice convenience to not have to leave the Web User Interface and SSH into another machine to make changes / updates.

What You'll Need

- Docker-CE
- Docker-Compose
- (Optional) NGinX Proxy Manager
- (Optional) Portainer-CE
- About 10 minutes of your time (not including adding items to the Dashboard)

Installation

Check out this video on installing Docker, Docker-Compose, NGinX Proxy Manager, and Portainer-CE with a single script in under 5 minutes.

For organizational purposes, you should run your docker containers (docker or docker-compose) from a folder called docker. Inside that folder you should create sub-folders for each application you run. So, inside the "docker" folder, we'll create a new folder called "dashy".

`cd docker`

`mkdir dashy`

`cd dashy`

Now, we want to make a couple of sub-folders inside of our new "dashy" folder. They'll be called "public" and "icons".

`mkdir {public,icons}`

NOTE: You can create your "dashy" sub-folder, and your "public" and "icons" sub-folders in a single command with:

`mkdir -p dashy/{public,icons}`

You can do a quick listing of your "dashy" directory with the command:

`ls`

Mkae sure both folders show up.

Now we'll create a simple text file called "docker-run.txt"

`nano docker-run.txt`

Inside of the file, you'll want to paste the code block below, and then we'll make any necessary modifications for your system.

```
docker run -d \
  -p 8295:8080 \
  --volume </path/to/your>/docker/dashy/public/conf.yml:/app/public/conf.yml \
  --volume </path/to/your>/docker/dashy/icons:/app/public/item-icons/icons \
  --name dashy \
  --restart=unless-stopped \
  lissy93/dashy:latest
```

or you can use this docker compose file.

```
name: dashy
services:
    dashy:
        ports:
            - 8295:8080
        volumes:
            - </path/to/your>/docker/dashy/public/conf.yml:/app/public/conf.yml
            - </path/to/your>/docker/dashy/icons:/app/public/item-icons/icons
        container_name: dashy
        restart: unless-stopped
        image: lissy93/dashy:latest
```

and save it as "docker-compose.yml".

Now, we'll make a couple of modifications as needed. First, if your host machine has port 8290 in use, change the port on the left side of the colon ':' to one that is not in use on your host.

Next, change the portion in the two volume mappings with '< >' around it to be the proper path of your docker folder.

In my case I have

`/home/brian/docker/dashy/public/conf.yml`

and

`/home/brian/docker/dashy/public/icons`

Again, only change the left side of the colon.

Now, save the file with CTRL+O, then press Enter to confirm, and exit the nano editor with CTRL+X.

Our Initial Configuration File

For us to start with a fairly clean Dashy install, we'll want to use a fairly small conf.yml file.

So make a conf.yml text file in the public sub-folder with the command:

`nano public/conf.yml`

Inside this file, you'll paste the following code block:

```
appConfig:
  theme: colorful
  layout: auto
  iconSize: medium
  language: en
pageInfo:
  title: Home Lab
  description: Welcome to your Home Lab!
  navLinks:
    - title: GitHub
      path: https://github.com/Lissy93/dashy
    - title: Documentation
      path: https://dashy.to/docs
  footerText: ''
sections:
  - name: Starter Only
    icon: fas fa-server
    items:
      - title: Google
        description: Search
        url: https://google.com
```

Now, we'll save this file with CTRL+O, press Enter to confirm, and use CTRL+X to exit the nano editor.

Authentication

If you want to add some Authentication on top of Dashy, I highly recommend Authelia, but there is a built in authentication piece as well.

You can modify the above block as follows:

```
appConfig:
  theme: colorful
  layout: auto
  iconSize: medium
  language: en
  auth:
    users:
      - user: your-preferred-username
```

```
        hash: hash-of-a-password-you-choose-using-sha256-hashing
        type: admin
pageInfo:
  title: Home Lab
  description: Welcome to your Home Lab!
  navLinks:
    - title: GitHub
      path: https://github.com/Lissy93/dashy
    - title: Documentation
      path: https://dashy.to/docs
  footerText: ''
sections:
  - name: Starter Only
    icon: fas fa-server
    items:
      - title: Google
        description: Search
        url: https://google.com
```

Getting Icons for your Dashboard

We'll be pulling a really great github repo down that has tons of icons for self-hosted applicationis, adn this just really pulls together any dashboard, regardless of which one you use.

So, move into the "icons" sub-folder we create earlier, and use the command:

`git clone https://github.com/walkxcode/dashboard-icons.git`

Clone the github repository.

Once pulled down, do an `ls` and you'll see a new folder called "dashboard-icons". Inside that you'll find several folders, but you'll likely be most interested in the "png" folder, which holds the png files for all of the icons in the repository.

Now, move back one step into your "dashy" folder with

`cd ..`

Pull and Run Dashy

Let's get our `docker run` command by running the command:

```
cat docker-run.txt
```

This will list out your docker-run command in the terminal. Highlight it, copy it, and paste it into the waiting terminal prompt. Press Enter, and docker will pull down the latest dashy image, and start it running using our starter conf.yml file.

Once it runs, give it about 1 minute, then go to your browser and enter the host machine's IP address, and enter the port you set on the left side of the port mapping in the docker run command.

For instance, my host is at 192.168.10.26, and I used port 8295, so I went to

```
http://192.168.10.26:8295
```

If all went according to plan, you should see your Dashy starter page load up.

Now you can start configuring, theming, and making your own special dashy dashboard page. Check out the video for more on how to navigate the Dashy User Interface, and how to configure your ultimate dashboard!

Homepage

Install Homepage

https://www.youtube.com/embed/3Ux7zfCCM1A

Self hosted dashboards are incredibly useful for a myriad of reasons. They provide quick access to all of your self hosted services, quick links to bookmarks of your most used online sites and services, and can offer some very useful monitoring information about your services at a glance.

Homepage is no different in this regard, and is a fast loading piece of software to boot. One of the nicest features is that as you save changes to the configuration file, the page reloads quickly to show you how your changes have affected the page layout. This is something I've struggled with in Dashy since I started using it, having to work around caching in the browser by opening the dev tools and refreshing the page multiple times.

What you'll need

- Docker and Docker Compose
- The most general familiarity with how to open a terminal emulator (command prompt)
- About 10 minutes of your time

Installation

Installation of Docker and Docker Compose via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget https://gitlab.com/bmcgonag/docker_installs/-/raw/main/install_docker_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install_docker_nproxyman.sh
```

and then run the script with the command:

```
./install_docker_nproxyman.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Installing Homepage

Now that we have docker and docker-compose installed, let's get ready to install Homepage. First we'll setup our desired folder structure. I put all of my docker applications inside of a top level (parent) folder called "docker". This makes it very easy to run a script that will backup all of my docker applications and compress them into a zipped format.

`mkdir -p docker/homepage`

This command will create the docker folder if it does not already exist, and will use the existing one if it does. Then it will create the homepage folder inside of that parent docker folder.

Now we'll move into the homepage folder and create a new file called docker-compose.yml.

`cd docker/homepage`

`nano docker-compose.yml`

In this case, I'm using the nano text editor directly in the terminal, but you can use any text editor you are more comfortable with. Make sure the editor is not a rich-text editor by default, as the yaml we will be adding is space dependent, and can be messed up with hidden rtf characters.

> Adding a new environment variable based on latest updates to Homepage. In this newer version you need to tell Homepage where it should expect to be accessed at. So if you use IP and port, you'll want o put that, but if you use a sub-domain like 'homepage.mydomain.net', then you'll want to enter that subdomain.

Copy the yaml code from the block below, and paste it into your new docker-compose.yml document.

```
---
services:
  homepage:
    image: ghcr.io/benphelps/homepage:latest
```

```
    container_name: homepage
    envrionment:
       - PUID: 1000
       - PGID: 1000
       - HOMEPAGE_ALLOWED_HOSTNAMES: "192.168.1.21:8921" # can be comma separated for multiple
entries if needed.
    ports:
      - 8921:3000
    volumes:
      - ./config:/app/config # Make sure your local config directory exists
      - /var/run/docker.sock:/var/run/docker.sock:ro # (optional) For docker integrations
    # user: 1000:1000 optional, not compatibile with direct socket see
https://gethomepage.dev/en/configs/docker/#using-socket-directly
    restart: unless-stopped
```

The changes you may want / need to make is to the port mapping (8921:3000), and to the environment variable HOMEPAGE_ALLOWED_HOSTNAMES. You can always change the left side port in a docker port mapping, just not the right side. The left side of the mapping is how you'll access the application once it's up and running. You simply want to make sure that it's not trying to use a port already in use on the host system. In this case I changed 3000 (the original left side port number) to 8921 in order to avoid a very common port used in nodejs based applications. As long as port 8921 is free on your host system, you won't need to change it.

Save the docker-compose.yml document with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Run the Application

Now we are ready to run our application for the first time. We'll use the following command to run it:

`docker compose up -d`

If you have an older version of docker-compose, you may need to put a hyphen (-) between the words docker and compose, like this:

`docker-compose up -d`

NOTE: the un-hyphenated version is the newer version of docker-compose.

Allow docker to pull down the latest image of Homepage, and start up the container. When it's complete you'll be back at a normal terminal prompt.

Now go to your favorite modern browser and enter the IP address of your host machine (if you're running the application on the physical machine you're working on, you can use

http://localhost:8921) and the port 8921 (unless you changed the left side of the port mapping to another number, in which case you should use that port number. In my case, I went to:

http://192.168.10.154:8921

You should now be presented with the default Homepage layout screen. After loading the web page for the first time, you'll find that back in the terminal you now have multiple yaml files in the "config" folder. These are initially created after you load the page for the first time.

You can now start modifying the yaml files to customize your Homepage as you please. I highly recommend reading the documentation on the configuration options, as there are many, and you can get very detailed on what can be displayed.

https://gethomepage.dev/en/installation/

For a solid starting spot, I recommend opening the "services.yaml" and "bookmarks.yaml" files and playing with those a bit. You can add multiple service sections under each group, and you can add more groups as well in the services.yaml file. The same goes for the bookmarks.yaml file.

Make sure to check out the video for more detail on how to do that as well.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:
https://www.patreon.com/awesomeopensource

Labdash

Labdash

Install and Configure Labdash

https://www.youtube.com/embed/CCEfU-ch9i4

Install Updates

Update and upgrade your server's packages using the following commands:

- For Ubuntu/Debian: `sudo apt update && sudo apt upgrade -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf update -y`

Create a Non-Root User

Create a non-root user with superuser (sudo) privileges:

1. Add a new user using `adduser <username>`
2. Set the password for this user.
3. Enter the relevant information (optional)
4. Enter 'Y', then press Enter.
5. Add the user to the "sudo" group:
   - For Ubuntu/Debian: `usermod -aG sudo <username>`
   - For RedHat/CentOS/Fedora/Alma/Rocky: `usermod -aG wheel <username>`

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.

Step 2: Install Docker and Docker Compose

Install Docker and Docker Compose on your server:

1. Install the `curl` utility:

- For Ubuntu/Debian: `sudo apt install curl -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf install curl -y`

2. Run the command to install Docker and Docker Compose: `curl https://get.docker.com | sh`

Add Your User to the Docker Group

Add your non-root user to the docker group so you can use Docker commands without sudo:

`sudo usermod -aG docker <username>`

Install Labdash in Docker

First we'll create a new folder to hold our applications and services files.

`mkdir -p docker/labdash`

Now we'll move into that folder.

`cd docker/labdash`

Here we'll create a new file in this location. It will be `compose.yaml`, which is our docker compose configuration file for the application.

```
---
services:
  lab-dash:
      container_name: lab-dash
      image: ghcr.io/anthonygress/lab-dash:latest
      privileged: true
      ports:
        - 2022:2022
      environment:
        - SECRET=<some random string of numbers and letters>
        # You can run `openssl rand -base64 32` to generate a key
      volumes:
        - ./sys:/sys:ro
        - ./config:/config
        - ./uploads:/app/public/uploads
        - /var/run/docker.sock:/var/run/docker.sock
      restart: unless-stopped
```

Notice any values in the file above surrounded by less than '<' and greater than '>' signs. These values you need to replace with actual values. In this particular docker compose file, you need to create a secret key. We can do this by exiting the nano editor with CTRL + x.

Next, run the command:

`openssl rand -base64 32`

Highlight and copy the value generated. Go back into your docker-compose.yml file with

`nano docker-compose.yml`

Move to the SECRET line, and clear any placeholder text. Now paste the value you just copied into the file. Save your changes with CTRL + O, then press Enter to confirm, and exit nano once again with CTRL + x.

Now we'll pull our images for Labdash with the command:

`docker compose pull`

Once it's pulled down, we can start our Labdash virtual machine with:

`docker compose up -d`.

I like to watch the logs the first time I start up a new docker virtual machine, to check for any errors I may need to correct. If you'd like to do that then use the command:

`docker compose logs -f`

after bringing up the virtual machine.

You can also just type them both in one line by concatenating the commands with `&&`.

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

When you are done checking the logs, you can stop following them with CTRL + C.

Now, open your favorite browse, and go to the IP address of the **host** machine you are running the docker virtual machine on, and port 2022. This will bring you to the initial page of Labdash. You'll run through a short informational wizard, then it will prompt you to get signed up as the administrative user.

Once you're signed in, you can start building out a dashboard to help you quickly access all of your self-hosted services and applications, and to help you keep an eye on their status and all kinds of other great information available through a multitude of widgets.

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://patreon.com/awesomeopensource

Paypal

https://paypal.me/brianmcgonagill

Merch
https://osia.printify.me

Homarr

Homarr

Install Homarr Dashbaord

https://www.youtube.com/embed/YoVlZ_HvT9w

Install Updates

Update and upgrade your server's packages using the following commands:

- For Ubuntu/Debian: `sudo apt update && sudo apt upgrade -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf update -y`

Create a Non-Root User

Create a non-root user with superuser (sudo) privileges:

1. Add a new user using `adduser <username>`
2. Set the password for this user.
3. Enter the relevant information (optional)
4. Enter 'Y', then press Enter.
5. Add the user to the "sudo" group:
   - For Ubuntu/Debian: `usermod -aG sudo <username>`
   - For RedHat/CentOS/Fedora/Alma/Rocky: `usermod -aG wheel <username>`

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.

Step 2: Install Docker and Docker Compose

Install Docker and Docker Compose on your server:

1. Install the `curl` utility:

- For Ubuntu/Debian: `sudo apt install curl -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf install curl -y`

2. Run the command to install Docker and Docker Compose: `curl https://get.docker.com | sh`

Add Your User to the Docker Group

Add your non-root user to the docker group so you can use Docker commands without sudo:

`sudo usermod -aG docker <username>`

Install Homarr in Docker

First we'll create a new folder to hold our applications and services files.

`mkdir -p docker/homarr`

Now we'll move into that folder.

`cd docker/homarr`

Here we'll create a new file in this location. It will be `compose.yaml`, which is our docker compose configuration file for the application. Copy the configuration text below, and paste it into the new file you just created.

```
---
services:
  homarr:
    container_name: homarr
    image: ghcr.io/homarr-labs/homarr:latest
    restart: unless-stopped
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock # Optional, only if you want docker
integration
      - ./homarr/appdata:/appdata
    environment:
      - SECRET_ENCRYPTION_KEY=< use 'openssl rand -hex 32' >
    ports:
      - '7575:7575'
```

In the configuration above, you'll notice any values surrounded by less than '<' and greater than '>' signs. These need to be replaced with the actual values you need.

In this case you only need to generate a secret encryption key. We can do this by exiting nano with CTRL + x, then using the command:

```
openssl rand -hex 32
```

in the command line to generate a 32 character hexadecimal value. Once generated, copy the value and go back into the file in nano with

```
nano compose.yaml
```

Now, erase the placeholder for the SECRET_ENCRYPTION_KEY value between '<' and '>', including the less tan and greater than signs, and paste in your generated key. It should look like this, when you've done it.

```
SECRET_ENCRYPTION_KEY=f387d2a8bdae825763...
```

Now save you file with CTRL + O, then press Enter to confirm. Exit nano with CTRL + X.

Next, we'll pull down our image for Homarr with

```
docker compose pull
```

Once pulled down, we will start our docker virtual machine with

`docker compose up -d`.

I like to watch the logs the first time I start up a new docker virtual machine, to check for any errors I may need to correct. If you'd like to do that then use the command:

```
docker compose logs -f
```

after bringing up the virtual machine.

You can also just type them both in one line by concatenating the commands with `&&`.

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

When you are done checking the logs, you can stop following them with CTRL + C.

Now, open your favorite modern browser, and go to the IP address of your **host** machine where you have the docker virtual machine running on port 7575.

I went to `http://192.168.50.115:7575` and was greeted with a login screen.

You'll be greeted and put through a quick on-boarding where you'll create your initial (administrative) account. Once done, you can go through, setup the system the way you prefer, and get started setting up your dashboards.

Support this Channel and Content

Become a Patron at Patreon

https://patreon.com/awesomeopensource

Be me a Coffee or Beer at Paypal

https://paypal.me/brianmcgonagill

Get the Awesome Open Source Merchandise

https://osia.printify.me