Network / Internet Speedtests

- Speedtest-Tracker
  - Setting up Speedtest-Tracker
- LibreSpeed and OpenSpeedTest

Speedtest-Tracker

Setting up Speedtest-Tracker

https://www.youtube.com/embed/iyRUj77cjKg

> Since this video was made, the process has changed just a bit. The docker-compose now includes a Maria DB database, but overall it will works just as well. I did, however have to delete my old config folder, and start fresh.

I've shown you several different open source, and self hosted, speedtest applications over the years. I showed you one called Docker Sppedtest Grafana a couple of years ago, but recently the install doesn't work as expected for me anymore, and I've heard similar reports from others who've tried it as well. So, I started looking for another option, and found Sppedtest Tracker by henrywhitaker3 over on Github. I've used it for about 6 months, and it's been very useful, but again it's gone unmaintained for over a year now. Luckily, another person picked it up and continued the project by forking it, and enhancing it quite a bit. So today we'll be looking at Speedtest Tracker by alexjustesen over at Github.

What You'll Need

- Docker-CE and Docker-Compose installed on the server / machine you want to use for running speed tests.
  - it's better to run them from a wired connection if at all possible.
- (Optional) If you want to setup a domain name for your Speed Test web interface, you'll want a reverse proxy (I use NginX Proxy Manger)
- (Optional) SMTP Details for an Email Server if you want email notifications of speed test data
- About 30 minutes of your time

Installation 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 Speedtest Tracker

SSH to your chosen machine, or if you can sit in front of it, and access the terminal / command line, then do that. Once connected, I like to create a folder structure that allows me to backup my docker applications easily. I put everything inside a parent "docker" folder. So, we'll make sure we put our application folder "speedtest-tracker" into our docker folder. If you don't have a "docker" folder already, don't worry, we'll create both with 1 command.

```
mkdir -p docker/speedtest-tracker
```

This command will create the folders that don't exist already, but will not duplicate them if they do.

Once we've created this structure, we'll move into the 'speedtest-tracker' folder, and create our 'docker-compose.yml' file.

```
cd docker/speedtest-tracker
```

```
nano docker-compose.yml
```

Once you've opened the new 'docker-compose.yml' file, you'll want to copy and paste the following yaml code into it. You'll need to make some value updates based on your system needs. I'm going to put two versions of the yaml code below. The first version will be to run the application **without** email. The second will be to run it with email sending functions, and for it you'll need the SMTP server information for your email.

No Email Notifications:

```
version: '3.3'
services:
   speedtest-tracker:
       container_name: speedtest-tracker
       ports:
           - '8982:80'
           - '8943:443'
       environment:
           - PUID=1000
           - PGID=1000
           - DB_CONNECTION=mysql
           - DB_HOST=db
           - DB_PORT=3306
           - DB_DATABASE=speedtest_tracker
           - DB_USERNAME=speedy
           - DB_PASSWORD=password
       volumes:
           - './config:/config'  # left side can be any path you want, but this keeps it in
the folder
           - '/etc/localtime:/etc/localtime:ro'
           - './web:/etc/ssl/web'
       image: 'ghcr.io/alexjustesen/speedtest-tracker:latest'
       restart: unless-stopped
       depends_on:
         - db
     db:
       image: mariadb:10
       restart: always
       environment:
         - MARIADB_DATABASE=speedtest_tracker
         - MARIADB_USER=speedy
         - MARIADB_PASSWORD=password
         - MARIADB_RANDOM_ROOT_PASSWORD=true
       volumes:
         - ./speedtest-db:/var/lib/mysql
```

With Email Notifications

```
version: '3.3'
services:
```

```
   speedtest-tracker:
       container_name: speedtest-tracker
       ports:
           - '8982:80'   # feel free to change the port on the left side of the colon
           - '8943:443'  # feel free to change the port ont he left side of the colon
       environment:
           - PUID=1000
           - PGID=1000
           - MAIL_HOST=smtp.myemailserver.io
           - MAIL_PORT=587
           - MAIL_USERNAME=no-reply@myemailserver.io
           - MAIL_PASSWORD=a-long-st0n6-pa55w0rd!
           - MAIL_ENCRYPTION=tls
           - MAIL_FROM_ADDRESS="my-email@myemailserver.io"
           - DB_CONNECTION=mysql
           - DB_HOST=db
           - DB_PORT=3306
           - DB_DATABASE=speedtest_tracker
           - DB_USERNAME=speedy
           - DB_PASSWORD=password
       volumes:
           - './config:/config'  # left side can be any path you want, but this keeps it in
the folder
           - '/etc/localtime:/etc/localtime:ro'
           - './web:/etc/ssl/web'
       image: 'ghcr.io/alexjustesen/speedtest-tracker:latest'
       restart: unless-stopped
       depends_on:
         - db
     db:
       image: mariadb:10
       restart: always
       environment:
         - MARIADB_DATABASE=speedtest_tracker
         - MARIADB_USER=speedy
         - MARIADB_PASSWORD=password
         - MARIADB_RANDOM_ROOT_PASSWORD=true
       volumes:
         - ./speedtest-db:/var/lib/mysql
```

Once you've made adjustments to any ports (on the left side of the colon ':' only), and if you elect to use SMTP email, make sure you update all MAIL_* values to match your SMTP server settings, you can then save the file with CTRL + O, then Enter to confirm, and CTRL + X to exit.

> If you're updating from the version that did not include the 'db' section or the MariaDB database, you'll probably need to delete your current config folder.
>
> `sudo rm -rf config`

Starting Our Speedtest Tracker Server

We now start our server with the command:

`docker-compose up -d`

I like to run a second command, which you can run separately after the container is up, or you can run it with two ampersands '&' between the commands:

`docker-compose logs -f`

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

Either way, you'll be able to watch the logs as the application starts up. You're looking for anything that stands out as an error. If you happen to hit errors, you can get some information that may help you fix the issues, and at the very least to report issues to the Github project.

After about 30 seconds to a minute (depending on the hardware you are running on), the application should be up and running. You can now navigate to the site in a modern web browser by entering the IP Address and the port number of the application.

`http://192.168.10.154:8982`

is where I would go. You'll want to use your server's IP address, and the port you entered on the left side of the colon ':' in the port mapping in your docker-compose file.

Once at the login page, you'll want to login using the default credentials. At the time of writing, the default login will be

username: admin@example.com

password: password

First thing you'll want to do, once you're logged in, is update the Admin user. On the left panel, find the 'Users' section, and click on it. In the users table, click the pencil icon in order to edit the default Administrative user. In the form, change the default email address to your email, and update the password to a long, strong password. Save it in a password manager like Vaultwarden / Bitwarden.

Next, you'll likely want to run your first speed test. Back on the Dashboard screen, click on the 'Queue a Speedtest' button, then patiently wait a minute or so, and the interface should update to show you the results.

Finally, go through the settings, and make sure you set the system up the way you want it. Take a look at setting your time zone, the frequency of tests, and make sure to setup notifications if you want them.

You've done it, and now you have a great tool to help you ensure you are getting the speeds you expect, and most likely pay your ISP for.

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

https://www.patreon.com/awesomeopensource

LibreSpeed and OpenSpeedTest

https://www.youtube.com/embed/7S8f_xEpCaQ

In the past we've gone over some great tools for checking your internet and network speeds from the command line, but there is really nothing quite like a really nice GUI (Graphical User Interface) for doing something like a speed test.

Today, I wantt o introduce you to not one, but two really great, really stratight-forward GUI's for running network and internet speedtests on your own hardware or VPS.

No more running a speedtest provided by your ISP when you are experiencing speed issues, and letting them hang up with the presumption that the problem is all on you.

What You'll Need

- Docker-CE
- Docker-Compose
- (Optional) NGinX Proxy Manager
- (Optional) a domain or subdomain for your speed test site(s).
- (Optional) Portainer-CE

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

Install and Setup

OpenSpeedTest

First, we'll setup OpenSpeedTest. It's a single `docker run` command, and really fast to get going. As always, I like to begin with a bit of organization. If you're just starting out, or a docker pro, it never hurts to keep your docker installs organized. I like to create a "docker" directory, then place folders inside that directory for each docker application / stack I run.

If you're new to the command line, then here are some simple commands to get you setup, if you're a pro, and already have your docker setup organized, feel free to skip to the "docker run" section below.

First let's create our main docker folder:

`mkdir docker`

Now let's move into that folder, and create our folders for both of the speedtest apps we want to install today.

`cd docker`

`mkdir {openspeedtest, librespeed}`

That command will create two folders. You can verify they were created with command:

`ls`

Make sure you see each folder listed. If so, let's move on to setting up our docker run command for OpenSpeedTest.

Docker Run

We'll move into our openspeedtest directory, and create a text file for our docker run command. I like to do this so I can reference it later (in case I need to re-build or re-install for any reason). This also helps me keep track of changes I may make as I go.

`cd openspeedtest`

Now we'll create our text file with

`nano docker-run.txt`

Inside that file copy the text below and paste it in:

```
docker run --restart=unless-stopped \
--name openspeedtest -d \
-p 8191:3000 \
-p 0192:3001 \
openspeedtest/latest
```

Once pasted, you can save the file with CTRL + O, then Enter to confirm, and exit the nano text editor with CTRL + X.

Before exiting, however, make sure you change the ports listed after the `-p` arguments to ports that are open on your host machine. You can change the number on the left side of the colon ':', but don't change the right side.

Once you've made changes, saved, and exited the file, do the following command to list the file contents to your terminal:

`cat docker-run.txt`

Highlight the listed `docker run` command fully, all the way through the word "latest", then paste it at the terminal prompt.

Press enter, and the command should run. This will pull down the openspeedtest image, and start the container on the ports you specify.

As long as you dont' see any error messages, you can go to the IP and port of your host machine in the web browser, and you should see the Open Speed Test GUI displayed. Click the start button to run your test.

For instance, I installed on my host with an ip of 192.168.10.26, so I went to `http://192.168.10.26:8192`

LibreSpeed

For LibreSpeed, we'll be using docker-compose, an excellent tool for providing a little bit more order to the docker-run command, and it has some great tooling built around it for updates in place, restarts, and so on.

first, let's back out of the "openspeedtest" folder, and get into the librespeed folder we created above.

`cd ..` will take you back one level in the folders, then you can use

`cd librespeed` to get into the librespeed folder.

Now we want to create a new "docker-compose.yml" file and put some yaml code in it that will tell docker how to run the application.

`nano docker-compose.yml`

Once the file is open, paste the following code into it:

```
---
version: "2.1"
services:
  librespeed:
    image: lscr.io/linuxserver/librespeed
    container_name: librespeed
    environment:
      - PUID=1000  # change this to your user id
      - PGID=1000  # change this to your group id
      - TZ=America/Chicago # change this to your timezone
```

```
      - PASSWORD=PASSWORD
    volumes:
      - /home/<your user name>/docker/librespeed/config:/config
    ports:
      - 8190:80
    restart: unless-stopped
```

Once you've pasted the above text into the file, you'll want to change a few things potentially. First, let's just save what you have with CTRL + O, then enter to confirm. Now, let's briefly exit nano with CTRL + X.

Now, let's check your user's UID and GID in the command line. Type the "id" command:

`id`

From the output identify your UID and GID numbers, and make note of them.

Next enter the 'pwd' command

`pwd`

and highlight / copy your current directory. This is the path we'll want for our volume mapping.

Now go back into your docker-compose file

`nano docker-compose.yml`

Replace the PUID with your UID number, and the PGID with yoru GID numbeer if needed.

Next, make sure to change the timezone to your timezone.

Finally, in the volume mapping make sure to paste the path to your librespeed directory, and add /config to it.

My path from the `pwd` command was /home/brina/docker/librespeed, andn I added /config to it to make the full path:

`/home/brian/docker/librespeed/config`

Once you've made the necessary changes, save the file with CTRL+O, then press Enter to confirm, and exit nano with CTRL + X.

Now we'll run our docker-compose file to pull down LibreSpeed and start it running.

`docker-compose up -d`

Give it time to pull down, and when you see the "done" message, you can navigate to your IP and the port specified in the compose file on your host machine. You should see the LibreSpeed GUI, and you can start a speed test.

NGinX Proxy Manager

If you want to run your speedtest(s) on a VPS, you can absolutely do so using the IP address alone, but if you'd prefer to set it up with a domian / subdomain name, you may want to use a tool like NGinX Proxy Manager. You can then proxy the traffic to your speedtest(s). In particular, a proxy manager is handy if you are wanting to run more than one speed test app on a single server.

I have several videos and how to's on using NGinX Proxy Manager, so definitely get out there and give it a try if you'd like to.

Support my Channel and Content

Support my Channel and ongoing efforts through Patreon:

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