WGDashboard Documentation

Table of contents

📣 What's New: v4.1

🎉 New Features

- **Multi-Language Support**: Now WGDashboard support the following language on its user interface, big thanks to our user's contribution!
    - Czech (@jursed)
    - German (@orangeferdi)
    - English
    - Italian (@3vis97)
    - Dutch (@DaanSelen)
    - Russian (Pixnet)
    - Ukrainian (@shuricksumy)

🛠️ Some Adjustments

🧐 Bugs Fixed

- Mobile UI issues in #353
- Removed WireGuard configuration error alert from Gunicorn start in #328
- Sometimes restrict peer might not be success in #357
- Weird SQLite error causing WGDashboard to crash in #366

🥘 Experimental Features

- **Cross-Server Access**: Now you can access other servers that installed `v4` of

WGDashboard through API key.

- **Desktop App**: Thanks to **Cross-Server Access**, you can now download an ElectronJS based desktop app of WGDashboard, and use that to access WGDashboard on different servers.

> ⚠ For more information, please visit 🥘 Experimental Features

💡 Features

- Automatically look for existing WireGuard configuration under `/etc/wireguard`
- Easy to use interface, provided credential and TOTP protection to the dashboard
- Manage peers and configuration
    - Add Peers or by bulk with auto-generated information
    - Edit peer information
    - Delete peers with ease
    - Restrict peers
    - Generate QR Code and `.conf` file for peers, share it through a public link
    - Schedule jobs to delete / restrict peer when conditions are met
- View real time peer status
- Testing tool: Ping and Traceroute to your peer

📝 Requirements

1. Supported operating systems. Please view the list below.
2. WireGuard & WireGuard-Tools (`wg-quick`)
3. Python 3.10 / 3.11 / 3.12
4. `git`, `net-tools`, `sudo` (*This should only apply to RHEL 9 & 8, interestingly it doesn't have it preinstalled)*

Supported Operating Systems

> ℹ️ All operating systems below are tested by myself. All are **ARM64** ran in UTM Virtual Machine.

Ubuntu

- 20.04 LTS
- 22.04 LTS
- 24.02 LTS

Debian

- 12.6
- 11.10

Red Hat Enterprise Linux

- 9.4

CentOS

- 9-Stream

Fedora

- 40
- 39
- 38

Alpine Linux

- 3.20.2

⚠ If you installed WGDashboard on other systems without any issues, please let me know. Thank you!

Existing WireGuard Configurations

ℹ This only applies to existing WireGuard Configuration under `/etc/wireguard`

```
[Interface]
...
SaveConfig = true
...
```

```
[Peer]
#Name# = Donald's iPhone
PublicKey = abcd1234
AllowedIPs = 1.2.3.4/32
```

⚠ With `v4`, WGDashboard will look for entry with `#Name# = abc...` in each peer and use that for the name.

🛠️ Install

Install Commands

These commands are tested by myself in each OS. It contains commands to install WireGuard, Git, Net Tools, and even Python on some OS.

⚠️ Please make sure you understand these commands before you run them.

Ubuntu

20.04 LTS

```
sudo add-apt-repository ppa:deadsnakes/ppa -y && \
sudo apt-get update -y && \
sudo apt-get install python3.10 python3.10-distutils
wireguard-tools net-tools --no-install-recommends -y && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p
```

22.04 LTS and 24.02 LTS

```
sudo apt-get update -y && \
sudo apt install wireguard-tools net-tools --no-install-
recommends -y && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf
```

Debian

12.6

```
apt-get install sudo git iptables -y && \
sudo apt-get update && \
sudo apt install wireguard-tools net-tools && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf
```

11.10

⚠ This commands will download Python 3.10's source code and build from it, since Debian 11.10 doesn't come with Python 3.10

```
apt-get install sudo -y && \
sudo apt-get update && \
sudo apt install -y git iptables build-essential zlib1g-dev
libncurses5-dev libgdbm-dev libnss3-dev libssl-dev
libreadline-dev libffi-dev libsqlite3-dev wget libbz2-dev
wireguard-tools net-tools && \
wget https://www.python.org/ftp/python/3.10.0/Python-
3.10.0.tgz && \
tar -xvf Python-3.10.0.tgz && \
cd Python-3.10.0 && \
sudo ./configure --enable-optimizations && \
sudo make && \
sudo make altinstall && \
cd .. && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf
```

Red Hat Enterprise Linux

9.4

```
sudo yum install wireguard-tools net-tools git python3.11 -y
&& \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf && \
firewall-cmd --add-port=10086/tcp --permanent && \
firewall-cmd --add-port=51820/udp --permanent && \
firewall-cmd --reload
```

CentOS

9-Stream

```
sudo yum install wireguard-tools net-tools git python3.11 -y
&& \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf && \
firewall-cmd --add-port=10086/tcp --permanent && \
firewall-cmd --add-port=51820/udp --permanent && \
firewall-cmd --reload
```

Fedora

40, 39 and 38

```
sudo yum install wireguard-tools net-tools git -y && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf && \
firewall-cmd --add-port=10086/tcp --permanent && \
firewall-cmd --add-port=51820/udp --permanent && \
firewall-cmd --reload
```

Alpine Linux

3.20.2

```
sudo yum install wireguard-tools net-tools git -y && \
git clone https://github.com/donaldzou/WGDashboard.git && \
cd ./WGDashboard/src && \
chmod +x ./wgd.sh && \
./wgd.sh install && \
sudo echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf && \
sudo sysctl -p /etc/sysctl.conf && \
firewall-cmd --add-port=10086/tcp --permanent && \
firewall-cmd --add-port=51820/udp --permanent && \
firewall-cmd --reload
```

Manual Installation

⚠ To ensure a smooth installation process, please make sure you have the following installed:

- Python 3.10 or above
- `git`
- `wireguard-tools`
- `net-tools`

1. Download WGDashboard

```
git clone https://github.com/donaldzou/WGDashboard.git
wgdashboard
```

2. Open the WGDashboard folder

```
cd wgdashboard/src
```

3. Install WGDashboard

```
sudo chmod u+x wgd.sh && \
sudo ./wgd.sh install
```

4. Give read and execute permission to root of the WireGuard configuration folder, you can change the path if your configuration files are not stored in `/etc/wireguard`

```
sudo chmod -R 755 /etc/wireguard
```

5. Run WGDashboard

    ```
    sudo ./wgd.sh start
    ```

6. Access dashboard

    Access your server with port `10086` (e.g. http://your_server_ip:10086), using username `admin` and password `admin`. See below how to change port and ip that the dashboard is running with.

🪜 Usage

⚠️ The following commands are assumed you're in `wgdashboard/src`

Start

This command will start WGDashboard and run it in the background.

```
./wgd.sh start
```

Stop

This command will stop WGDashboard.

```
./wgd.sh stop
```

Restart

This command will stop and start WGDashboard in the background.

```
./wgd.sh restart
```

Debug Mode

This command will run WGDashboard in the foreground and output logs on the screen.

```
./wgd.sh debug
```

Run WGDashboard as a system service

⚠️ This feature require WGDashboard v2.0 or above

In the src folder, it contained a file called wg-dashboard.service, we can use this file to let our system autostart the dashboard after reboot. The following guide has tested on **Ubuntu**, most **Debian** based OS might be the same, but some might not. Please don't hesitate to provide your system if you have tested the autostart on another system.

1. Changing the directory to the dashboard's directory

   ```
   cd wgdashboard/src
   ```

2. Get the full path of the dashboard's directory

   ```
   pwd
   #Output: /root/wgdashboard/src
   ```

   For this example, the output is /root/wireguard-dashboard/src, your path might be different since it depends on where you downloaded the dashboard in the first place. **Copy the the output to somewhere, we will need this in the next step.**

3. Edit the service file, the service file is located in wireguard-dashboard/src, you can use other editor you like, here will be using nano

   ```
   nano wg-dashboard.service
   ```

   You will see something like this:

   ```
   [Unit]
   After=syslog.target network-online.target
   Wants=wg-quick.target
   ConditionPathIsDirectory=/etc/wireguard

   [Service]
   Type=forking
   PIDFile=<absolute_path_of_wgdashboard_src>/gunicorn.pid
   WorkingDirectory=<absolute_path_of_wgdashboard_src>
   ExecStart=<absolute_path_of_wgdashboard_src>/wgd.sh start
   ExecStop=<absolute_path_of_wgdashboard_src>/wgd.sh stop
   ExecReload=<absolute_path_of_wgdashboard_src>/wgd.sh restart
   ```

```
TimeoutSec=120
PrivateTmp=yes
Restart=always

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

Now, we need to replace all `<absolute_path_of_wgdashboard_src>` to the one you just copied from step 2. After doing this, the file will become something like this, your file might be different:

Be aware that after the value of `WorkingDirectory`, it does not have a `/` (slash). And then save the file after you edited it

4. Copy the service file to systemd folder

```
$ sudo cp wg-dashboard.service /etc/systemd/system/wg-
dashboard.service
```

To make sure you copy the file successfully, you can use this command `cat /etc/systemd/system/wg-dashboard.service` to see if it will output the file you just edited.

5. Enable the service

```
$ sudo chmod 664 /etc/systemd/system/wg-dashboard.service
$ sudo systemctl daemon-reload
$ sudo systemctl enable wg-dashboard.service
$ sudo systemctl start wg-dashboard.service  # <-- To start the
service
```

6. Check if the service run correctly

```
$ sudo systemctl status wg-dashboard.service
```

And you should see something like this

```
● wg-dashboard.service
Loaded: loaded (/etc/systemd/system/wg-dashboard.service;
enabled; vendor preset: enabled)
Active: active (running) since Wed 2024-08-14 22:21:47 EDT; 55s
ago
Process: 494968 ExecStart=/home/donaldzou/Wireguard-
Dashboard/src/wgd.sh start (code=exited, status=0/SUCCESS)
Main PID: 495005 (gunicorn)
Tasks: 5 (limit: 4523)
Memory: 36.8M
CPU: 789ms
CGroup: /system.slice/wg-dashboard.service
├─495005 /home/donaldzou/Wireguard-
Dashboard/src/venv/bin/python3 ./venv/bin/gunicorn --config
./gunicorn.conf.py
└─495007 /home/donaldzou/Wireguard-
Dashboard/src/venv/bin/python3 ./venv/bin/gunicorn --config
./gunicorn.conf.py

Aug 14 22:21:40 wg sudo[494978]:     root :
PWD=/home/donaldzou/Wireguard-Dashboard/src ; USER=root ;
COMMAND=./venv/bin/gunicorn --config ./gunicorn.conf.py
Aug 14 22:21:40 wg sudo[494978]: pam_unix(sudo:session): session
opened for user root(uid=0) by (uid=0)
Aug 14 22:21:40 wg wgd.sh[494979]: [WGDashboard] WGDashboard w/
Gunicorn will be running on 0.0.0.0:10086
Aug 14 22:21:40 wg wgd.sh[494979]: [WGDashboard] Access log file
is at ./log/access_2024_08_14_22_21_40.log
Aug 14 22:21:40 wg wgd.sh[494979]: [WGDashboard] Error log file
is at ./log/error_2024_08_14_22_21_40.log
Aug 14 22:21:40 wg sudo[494978]: pam_unix(sudo:session): session
closed for user root
Aug 14 22:21:45 wg wgd.sh[494968]: [WGDashboard] Checking if
WGDashboard w/ Gunicorn started successfully
Aug 14 22:21:47 wg wgd.sh[494968]: [WGDashboard] WGDashboard w/
Gunicorn started successfully
Aug 14 22:21:47 wg wgd.sh[494968]: -----------------------------
```

```
--------------------------------
Aug 14 22:21:47 wg systemd[1]: Started wg-dashboard.service.
```

If you see `Active:` followed by `active (running) since...` then it means it run correctly.

7. Stop/Start/Restart the service

```
sudo systemctl stop wg-dashboard.service      # <-- To stop the service
sudo systemctl start wg-dashboard.service     # <-- To start the service
sudo systemctl restart wg-dashboard.service   # <-- To restart the service
```

8. **And now you can reboot your system, and use the command at step 6 to see if it will auto start after the reboot, or just simply access the dashboard through your browser. If you have any questions or problem, please report it in the issue page.**

🗂️ User Guides

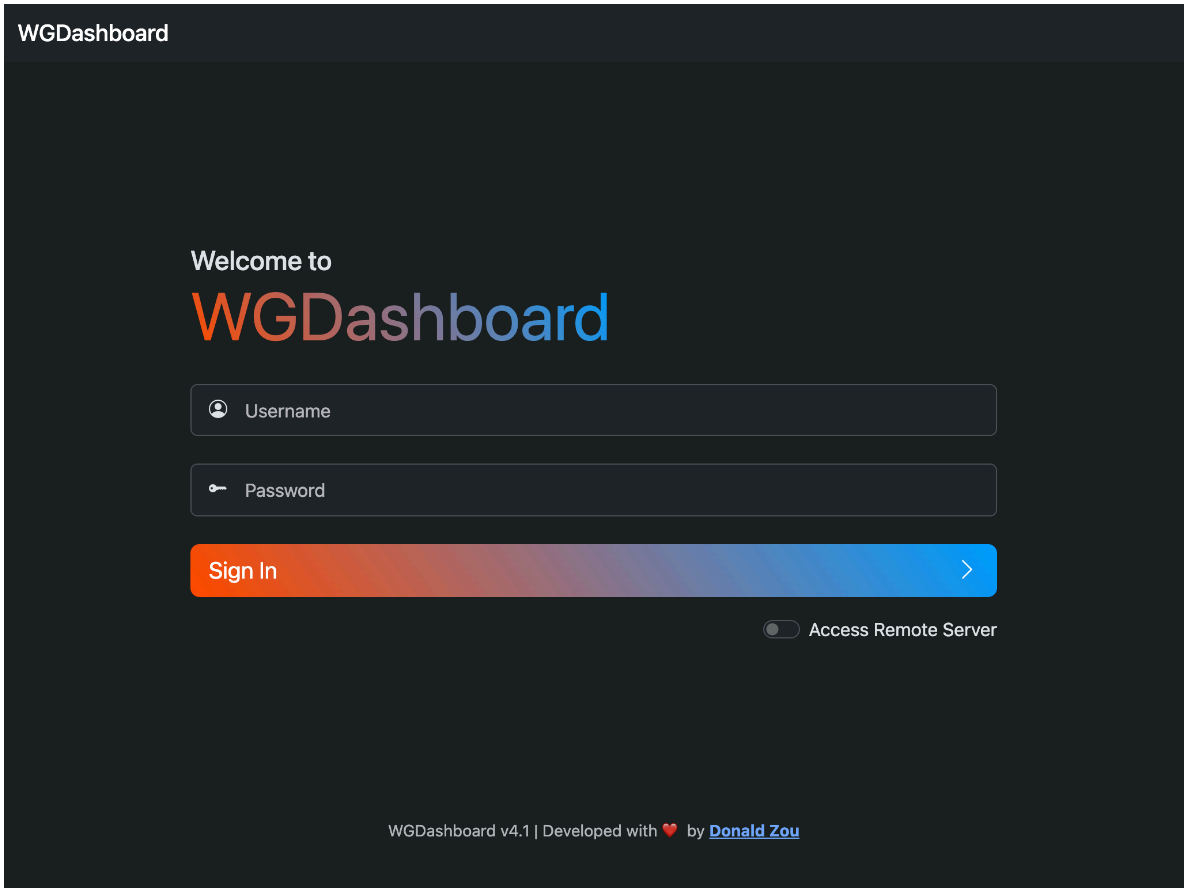
I will be providing some step-by-step guides on how to use WGDashboard, specifically on the app itself. For information on how to run WGDashboard, please visit 🪜 Usage.

You can use the table of contents below, or the navigation bar on the left to find the content you're looking for :)

Sign In

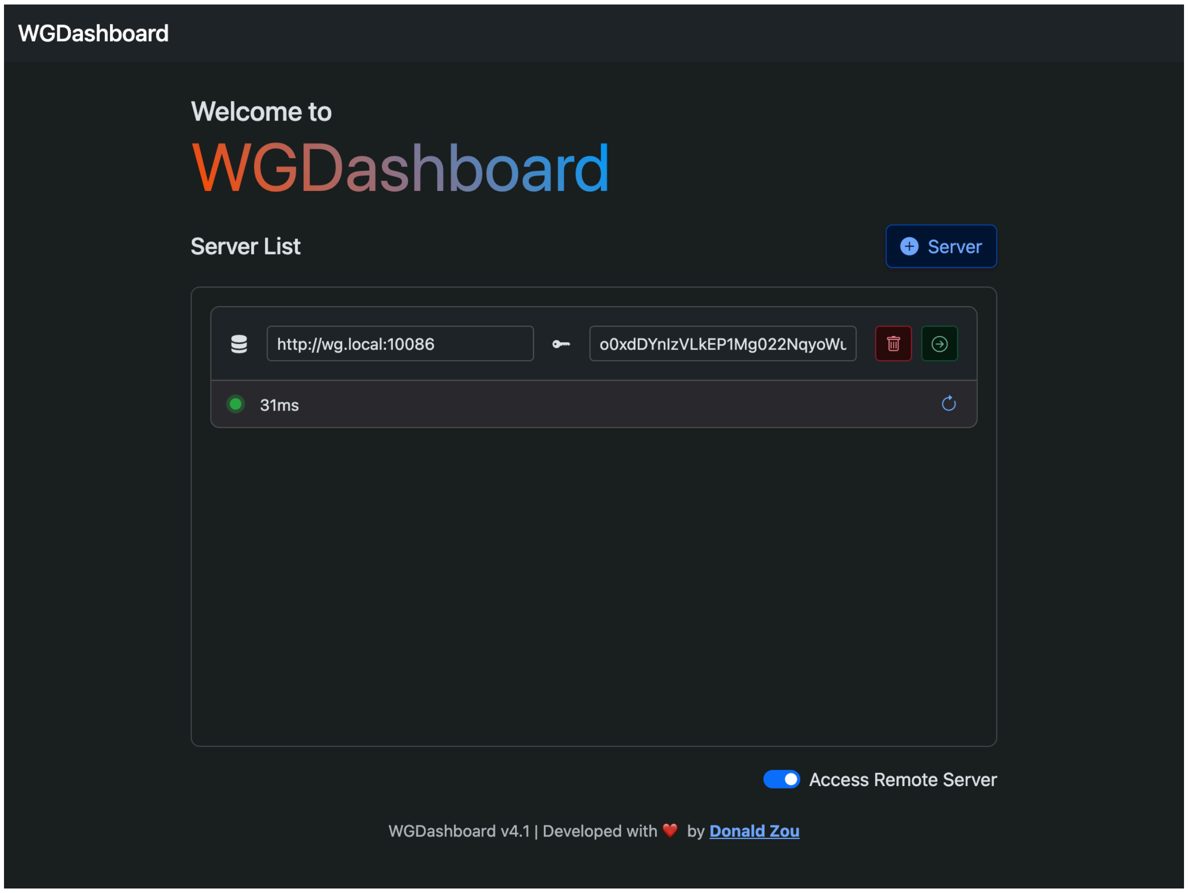
To sign in to WGDashboard, enter the username in the **Username** field, password in the **Password** field, and click the **Sign In** button.

⚠ By default, both **Username** and **Password** is `admin`



Sign in

Access Remote Server



access-remote-server.png

Enable

To access remote server, simply toggle the **switch** under **Sign In** button in the Sign In page

Add Remote Server

To add remote see

✂️ Configure

Configuration File

Since version 2.0, WGDashboard will be using a configuration file called `wg-dashboard.ini`, (It will generate automatically after first time running the dashboard). More options will include in future versions, and for now it included the following configurations:

	Description	Default
[Account]	*Configuration on account*	
username	Dashboard login username	admin
password	Password, will be hash with SHA256	admin hashed in SHA256
[Server]	*Configuration on dashboard*	
wg_conf_path	The path of all the Wireguard configurations	/etc/wireguard
app_prefix	Prefix before each path	(blank)
app_ip	IP address the dashboard will run with	0.0.0.0
app_port	Port the the dashboard will run with	10086
auth_req	Does the dashboard need authentication to access, if auth_req = false, user will not be access the **Setting** tab due to security consideration. **User can only edit the file directly in system.**	true
version	Dashboard Version	v4.0
dashboard_refresh	How frequent the dashboard will refresh on the configuration page	60000ms

`_interval`		
`dashboard_sort`	How configuration is sorting	`status`
`dashboard_theme`	Dashboard Theme	`dark`
`dashboard_api_key`	WGDashboard API Key Function	`false`
`dashboard_language`	WGDashboard Language	`en`
`[Peers]`	*Default Settings on a new peer*	
`peer_global_dns`	DNS Server	`1.1.1.1`
`peer_endpoint_allowed_ip`	Endpoint Allowed IP	`0.0.0.0/0`
`peer_display_mode`	How peer will display	`grid`
`remote_endpoint`	Remote Endpoint (i.e where your peers will connect to)	*depends on your server's default network interface*

`peer_mtu`	Maximum Transmit Unit	`1420`
`peer_keep_alive`	Keep Alive	`21`

Generating QR code and peer configuration file (.conf)

Starting version 2.2, dashboard can now generate QR code and configuration file for each peer. Here is a template of what each QR code encoded with and the same content will be inside the file:

```
[Interface]
PrivateKey = QWERTYUIOPO234567890YUSDAKFH10E1B12JE129U21=
Address = 0.0.0.0/32
DNS = 1.1.1.1

[Peer]
PublicKey = QWERTYUIOPO234567890YUSDAKFH10E1B12JE129U21=
AllowedIPs = 0.0.0.0/0
Endpoint = 0.0.0.0:51820
PresharedKey = QWERTYUIOPO234567890YUSDAKFH10E1B12JE129U21=
```

	Description	Default Value	Available in Peer setting
[Interface]			
PrivateKey	The private key of this peer	Private key generated by WireGuard (wg genkey) or provided by user	Yes
Address	The allowed_ips of your peer	N/A	Yes
DNS	The DNS server your peer will use	1.1.1.1 - Cloud flare DNS, you can change it when you adding the peer or in the peer setting.	Yes
[Peer]			
PublicKey	The public key of your server	N/A	No
AllowedIPs	IP ranges for which a peer will route traffic	0.0.0.0/0 - Indicated a default route to send all internet and VPN traffic through that peer.	Yes
Endpoint	Your wireguard server ip and port, the dashboard will search for your server's default interface's ip.	<your server default interface ip>:<listen port>	Yes

PresharedKey	The Pre-Shared Key of this peer *(if available)*	N/A	Yes

❓ Update

> ⚠️ **Notice to users who are using `v3 - v3.0.6` and want to update to `v4.0`**

Although theoretically updating through `wgd.sh` should work, but I still suggest you to update the dashboard manually.

> ⚠️ **Notice to users who are using `v2.3.1` or below**

For user who is using `v2.3.1` or below, please notice that all data that stored in the current database will **not** transfer to the new database. This is hard decision to move from TinyDB to SQLite. But SQLite does provide a thread-safe access and TinyDB doesn't. I couldn't find a safe way to transfer the data, so you need to do them manually... Sorry about that 😔 。 But I guess this would be a great start for future development 😎 .

How to update

1. Change your directory to `wgdashboard`

```
cd wgdashboard/src
```

2. Update the dashboard

```
git pull https://github.com/donaldzou/WGDashboard.git --force
```

3. Install

```
sudo ./wgd.sh install
```

Starting with `v3.0`, you can simply do `sudo ./wgd.sh update`!! (I hope)

🐬 Docker Solutions

Current, we have 2 beloved contributors provided solutions for hosting WGDashboard with Docker

Solution 1 from @DaanSelen

> Please visit `Docker-explain.md` (https://github.com/donaldzou/WGDashboard/blob/main/docker/README.md) for most up-to-date information

Solution 2 from @shuricksumy

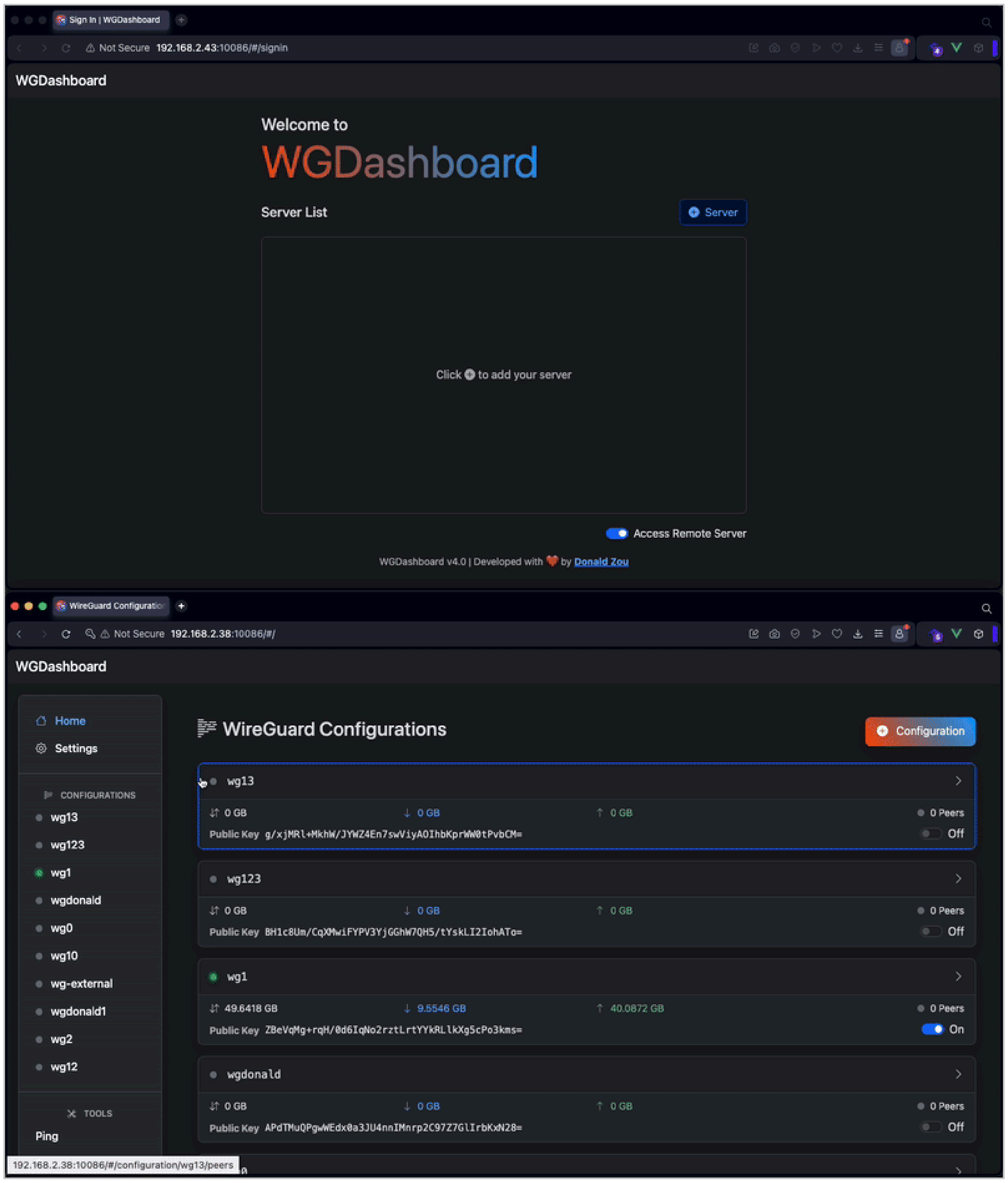
> Please visit `shuricksumy/docker-wgdashboard` (https://github.com/shuricksumy/docker-wgdashboard) for most up-to-date information

> ℹ️ For questions or issues related to Docker, please visit issue #272 (https://github.com/donaldzou/WGDashboard/issues/272)

🥘 Experimental Features

Cross-Server Access

Starting with `v4.0`, you can access WGDashboards on other server through one WGDashboard with API Keys



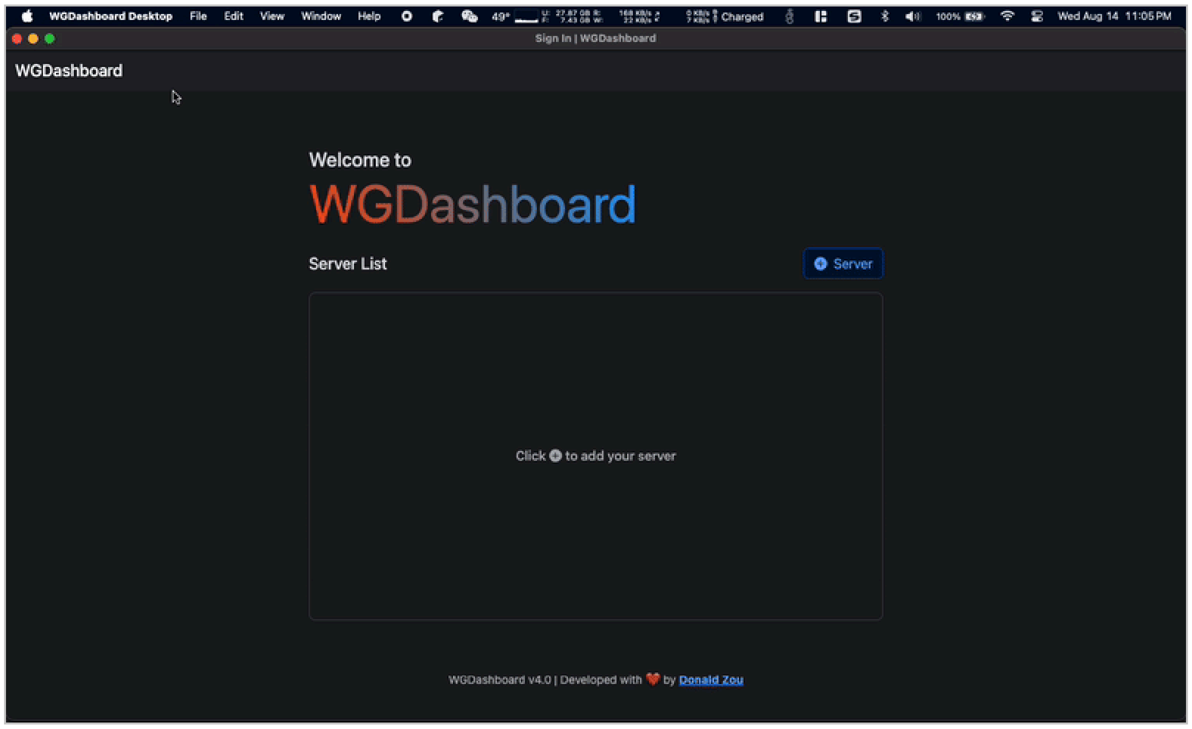
```
wg13
0 GB          0 GB          0 GB          0 Peers
Public Key g/xjMRl+MkhW/JYWZ4En7swViyAOIhbKprWW8tPvbCM=          Off

wg123
0 GB          0 GB          0 GB          0 Peers
Public Key BH1c8Um/CqXMwiFYPV3YjGGhW7QH5/tYskLI2IohATo=          Off

wg1
49.6418 GB    9.5546 GB     40.0872 GB    0 Peers
Public Key ZBeVqMg+rqH/0d6IqNo2rztLrtYYkRLlkXg5cPo3kms=          On

wgdonald
0 GB          0 GB          0 GB          0 Peers
Public Key APdTMuQPgwWEdx0a3JU4nnIMnrp2C97Z7GlIrbKxN28=          Off
```

Cross Server Example

Desktop App

Since the major changes for `v4.0` is to move the whole front-end code to Vue.js. And with this change, we can take the advantage of combining ElectronJS and Vue.js to create a Desktop version of WGDashboard. Currently, we provide an Universal macOS app and a Windows app.

To download the app, please visit the latest release (https://github.com/donaldzou/WGDashboard/releases).



ElectronJS App Demo

📖 API Documentation

I will try my best to keep this up-to-date 😄

Get Started

To use the REST API of your WGDashboard, you need to obtain an API Key.

Create API Key

1. To request an API Key, simply login to your WGDashboard, go to **Settings**, scroll to the very bottom. Click the **switch** on the right to enable API Key.
2. Click the blur **Create** button, set an **expiry date** you want or **never expire**, then click **Done.**

Use API Key

- Simply add `wg-dashboard-apikey` with the value of your API key into the HTTP Header.

Default

```
fetch('http://server:10086/api/handshake', {
    headers: {
        'content-type': 'application/json',
         'wg-dashboard-apikey': 'insert your api key here'
    },
    method: "GET"
})
```

With APP_PREFIX set

For more information, please visit ✂️ Configure

```
fetch('http://server:10086/[app_prefix]/api/handshake', {
    headers: {
       'content-type': 'application/json',
        'wg-dashboard-apikey': 'insert your api key here'
    },
    method: "GET"
})
```

Now you're ready

Please visit this page for details of Endpoints

Endpoints

Handshake to Server

This endpoint is designed for a simple handshake when using API key to connect. If `status` is `true` that means

Request

`GET /api/handshake`

Response

`200 - OK`

```
{
    "data": null,
    "message": null,
    "status": true
}
```

`401 - UNAUTHORIZED`

```
{
    "data": null,
    "message": "Unauthorized access.",
    "status": false
}
```

⚠ Notice: this `401` response will return at all endpoint if your API Key or session is invalid.

Validate Authentication

This endpoint if needed for non-cross-server access. This will check if the cookie on the client side is still valid on the server side.

Request

`GET /api/validateAuthentication`

Response

`200 - OK`

Session is still valid

```json
{
    "data": null,
    "message": null,
    "status": true
}
```

Session is invalid

```json
{
    "data": null,
    "message": "Invalid authentication.",
    "status": false
}
```

Authenticate

This endpoint is dedicated for non-cross-server access. It is used to authenticate user's username, password and TOTP

Request

`POST /api/authenticate`

Body Parameters

```
{
    "username": "admin",
    "password": "admin",
    "totp": "123456"
}
```

Parameter	Type
`username`	string
`password`	string
`totp`	string

Response

`200 - OK`

If username, password and TOTP matched

```
{
    "data": null,
    "message": null,
    "status": true
}
```

If username, password or TOTP is not match

```
{
    "data": null,
    "message": "Sorry, your username, password or OTP is
incorrect.",
    "status": false
}
```

Sign Out

To remove the current session on server side

Request

`GET /api/signout`

Response

`200 - OK`

```
{
    "data": null,
    "message": null,
    "status": true
}
```

Get WGDashboard Configuration

Get the WGDashboard Configuration, such as `dashboard_theme`...

Request

`GET /api/getDashboardConfiguration`

Response

`200 - OK`

```
{
    "data": {
        "Account": {
            "enable_totp": false,
            "password": "some hashed value :(",
            "totp_verified": false,
            "username": "admin"
        },
        "Database": {
            "type": "sqlite"
        },
        "Other": {
            "welcome_session": false
        },
        "Peers": {
            "peer_display_mode": "grid",
            "peer_endpoint_allowed_ip": "0.0.0.0/0",
            "peer_global_dns": "1.1.1.1",
            "peer_keep_alive": "21",
            "peer_mtu": "1420",
            "remote_endpoint": "192.168.2.38"
        },
        "Server": {
            "app_ip": "0.0.0.0",
            "app_port": "10086",
            "app_prefix": "",
            "auth_req": true,
            "dashboard_api_key": true,
            "dashboard_refresh_interval": "5000",
            "dashboard_sort": "status",
            "dashboard_theme": "dark",
            "version": "v4.0",
            "wg_conf_path": "/etc/wireguard"
        }
    },
    "message": null,
    "status": true
}
```

Update WGDashboard Configuration Item

Request

`POST /api/updateDashboardConfigurationItem`

Body Parameters

```
{
    "section": "Server",
    "key": "dashboard_theme",
    "value": "dark"
}
```

Parameter	Type	
`section`	string	Each section in the `wg-dashboard.ini`
`key`	string	Each key/value pair under each in the `wg-dashboard.ini`
`value`	string	Value for this key/value pair

Response

`200 - OK`

If update is success

```
{
    "data": true,
    "message": null,
    "status": true
}
```

If update failed

```json
{
    "data": true,
    "message": "Message related to the error will appear here",
    "status": false
}
```

Get WireGuard Configurations

To get all WireGuard configurations in `/etc/wireguard`

Request

`GET /api/getWireguardConfigurations`

Response

`200 - OK`

```
{
    "data": [
        {
            "Address": "10.200.200.1/24",
            "ConnectedPeers": 0,
            "DataUsage": {
                "Receive": 0.1582,
                "Sent": 2.1012999999999997,
                "Total": 2.2595
            },
            "ListenPort": "51820",
            "Name": "wg0",
            "PostDown": "iptables -D FORWARD -i wg0 -j ACCEPT;
iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t nat -D
POSTROUTING -o enp0s1 -j MASQUERADE;",
            "PostUp": "iptables -A FORWARD -i wg0 -j ACCEPT;
iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A
POSTROUTING -o enp0s1 -j MASQUERADE;",
            "PreDown": "",
            "PreUp": "",
            "PrivateKey":
"8DsSMli3okgUx5frKbFQ0fMW5ZMyqyxOdOW7+g21L18=",
            "PublicKey":
"GQlGi8QJ93hWY7L2xlJyh+7S6+ekER9xP11T92T0O0Q=",
            "SaveConfig": true,
            "Status": false
        }
    ],
    "message": null,
    "status": true
}
```

Add WireGuard Configuration

Add a new WireGuard Configuration

Request

`POST /api/addWireguardConfiguration`

Body Parameters

```
{
    "ConfigurationName": "wg0",
    "Address": "10.0.0.1/24",
    "ListenPort":  51820,
    "PrivateKey": "eJuuamCgakVs2xUZGHh/g7C6Oy89JGh7eE2jjEGbbFc=",
    "PublicKey":  "3Ruirgw9qNRwNpBepkiVjjSe82tY+lDZr6WaFC4wO2g=",
    "PresharedKey":
"GMMLKWdJlgsKVoR26BJPsNbDXyfILL+x1Nd6Ecmn4lg=",
    "PreUp":  "",
    "PreDown": "iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D
FORWARD -o wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o enp0s1
-j MASQUERADE;",
    "PostUp":  "iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A
FORWARD -o wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o enp0s1
-j MASQUERADE;",
    "PostDown": ""
}
```

Response

`200 - OK`

If everything is good

```
{
    "data": null,
    "message": null,
    "status": true
}
```

If the new configuration's `ConfigurationName` is already existed

```
{
    "data": null,
    "message": "Already have a configuration with the name
\"wg0\"",
    "status": false
}
```

If the new configuration's `ListenPort` is used by another configuration

```
{
    "data": null,
    "message": "Already have a configuration with the port
\"51820\"",
    "status": false
}
```

If the new configuration's `Address` is used by another configuration

```
{
    "data": null,
    "message": "Already have a configuration with the address
\"10.0.0.1/24\"",
    "status": false
}
```

Toggle WireGuard Configuration

To turn on/off of a WireGuard Configuration

Request

`GET /api/toggleWireguardConfiguration/?configurationName=`

Query String Parameter

Parameter	Type
`configurationName`	string

Response

`200 - OK`

If toggle is successful, server will return the current status in `status`: `true` or `false` indicating if the configuration is up or not.

```
{
    "data": true,
    "message": null,
    "status": true
}
```

If the `configurationName` provided does not exist

```
{
    "data": null,
    "message": "Please provide a valid configuration name",
    "status": false
}
```

Update WireGuard Configuration

Update WireGuard configuration information

Request

`POST /api/updateWireguardConfiguration`

Body Parameter

```
{
    "Name": "wg88",
    "Address": "10.24.4.0/23",
    "ListenPort": 56768,
    "PostDown": "iptables -D FORWARD -i wg0 -j ACCEPT; iptables -D FORWARD -o wg0 -j ACCEPT; iptables -t nat -D POSTROUTING -o enp0s1 -j MASQUERADE;",
    "PostUp": "iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT; iptables -t nat -A POSTROUTING -o enp0s1 -j MASQUERADE;",
    "PreDown": "",
    "PreUp": ""
}
```

Response

`200 - OK`

```
{
    "data": true,
    "message": null,
    "status": true
}
```

If the `Name` provided does not exist or `Name` is not in body parameter

```
{
    "data": null,
    "message": "Please provide a valid configuration name",
    "status": false
}
```

Delete WireGuard Configuration

Request

`POST /api/deleteWireguardConfiguration`

Get WireGuard Configuration Backup

Request

```
GET /api/getWireguardConfigurationBackup?configurationName=
```

Get All WireGuard Configuration Backup

Request

`GET /api/getAllWireguardConfigurationBackup`

Delete WireGuard Configuration Backup

Request

`POST /api/deleteWireguardConfigurationBackup`

Get WGDashboard API Keys

Get a list of active API key in WGDashboard

Request

`GET /api/getDashboardAPIKeys`

Response

`200 - OK`

If API Key function is enabled and there are active API keys

⚠ If `ExpiredAt` is `null`, that means this API key will never expire

```
{
    "data": [
        {
            "CreatedAt": "2024-08-15 00:42:31",
            "ExpiredAt": null,
            "Key": "AXt1x3TZMukmA-eSnAyESy08I14n20boppSsknHOB-Y"
        },
        {
            "CreatedAt": "2024-08-14 22:50:44",
            "ExpiredAt": "2024-08-21 22:50:43",
            "Key": "ry0Suo0BrypSMzbq0C_TjkEcgrFHHj6UBZGmC2-KI2o"
        }
    ],
    "message": null,
    "status": true
}
```

If API key function is disabled

```
{
    "data": null,
    "message": "Dashboard API Keys function is disabled",
    "status": false
}
```

Add WGDashboard API Key

Add a new API Key in WGDashboard

Request

`POST /api/newDashboardAPIKey`

Body Parameters

```
{
    "neverExpire": false,
    "ExpiredAt": "2024-12-31 16:00:00"
}
```

Parameter	Type	
`neverExpire`	bool	If this is `false`, please specify a date in `ExpiredAt`
`ExpiredAt`	string	If `neverExpire` is `true`, this can be omitted. Format is `YYYY-MM-DD hh:mm:ss`.

Response

`200 - OK`

If success, it will return the latest list of API Keys

```
{
  "data": [
    {
      "CreatedAt": "2024-08-15 00:42:31",
      "ExpiredAt": null,
      "Key": "AXt1x3TZMukmA-eSnAyESy08I14n20boppSsknHOB-Y"
    },
    {
      "CreatedAt": "2024-08-14 22:50:44",
      "ExpiredAt": "2024-12-31 16:50:43",
      "Key": "ry0Suo0BrypSMzbq0C_TjkEcgrFHHj6UBZGmC2-KI2o"
    }
  ],
  "message": null,
  "status": true
}
```

If API key function is disabled

```
{
    "data": null,
    "message": "Dashboard API Keys function is disabled",
    "status": false
}
```

Delete WGDashboard API Key

Delete an existing API Key in WGDashboard

Request

`POST /api/deleteDashboardAPIKey`

Body Parameters

```
{
    "key": "ry0Suo0BrypSMzbq0C_TjkEcgrFHHj6UBZGmC2-KI2o"
}
```

Parameter	Type	
`key`	string	The API Key you want to delete

Response

`200 - OK`

If success, it will return the latest list of API Keys after deleting the one you provided

```
{
  "data": [
    {
      "CreatedAt": "2024-08-15 00:42:31",
      "ExpiredAt": null,
      "Key": "AXt1x3TZMukmA-eSnAyESy08I14n20boppSsknHOB-Y"
    }
  ],
  "message": null,
  "status": true
}
```

If API key function is disabled

```json
{
    "data": null,
    "message": "Dashboard API Keys function is disabled",
    "status": false
}
```

Reset Peer Data Usage

Reset peer's total, sent or receive data

Request

`POST /api/resetPeerData/<configName>`

URL Parameter

Parameter	Type	Description
`configName`	string	The config name of the peer is in

Body Parameter

```
{
  "id": "mCP70rKd4iumKptwTgzvAR3g8/D74ZDkwR0EuI10uk4=",
  "type": "total"
}
```

Parameter	Type	Description
`id`	string	Public Key of the peer you want to update
`type`	string	Type of data you want to reset. It can be `total`, `receive` or `sent`

Response

Request Success

```
{
  "data": null,
  "message": null,
  "status": true
}
```

⚠ Request Failed

If `id` is not provided, `configName` does not exist, or peer does not exist

```
{
  "data": null,
  "message": "Configuration/Peer does not exist",
  "status": false
}
```

Delete Peers

To delete peers individually or in bulk

Request

`POST /api/deletePeers/<configName>`

URL Parameter

Parameter	Type	
`configName`	string	The config name of the peer(s) is in

Body Parameter

```
{
  "peers": [
    "mCP70rKd4iumKptwTgzvAR3g8/D74ZDkwR0EuI10uk4=",
    "lKptwTgzvAR3gmCP70rKd4iu8/D74ZDkwR0EuI10uk4=",
    "pCP70rKd4iumK0uk4ptwTgzvAR3g8/D74ZDkwR0EuI1="
  ]
}
```

Parameter	Type	
`peers`	list[string]	List of strings contain public key(s) you want to delete

Response

Request Success

```
{
  "data": null,
  "message": "Deleted 3 peer(s)",
  "status": true
}
```

⚠ Request Failed

If configuration name provided in `configName` does not exist

```
{
  "data": null,
  "message": "Configuration does not exist",
  "status": false
}
```

If length of `peers` is 0

```
{
  "data": null,
  "message": "Please specify one or more peers",
  "status": false
}
```

If failed to save to WireGuard

```
{
  "data": null,
  "message": "Failed to save configuration through WireGuard",
  "status": false
}
```

If failed delete some of the peers

```
{
  "data": null,
  "message": "Deleted 3 peer(s) successfully. Failed to delete 1 
peer(s)",
  "status": false
}
```

Restrict Peers

Description

Request

`POST /api/restrictPeers/<configName>'`

URL Parameter

Parameter	Type	
`configName`	string	The config name of the peer is in

Body Parameter

```
{
    "peers": [
      "mCP70rKd4iumKptwTgzvAR3g8/D74ZDkwR0EuI10uk4=",
      "lKptwTgzvAR3gmCP70rKd4iu8/D74ZDkwR0EuI10uk4=",
      "pCP70rKd4iumK0uk4ptwTgzvAR3g8/D74ZDkwR0EuI1="
    ]
}
```

Parameter	Type	
`peers`	list[string]	List of strings contain public key(s) you want to delete

Response

Request Success

Description

```
{
  "data": null,
  "message": "Restricted 3 peer(s)",
  "status": true
}
```

⚠ Request Failed

If configuration name provided in `configName` does not exist

```
{
  "data": null,
  "message": "Configuration does not exist",
  "status": false
}
```

If length of `peers` is 0

```
{
  "data": null,
  "message": "Please specify one or more peers",
  "status": false
}
```

If failed to save to WireGuard

```
{
  "data": null,
  "message": "Failed to save configuration through WireGuard",
  "status": false
}
```

If failed restrict some of the peers

```
{
  "data": null,
  "message": "Restricted 3 peer(s) successfully. Failed to
restrict 1 peer(s)",
  "status": false
}
```

Create Share Peer URL

Create a URL to share peer's configuration file and QR Code

Request

`POST /api/sharePeer/create`

URL Parameter

Parameter	Type	
``	string	The config name of the peer is in

Body Parameter

```json
{
    "Configuration": "wg0",
    "Peer": "mCP70rKd4iumKptwTgzvAR3g8/D74ZDkwR0EuI10uk4=",
    "ExpireDate": "2024-10-01 23:50:52"
}
```

Parameter	Type	
`Configuration`	string	The config name of the peer is in
`Peer`	string	The peer you want to share
`ExpireDate`	string	Expire date for the URL, format is `YYYY-MM-DD hh:mm:ss` (24-hours). This field is **optional**

Response

Request Success

If create successfully, it will return an object in the data, it contains information about the share URL

```
{
    "data": [
        {
          "Configuration": "wg88",
          "ExpireDate": "2024-10-01 23:50:52",
          "Peer": "72y7R7deXRLZoIb+BYNFWe5RuCmXnWxTSMHfv4Kjay8=",
          "ShareID": "448a6f79-f6fc-4ce9-802a-1a3142ae7253"
        }
    ],
    "message": null,
    "status": true
}
```

Request Failed

If `Configuration` or `Peer` did not provide

```
{
  "data": null,
  "message": "Please specify configuration and peers",
  "status": false
}
```

If this peer have an existing URL

```
{
  "data": null,
  "message": "This peer is already sharing, please stop sharing
first.",
  "status": false
}
```

Endpoint Name

Description

Request

`POST /api/`

URL Parameter

Parameter	Type	
``	string	The config name of the peer is in

Body Parameter

```json
{
  "id": "mCP70rKd4iumKptwTgzvAR3g8/D74ZDkwR0EuI10uk4=",
  "type": "total"
}
```

Parameter	Type	
``	string	The config name of the peer is in

Response

Request Success

Description

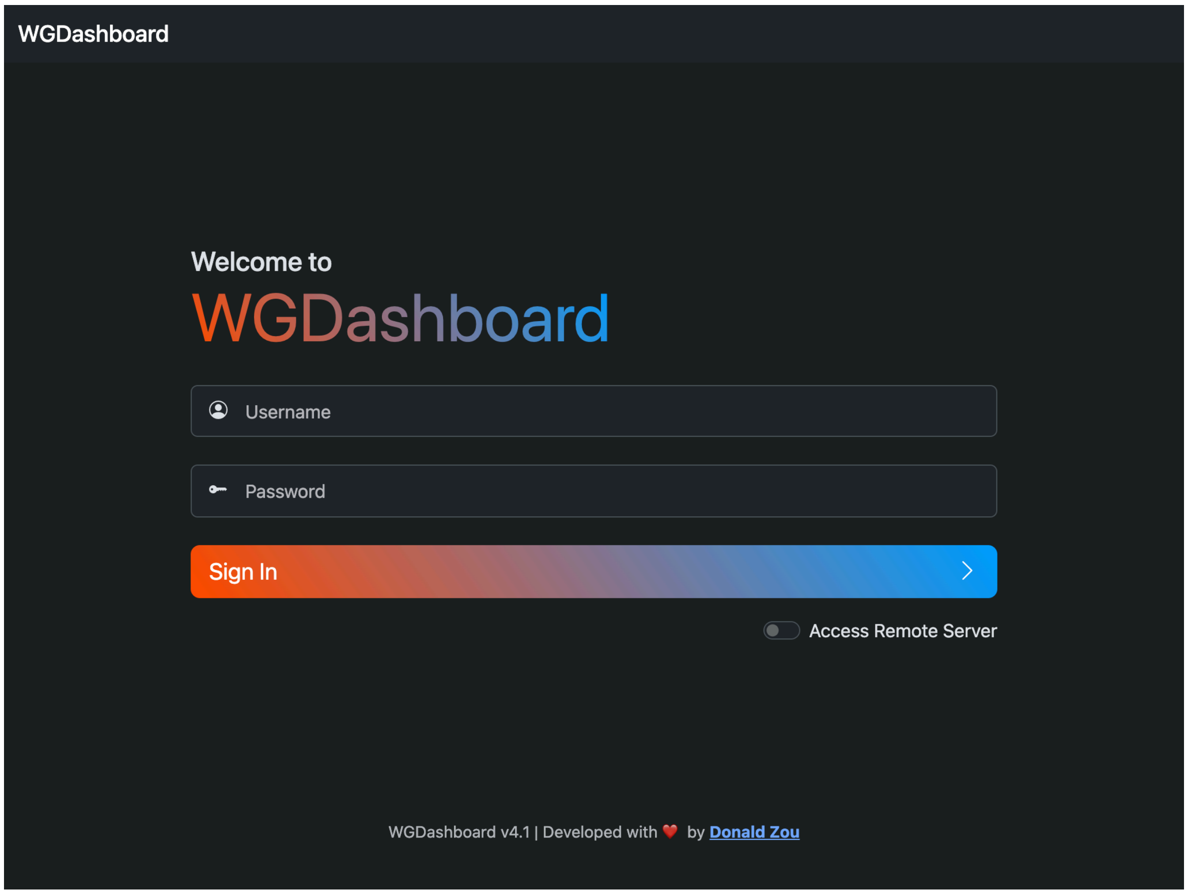
```json
{
  "data": null,
  "message": null,
  "status": true
}
```

⚠ Request Failed

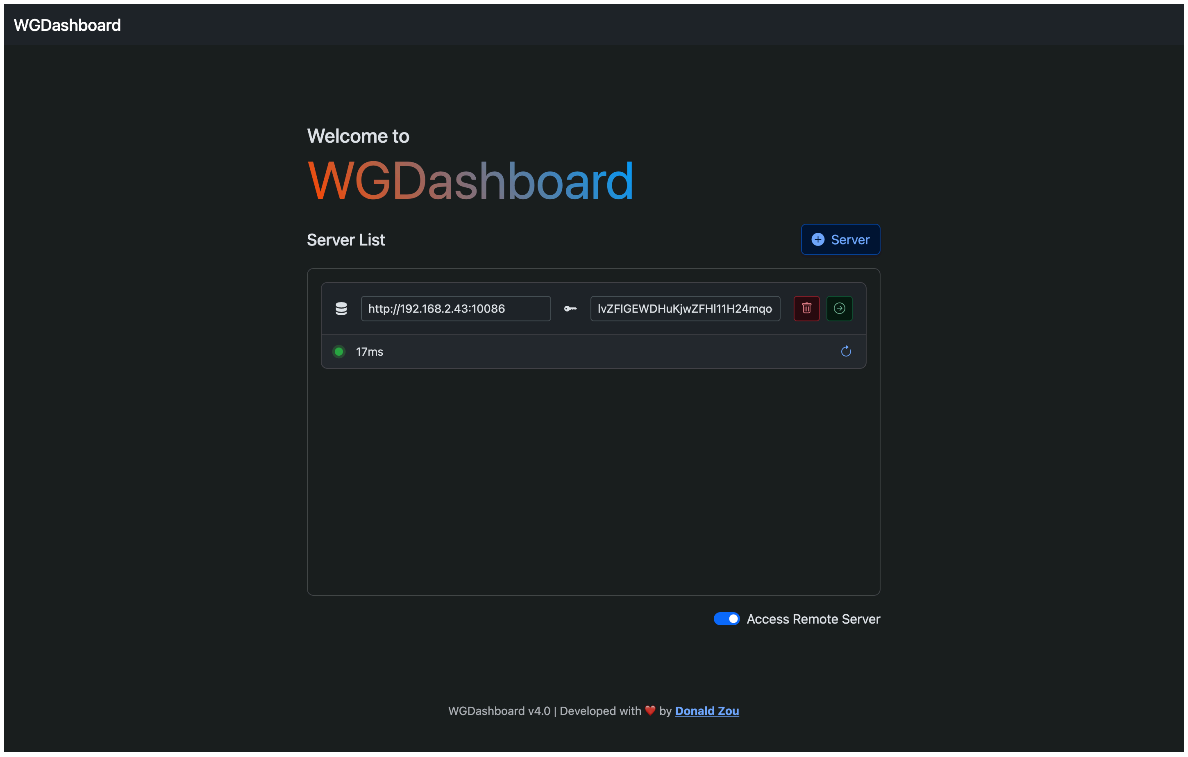
Description

```
{
  "data": null,
  "message": null,
  "status": true
}
```

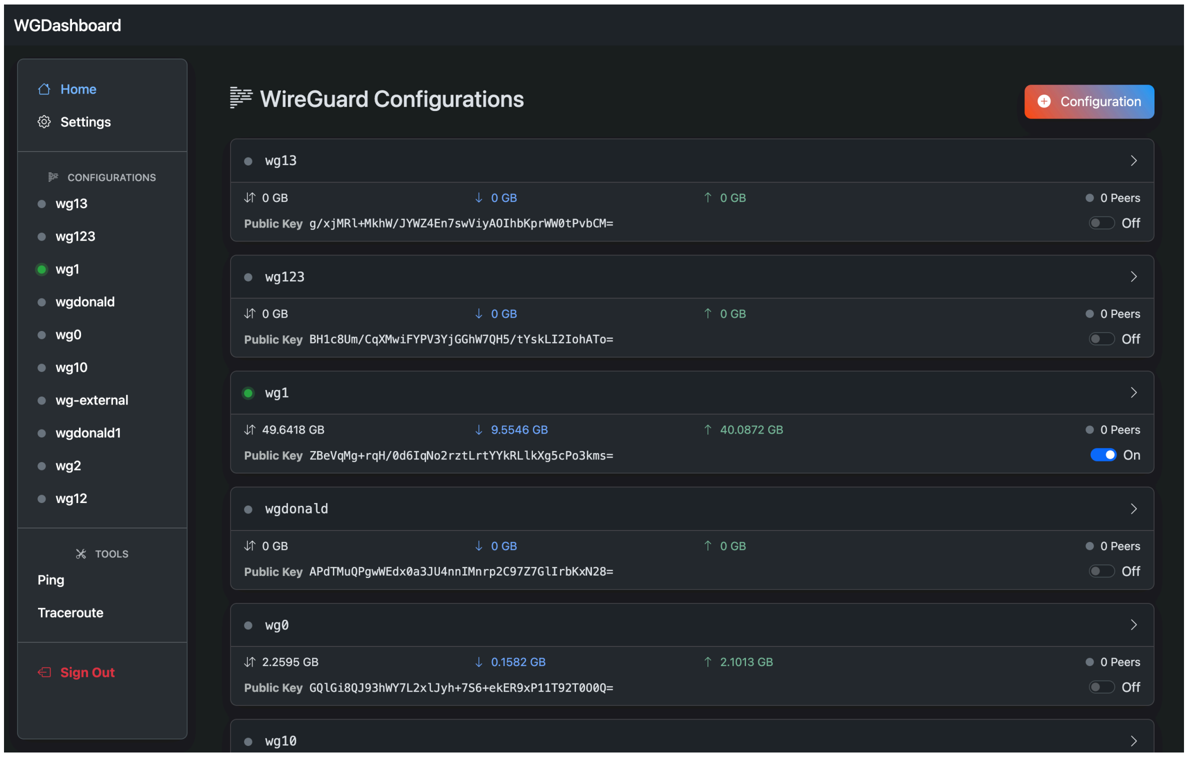
🔍 Screenshot



Sign in

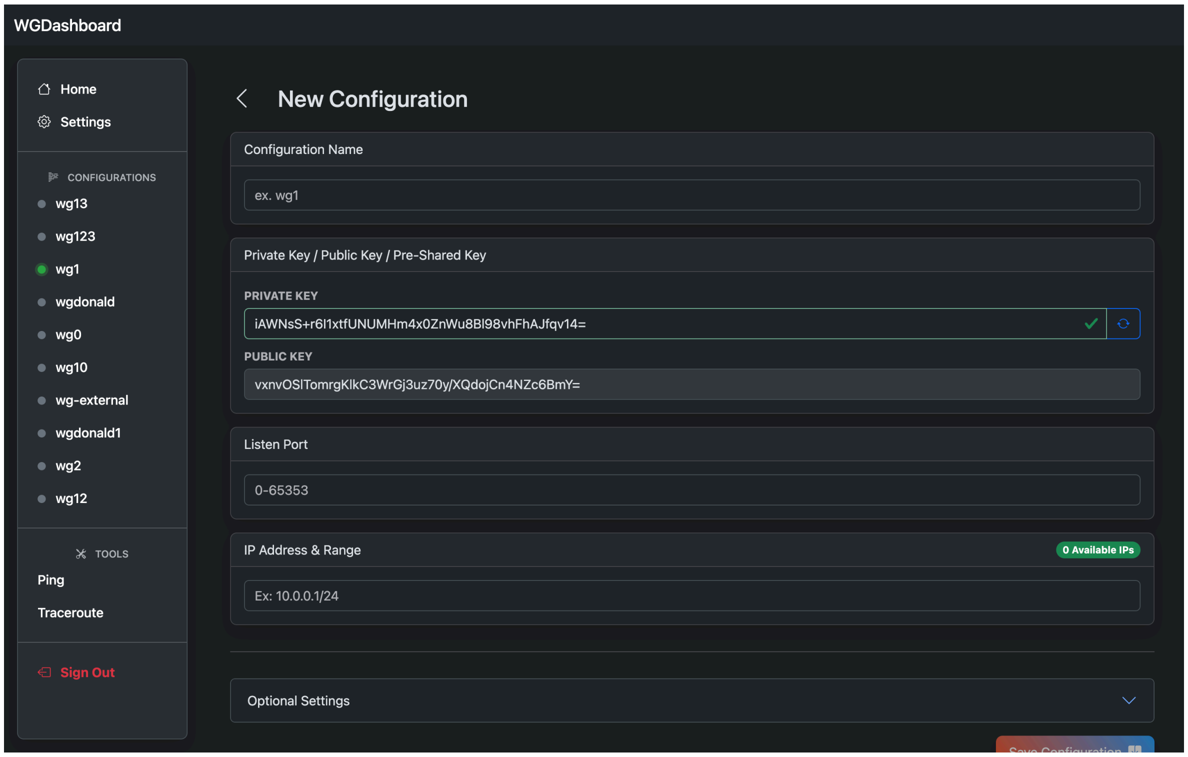

WGDashboard
Welcome to
WGDashboard
Server List
Server
http://192.168.2.43:10086
lvZFIGEWDHuKjwZFHI11H24mqo
17ms
Access Remote Server
WGDashboard v4.0 | Developed with ❤️ by Donald Zou

Cross server



```
● wg13
⇅ 0 GB        ↓ 0 GB        ↑ 0 GB        ● 0 Peers
Public Key g/xjMRl+MkhW/JYWZ4En7swViyAOIhbKprWW0tPvbCM=        Off

● wg123
⇅ 0 GB        ↓ 0 GB        ↑ 0 GB        ● 0 Peers
Public Key BH1c8Um/CqXMwiFYPV3YjGGhW7QH5/tYskLI2IohATo=        Off

● wg1
⇅ 49.6418 GB  ↓ 9.5546 GB   ↑ 40.0872 GB  ● 0 Peers
Public Key ZBeVqMg+rqH/0d6IqNo2rztLrtYYkRLlkXg5cPo3kms=        On

● wgdonald
⇅ 0 GB        ↓ 0 GB        ↑ 0 GB        ● 0 Peers
Public Key APdTMuQPgwWEdx0a3JU4nnIMnrp2C97Z7GlIrbKxN28=        Off

● wg0
⇅ 2.2595 GB   ↓ 0.1582 GB   ↑ 2.1013 GB   ● 0 Peers
Public Key GQlGi8QJ93hWY7L2xlJyh+7S6+ekER9xP11T92T000Q=        Off

● wg10
```

Index



New Configuration

Configuration Name

ex. wg1

Private Key / Public Key / Pre-Shared Key

PRIVATE KEY

iAWNsS+r6I1xtfUNUMHm4x0ZnWu8BI98vhFhAJfqv14=

PUBLIC KEY

vxnvOSlTomrgKlkC3WrGj3uz70y/XQdojCn4NZc6BmY=

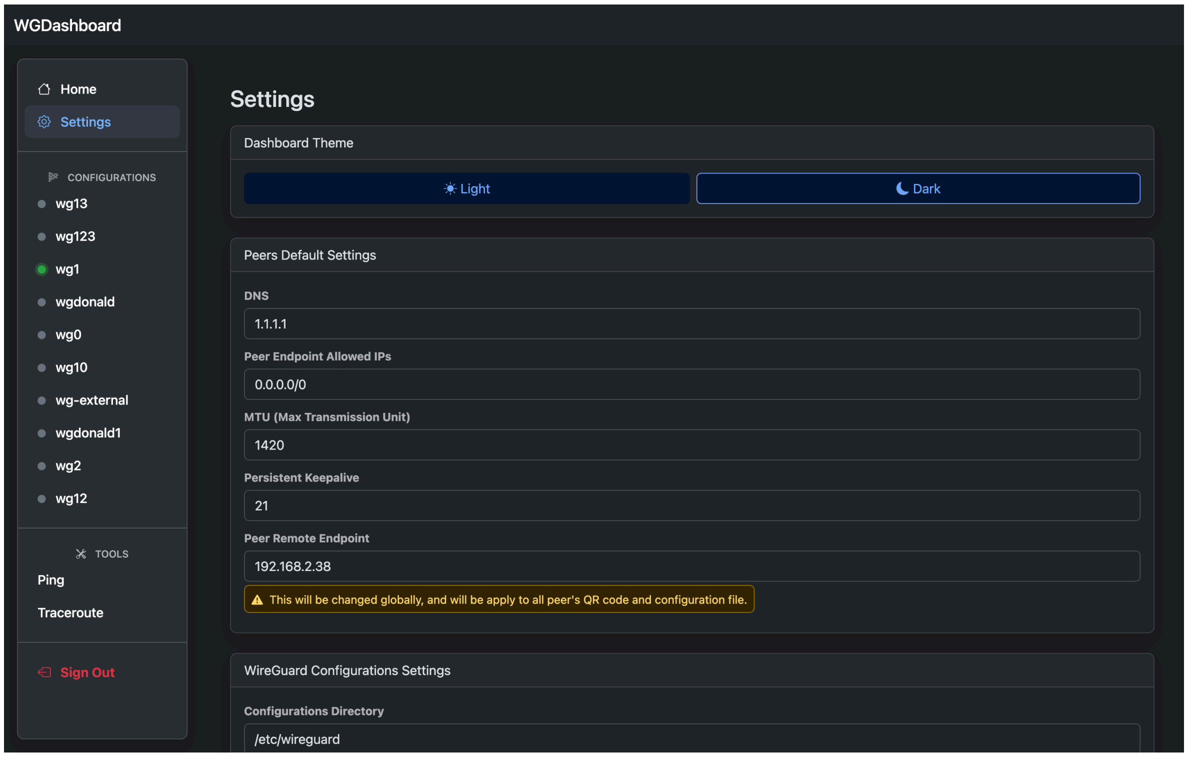
Listen Port

0-65353

IP Address & Range — 0 Available IPs

Ex: 10.0.0.1/24

New configuration



Settings

Dashboard Theme

Peers Default Settings

DNS: 1.1.1.1

Peer Endpoint Allowed IPs: 0.0.0.0/0

MTU (Max Transmission Unit): 1420

Persistent Keepalive: 21

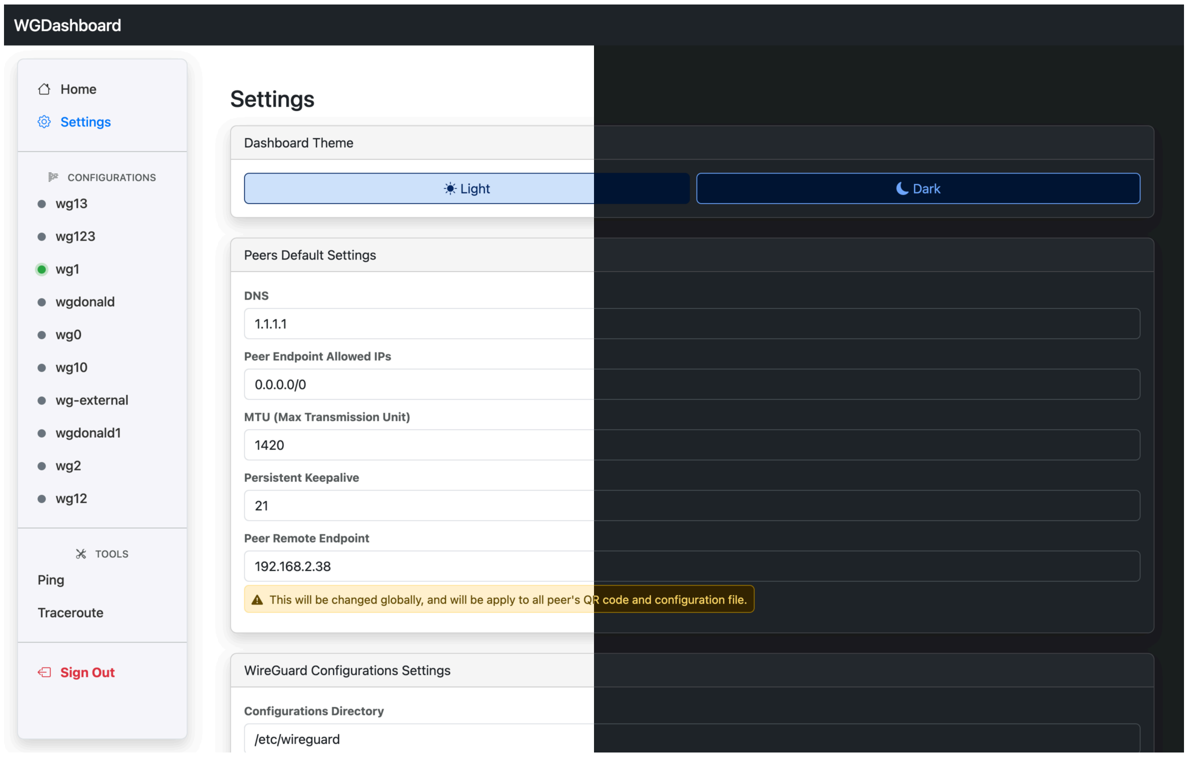
Peer Remote Endpoint: 192.168.2.38

This will be changed globally, and will be apply to all peer's QR code and configuration file.

WireGuard Configurations Settings

Configurations Directory: /etc/wireguard

Settings



Settings

Dashboard Theme

Light [✓ Dark]

Peers Default Settings

DNS
1.1.1.1

Peer Endpoint Allowed IPs
0.0.0.0/0

MTU (Max Transmission Unit)
1420

Persistent Keepalive
21

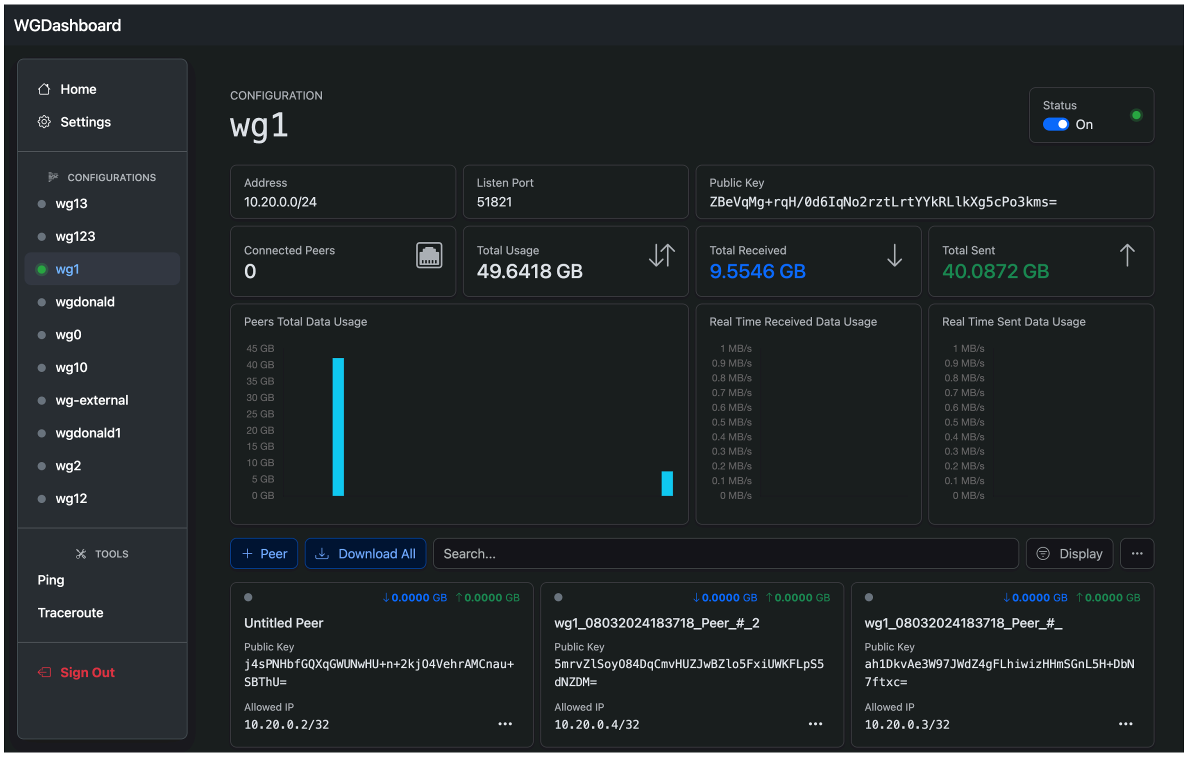
Peer Remote Endpoint
192.168.2.38

This will be changed globally, and will be apply to all peer's QR code and configuration file.

WireGuard Configurations Settings

Configurations Directory
/etc/wireguard

Light dark



CONFIGURATION
wg1

Status
On

Address
10.20.0.0/24

Listen Port
51821

Public Key
ZBeVqMg+rqH/0d6IqNo2rztLrtYYkRLlkXg5cPo3kms=

Connected Peers
0

Total Usage
49.6418 GB

Total Received
9.5546 GB

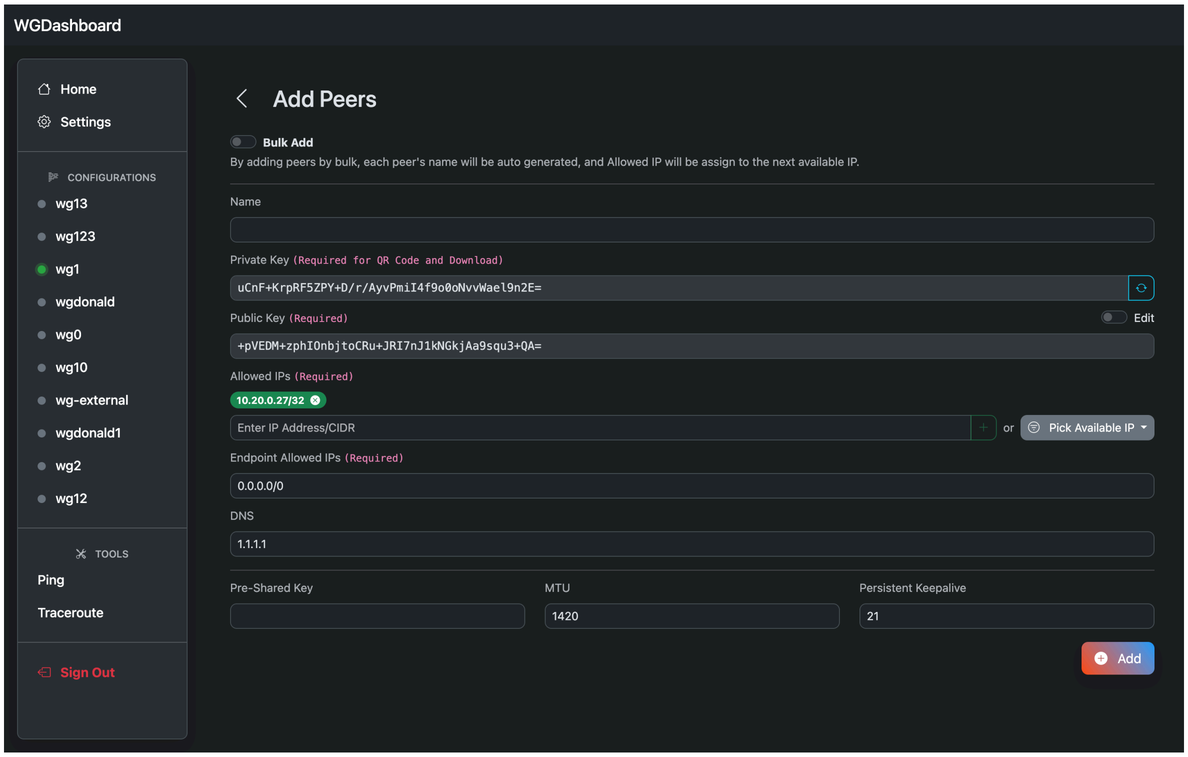
Total Sent
40.0872 GB

↓0.0000 GB ↑0.0000 GB
Untitled Peer
Public Key
j4sPNHbfGQXqGWUNwHU+n+2kjO4VehrAMCnau+SBThU=
Allowed IP
10.20.0.2/32

↓0.0000 GB ↑0.0000 GB
wg1_08032024183718_Peer_#_2
Public Key
5mrvZlSoy084DqCmvHUZJwBZlo5FxiUWKFLpS5dNZDM=
Allowed IP
10.20.0.4/32

↓0.0000 GB ↑0.0000 GB
wg1_08032024183718_Peer_#_
Public Key
ah1DkvAe3W97JWdZ4gFLhiwizHHmSGnL5H+DbN7ftxc=
Allowed IP
10.20.0.3/32

Configuration



Add Peers

Bulk Add

By adding peers by bulk, each peer's name will be auto generated, and Allowed IP will be assign to the next available IP.

Name

Private Key (Required for QR Code and Download)

`uCnF+KrpRF5ZPY+D/r/AyvPmiI4f9o0oNvvWael9n2E=`

Public Key (Required) Edit

`+pVEDM+zphIOnbjtoCRu+JRI7nJ1kNGkjAa9squ3+QA=`

Allowed IPs (Required)

10.20.0.27/32

Enter IP Address/CIDR or Pick Available IP

Endpoint Allowed IPs (Required)

0.0.0.0/0

DNS

1.1.1.1

Pre-Shared Key

MTU

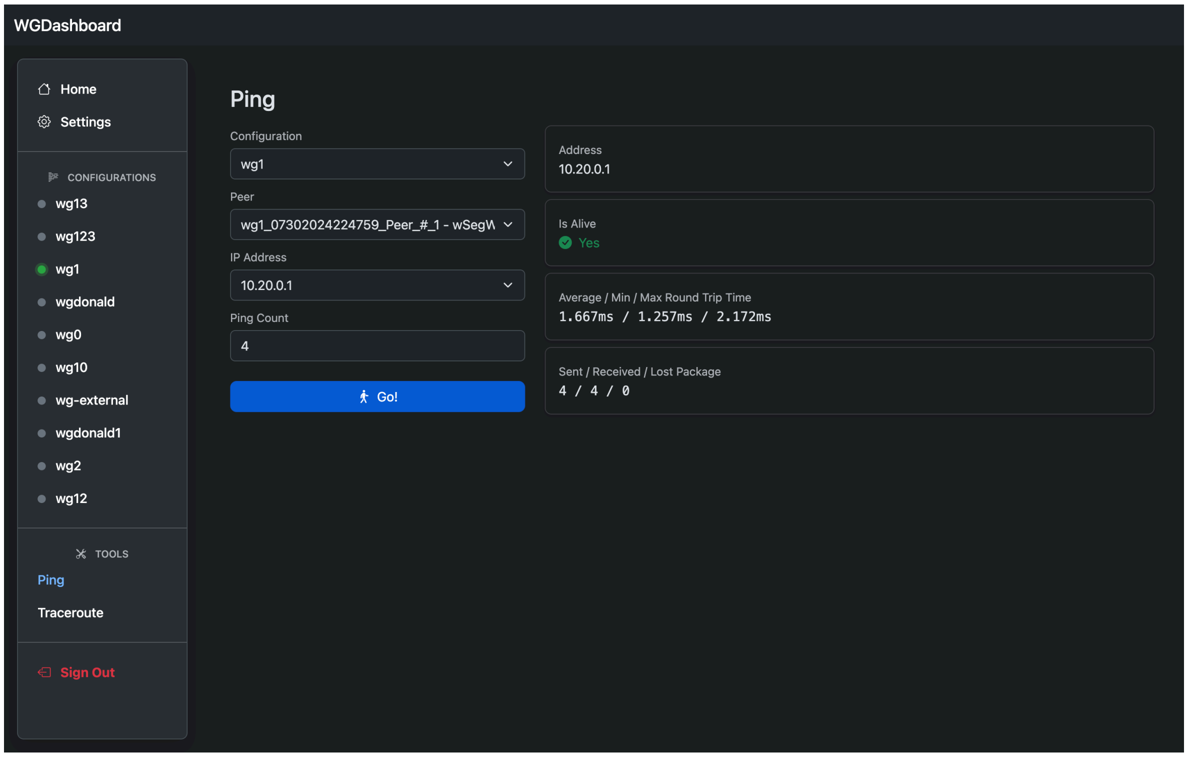
1420

Persistent Keepalive

21

Add

Add peers



Ping

Configuration

wg1

Peer

wg1_07302024224759_Peer_#_1 - wSegW

IP Address

10.20.0.1

Ping Count

4

Go!

Address

10.20.0.1

Is Alive

Yes

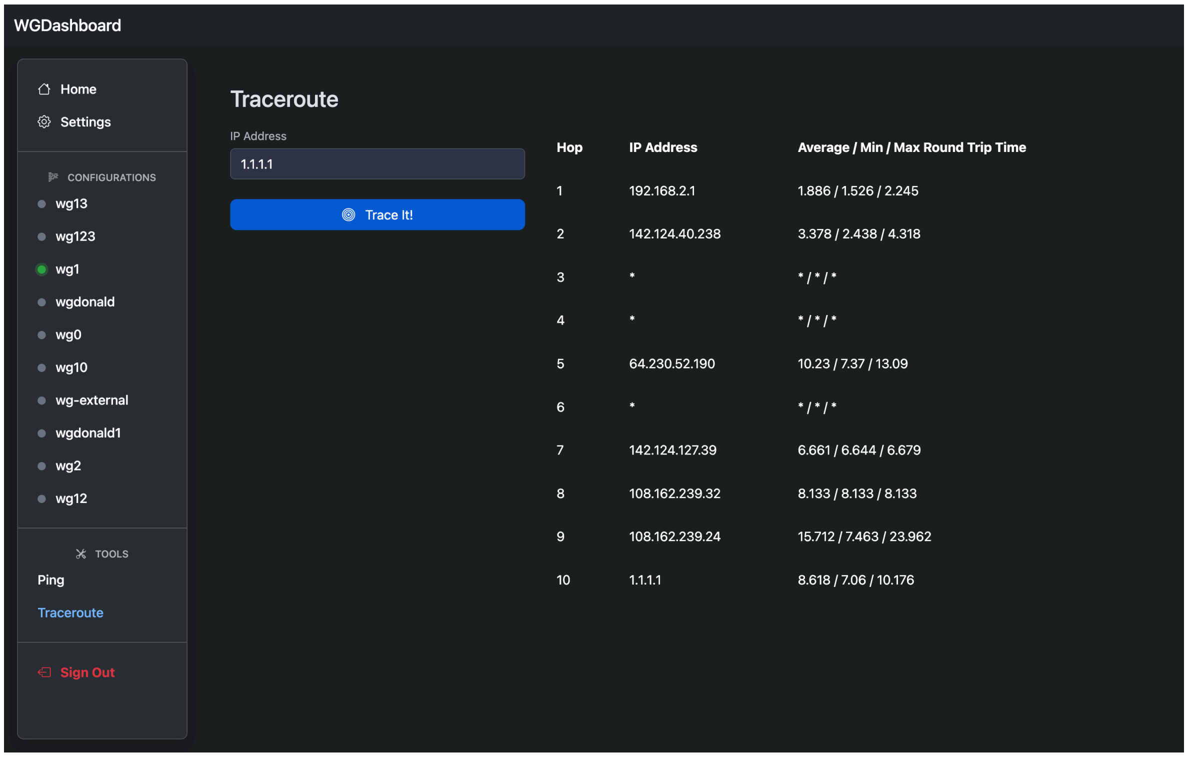
Average / Min / Max Round Trip Time

`1.667ms / 1.257ms / 2.172ms`

Sent / Received / Lost Package

`4 / 4 / 0`

Ping



Hop	IP Address	Average / Min / Max Round Trip Time
1	192.168.2.1	1.886 / 1.526 / 2.245
2	142.124.40.238	3.378 / 2.438 / 4.318
3	*	* / * / *
4	*	* / * / *
5	64.230.52.190	10.23 / 7.37 / 13.09
6	*	* / * / *
7	142.124.127.39	6.661 / 6.644 / 6.679
8	108.162.239.32	8.133 / 8.133 / 8.133
9	108.162.239.24	15.712 / 7.463 / 23.962
10	1.1.1.1	8.618 / 7.06 / 10.176

Traceroute

🕰️ Changelogs

Please use the navigation bar on the left to find each changelog

v3.0.0 - v3.0.6.2

- 🎉 New Features
  - **Moved from TinyDB to SQLite**: SQLite provide a better performance and loading speed when getting peers! Also avoided crashing the database due to **race condition.**
  - **Added Gunicorn WSGI Server**: This could provide more stable on handling HTTP request, and more flexibility in the future (such as HTTPS support). **BIG THANKS to @pgalonza ❤**
  - **Add Peers by Bulk:** User can add peers by bulk, just simply set the amount and click add.
  - **Delete Peers by Bulk**: User can delete peers by bulk, without deleting peers one by one.
  - **Download Peers in Zip**: User can download all *downloadable* peers in a zip.
  - **Added Pre-shared Key to peers:** Now each peer can add with a pre-shared key to enhance security. Previously added peers can add the pre-shared key through the peer setting button.
  - **Redirect Back to Previous Page:** The dashboard will now redirect you back to your previous page if the current session got timed out and you need to sign in again.
  - **Added Some** 🥘 **Experimental Features**
- 🪚 Bug Fixed
  - IP Sorting range issues #99 (https://github.com/donaldzou/WGDashboard/issues/99) [❤️ @barryboom]
  - INvalid character written to tunnel json file #108 (https://github.com/donaldzou/WGDashboard/issues/108) [❤️ @ikidd]
  - Add IPv6 #91 (https://github.com/donaldzou/WGDashboard/pull/91) [❤️ @pgalonza]

  - Added MTU and PersistentKeepalive to QR code and download files #112 (https://github.com/donaldzou/WGDashboard/pull/112) [❤ @reafian]
  - **And many other bugs provided by our beloved users ❤**
- 🧐 **Other Changes**
  - **Key generating moved to front-end**: No longer need to use the server's WireGuard to generate keys, thanks to the `wireguard.js` from the official repository (https://git.zx2c4.com/wireguard-tools/tree/contrib/keygen-html/wireguard.js)!
  - **Peer transfer calculation**: each peer will now show all transfer amount (previously was only showing transfer amount from the last configuration start-up).
  - **UI adjustment on running peers**: peers will have a new style indicating that it is running.
  - **`wgd.sh` finally can update itself**: So now user could update the whole dashboard from `wgd.sh`, with the `update` command.
  - **Minified JS and CSS files**: Although only a small changes on the file size, but I think is still a good practice to save a bit of bandwidth ;)

And many other small changes for performance and bug fixes! 😆

v2.3.1

- Updated dashboard's name to **WGDashboard**!!

v2.3

- 🎉 **New Features**
  - **Update directly from `wgd.sh`:** Now you can update WGDashboard directly from the bash script.
  - **Displaying Peers:** You can switch the display mode between list and table in the configuration page.
- 🪚 **Bug Fixed**
  - Peer DNS Validation Fails #67 (https://github.com/donaldzou/WGDashboard/issues/67): Added DNS format check. [❤️ @realfian]
  - configparser.NoSectionError: No section: 'Interface' #66 (https://github.com/donaldzou/WGDashboard/issues/66): Changed permission requirement for `etc/wireguard` from `744` to `755`. [❤️ @ramalmaty]
  - Feature request: Interface not loading when information missing #73 (https://github.com/donaldzou/WGDashboard/issues/73): Fixed when Configuration Address and Listen Port is missing will crash the dashboard. [❤️ @js32]
  - Remote Peer, MTU and PersistentKeepalives added #70 (https://github.com/donaldzou/WGDashboard/pull/70): Added MTU, remote peer and Persistent Keepalive. [❤️ @realfian]
  - Fixes DNS check to support search domain #65 (https://github.com/donaldzou/WGDashboard/pull/65): Added allow input domain into DNS. [❤️ @davejlong]
- 🧐 **Other Changes**
  - Moved Add Peer Button into the right bottom corner.

v2.2.1

Bug Fixed:

- Added support for full subnet on Allowed IP
- Peer setting Save button

v2.2

- 🎉 New Features
  - **Add new peers**: Now you can add peers directly on dashboard, it will generate a pair of private key and public key. You can also set its DNS, endpoint allowed IPs. Both can set a default value in the setting page. [❤️ in #44 (https://github.com/donaldzou/wireguard-dashboard/issues/44)]
  - **QR Code:** You can add the private key in peer setting of your existed peer to create a QR code. Or just create a new one, dashboard will now be able to auto generate a private key and public key ;) Don't worry, all keys will be generated on your machine, and **will delete all key files after they got generated.** [❤️ in #29 (https://github.com/donaldzou/wireguard-dashboard/issues/29)]
  - **Peer configuration file download:** Same as QR code, you now can download the peer configuration file, so you don't need to manually input all the details on the peer machine! [❤️ in #40 (https://github.com/donaldzou/wireguard-dashboard/issues/40)]
  - **Search peers**: You can now search peers by their name.
  - **Autostart on boot:** Added a tutorial on how to start the dashboard to on boot! Please read the tutorial below ("Run WGDashboard as a system service" in "🪜 Usage"). [❤️ in #29 (https://github.com/donaldzou/wireguard-dashboard/issues/29)]
  - **Click to copy**: You can now click and copy all peer's public key and configuration's public key.
- 🪚 Bug Fixed
  - When there are comments in the wireguard config file, will cause the dashboard to crash.
  - Used regex to search for config files.
- 🧐 Other Changes

- Moved all external CSS and JavaScript file to local hosting (Except Bootstrap Icon, due to large amount of SVG files).
- Updated Python dependencies
    - Flask: `v1.1.2 => v2.0.1`
    - Jinja: `v2.10.1 => v3.0.1`
    - icmplib: `v2.1.1 => v3.0.1`
- Updated CSS/JS dependencies
    - Bootstrap: `v4.5.3 => v4.6.0`
- UI adjustment
    - Adjusted how peers will display in larger screens, used to be 1 row per peer, now is 3 peers in 1 row.

v2.1

- Added **Ping** and **Traceroute** tools!
- Adjusted the calculation of data usage on each peers
- Added refresh interval of the dashboard
- Bug fixed when no configuration on fresh install (#23 (https://github.com/donaldzou/wireguard-dashboard/issues/23))
- Fixed crash when too many peers (#22 (https://github.com/donaldzou/wireguard-dashboard/issues/22))

v2.0

- Added login function to dashboard
    - ***I'm not using the most ideal way to store the username and password, feel free to provide a better way to do this if you any good idea!***
- Added a config file to the dashboard
- Dashboard config can be changed within the **Setting** tab on the sidebar
- Adjusted UI
- And much more!

v1.1.2

- Resolved issue #3 (https://github.com/donaldzou/wireguard-dashboard/issues/3).

v.1.1.1

- Able to add a friendly name to each peer. Thanks #2 (https://github.com/donaldzou/wireguard-dashboard/issues/2)!

v1.0 - Journey starts here!

- Added the function to remove peers