

---

# **Linguard**

***Release 1.0.0***

**José Antonio Mazón San Bartolomé**

**Oct 21, 2021**



# CONTENTS

|          |                             |           |
|----------|-----------------------------|-----------|
| <b>1</b> | <b>Key features</b>         | <b>3</b>  |
| <b>2</b> | <b>Contents</b>             | <b>5</b>  |
| 2.1      | Installation . . . . .      | 5         |
| 2.2      | How does it look? . . . . . | 5         |
| 2.3      | In depth . . . . .          | 12        |
| 2.4      | Contributing . . . . .      | 17        |
| 2.5      | Changelog . . . . .         | 19        |
| <b>3</b> | <b>Indices and tables</b>   | <b>23</b> |



Linguard aims to provide a clean, simple yet powerful web GUI to manage your WireGuard server, and it's powered by Flask.



## **KEY FEATURES**

- Management of Wireguard interfaces and peers via web. Interfaces can be created, removed, edited, exported and brought up and down directly from the web GUI. Peers can be created, removed, edited and downloaded at anytime as well.
- Display stored and real time traffic data using charts (storage of traffic data may be manually disabled).
- Display general network information.
- Encrypted user credentials (AES).
- Easy management through the `linguard systemd` service.



**CONTENTS**

## 2.1 Installation

### 2.1.1 As a systemd service

1. Download [any release](#).
2. Extract it and run the installation script:

```
chmod +x install.sh  
sudo ./install.sh
```

3. Run Linguard:

```
sudo systemctl start linguard.service
```

### 2.1.2 Using docker

1. Download the `docker-compose.yaml` file [from the repository](#).
2. Run Linguard:

```
sudo docker-compose up -d
```

---

**Note:** You can check all available tags [here](#).

---

## 2.2 How does it look?

Here are a bunch of screenshots:

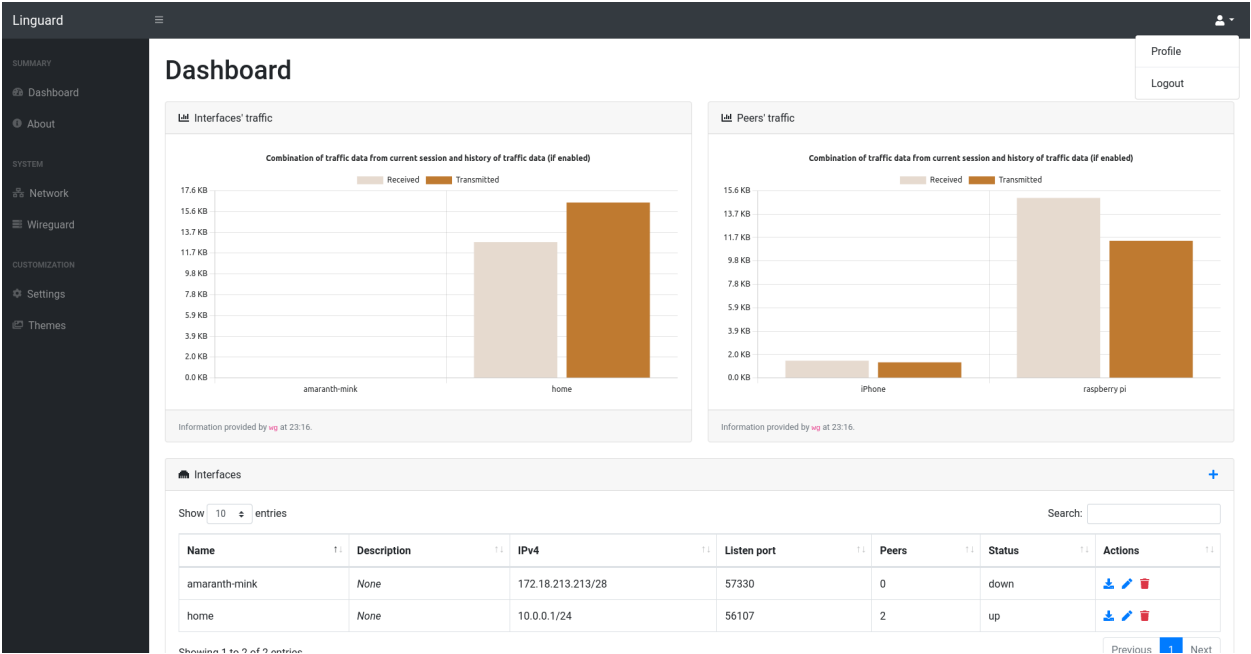


Fig. 1: Dashboard (1)

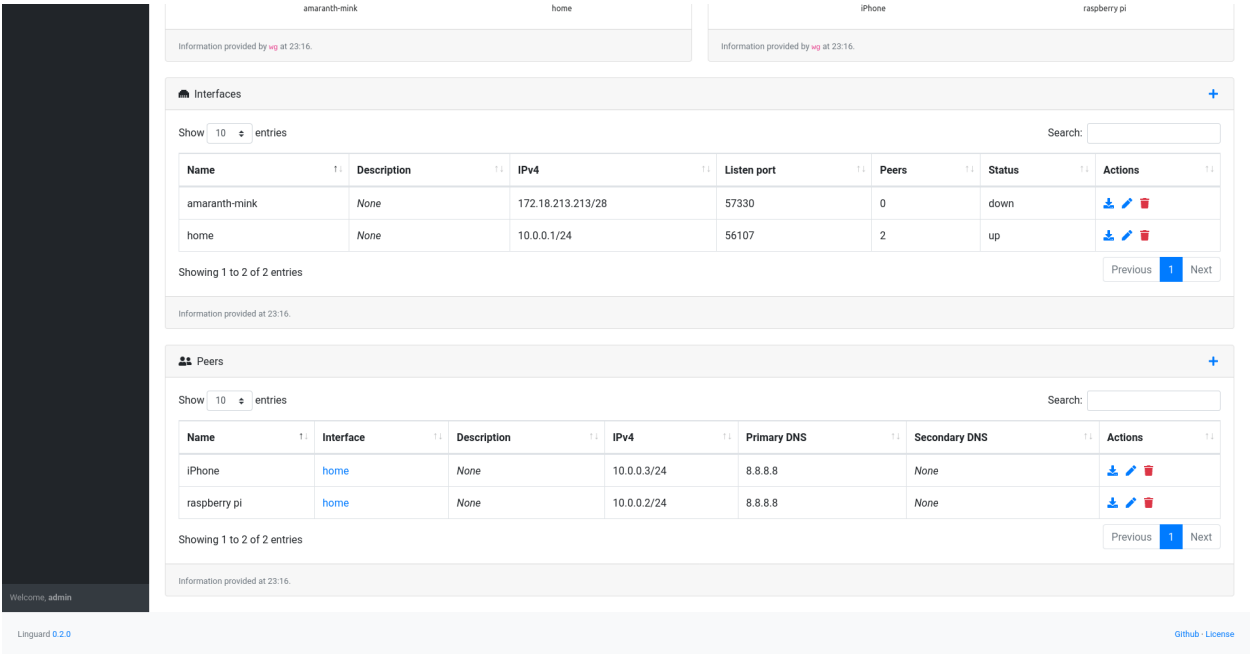


Fig. 2: Dashboard (2)

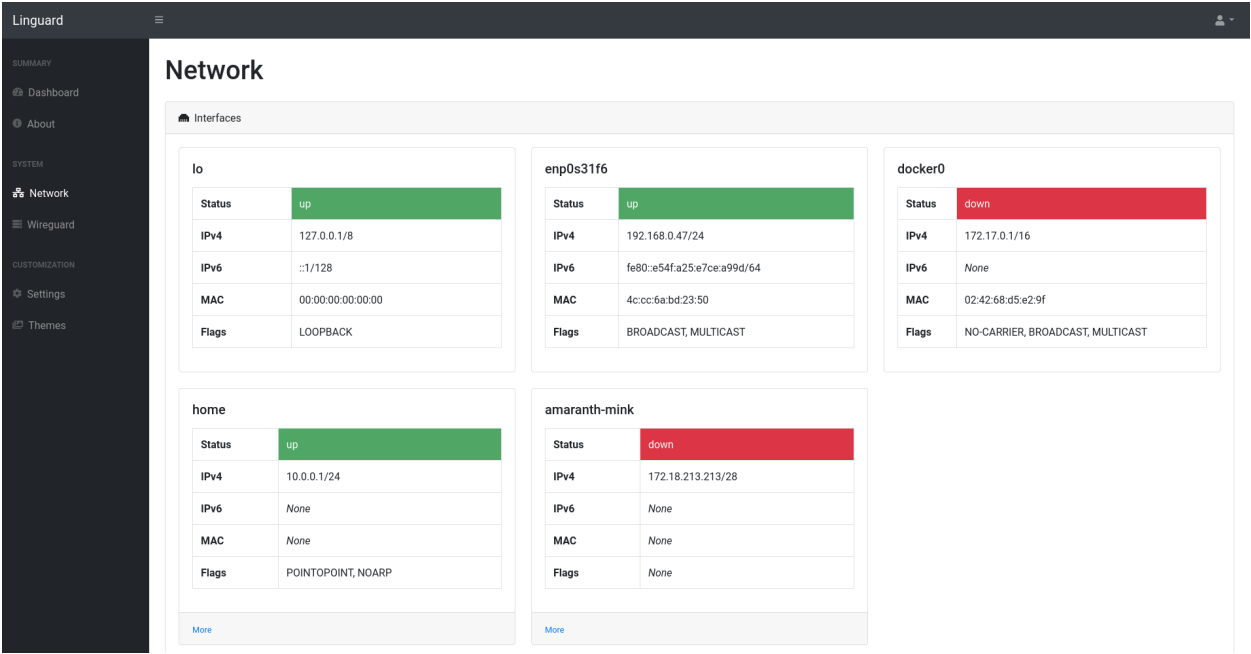


Fig. 3: Network interfaces

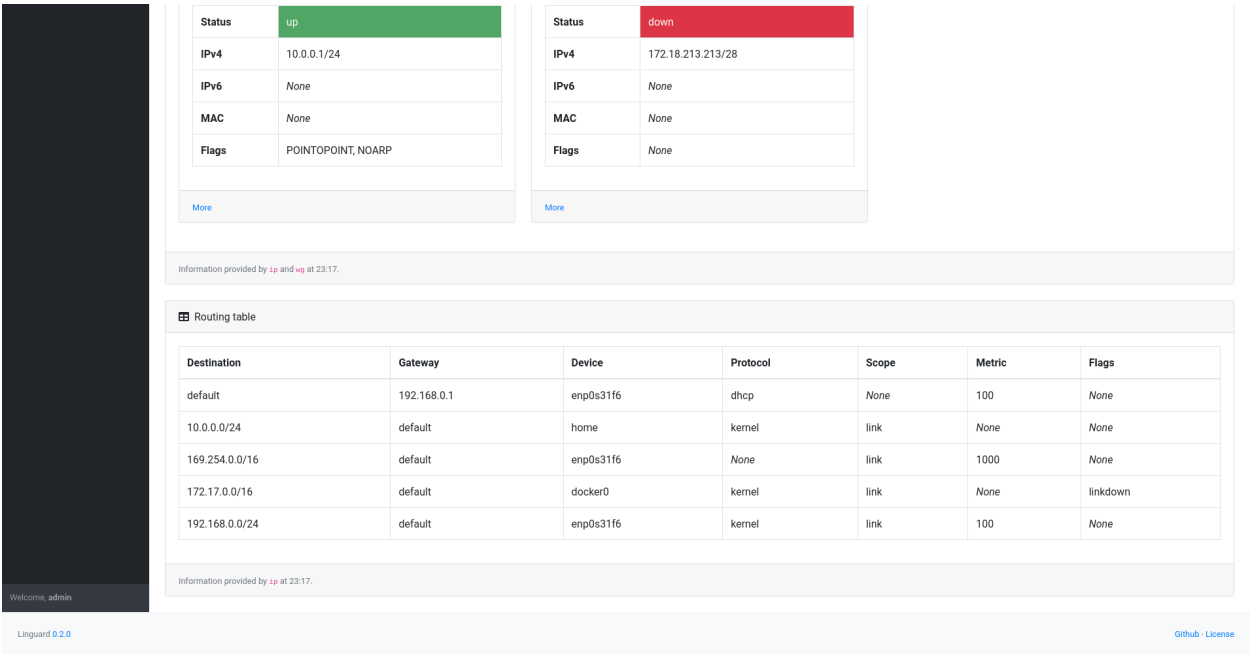


Fig. 4: Routing information

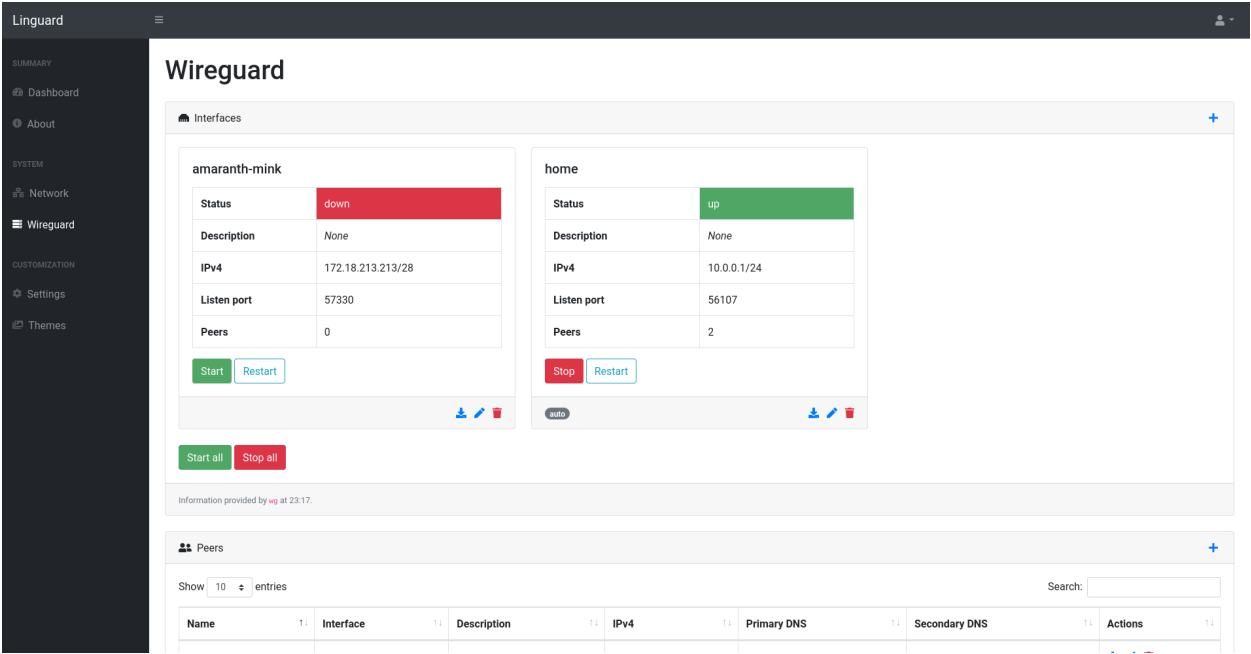


Fig. 5: Wireguard interfaces

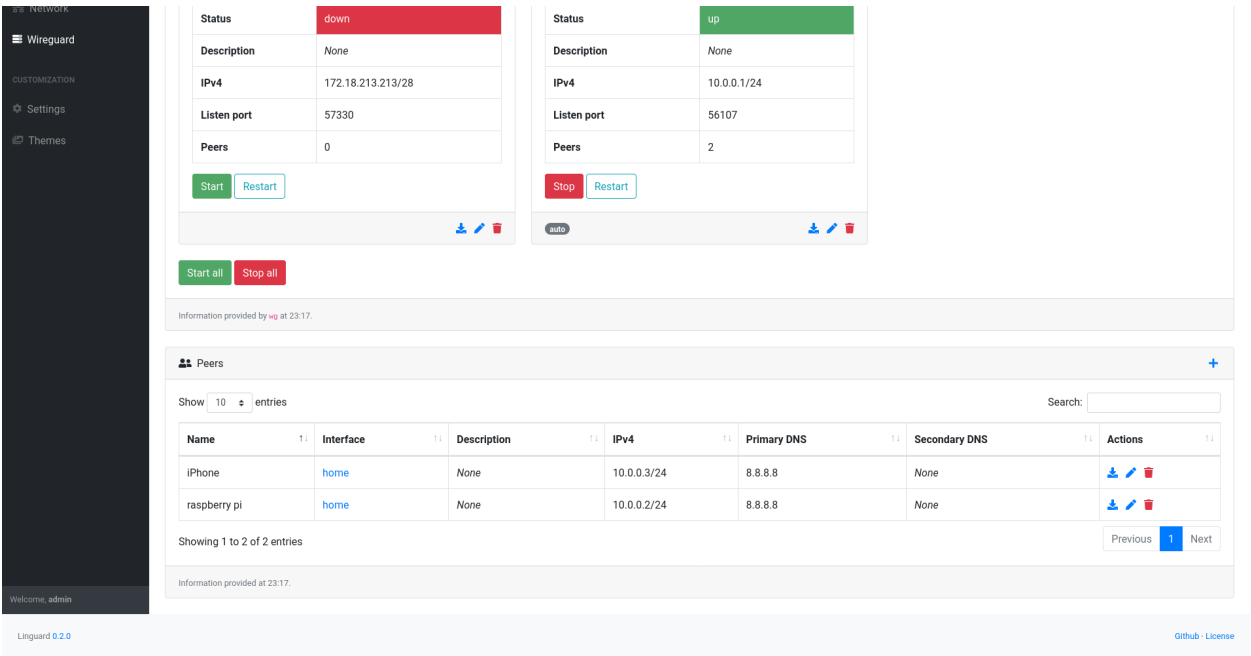


Fig. 6: Wireguard peers

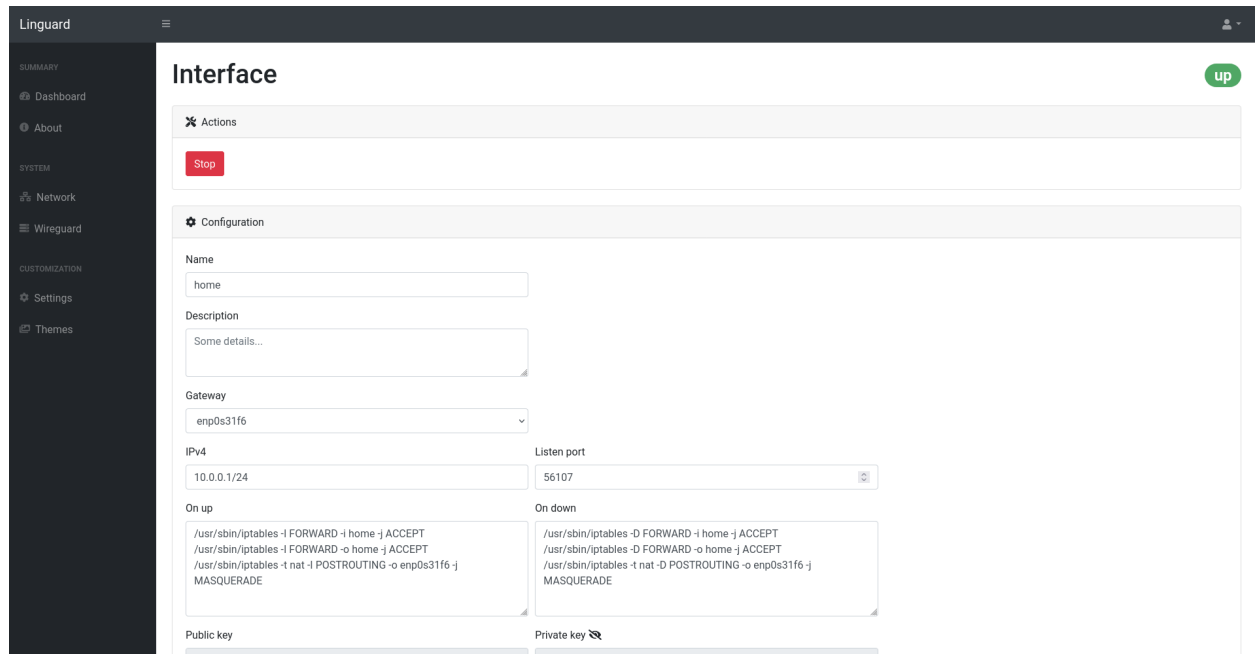


Fig. 7: Interface's actions and configuration

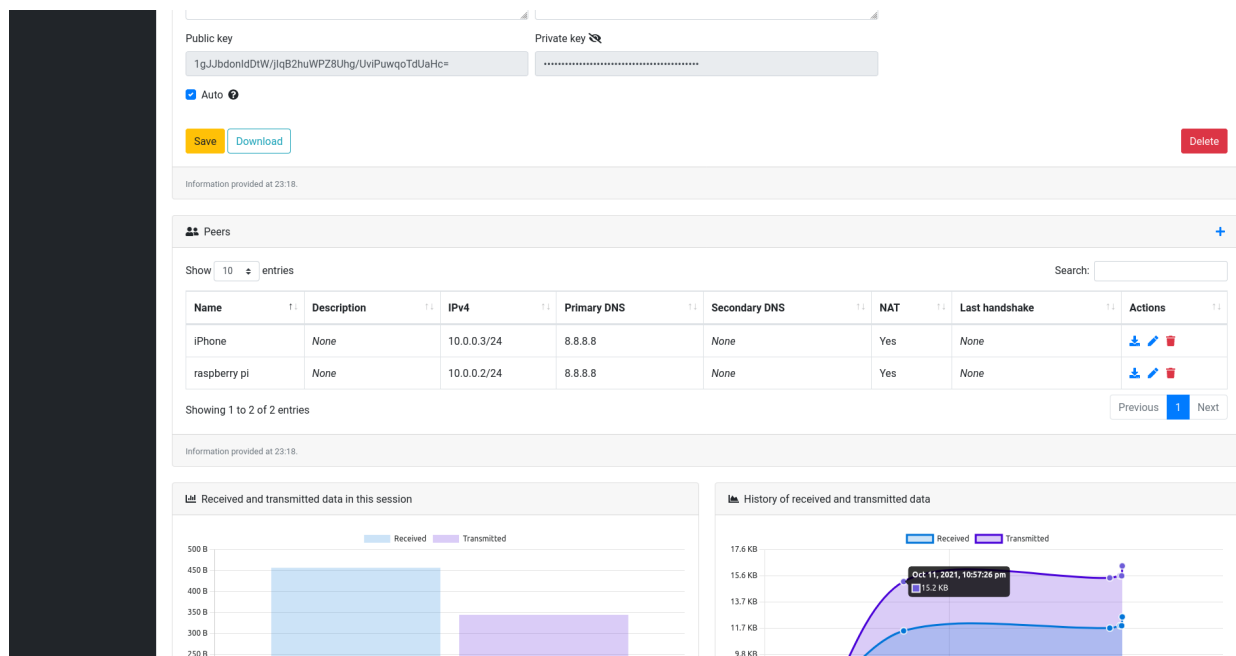


Fig. 8: Interface's peers and traffic data (1)

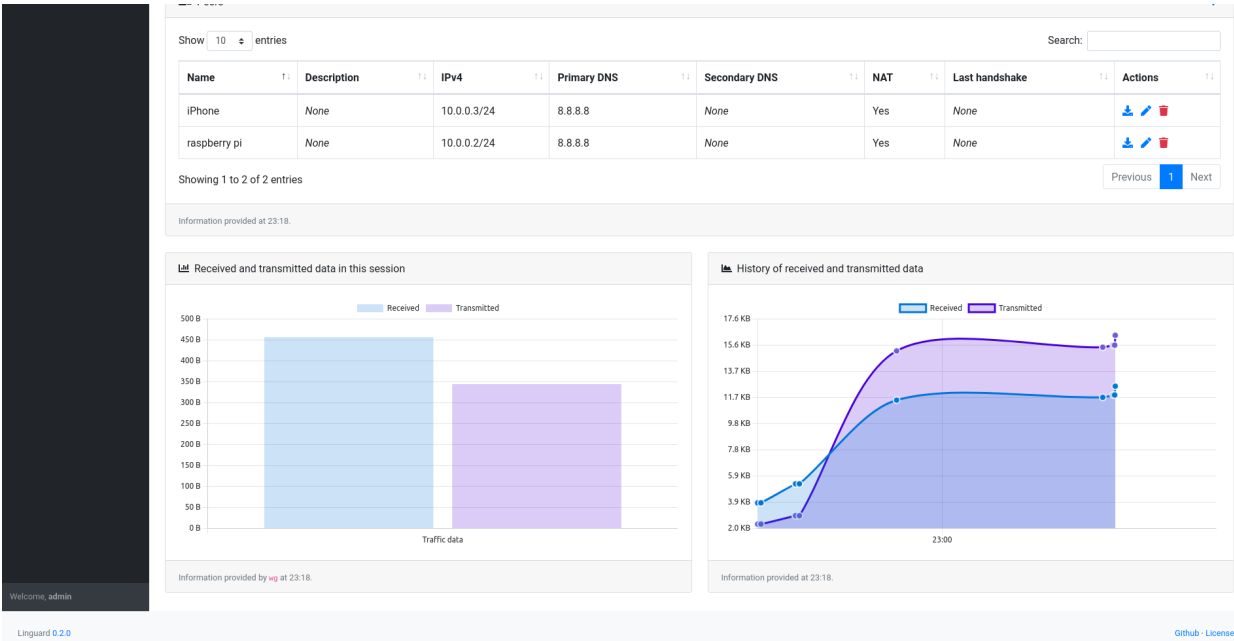


Fig. 9: Interface's peers and traffic data (2)

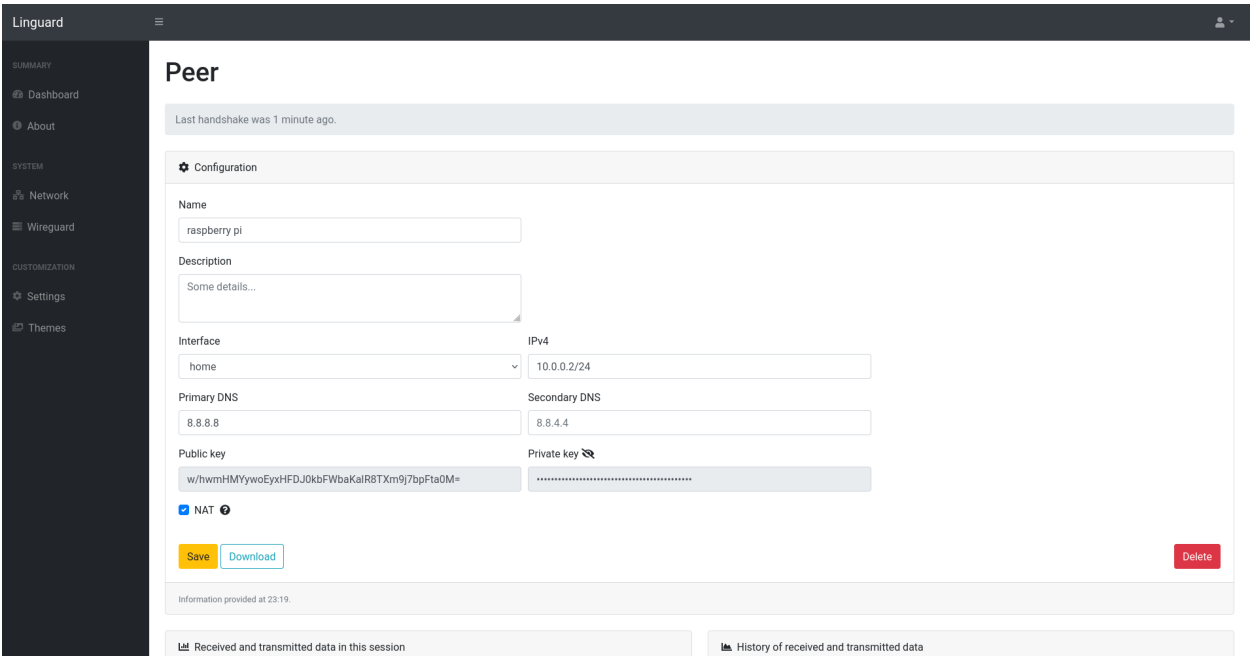


Fig. 10: Peer's configuration

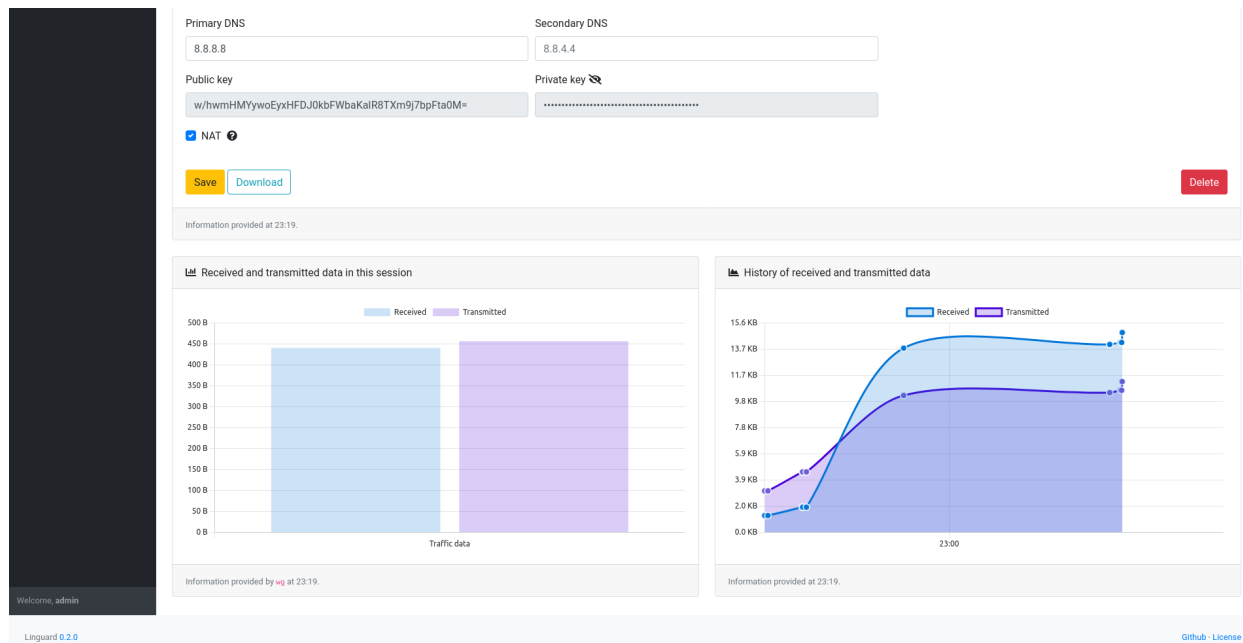


Fig. 11: Peer's traffic data

## Settings

[Back](#)

**Web**

Maximum login attempts: 0

Secret key: [REDACTED]

Credentials file: /var/www/linguard/data/.credentials

**Wireguard**

Endpoint: 192.168.0.47

Interfaces' directory: /var/www/linguard/data/interfaces

Configuration file: /var/www/linguard/data/linguard.yaml

wg bin: /usr/bin/wg

wg-quick bin: /usr/bin/wg-quick

iptables bin: /usr/sbin/iptables

**Logging**

Logfile: /var/www/linguard/data/linguard.log

Log Level: debug

☐ Overwrite

**Traffic data collection**

☒ Enabled

Fig. 12: Settings (1)

The screenshot shows the Linguard settings interface. On the left is a dark sidebar with a 'Themes' link and a 'Welcome, admin' message. The main content area has a light gray background and contains several configuration sections:

- Network Configuration:** A grid of input fields for IP address (192.168.0.47), interfaces (/var/www/linguard/data/interfaces), linguard.yaml (/var/www/linguard/data/linguard.yaml), wg bin (/usr/bin/wg), wg-quick bin (/usr/bin/wg-quick), and iptables bin (/usr/sbin/iptables).
- Logging:** A section with a 'Logfile' field (/var/www/linguard/data/linguard.log), a 'Log Level' dropdown menu set to 'debug', and an 'Overwrite' checkbox.
- Traffic data collection:** A section with an 'Enabled' checkbox (checked), a 'Driver' dropdown menu set to 'JSON', and a 'Driver configuration' text area containing a JSON object: 

```
{ "timestamp_format": "%d/%m/%Y %H:%M:%S" }
```

.

At the bottom of the settings area is a yellow 'Save' button. The footer of the interface shows 'Linguard 0.2.0' on the left and 'Github License' on the right.

Fig. 13: Settings (2)

## 2.3 In depth

### 2.3.1 Arguments

The following table describes every argument accepted by Linguard:

| Argument                  | Type       | Explanation                              | Notes                                                |
|---------------------------|------------|------------------------------------------|------------------------------------------------------|
| <i>workdir</i>            | Positional | Path to the Linguard's working directory | Linguard will store here everything it needs to work |
| <i>-h</i>   <i>--help</i> | Optional   | Display Linguard's CLI help and exit     |                                                      |
| <i>--debug</i>            | Optional   | Start the Flask backend in debug mode    | Default value is <code>False</code>                  |

### 2.3.2 Configuration

Two sample configuration files are provided, `uwsgi.sample.yaml` and `linguard.sample.yaml`, although the most interesting one is the second, since the first only contains options for a third party software, [UWSGI](#).

Nonetheless, it is worth noting that the path to the Linguard's working directory (which will be used by Linguard to store stuff) needs to be provided through uwsgi's configuration, using the field `pyargv`. Moreover, to edit the port and/or the interface in which the web server is running you will need to edit the field `http-socket` of uwsgi's configuration file.

For now on, we will only discuss Linguard's configuration values. Although the file `linguard.sample.yaml` contains every possible option, the following tables explain each one of them and detail all possible values.

## Logging configuration

These options must be specified inside a `logger` node.

| Option            | Explanation                                                          | Values                             | Default |
|-------------------|----------------------------------------------------------------------|------------------------------------|---------|
| <i>level</i>      | Set the minimum level of messages to be logged                       | debug, info, warning, error, fatal | info    |
| <i>over-write</i> | Whether to overwrite the log file when the application starts or not | true, false                        | false   |

## Web configuration

These options must be specified inside a `web` node.

| Option                | Explanation                                                                 | Values                      | Default                            |
|-----------------------|-----------------------------------------------------------------------------|-----------------------------|------------------------------------|
| <i>login_attempts</i> | Maximum number of login attempts within <code>login_ban_time</code>         | (almost) Any integer        | 0 (unlimited attempts)             |
| <i>login_ban_time</i> | Amount of seconds an IP will be banned after too many failed login attempts | (almost) Any integer        | 120                                |
| <i>secret_key</i>     | Key used to secure the authentication process                               | A 32 characters long string | A random 32 characters long string |

## Traffic data collection configuration

These options must be specified inside a `traffic` node.

| Option         | Explanation                                   | Values                                                                                                                                                                                  | Default                                               |
|----------------|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| <i>driver</i>  | Driver used to save and load traffic data     | Any registered driver. You can even craft your own driver using the base class <code>TrafficStorageDriver</code> . Further information will be available through the code documentation | The JSON driver, which stores data serialized as JSON |
| <i>enabled</i> | Whether the data collection is enabled or not | true, false                                                                                                                                                                             | true                                                  |

**Note:** Linguard will only store the **amount of bytes received and transmitted** by peers, and only if `enabled` is set to `true`.

## Wireguard configuration

These options must be specified inside a `wireguard` node.

### Global options

| Option              | Explanation                                                       | Values                                                                                      | Default                                                                       |
|---------------------|-------------------------------------------------------------------|---------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| <i>end-point</i>    | Endpoint for all peers                                            | Should be something like <code>vpn.example.com</code> , though it may also be an IP address | Default value will be your computer's public IP (if it can be obtained)       |
| <i>wg_bin</i>       | Path to the WireGuard binary file ( <code>wg</code> )             | <code>path/to/file</code>                                                                   | If not specified, it will be retrieved using the <code>whereis</code> command |
| <i>wg_quick_bin</i> | Path to the WireGuard quick binary file ( <code>wg-quick</code> ) | <code>path/to/file</code>                                                                   | If not specified, it will be retrieved using the <code>whereis</code> command |
| <i>interfaces</i>   | Dictionary containing all interfaces of the server                | A number of <code>interface</code> nodes whose keys are their own UUIDs                     |                                                                               |
| <i>iptables_bin</i> | Path to the iptables binary file ( <code>iptables</code> )        | <code>path/to/file</code>                                                                   | If not specified, it will be retrieved using the <code>whereis</code> command |

## Interface configuration

These options must be specified inside an `interface` node.

| Option             | Explanation                                                                          | Values                                                | Default                                                                                                                                           |
|--------------------|--------------------------------------------------------------------------------------|-------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>auto</i>        | Whether the interface will be automatically brought up when the server starts or not | true, false                                           | Default value is true                                                                                                                             |
| <i>description</i> | A description of the interface                                                       | A character string                                    |                                                                                                                                                   |
| <i>gw_iface</i>    | Interface used to connect the WireGuard interface to your network                    | A valid network device                                | Your computer's default gateway                                                                                                                   |
| <i>ipv4_addr</i>   | IPv4 address assigned to the interface                                               | A valid IPv4 address                                  |                                                                                                                                                   |
| <i>listen_port</i> | UDP port used by WireGuard to communicate with peers                                 | 1-65535                                               |                                                                                                                                                   |
| <i>name</i>        | The interface's name                                                                 | A character string                                    | It may only contain alphanumeric characters, underscores and hyphens. It must also begin with a letter and cannot be more than 15 characters long |
| <i>on_up</i>       | Linux commands to be executed when the interface is going to be brought up           | Any linux command in path                             | By default, it will add FORWARD and POSTROUTING rules related to the interface                                                                    |
| <i>on_down</i>     | Linux commands to be executed when the interface is going to be brought down         | Any linux command in path                             | By default, it will remove FORWARD and POSTROUTING rules related to the interface                                                                 |
| <i>peers</i>       | Dictionary containing all peers of the interface                                     | A number of peer nodes whose keys are their own UUIDs |                                                                                                                                                   |
| <i>private_key</i> | Private key used to authenticate the interface                                       | A valid private key generated via wg                  |                                                                                                                                                   |
| <i>public_key</i>  | Public key used to authenticate the interface                                        | A valid private key generated via wg                  |                                                                                                                                                   |
| <i>uuid</i>        | Unique identifier                                                                    | A valid Version 4 UUID                                |                                                                                                                                                   |

## Peer configuration

These options must be specified inside an **peer** node.

| Option              | Explanation                                                                | Values                               | Default                                                                                         |
|---------------------|----------------------------------------------------------------------------|--------------------------------------|-------------------------------------------------------------------------------------------------|
| <i>dns1</i>         | Main DNS used by the peer                                                  | A valid IPv4 address                 |                                                                                                 |
| <i>dns2</i>         | Secondary DNS used by the peer                                             | A valid IPv4 address                 |                                                                                                 |
| <i>ipv4_address</i> | IPv4 address assigned to the peer                                          | A valid IPv4 address                 |                                                                                                 |
| <i>name</i>         | The peer's name                                                            | A character string                   |                                                                                                 |
| <i>nat</i>          | Linux commands to be executed when the interface is going to be brought up | Any linux command in path            | Default value is false. If true, this option will enable the PersistentKeepalive WireGuard flag |
| <i>private_key</i>  | Private key used to authenticate the peer                                  | A valid private key generated via wg |                                                                                                 |
| <i>public_key</i>   | Public key used to authenticate the peer                                   | A valid private key generated via wg |                                                                                                 |
| <i>uuid</i>         | Unique identifier                                                          | A valid Version 4 UUID               |                                                                                                 |

### 2.3.3 Security

Although Linguard stores users' credentials encrypted, it does not implement end-to-end encryption and instead, it relays on TLS to secure the communication between the user and the server. This means you should never run Linguard on its own, but use the `https` option of uWSGI or set up a reverse proxy if you wish to use plain HTTP with uWSGI. Don't worry, here's how:

#### uWSGI with HTTPS socket

```
uwsgi:
  https: 0.0.0.0:8443,foobar.crt,foobar.key # More info at https://uwsgi-docs.
  ↪readthedocs.io/en/latest/HTTPS.html
  master: true
  enable-threads: true
  chdir: /var/www/linguard
  venv: venv
  wsgi-file: linguard/__main__.py
  pyargv: data
  need-plugin: python3
  callable: app
  die-on-term: true
  chmod-socket: 660
  vacuum: true
```

## Apache reverse proxy

```
<VirtualHost *:443>
    ServerName vpn.example.com

    ErrorLog ${APACHE*LOG*DIR}/error.log
    CustomLog ${APACHE*LOG*DIR}/access.log combined

    SSLEngine on
    SSLCertificateFile /path/to/crt
    SSLCertificateKeyFile /path/to/key
    SSLProtocol -all +TLSv1.2 +TLSv1.3

    ProxyPreserveHost On
    ProxyPass / http://10.0.0.1:8080/
    ProxyPassReverse / http://10.0.0.1:8080/
</VirtualHost>
```

## Nginx reverse proxy

```
server {
    listen 443;
    server_name      vpn.example.com;

    ssl_certificate   /path/to/crt;
    ssl_certificate_key /path/to/key;
    ssl_protocols     TLSv1.2 TLSv1.3;

    location / {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://10.0.0.1:8080;
    }
}
```

## 2.4 Contributing

---

**Note:** Linguard is and will always be open source.

---

You may contribute by opening new issues, commenting on existent ones and creating pull requests with new features and bugfixes. Any help is welcome, just make sure you read the following sections, which will guide you to set up the development environment.

## 2.4.1 Git flow

You should never work directly on the main branch. This branch is only used to gather new features and bugfixes previously merged to the dev branch and publish them in a single package. In other words, its purpose is to release new versions of Linguard.

Hence, the dev branch **should always be your starting point and the target of your pull requests**.

```
git clone https://github.com/joseantmazonsb/linguard.git
cd linguard
git checkout dev
```

## 2.4.2 Requirements

You will need to install the following Linux packages:

```
sudo iproute2 python3 python3-venv wireguard iptables libpcrc3 libpcrc3-dev uwsgi uwsgi-
↳plugin-python3
```

## 2.4.3 Dependency management

Poetry is used to handle packaging and dependencies. You will need to install it before getting started to code:

```
curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/install-poetry.
↳py | python3 -
```

Once you have checked out the repository, you'd install the python requirements this way:

```
poetry config virtualenvs.in-project true
poetry install
```

Then, you would only need to run `poetry shell` and voilà, ready to code!

---

**Note:** Actually, you should always run `poetry run pytest` before getting started to code in order to check that everything's all right.

---

## 2.4.4 Configuration files

Linguard has a setup assistant and does not require you to have an existing configuration file in its working directory. Nonetheless, you may use your own existing file as long as it is valid and named `linguard.yaml`.

As for the UWSGI configuration, Linguard provides a sample file (`uwsgi.sample.yaml`) for you to play around with it. Just make sure you run UWSGI using a valid file!

## 2.4.5 Testing

PyTest and Coverage are used to test Linguard and generate coverage reports, which are uploaded to Codecov.

TDD is enforced. Make sure your code passes the existing tests and provide new ones to prove your new features/bugfixes actually work when making pull requests.

All tests should be anywhere under `linguard/tests`, and you can run them all using Poetry:

```
poetry run pytest
```

You may as well generate a coverage report using poetry:

```
poetry run coverage run -m pytest && poetry run coverage report
```

## 2.4.6 Building

To build Linguard you may use the `build.sh` script, which automatically generates a `dist` folder containing a compressed file with all you need to publish a release.

## 2.4.7 Versioning

Linguard is adhered to Semantic Versioning.

All releases must follow the format `{MAJOR} . {MINOR} . {PATCH}`, and git tags linked to releases must follow the format `v{MAJOR} . {MINOR} . {PATCH}`. Thus, release `1.0.0` would be linked to the `v1.0.0` git tag.

## 2.4.8 CI/CD

Github Workflows are used to implement a CI/CD pipeline. When pull requests targeting the `main` or `dev` branches are opened, a series of tests will automatically be ran to ensure everything is working properly.

**Warning:** The `main` branch is used to automatically deploy new releases, and **should never be the target of external pull requests**.

## 2.5 Changelog

All notable changes to this project will be documented here.

**Note:** Linguard is adhered to Semantic Versioning.

## 2.5.1 1.1.0

### What's new

- Ban time is now editable and applies to individual IP addresses instead of globally (which makes much more sense).

### Fixes

- Fixed a bug with the settings page which caused the display of default/last saved settings everytime the page was reloaded, even though the values were actually being stored in the configuration file and applied.

### Docs

- Added entry for ban time.

## 2.5.2 1.0.1

### Fixes

- Fixed a bug related to versioning which caused the app to start in dev mode.

### Docs

- Removed "Versions" empty section from index.

## 2.5.3 1.0.0

### What's new

- QR codes! You can scan a QR code to get the WireGuard configuration of any peer or interface.
- Docker is finally here! For now on, there will be official docker images available for every release.
- Display the IP address of the interface to be used when adding or editing a peer.
- Updating the name of an interface also updates all references inside the "On up" and "On down" text areas.
- Delete buttons have been relocated in the Interface and Peer views.

### Fixes

- Fixed a bug when updating the username or password which made the "Logged in {time} ago" sign show no time at all.
- Removed the possibility to add peers if there are no WireGuard interfaces.
- Ensured that peers can only be assigned valid, unused and not reserved IP addresses.
- Ensured that peers' IP addresses are in the same network of their interface.
- Ensured that interfaces can only be assigned valid, unused and not reserved IP addresses.

- Ensured that interfaces' cannot be assigned an IP address belonging to a network which already has an interface.
- Fixed a bug when updating an interface's gateway, which only updated one appearance of the previous gateway in the "On up" and "On down" text areas.
- Fixed the behaviour of the `overwrite` flag regarding the logging settings which was causing to overwrite the log file each time the settings were saved instead of every time Linguard boots up.

## Docs

- Improved documentation about the development environment.
- Fixed a bunch of typos.
- Fixed the Traffic Data Driver table.

## 2.5.4 0.2.0

- Easy first time setup, which automatically detects the location of the required binaries and sets the public IP as endpoint by default.
- Everything in one place: `workdir`-based architecture.
- Removed option to log to standard output.
- Includes a ready-to-go uWSGI configuration file.
- Removed the `linguard.sample.yaml` file in favour of the first time setup.
- Settings are now accessible through the side navbar.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`