upribox Documentation

Release 1.0

Markus Huber

Nov 13, 2017

Contents

Note: If you bought a pre-assembled upribox from our online store you do not have to read this guide since everything is already set up and you only have to complete the simple tasks described in the enclosed manual.

In a few easy steps you can build your own upribox and won't get bothered anymore by annoying ads and trackers (see *Silent Mode*). Additionally you can use your upribox to surf the web anonymously via Tor (see *Ninja Mode*) and set it up to be your OpenVPN server which secures your connection in unprotected WiFis and lets you benefit from the ad-blocking features on the road (see *VPN Server*).

CHAPTER 1

Installation

You can download the latest upribox image from Github and verify its integrity and authenticity with the provided signature file and PGP key (see *Signed Releases*). See the official Raspberry Pi documentation for pointers on how to install the upribox image on the SD card. Upon the first boot the SSH/VPN keys are automatically re-generated (this will take a couple of minutes).

1.1 Hardware requirements

In the following you will find a list of required (and tested) hardware for the upribox software. On the Raspberry Pi 2 make sure that you use a compatible USB WiFi dongle!

1.1.1 Table of recommended hardware

Note: Since upribox v0.5 we support the Pi3 natively, so no additional WiFi dongle is required. The Pi3 WiFi however supports only one active Hotspot. You cannot activate the **Ninja WiFi** on the Pi3 without an additional TL-WN722N v1 WiFi dongle.

	Raspberry Pi 3[1]	Raspberry Pi 2[2]
SD Card	microSDHC Class 10 (min. 4GB)[3]	
Power Supply	Micro USB 5V/2A[4]	
WiFi	onboard	TL-WN722N **v1**[5]

[1] Raspberry Pi 3 [Element14] [Amazon.com]
[2] Raspberry Pi 2 [Element14] [Adafruit]
[3] Sandisk SDHC 8GB [Amazon.com]
[4] Power Supply [Amazon.com] [Adafruit]
[5] TL-WN722N Wireless USB adapter [Amazon.com]

Warning: Please note that version 2 of the TL-WN722N WiFi dongle uses a different WiFi chipset and is not supported.

The upribox software works with Raspberry Pi 1 as well, but the performance for ad-blocking is considerable worse.

CHAPTER 2

User manual

2.1 Web Interface

Once you are connected to either of the upribox WiFi networks (Silent or Ninja) you can access the upribox web interface via the following URI: **http://upri.box <http://upri.box/>**. (see *Modes*)

2.2 Default passwords

If you used the latest community upribox image for setting up your own privacy box you need the following passwords for accessing it:

Login	User	Default Password
Wifi (SSID: upribox)	•	changeme
Web Interface / SSH	upri	changethedefaults!

It is important that you change the passwords upon the first login in the admin section of the web interface. New passwords have to be at least 8 characters long containing lower-case, upper-case, numbers and special characters.

The upribox performs an auto-update every **four hours**. This includes:

- Blocking rules for privoxy and DNS
- Software updates via ansible + updates from github

Please note that this process overwrites manual changes to configuration files. To conduct persistent manual changes you have to use *custom facts* (see *Customization*).

2.2.1 Features

Silent Mode

The Silent mode automatically blocks advertisement from websites and mobile apps. In addition common tracking services are blocked as well.

What are trackers?

No one likes annoying online advertisements. But this is just the tip of the iceberg when it comes to negative side effects of your internet browsing. Much more harmful and not directly visible to you websites include tools that gather your personal information and use them across multiple websites: so called trackers.

... and why should we block them?

Trackers collect your private data and create a profile of you. This can include sensitive data like your name, gender, age or sexual orientation. Apart from the loss of sensitive information itself this can also be to your disadvantage when websites use this profile to decline or adapt their offer to you based on gathered information that my or may not be appropriate.

Ninja Mode

By activating Ninja mode the upribox will create another network (on Raspberry Pi 3 only with an additional WiFi dongle). Connected devices benefit from the same ad- and tracker blocking mechanisms as in the Silent mode but additionally the traffic will be anonymized.

The upribox Ninja mode uses the Tor anonymity network to hinder tracking of your internet usage. For example, your internet provider will not be able to see which websites you access. Your requests are tunnelled through multiple Tor relays which may cause reduction of your internet speed and websites being displayed in foreign languages.

For the best protection of your anonymity we recommend the usage of the Tor Browser (download for Windows or macOS) or alternatively Tails. If you cannot use the Tor Browser, activate and use the upribox Ninja mode.

VPN Server

You are on the move and want to access your E-mails or online banking securely? Use the upribox VPN to protect your data when using public WiFi hotspots.

VPN (virtual private network) makes it possible to surf with your upribox while on the road. Access to your upribox is protected via VPN user profiles. Create a VPN profile in the VPN section of the upribox user interface and download it to your device (e.g. smart phone).

Usage of the upribox VPN requires the creation of VPN profiles and installation of OpenVPN. For your smartphone download the official OpenVPN software for Android or iOS. For your desktop computer we recommend Tunnelblick for Mac OS or OpenVPN GUI for Windows.

2.2.2 Modes

To integrate the upribox in your home network and access the web interface via the URI *http://upri.box <http://upri.box/>* we implemented three different ways. This decision is motivated by the nature of these access modes. The most comfortable mode for the user, Apate, sometimes does not work with more complex home networks

or specific routers we don't have control over. To give you an alternative your upribox can be accessed by configuring your upribox as the DHCP server on the network which requires a small configuration on your router at home. The third mode is to connect directly to the upribox WiFi and access the web interface from the connected device.

The easiest way to configure Apate or DHCP server mode is by using the WiFi mode and connecting a device to the upribox Silent WiFi. You then can choose the mode that best fits your needs and expertise and change it whenever you like in the admin section of the upribox web interface. In the following the three modes are described in more detail:

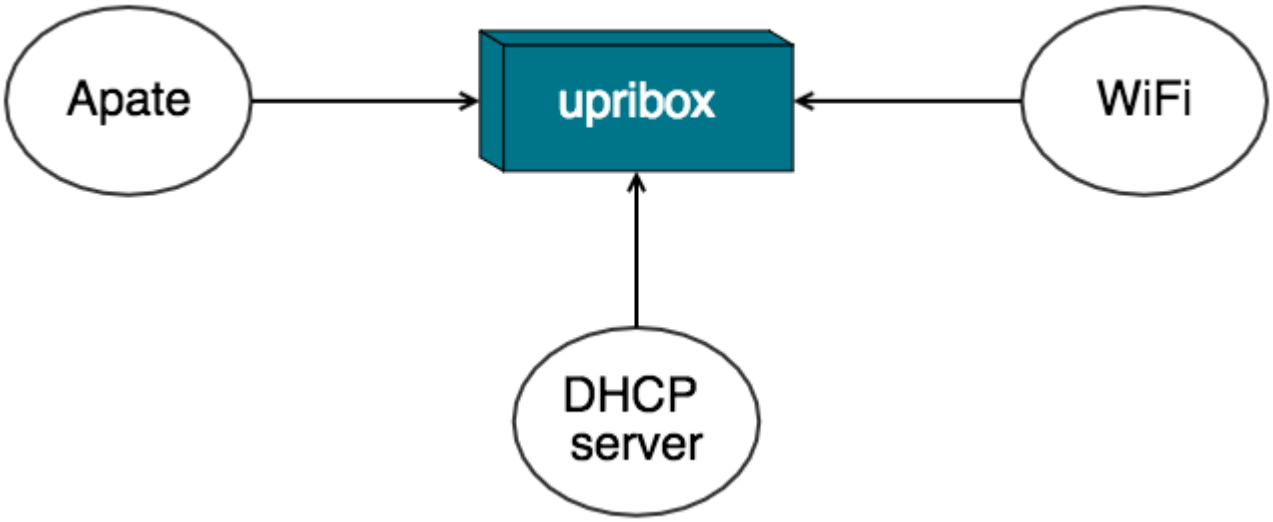

Fig. 2.1: The three different modes to access your upribox: Apate, DHCP server and WiFi

Apate

After activating Apate you don't have to configure anything else. Ads and trackers on all your devices will be blocked.

Apate is an ARP spoofing daemon which means that the upribox will trick all devices on your network into thinking that it is the real recipient of the packets. Your phone, laptop or any other device then sends the traffic over the upribox, which acts as a Man-In-The-Middle and forwards the packets. Before returning requested websites to your device the upribox will always remove advertisements and trackers for you. Some routers can detect ARP spoofing and treat it as an attack. In this case Apate won't work on your network. We therefore recommend to test Apate by activating it and trying to access the internet with another device (e.g. your smart phone).

DHCP Server

By setting up your upribox as the DHCP server of your network it will act as the distributor of IP addresses for all other devices. Additionally your upribox is again Man-In-The-Middle and can filter ads and trackers from your traffic. To activate this mode you have to make some adjustments on your upribox and your router:

- The upribox needs a static IP address. This can be set in the admin section of the upribox web interface. Additionally you have to configure the subnet mask, the DNS server and the gateway. The form is going to be filled automatically and in most cases you can just accept the provided values.
- Activate the DHCP server mode in the admin section of the upribox web interface.
- Up to this point your home router had the role of the DHCP server. But now there's a new sheriff in town. Go to the web interface of your home router and deactivate the DHCP server.

WiFi Mode

If your router does not support Apate or you don't want to change any settings on your router you can still access your upribox web interface by connecting to one of the WiFis (Silent or Ninja). Please note that this way only devices directly connected to the WiFis will benefit from the upribox' ad-blocking mechanisms.

What about IPv6?

If you are using IPv6 on your home network we recommend using the WiFi mode or alternatively setting up an additional IPv4 only WiFi router which is connected to your upribox in Apate or DHCP server mode. To use the latter please don't forget to deactivate the DHCP server on the new router.

2.2.3 Security

When we designed the upribox architecture and features security was the first thing on our minds. We wanted to create a box that protects your privacy while surfing the web and we knew that our ambitions would be in vain if we endangered this valuable asset with the mechanisms that were supposed to guard it. For this reason every new feature has to undergo a strict security evaluation before it is rolled out to you. In this process we test the new feature in its entirety and validate that the other security mechanism are still intact. The following chapter covers these security mechanisms and features and tries to give you additional understanding of our thoughts behind them:

General Security

Unattended Upgrades

The software package unattended-upgrades is responsible for automatically keeping the upribox current with the latest security updates. We configured your upribox in a way that it searches for important updates every day and installs them without the need of any interaction with you. In more detail the package will perform the following tasks:

- update the package lists
- download upgradeable packages
- upgrade packages
- remove downloaded packages that are not available anymore in the sources (every three weeks)

Passwords

If you ordered a fully assembled upribox from our website you will find your passwords for the web interface, SSH and both WiFis in the included manual. These passwords were generated with a cryptographically secure pseudo-random number generator (CSPRNG) which means that the program that created the passwords used operating system specific randomness sources which leads to high unpredictability. Upriboxes that you assembled yourself with the provided community image are pre-configured with passwords that can be changed in the web interface. Only secure passwords with a minimum of 8 characters containing lower-case, upper-case, numbers and special characters are accepted by the system.

SSH/VPN Keys

When generating a new profile for OpenVPN we create a new pair of certificate (also known as public key) and private key for this profile. Furthermore, we ensure that every upribox has different SSH and VPN keys by re-generating them

automatically upon the first boot on every bought upribox but also on upriboxes that you built with the community image.

Least Privilege

The upribox architecture follows the principle of least privilege. This means that every part of the system (such as a process, script or user) is only able to access and modify those parts of the system that are necessary for the completion of its tasks. One example for this implementation is that only the central configuration script *upri-config.py* (see *django*) is able to modify files with root privileges.

Privacy

Logs

Your upribox saves log files in memory instead of the SD card. This helps to extend the life time of the card and protects your privacy since data on the RAM disk are deleted regularly.

User Statistics

To calculate the necessary data for the statistics the upribox aggregates and anonymizes information by calculating the sum of blocked contents and filtered domains over a specific time. This procedure ensures that nobody can make assumptions about another user's internet behaviour. Furthermore, the calculated information is only stored on the upribox itself and never leaves it.

Cryptography

Raspberry Pi Hardware RNG

A common problem with small devices like the upribox is that the hardware and the operating system do not provide enough entropy for a cryptographically secure generation of random numbers. This is a crucial requirement for a fast and secure calculation of keys. The Raspberry Pi has an included random number generator in its hardware. To feed the Raspberry Pi hardware RNG to the entropy pool at */dev/random* that is used in the upribox' Python scripts the package *rng-tools* is installed.

Certificate Pinning

To get the latest filter rules your upribox communicates with our backend server which provides a self-signed SSL certificate to ensure the authenticity of the received updates. Your upribox uses certificate pinning to associate the server with the expected certificate and is so able to detect an attempted attack immediately.

Strong Ciphers/Hashes for OpenVPN

For the VPN feature of the upribox we use OpenVPN which is *not* executed as root and uses the strong cryptographic hash function SHA384 for the packet HMAC authentication and AES-256 (CBC mode) for encryption.

Signed Releases

On Github you can download our latest community image and verify its integrity and authenticity with the provided signature file.

2.2.4 Customization

There are two possible ways to adapt the settings of your upribox by writing to *custom facts*: use the Web Interface or edit the *custom facts* manually. Configuration options that are important for all users are available in the Web Interface, special configuration options for tech-savvy users can be set manually using SSH.

Note: The upribox Software update mechanisms ensures that the system remains in a consistent state. Manual changes to configuration files are therefore overwritten by the periodic software update process of the upribox. **The only way the persist your changes is by writing to the correspondent custom fact.**

The custom configuration options of the upribox Software are stored in **/etc/ansible/fact.d/**. Example for these configuration facts can be found here: `local_facts.tar.gz`.

Advanced Network Settings

static network configuration

Connect to your upribox via SSH and create an **interfaces.fact** file in the */etc/ansible/facts.d* directory. The following interfaces configuration, will set the upribox to use a static IP configuration:

```
{
    "general": {
        "mode": "static"
    },
    "static": {
        "ip": "10.203.95.160",
        "netmask": "255.255.255.0",
        "gateway": "10.203.95.254",
        "dns": "10.203.50.233 10.203.95.250"
    }
}
```

Make sure to adapt the *ip*,*netmask*,*gateway*, and *dns* values to reflect your setup. Once you created the *interfaces.fact* file, run

```
sudo upri-config.py restart_network
sudo reboot
```

to configure the network device and to restart the upribox with the static IP setup. (see *CLI Tool*)

custom VPN server port

Connect to your upribox via SSH and use the following commands to set a custom *port* and *protocol* for the upribox OpenVPN server:

```
sudo upri-config.py set_vpn_connection 1194/UDP
sudo upri-config.py restart_vpn
sudo upri_conifg.py restart_firewall
```

Make sure to use a correct port - protocol combination: valid ports are between *1025* and *65535* (**unprivileged ports**), and protocol can be either **UDP** or **TCP**. If you want to access your upribox's VPN server over **443/TCP** (standard HTTPS port) you need to set a custom port-forwarding rule in your router: set your VPN server to an unprivileged TCP port e.g. 4300/TCP and then forward port 443/TCP to port 4300/TCP of your upribox. (see *CLI Tool*)

custom wifi channel

Connect to your upribox via SSH and use the following commands to set a custom *channel* for the upribox WiFi:

```
sudo upri-config.py set_wifi_channel 3
sudo upri-config.py restart_wlan
```

Valid WiFi channels are numbers between 1 and 10. (see *CLI Tool*)

de/activate WiFi

If you have ssh enabled you can connect to your upribox and deactivate both, Ninja and Silent WiFi:

```
sudo upri-config.py enable_silent no
sudo upri-config.py restart_silent
sudo upri-config.py enable_tor no
sudo upri-config.py restart_tor
```

To activate them again replace "no" with "yes". If you activate Ninja WiFi, you have to activate Silent WiFi as well. (see *CLI Tool*)

2.2.5 Architecture

A core technology used in the upribox software is Ansible[1]: a python-based configuration management software. Our rationale behind using Ansible is twofold:

Reproducibility Every setting, installed package etc. should be documented in code. We use Ansible's default push mode to configure the base image in order to deploy the latest upribox software and harden the base image. All changes we perform on a given base images can be reproduced (see *Development*).

Continuous delivery Ansible enables us to roll out bugfixes as well as new features continuously. Once the upribox software is deployed it automatically gets changes from our Github repository and deploys them using Ansible's pull mode.

Note: Config files are overwritten periodically (see *Customization*).

---

[1] https://www.ansible.com

Modules

Base setup

init

This role is responsible for basic configurations of the operating system such as

- expanding disk space
- configuring locale
- user management
  - creating sudo group
  - creating remote user
  - removing default user (*pi*)
- setting the hostname (*upri*)
- adding authorized key

common

The *common* role lays the groundwork for the following more specific roles. The main parts of this role are the following:

- building the infrastructure for logging
- creating the logging directory (`/var/tmp/log` for production and `/var/log/log` for development mode, see *Development vs. Production mode*)
- updating rsyslog config and deleting old rsyslog logfiles
- configuring logrotate
- settings and configurations
- writing default settings
- copying ansible config
- creating directory for local facts (see *Customization*)
- The upribox updates every 4 hours to the latest version on github via ansible. For this purpose the common role needs to execute among others the following tasks before updating
- installing ansible
- configuring a cron job
- copying the update script
- copying git deployment key
- update of the filter rules
- creating crontab entry to parse user-agents which are used to fingerprint the devices connected to the upribox

unattended_upgrades

Unattended Upgrades[2] provides the upribox automatically with the latest security updates.

Networking

arp

The upribox provides a zero-config service called *Apate* (see *Apate*) which allows you to benefit from the ad-blocking functionality on every device in your network not just when connected to the upribox WiFi. This works with a technique called *ARP spoofing*. In this role the Apate daemon files are copied and configured, requirements are installed to a virtual environment and eventually the daemon is (re)started.

iptables

In order to be able to configure ad-blocking (Silent Mode) and Tor (Ninja Mode) for each device separately the upribox adds and removes iptables rules dynamically. Two lists of MAC addresses - one for devices which don't need ad-blocking and one for devices with Tor enabled - are stored in local facts. If a user deactivates ad-blocking or activates Tor for a device in the user interface the MAC address will be added to the no_adblocking list and the tor list, respectively. The iptables rules are immediately copied to `/etc/iptables/` and to take effect the service is restarted.

vpn

The upribox uses OpenVPN[3] as a VPN service to protect your communication security on the road. This can be used to protect your sensitive information when using public WiFi hotspots. The role creates the necessary certificates and keys and installs and sets up the service.

Note: For security reasons OpenVPN is *not* executed as root and uses SHA384 for the packet HMAC authentication and AES-256 (CBC mode) for encryption.

---

wlan

This role is responsible for installing and configuring all WiFi and network related services such as hostapd or isc-dhcp-server. Interface wlan0 is used as the WiFi interface for the Silent Mode whereas a virtual interface wlan0_0 is added in case of an activated Ninja Mode.

Silent Mode

Silent Mode creates a wireless network with the default SSID *upribox*. If your device is connected to this network ads and trackers will automatically be blocked.

[2] https://wiki.debian.org/UnattendedUpgrades
[3] https://openvpn.net/

Ninja Mode

Ninja Mode results in a separate wireless network with the default SSID *upribox-ninja* and blocks ads and trackers as well but in addition the traffic will be routed through the Tor[5] network.

Privacy

dns

The upribox uses the *dnsmasq* daemon to filter DNS requests. This role set-ups *dnsmasq* on all interfaces and listens for requests. Filtered domains are loaded from */etc/dnsmasq.d*.

dns_ninja

The DNS ninja *dnsmasq* daemon filters domains and in additions resolves all requests via the Tor network. The daemons listens for requests on port *5300/UDP*. This setup also does not log any DNS requests it receives.

dns_unfiltered

The upribox needs another instance of the dnsmasq service which is responsible for handling DNS requests from devices on which ad-blocking is deactivated.

nginx

This role is used to install and set-up nginx for the upribox. The nginx web server is responsible for a number of tasks:

upribox blackhole

The blackhole setup returns an empty response for any request it receives, depending on the type of request this could be either an empty HTML page, or a blank image file. The server in addition attempts to reset/delete browser cookies for filtered domains: for every cookie the server receives, the server responds with the same cookie with empty values and a validity of 0. This setup ensures that tracking cookies are deleted from the user's browser the moment the request for a domain filtered by the upribox is made.

upribox CSS filter

The upribox serves custom CSS files to remove ad-content from websites. The custom domain to serve CSS files is: *filter.upri.box*. CSS filters are loaded from */etc/nginx/lua/css.lua* and periodically updated.

upribox web interface

The nginx role finally prepares the setup for the upribox web interface. The nginx configuration ensures that requests to http://upri.box are forwarded to the upribox Django web interface.

[5] https://www.torproject.org/

privoxy

This role deploys the Privoxy[4] filter proxy on the upribox. The upribox uses Privoxy to: filter unwanted content in HTTP requests such as advertisement or tracker code. In addition to content filtering, Privoxy injects a custom CSS file into websites to stop (filtered) ads from showing up in websites. The filter configuration for Privoxy is stored in */etc/privoxy* and updated periodically.

tor

This role setups the Tor network daemon for the upribox. The Tor daemon is configured for transparent proxying and offers its own DNS resolver to perform DNS queries through the Tor network.

User Interface

ssh

By default the upribox can be reached via SSH. This feature can be disabled in the admin page of the web interface or directly by calling the *enable_ssh* action of the configuration script *upri-config.py* (see *django*).

Note: In the web interface it is not possible to deactivate WiFi (Silent and Ninja), SSH and Apate, since this scenario would make the upribox unreachable. Therefore one of the services should always be up and running. (see *Modes*)

---

fingerprinting

The upribox provides a service called *registrar* which gathers MAC address, IP address and hostname of a device and saves the information into the database. A separate script uses the user-agents provided by squid and tries to extract a model name of the device. These names are later on suggested to the user in the web interface as a way to identify his or her device in a list of other devices on the network. Furthermore the chosen name acts as a label in the device overview.

squid

In order to gather user-agents of (and subsequently fingerprint) connected devices (see *fingerprinting*) the upribox uses squid[7]. The squid log file is later parsed and the information saved into the database.

django

The upribox user interface (see web_interface) is based on the Python Web framework Django[6]. The role is responsible for installing the requirements to a virtual environment, copying the web interface files, setting up the database and installing services like a supervisor (for the rqworker) and the application container uWSGI. By deploying this role the upribox also starts a cleanup process for the saved statistic files removing data older than 6 months.

[4] https://www.privoxy.org/
[7] http://www.squid-cache.org/
[6] https://www.djangoproject.com

Note: For privacy reasons the upribox does not keep the ad-blocking logfile with timestamps and URLs but tries to aggregate the information as soon as possible to store only the information that is needed for the statistics and to assure anonymity. (see *Logs*)

CLI Tool

All changes to the upribox configuration are perfomed via `upri-config.py`. This nifty command line tool can be used via SSH and also provides a secure way to perform a limited set of privileged command via the Django webinterface.

```
usage: upri-config.py [-h]
                      {set_ssid,set_password,set_tor_ssid,set_tor_password,restart_
↪wlan,enable_tor,enable_silent,restart_tor,restart_silent,enable_vpn,set_vpn_
↪connection,set_wlan_channel,restart_vpn,enable_ssh,restart_ssh,enable_apate,enable_
↪static_ip,restart_apate,parse_logs,parse_user_agents,generate_profile,delete_
↪profile,restart_firewall,enable_device,disable_device,set_ip,configure_devices,set_
↪dns_server,set_netmask,set_gateway,restart_network,set_dhcpd,restart_dhcpd,torify_
↪device,exclude_device,untorify_device,include_device}
                      ...
```

Actions

Actions cover tasks that are able to modify the configuration of the upribox

action Possible choices: set_ssid, set_password, set_tor_ssid, set_tor_password, restart_wlan, enable_tor, enable_silent, restart_tor, restart_silent, enable_vpn, set_vpn_connection, set_wlan_channel, restart_vpn, enable_ssh, restart_ssh, enable_apate, enable_static_ip, restart_apate, parse_logs, parse_user_agents, generate_profile, delete_profile, restart_firewall, enable_device, disable_device, set_ip, configure_devices, set_dns_server, set_netmask, set_gateway, restart_network, set_dhcpd, restart_dhcpd, torify_device, exclude_device, untorify_device, include_device

This script accepts the name of an action that shall be executed

Sub-commands:

set_ssid

Sets a new SSID for the Silent WiFi by writing to the fact *wlan*

```
upri-config.py set_ssid [-h] ssid
```

Positional Arguments

ssid The SSID for the Silent WiFi

set_password

Sets a new password for the Silent WiFi by writing to the fact *wlan*

```
upri-config.py set_password [-h] password
```

Positional Arguments

password The SSID for the Silent WiFi

set_tor_ssid

Sets a new SSID for the Ninja WiFi by writing to the fact *wlan*

```
upri-config.py set_tor_ssid [-h] ssid
```

Positional Arguments

ssid The SSID for the Silent WiFi

set_tor_password

Sets a new password for the Ninja WiFi by writing to the fact *wlan*

```
upri-config.py set_tor_password [-h] password
```

Positional Arguments

password The SSID for the Silent WiFi

restart_wlan

Triggers the Ansible tasks with the tag *ssid*

```
upri-config.py restart_wlan [-h]
```

enable_tor

Enables/disables the Ninja WiFi by writing to the fact *tor*

```
upri-config.py enable_tor [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not Ninja WiFi is enabled (“yes” or “no”)

enable_silent

Enables/disables the Silent WiFi by writing to the fact *wlan*

```
upri-config.py enable_silent [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not Silent WiFi is enabled (“yes” or “no”)

restart_tor

Triggers the Ansible tasks with the tag *toggle_tor*

```
upri-config.py restart_tor [-h]
```

restart_silent

Triggers the Ansible tasks with the tag *toggle_silent*

```
upri-config.py restart_silent [-h]
```

enable_vpn

Enables/disables the VPN by writing to the fact *vpn*

```
upri-config.py enable_vpn [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not VPN is enabled (“yes” or “no”)

set_vpn_connection

Sets a custom port and protocol for the upribox OpenVPN server by writing to the fact *vpn*

```
upri-config.py set_vpn_connection [-h] port_protocol
```

Positional Arguments

port_protocol The port and protocol used for the OpenVPN server (usage: "1194/udp")

set_wlan_channel

Sets a new WiFi channel for the Silent WiFi by writing to the fact *wlan*

```
upri-config.py set_wlan_channel [-h] channel
```

Positional Arguments

channel The channel for the Silent WiFi

restart_vpn

Triggers the Ansible tasks with the tag *toggle_vpn*

```
upri-config.py restart_vpn [-h]
```

enable_ssh

Enables/disables the ssh by writing to the fact *ssh*

```
upri-config.py enable_ssh [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not SSH is enabled ("yes" or "no")

restart_ssh

Triggers the Ansible tasks with the tag *toggle_ssh*

```
upri-config.py restart_ssh [-h]
```

enable_apate

Enables/disables the Apate (see arp) by writing to the fact *apate*

```
upri-config.py enable_apate [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not Apate is enabled (“yes” or “no”)

enable_static_ip

Sets the upribox to DHCP or static IP mode by writing to the fact *interfaces*

```
upri-config.py enable_static_ip [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not a static IP is enabled (“yes” or “no”)

restart_apate

Triggers the Ansible tasks with the tag *toggle_apate*

```
upri-config.py restart_apate [-h]
```

parse_logs

Parses the log files of the services and aggregates the statistics data

```
upri-config.py parse_logs [-h]
```

parse_user_agents

Parses the log file of the service squid containing MAC addresses, IP addresses and user-agents and saves the gathered information into the database

```
upri-config.py parse_user_agents [-h]
```

generate_profile

Generates openvpn client certificates and saves the generated openvpn client configuration into the database

```
upri-config.py generate_profile [-h] profile_id
```

Positional Arguments

profile_id The profile ID of a profile that was created in the web interface

delete_profile

Revokes previously generated openvpn client certificates

```
upri-config.py delete_profile [-h] profile_id
```

Positional Arguments

profile_id The profile ID of a profile that was created in the web interface

restart_firewall

Triggers the Ansible tasks with the tag *iptables*

```
upri-config.py restart_firewall [-h]
```

enable_device

Enables ARP spoofing via Apate (see arp) for a specific device

```
upri-config.py enable_device [-h] ip
```

Positional Arguments

ip The IP address of the device that shall be enabled

disable_device

Disables ARP spoofing via Apate (see arp) for a specific device

```
upri-config.py disable_device [-h] ip
```

Positional Arguments

ip The IP address of the device that shall be disabled

set_ip

Sets a static IP by writing to the fact *interfaces*

```
upri-config.py set_ip [-h] ip
```

Positional Arguments

ip The static IP address for the upribox

configure_devices

Triggers the Ansible tasks with the tag *configure_devices*

```
upri-config.py configure_devices [-h]
```

set_dns_server

Sets the DNS server by writing to the fact *interfaces*

```
upri-config.py set_dns_server [-h] dns
```

Positional Arguments

dns The DNS server for the upribox

set_netmask

Sets subnetmask by writing to the fact *interfaces*

```
upri-config.py set_netmask [-h] netmask
```

Positional Arguments

netmask The subnetmask for the upribox

set_gateway

Sets the gateway by writing to the fact *interfaces*

```
upri-config.py set_gateway [-h] gateway
```

Positional Arguments

gateway The gateway for the upribox

restart_network

Triggers the Ansible tasks with the tag *network_config*

```
upri-config.py restart_network [-h]
```

set_dhcpd

Enables/disables the DHCP server by writing to the fact *dhcpd*

```
upri-config.py set_dhcpd [-h] boolean
```

Positional Arguments

boolean Possible choices: yes, no

Whether or not the upribox acts as a DHCP server (“yes” or “no”)

restart_dhcpd

Triggers the Ansible tasks with the tag *dhcp_server*

```
upri-config.py restart_dhcpd [-h]
```

torify_device

Adds iptables rule to torify a specific device

```
upri-config.py torify_device [-h] mac
```

Positional Arguments

mac The MAC address of the device whose traffic shall be routed over the tor network

exclude_device

Adds iptables rule to disable ad-blocking for a specific device

```
upri-config.py exclude_device [-h] mac
```

Positional Arguments

mac The MAC address of the device whose traffic shall not be ad-blocked

untorify_device

Removes iptables rule to untorify a specific device

```
upri-config.py untorify_device [-h] mac
```

Positional Arguments

mac The MAC address of the device whose traffic shall not be routed over the tor network

include_device

Removes iptables rule to enable ad-blocking for a specific device

```
upri-config.py include_device [-h] mac
```

Positional Arguments

mac The MAC address of the device whose traffic shall be ad-blocked

2.2.6 Development

The current upribox image is based on Raspbian[1] Jessie Lite and customized with Ansible (see *Architecture*). The Raspbian image can be staged into *production* or *development* mode.

Development environment

The following guide assumes that you have a Raspberry Pi with the upribox image set-up. If you still need help with that task please read the intro guide. The following guide explains the steps necessary to setup a development environment for the upribox software.

Prerequisites [on your development machine]

- install *ansible* 1.9.6 (`sudo pip install ansible==1.9.6`) and *git*
- install requirements for the *ansible* modules (`sudo apt-get install python-pip python-dev libffi-dev libssl-dev libxml2-dev libxslt1-dev libjpeg8-dev zlib1g-dev`)
- make sure to log into your Raspberry via SSH once because ansible uses ~/.ssh/known_hosts for verification (or disable host verification)
- add your SSH public key to your Raspberry, e.g. with `ssh-copy-id`

If you successfully completed the prerequisites you should be able to login into your upribox via SSH without the need of a password. In addition you should have *ansible* installed on your computer. Next, clone the upribox software to your computer:

```
git clone https://github.com/usableprivacy/upribox.git
```

[1] https://www.raspberrypi.org/downloads/raspbian/

Development vs. Production mode

The development mode is intended for testing new features and debugging the upribox software. As such log files are persistent and auto software updates are disabled. The upribox images available for download are all set to production mode by default. In production mode log files are deleted on every reboot and the upribox configuration is automatically downloaded and updated via github.

Note: The production mode is also intended for the production of purchasable pre-assembled boxes. In this process we also create a new user and generate a cryptographically secure password. This happens out of scope of the production ansible playbook and therefore you have to create a user on your own when deploying in production mode from scratch.

Development Mode

- copy *environments/development/inventory.sample* to *environments/development/inventory*
- add your RaspberryPi address(es) in the [upriboxes] section in environments/development/inventory

Once you added the IP address of your Raspberry Pi to the development inventory, start changing the upribox source and deploy your local config with:

```
ansible-playbook -i environments/development/inventory site.yml
```

Production Mode

- copy *environments/production/inventory.sample* to *environments/production/inventory*
- add your Raspberry IP address(es) in the [upriboxes] section in *environments/production/inventory*
- from now on, the config can be deployed with `ansible-playbook -i environments/production/inventory site.yml`

Creating an image from scratch

If you want to create the entire upribox image from scratch you can use *setup.yml* ansible playbook. Download the latest Raspian Lite image, make you sure you have installed all the prerequisites (see *Prerequisites [on your development machine]*) and in addition install *sshpass*.

Set-up the initial upribox base image

- copy *environments/development/inventory.sample* to *environments/init/inventory*
- add your RaspberryPi address(es) in the [upriboxes] section in *environments/init/inventory*
- make sure you have a public/private key pair for ssh on your development machine. *~/.ssh/id_rsa.pub* will be automatically added to the authorized_hosts on the Raspberry
- run the initial setup with `ansible-playbook -i environments/init/inventory setup.yml` This command will log into your Raspberry with the default credentials *pi/raspberry*, create a new user (*upri*) and delete *pi*. Add `--ask-pass` if you change the default password.
- from now on, you can deploy the upribox software in production or development mode (see *Development vs. Production mode*).