

# 14.12.2021 // Install FabAccess

## Install FabAccess

### Minimal Hardware Requirements

#### Server

tl;dr No Hardware Requirements known so far. No all Platforms are supported with Docker Images.

- a CPU *would be nice* but we can technically run on a mainframe too.
  - At least 2MHz though. We need a speedy boi
- Some memory. The stick you found in your attic is fine
  - 30MB should be enough. But if you want to *really* sure, make that 40MB
- A network card. Any one is fine if it speaks Ethernet even better but your old bus card is dandy too

or if building a custom PC just so you can run FabAccess is too much effort; **a Raspberry Pi is fine, although RasPi 2 and up are recommended.** Any one really (except: Raspberry Pi 1 or Raspberry Pi Zero).

#### Client

Android SDK Version: 21 iOS Version: 13 Windows 10: 1709 - Build:16299

### Internal Release

Internal Release is only for our Space to develop FabAccess and distribute the Server and the Client easier to our users. We keep everything Open Source, but we not want to fear your users with to many bugs.

### Beta Release

To not overcomplicate our CI/CD and development process this actual "Beta" is more an "Alpha" for interested testers.

# Install the Server

## Option 1. `rustup && cargo install`

### Steps:

1. Install rustup.rs. Distribution packages for rustc are generally way too old
2. `$ rustup install stable`
3. Get yourself a directory to clone BFFH into
  - If you put this dir on a SSD you can speed up build times by 5-10 times.
4. `git clone --recursive --branch stable`
  - ... stable ... TODO...
  - You can also check out the `development` branch but keep in mind that until Beta it has no stability guarantee. It may work. It may make you a sandwich. But it may also set your hat on fire and fill your shoes with orange juice. *You have been warned.*
5. `cargo install --path .`
  - if you add `--debug` you get a debug build. It gives you much more logging output but it's slower to run and is almost spammy
6. Make yourself a coffee. Or tea. Or open \$beverage of your choice. You earned it! (And you'll be looking at "**Compiling**" for a while.)
  - If you get `error: failed to run custom build command for 'gsasl-sys v0.2.3'` or something like that with the stderr output reading "[...]Unable to find libclang[...]":
    - `export LIBCLANG_PATH=/usr/lib` Or wherever `libclang.so` is installed on your computer. It's usually `/usr/lib/libclang.so` or `/usr/lib/llvm/12/lib/libclang.so` or similar. If you can't find it, consult your package manager
  - If you get any other error open an issue

## Option 1.1 Install on Ubuntu for "Dummies"

- see further below, as this is a bit of a lengthy process...

## Option 2. Docker

Docker Image can not run on armv6 (Raspberry Pi 1 or Raspberry Pi Zero)

### Steps:

1. Install Docker On Raspberry Pi: <https://phoenixnap.com/kb/docker-on-raspberry-pi>
2. Install Docker-Compose On Raspberry Pi: <https://dev.to/elalemany0/how-to-install-docker-and-docker-compose-on-raspberry-pi-1mo>
3. Get Docker-Compose Files `git clone` <https://gitlab.com/fabinfra/fabaccess/dockercompose.git> The Dockerfile is in the root directory of the main repo docker-compose.yml is available in a separate [git repo](#)
4. Edit config files in `config` folder to taste
5. Start Server with `docker-compose up -d`

To make it easier to apply your changes in your config and keep the dockercompose up to date, you should "fork" this repository.

## Get Server Logs

```
docker-compose logs
```

## Install the Client

Currently only Windows(UWP), Android and iOS are directly supported.

### Option 1. Platform App Store

**Android:** [https://play.google.com/store/apps/details?id=org.fab\\_infra.fabaccess](https://play.google.com/store/apps/details?id=org.fab_infra.fabaccess) **iOS:** <https://testflight.apple.com/join/KfeUaHT4> **Windows:** <https://www.microsoft.com/de-de/p/fabaccess/9p69mwzjf2mv>

### Option 2. Build locally

Follow build instructions on: <https://gitlab.com/fabinfra/fabaccess/borepin>

... and now for something completely different...

## Option 1.1 Install on Ubuntu for "Dummies"

This description is how to compile and set up Diflouroborane on Ubuntu 20.04.3 LTS clean install. Other releases or distros might work as well. The process is quite lengthy but at the end you will have a FabAccess running to your needs. ... as I said: for complete dummies, if someone finds a better solution, please post suggestions on: <https://fabaccess.zulipchat.com/#narrow/stream/255963-General/topic/Demo>

## Steps

1. Get your system up-to-date `sudo apt-get update && sudo apt-get upgrade`
2. Get rustup `sudo apt install curl` `curl --proto 'https' --tlsv1.2 -sSf https://sh.rustup.rs | sh`  
**restart the console**
3. Get capnproto, gsas and git `sudo apt-get install gsasl` `sudo apt-get install capnproto`  
`sudo apt install git`
4. Create a target directory for BFFH there might be better places compared to where I created it, but it works... `mkdir BFFH` `cd BFFH`
5. Clone the Diflouroborane repository `git clone https://gitlab.com/fabinfra/fabaccess/bffh`
6. For compiling some dependencies were missing on Ubuntu `git submodule update --init` `sudo apt install libgsasl7-dev` `sudo apt install cmake` `sudo apt install libclang-dev` `sudo apt install libssl-dev` `sudo apt install cargo`
7. Open the subdirectory and start compiling `cd bffh` `cargo build --release` **if the compiler prompts something like "error: linker 'cc' not found":** `sudo apt install build-essential` `cargo build --release`
8. Copy the configuration files (best done with the GUI filemanager of Ubuntu) copy files from "bffh/examples" paste them into "bffh/target/release/examples"
9. Install mosquitto MQTT broker Diflouroborane uses mosquitto MQTT broker to communicate with the Shellies. `sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa`  
`sudo apt-get update` `sudo apt-get install mosquitto` `sudo apt-get install mosquitto-clients`  
`sudo apt clean` *find out which linux release is installed (for Ubuntu -> google)*  
`uname -a` `sudo wget http://repo.mosquitto.org/debian/mosquitto-bullseye.list` *change "bullseye" according to your distro: bullseye, buster, stretch, jessie, ...*
10. Configuring mosquitto broker for some reason, starting with version 2.x mosquitto does not allow external communication via the broker per default. This needs to be changed via a config file:
11. Stop mosquitto `sudo service mosquitto stop`
12. Change into the "etc/mosquitto/" directory (lots of `cd ..` then `cd etc`, `cd mosquitto`)
13. Create a configuration file: `sudo touch file test.conf`
14. Edit the configuration fil: `sudo nano -w test.conf` add: `listener 1883` `allow_anonymous true` Save (Strg-O) and close (Strg-X)
15. Start mosquitto `mosquitto -v -c /etc/mosquitto/test.conf`
16. Find the IP adress of your computer - **new console** `ip a`
17. Configure your Shelly as long as your Shelly has not been given the credentials for a WLAN, it will create an access point (AP) for configuration. This AP will appear in your list of WLAN. Connect to this Shelly-AP and connect to `192.168.33.1` in your browser. A configuration page should appear. If your Shelly is already connected to your WLAN, you must find the assigned IP-Adress (e.g. by looking into your router). Enter this IP Adress in your browser and you will get the configuration page.
18. MQTT Client setup goto "Internet & Security" -> "Advanced - Developer Settings" enable "MQTT" enter the IP-Adress from Step 16 in the field "IP-Adress" As we did

not define MQTT credentials in mosquitto yet, no credentials need to be filled in. To find the "ID" of your Shelly activate "Use custom MQTT prefix" (but do not change it!). This should be something like: `shelly1-123456789ABC` for a Shelly 1 `shelly1pm-123456` for a Shelly 1PM note this ID for later - **save - re-check the settings!**

19. WLAN Client setup goto "Internet & Security" -> "WIFI MODE - CLIENT" Set WLAN Credentials
20. Configure Diflouroborane Open the file "bfff.dhall" in the GUI Editor (just by double-clicking it) Change `Shelly_123` to your Shelly name, e.g. `shelly1-123456789ABC` (**case sensitive!, dash sensitive!**) in "Link up machines to actors" and in "actors". Change the third IP-adress under "listens" to the IP-adress of your computer. - **save**
21. start Diflouroburane change to the directory in the console where you checked for the ip-address `cd BFFH/bfff/target/release` load settings to Diflouroborane:  
`./diflouroborane -c examples/bfff.dhall --load examples` start Diflouroborane:  
`./diflouroborane -c examples/bfff.dhall`

**Important** every time you change the bfff.dhal you need to reload the settings (otherwise the App will not connect to the server on restart): `./diflouroborane -c examples/bfff.dhall --load examples` and restart start Diflouroborane: `./diflouroborane -c examples/bfff.dhall`

Download the borepin APP as described previously

- start the App
- press: "Connect to new Server"
- press: Enter the IP of your computer in the "Host"-Field
- Enter your Username and Password.

To connect to the demo instance

- start the App
- press: "Connect to new Server"
- press: "Demo Host Address"
- User: "Testuser"
- Passw: "secret"

Have fun and give feedback!

---

Version #1

Erstellt: 15 Oktober 2024 10:20:26 von Mario Voigt (Stadtfabrikanten e.V.)

Zuletzt aktualisiert: 25 Februar 2025 21:22:13 von Mario Voigt (Stadtfabrikanten e.V.)