FabAccess Setup - Schritt für Schritt

Dieses Dokument enthält eine Schritt-für-Schritt-Anleitung, wie Sie FabAcess zum Laufen bringen. Am Ende dieser Beschreibung werden Sie Folgendes haben:

- 1 oder mehrere Shellys in Ihrem System registriert
- 1 oder mehrere Benutzer, die inm System registriert sind
- QR-Codes für den Zugang zu einer Maschine generiert
- 1 Shelly konfiguriert als Türöffner
- 1 Shelly konfiguriert, um zu erkennen, ob eine Maschine nur eingeschaltet ist oder wirklich läuft (TO-DO)

Schritt 1: Installieren des BFFH-Servers

Es gibt mehrere Möglichkeiten, den BFFH-Server zu installieren. Dies kann entweder über

- Installation mit Docker, oder
- Installation aus dem Quellcode, oder
- Installation unter Ubuntu für Dummies

Schritt 2: Installieren der FabAccess App (Borepin)

Siehe Downloads / Demo

Schritt 3 App mit Server verbinden

Zuerst müssen Sie die IP-Adresse des Servers herausfinden, falls diese noch nicht bereits bekannt ist. Dies kann durch Eingabe von `ip a` auf der Konsole des Systems, auf dem der BFFH-Server läuft, erfolgen. Verwenden Sie die unter BROADCAST angegebene Adresse.

Start the server. If you are using the docker, this is done by using `docker-compose up -d`.
If you compiled the server on your system this is done by entering `./diflouroborane -c examples/bffh.dhall --load examples`
and then
`./diflouroborane -c examples/bffh.dhall`.

You will see some debug information, with probably some warnings.

Open the App. You will be asked to connect to a Host. Tap “DEMO HOST ADRESS” and change the IP to the IP of your Server, do not change the port number (everything after the IP. This should look like `192.168.1.15:59661`). Tap “SELECT HOST”.

You will be asked to sign in. For Version 0.2 only the Option “LOGIN WITH PASSWORD” ist available. Use `Testuser` and the passwort `secret` to log in.

You will find an overview of the installed machines including the option “SCAN QR-CODE”. Next step is setting up you machines so they can be switched on an off.

Schritt 4: Shellys vorbereiten

Solange Ihr Shelly noch keine Zugangsdaten für ein WLAN erhalten hat, erstellt er einen eigenen Access Point (AP) zur Konfiguration, wenn er an die Versorgungsspannung angeschlossen wird. Dieser AP wird in Ihrer WLAN-Liste erscheinen. Verbinden Sie sich mit diesem Shelly-AP und stellen Sie in Ihrem Browser eine Verbindung zu `192.168.33.1` her. Es sollte eine Konfigurationsseite erscheinen. Wenn Ihr Shelly bereits mit Ihrem WLAN verbunden ist, müssen Sie die zugewiesene IP-Adresse herausfinden (z.B. durch einen Blick in Ihren Router). Geben Sie diese IP-Adresse in Ihrem Browser ein und Sie erhalten die Konfigurationsseite.

Shelly MQTT-Client einrichten

goto “Internet & Security” -> “Advanced - Developer Settings” enable “MQTT” enter the IP-Adress from your Server in the field “IP-Adress” As we did not define MQTT credentials in mosquitto yet, no creadentials 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 somthing like: `shelly1-123456789ABC` for a Shelly 1 `shelly1pm-123456` for a Shelly 1PM note this ID for later **- save - re-check the settings!**

Shelly WLAN-Client einrichten

Gehen Sie zu „Internet & Sicherheit“ -> „WIFI-MODUS - CLIENT“ WLAN-Zugangsdaten festlegen

Adding a Shelly to your server To understand the underlaying concept of actors and machines, please see the “configuration part” of the documentation. Four our example we will assume we have one actor (shelly) per machine.

Tip Prior to modifying the configuration files the proper working of the MQTT broker should be tested. To test the broker it is the best to use a second (linux) computer with a different IP adress. To test if the broker allows access from an external IP address open a MQTT subscriber on the second computer by typing
`mosquitto_sub -h 192.168.1.15 -t /test/topic` (change the IP adress to the adress of your server).
Use
`mosquitto_pub -h localhost -t /test/topic -m "Hallo from BFFH-Server!"`
to send a message to the other computer. If the message appears, everything is ok. When not, this should be first solved, as a connection to the shellies will not be possible this way.
If you are interested in communication between the shellies and the BFFH-Server you can use
`mosquitto_sub -h 192.168.1.15 -t shellies/#`
(change the IP adress to your needs). You will see some values popping op from time to time.

Configure Diflouroborane Open the file “bffh.dhall” in the GUI Editor (just by double-clicking it) or use `nano bffh.dhall` in your console.

Den Server mit dem MQTT-Broker verbinden
find the line which starts with `, listens`. You will find three lines stating addresses. The third address needs to be changed to the adress of your MQTT broker (most likely the IP adress of your BFFH server)

First you have to make your “actors” (in our case the Shellies) know to the system.
Go to the line where it starts with `, actors =` and after the `{` you can enter your Shelly with
`shelly1-123456789ABC = { module = "Shelly", params = {=}}`
The ID of the Shelly should match the ID of your Shelly. Here you can enter as many actors as you want, each separated by a `,`.

Now you have to link a “machine” to an “actor”.
Go to the line starting with `{actors_connections =` and after the following `[` you add
`{ machine = "Identifier-of-your-Machine", actor = "shelly1-E8DB84A1CFF4" }`
using your own Name-of-your-Machine and the Shelly-ID of the related actor.

Now you have to set the “access-permissions” to your “machine”.
Go to the line starting with `, machines =`. and after the `{` you can add a machine:

```
Identifier-of-your-Machine =
  { description = Some "I am your first Testmachine"
  , disclose = "lab.test.read"
  , manage = "lab.test.admin"
  , name = "Name of the Machine"
```

```
    , read = "lab.test.read"
    , write = "lab.test.write"
    },
```

Bitte beachten Sie, dass „Identifier-of-your-Machine“ die interne ID für BFFH ist. Der Name der Maschine, der in der App angezeigt wird, ist „Name der Maschine“.
Die angegebenen Berechtigungen sind für den Anfang in Ordnung (wenn Sie die Rollen des Testbenutzers nicht geändert haben). Um mehr über das Berechtigungskonzept zu erfahren, lesen Sie den Teil „Konfiguration“ der Dokumentation.

- save (if you are using nano, this will be Ctrl-O)

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

Open the App, an you should see the newly created machine in the list. By tapping “USE” you will activate the machine (Shelly will click, the MQTT-listener should promp an “on”), by tapping “GIVEBACK” you will deactivat the machine.

Creating a QR-Code for your machine A QR code allows users to directly enter the UI of the machine, where the machine can be used or given back. The QR code should contain the following content:
`urn:fabaccess:resource:{MachineID}`
e.g.
`urn:fabaccess:resource:Identifyer-of-your-Machine`

QR-Codes können auf verschiedenen Seiten im Internet (z.B. https://www.qrcode-generator.de) generiert werden, der „Typ“ des QR-Codes sollte „Text“ sein. Der generierte Code kann direkt mit der FabAccess App in der Maschinenübersicht gescannt werden.

Adding a user Adding a user to the system consists of two steps

- creating the user
- provide permissions

Users are defined in the file users.toml. To add a user simply add
`[Name-of-the-User]`
`roles = ["Name-of-a-role/internal", "Name-of-another-role/internal"]`
`priority = 0`
`passwd = "the-chosen-password"`

`noot = "whatever-this-means"`
Adding users or changing existing users does NOT require to restart the system (tested?)

The permissions of the user are defined by the linked roles. The roles are defined in the file bffh.dhall. Open the file bffh.dhall an find the line starting with `, roles =`
The concept of the role management is described in the “configuration” part of the documentation. To keep it simple we create a role called “ChainsawUser” `ChainsawUser = { permissions = [ "lab.machines.chainsaw.write"` - allows the user to use the machine
`, "lab.machines.chainsaw.read"`- allows the user to read see the status of the machine
`, "lab.machines.chainsaw.disclose"` - allows the user to see the machine in the machine overview
`]`

If a user assinged to this role uses the chainsaw, no other user is able to use it until this user gives the chainsaw back. To unlock the machine from the user, admin permissions are needed. So there could be an admin role like `ChainsawAdmin = { parents = ["ChainsawUser"]`
- inherits all the permissions of the ChainsawUser `, permissions = ["lab.machines.chainsaw.admin"]`
- addinional admin permissions `}`

The `machine` should be defined as: `Identifier-of-your-Chainsaw =`

```
  { description = Some "Beware of Freddy!"
  , disclose = "lab.machine.chainsaw.disclose"
  , manage = "lab.machine.chainsaw.admin"
  , name = "Chainsaw"
  , read = "lab.machine.chainsaw.read"
  , write = "lab.machine.chainsaw.write"
  },
```

Wenn ein Benutzer „ChainsawUser/internal“ zugewiesen ist, kann er/sie die Kettensäge in FabAccess sehen und benutzen.

Using a Shelly as a door opener (electronic wise) In version 0.2 a door opener functionality is not implemented. The specific behaviour of a door opener is, to activate a door openeing relais only for a few seconds. This behaviour is not yet implemented in FabAccess, but there is decent way to implement it by other means. The simple Shellies (1, 1pm, 2.5) have an internal timer “AUTO-OFF” which can be set. To use this timer you have to access the settings of the Shelly via a browser on your computer. To do so, you have to know the IP adress your Shelly is assinged to. This can normally found out in the router of your Wifi. By entering this IP adress in your browser you will access the main

menu of your Shelly.

Go to “Timer” and set the “AUTO-OFF” to e.g. 3 seconds.
Define a machine called “door” in the bffh.dhall

- define the actor:
  `shelly1-123456789ABC = { module = "Shelly", params = {=}}`
- define the machine:
  `{ machine = "door", actor = "shelly1-123456789ABC" }`
- set permissions for the machine:
  `door =`
  `{ description = Some "close it firmly"`
  `, disclose = "lab.door.disclose"`
  `, manage = "lab.door.admin"`
  `, name = "Door to the Lab"`
  `, read = "lab.door.read"`
  `, write = "lab.door.write"`
  `},`
- create a role having ALL permissions to the door
  `DoorUser =`
  `{ permissions =`
  `[ "lab.door.write"` - allows the user to use the door
  `, "lab.door.read"`- allows the user to read see the status of the door
  `, "lab.door.disclose"` - allows the user to see the machine in the machine overview
  `, "lab.door.admin"`
  `]`
- assign the role DoorUser/internal to all users

Es ist wichtig, dass alle Benutzer über Admin- bzw. Verwaltungsrechte verfügen, da die Aufforderung zum Öffnen der Tür durch einen Benutzer dazu führt, dass die Tür von diesem Benutzer „in Gebrauch“ ist. Die Tür kann nur wieder aktiviert werden, wenn der vorherige Benutzer die Tür „ent-benutzt“ oder wenn ein anderer Benutzer die Tür „zwangsbefreien“ kann, bevor er sie selbst benutzt.

Note in this special case, where all users will need admin capabilities the role could also contain only the permission `lab.door.use` and all permissions (disclos, manage, read, write) assigned to the machine would simply match `lab.door.use` (e.g. disclose = “lab.door.use”`).

**Erkennen, ob eine Maschine nur eingeschaltet ist oder wirklich läuft (TO-DO)

Version #4
Erstellt: 23 Oktober 2024 22:51:00 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 6 November 2024 03:07:15 von Mario Voigt (Stadtfabrikanten e.V.)