Home Automation in Python

What is Home Automation

What is Home Automation

”Something I can control from my phone”

Something more?

What is Home Automation

”Something I can control from my phone”

Something more?

Connecting devices together to generate behavior


Problems
Complex
Expensive


Problems
Complex
Expensive
"Where do I setup everything?"
"Do I have to do everything on the phone?"
"There is a different app for each service, it's a nightmare to setup and maintain"
"I need to buy everything from the same brand ..."
"... and that brand is expensive!"

Solution

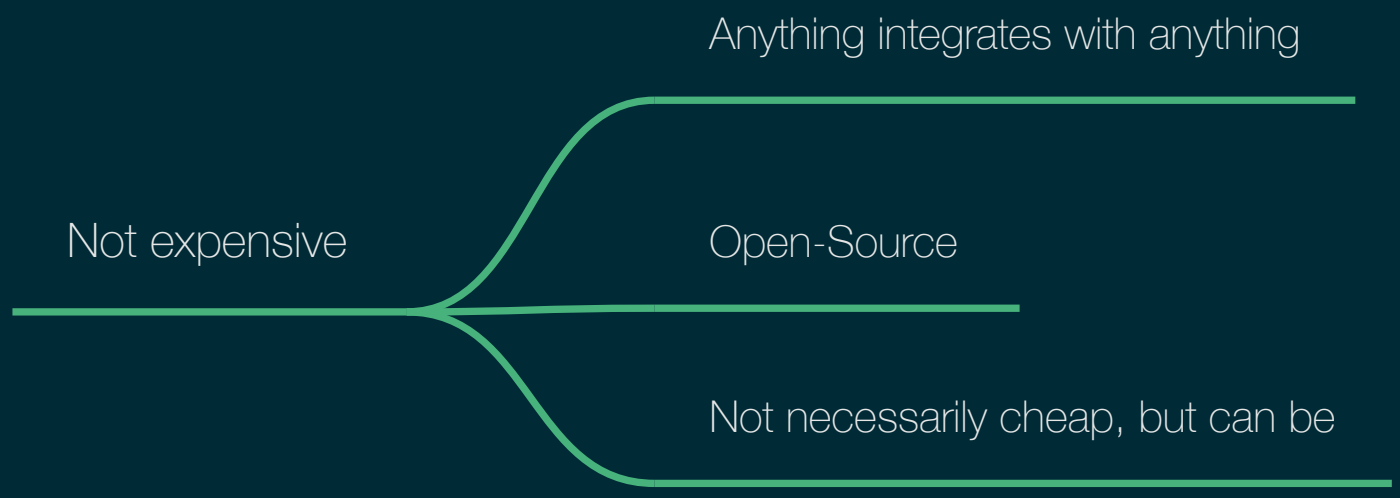
HomeAssistant & Appdaemon

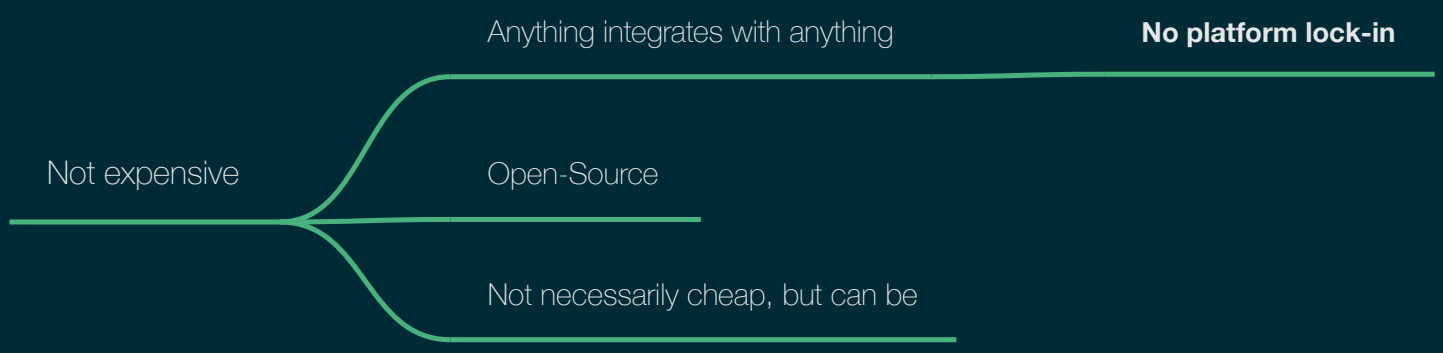
HomeAssistant & Appdaemon

- Simple
- Not expensive
- ... and

Simple

- One place to manage them all
- Version Controlled configuration
- TDD your automations 😍


Not expensive
Anything integrates with anything
Open-Source
Not necessarily cheap, but can be


Not expensive
Anything integrates with anything
No platform lock-in
Open-Source
Not necessarily cheap, but can be

- Not expensive
  - Anything integrates with anything
    - **No platform lock-in**
  - Open-Source
  - Not necessarily cheap, but can be
    - The setup for today costs 7€
    - (or 2607€ if we count my macbook and the phone I used as hotspot 😁)

... and

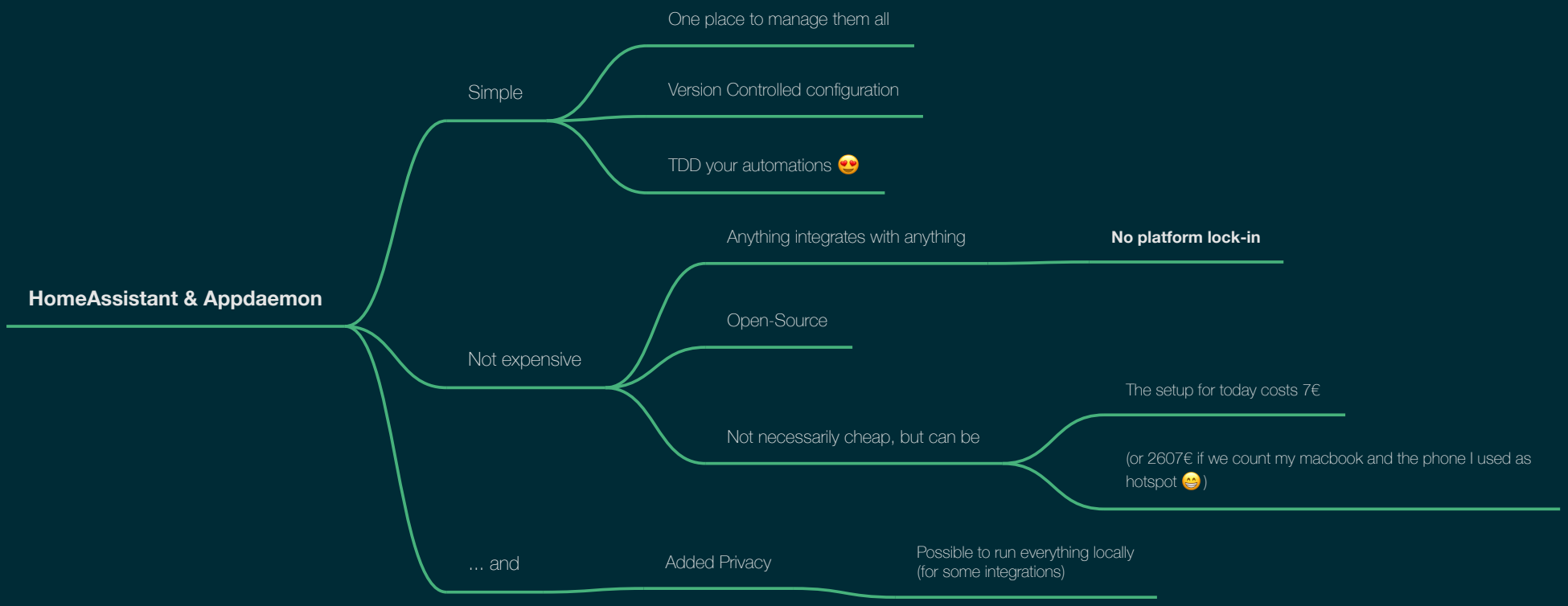
... and

Added Privacy

... and

Added Privacy

Possible to run everything locally
(for some integrations)


HomeAssistant & Appdaemon
Simple
One place to manage them all
Version Controlled configuration
TDD your automations 😍
Not expensive
Anything integrates with anything
No platform lock-in
Open-Source
Not necessarily cheap, but can be
The setup for today costs 7€
(or 2607€ if we count my macbook and the phone I used as hotspot 😄)
... and
Added Privacy
Possible to run everything locally (for some integrations)

Home Assistant & Appdaemon

- Architecture
- Everything is translated by **HomeAssistant**
- Everything is orchestrated by **Appdaemon**

Home Assistant & Appdaemon

- Architecture
  - Drawing on Board
- Everything is translated by **HomeAssistant**
- Everything is orchestrated by **Appdaemon**

Tower of Babel

- Many brands with many different protocols to implement essentially the same thing
- They’re obviously not compatible
- For instance

For instance

- A Philips Hue Bulb
- An Ikea LED Strip

Essentially the same thing: **A light**

On / off

Brightness

(Maybe) color temperature

Yet, 2 completely different **Apps** or **API** to control them

Enters **HomeAssistant** ...

Enters **HomeAssistant** ...

HomeAssistant would translate a light, *no matter the brand*, as ...

Enters **HomeAssistant** ...

HomeAssistant would translate a light, *no matter the brand*, as ...

... a `light` `entity`

and offer a standardised interface to access it

Entities, Services and Components

Every “thing” is represented by **HomeAssistant** as an `entity`

Every “thing” is represented by **HomeAssistant** as an `entity`

- Can have **state**
- Can be the **target of service calls**
- Examples

Every “thing” is represented by **HomeAssistant** as an `entity`

- Can have **state**
- Can be the **target of service calls**
- Examples
  - A connected switch
  - A lightbulb
  - Whether or not the sun is up
  - Current weather

HomeAssistant offers an API to manipulate entities via `Services`

HomeAssistant offers an API to manipulate entities via `Services`

`Service` is an endpoint that can be called and which performs an action

Examples

HomeAssistant offers an API to manipulate entities via `Services`

`Service` is an endpoint that can be called and which performs an action

Examples

- Turn On / Off
- Change Brightness
- Increase Volume

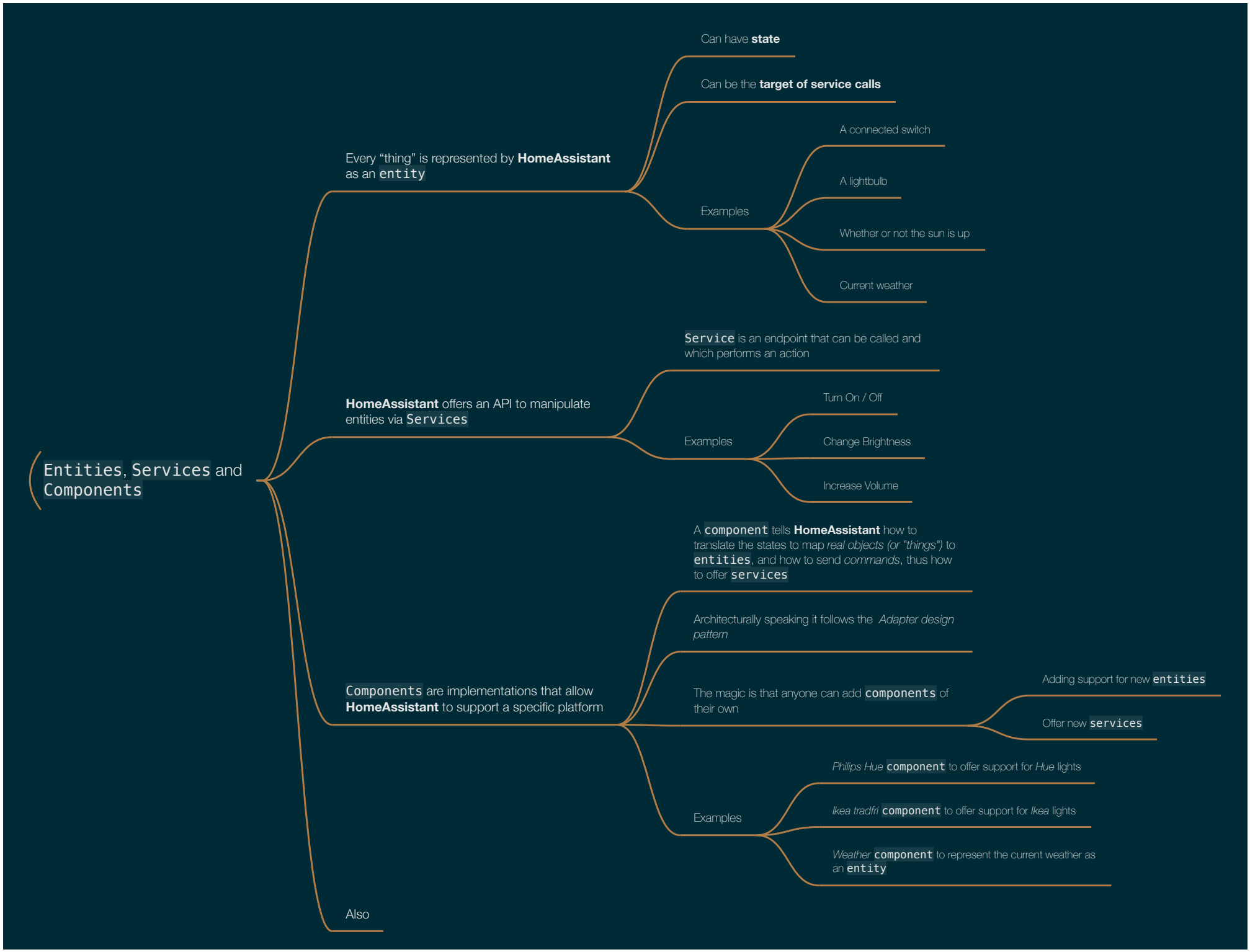
`Components` are implementations that allow **HomeAssistant** to support a specific platform

`Components` are implementations that allow **HomeAssistant** to support a specific platform

- A `component` tells **HomeAssistant** how to translate the states to map *real objects (or "things")* to `entities`, and how to send *commands*, thus how to offer `services`
- Architecturally speaking it follows the *Adapter design pattern*
- The magic is that anyone can add `components` of their own
  - Adding support for new `entities`
  - Offer new `services`
- Examples

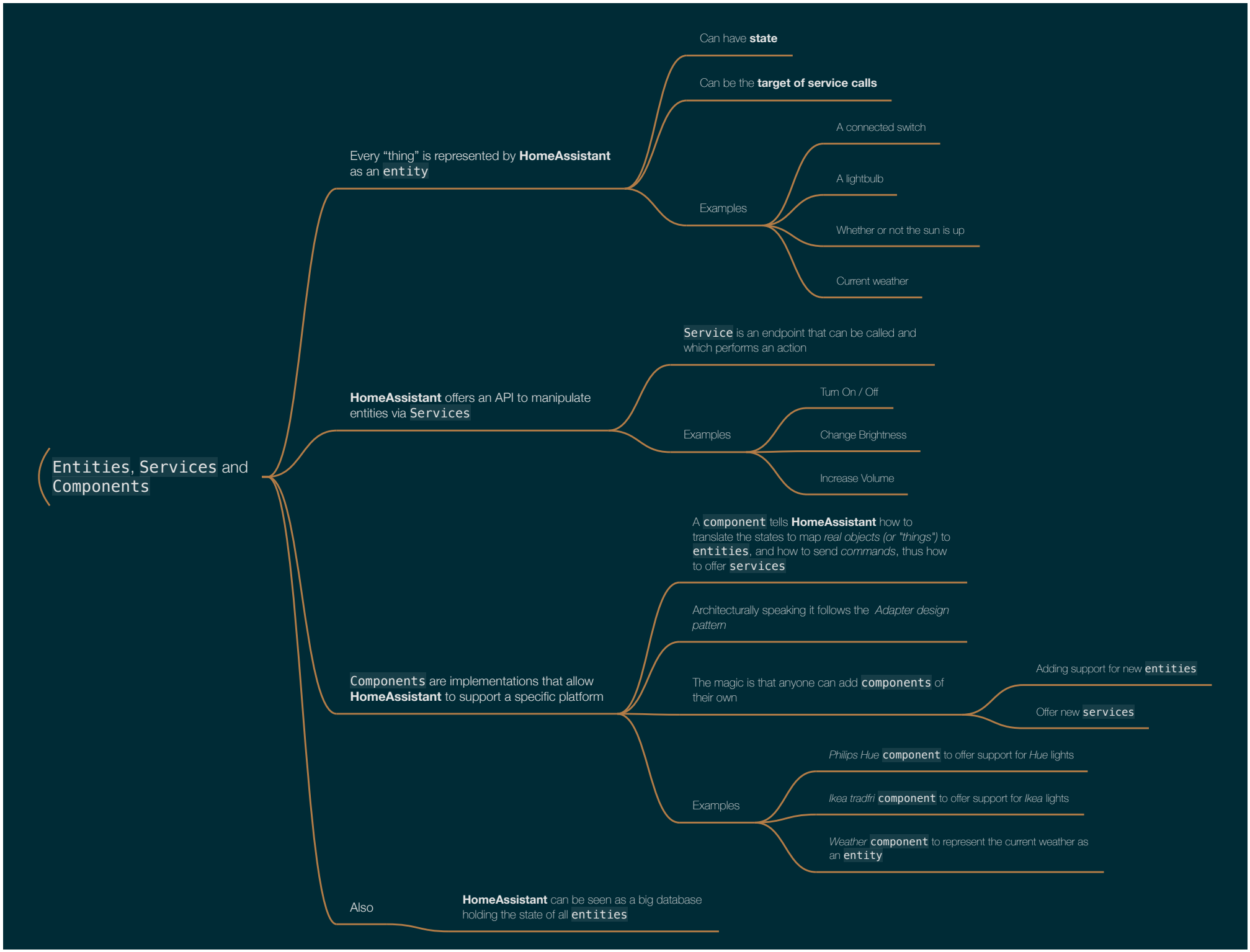
`Components` are implementations that allow **HomeAssistant** to support a specific platform

- A `component` tells **HomeAssistant** how to translate the states to map *real objects (or "things")* to `entities`, and how to send *commands*, thus how to offer `services`
- Architecturally speaking it follows the *Adapter design pattern*
- The magic is that anyone can add `components` of their own
  - Adding support for new `entities`
  - Offer new `services`
- Examples
  - *Philips Hue* `component` to offer support for *Hue* lights
  - *Ikea tradfri* `component` to offer support for *Ikea* lights
  - *Weather* `component` to represent the current weather as an `entity`


Entities, Services and Components
Every "thing" is represented by HomeAssistant as an entity
Can have state
Can be the target of service calls
Examples
A connected switch
A lightbulb
Whether or not the sun is up
Current weather
HomeAssistant offers an API to manipulate entities via Services
Service is an endpoint that can be called and which performs an action
Examples
Turn On / Off
Change Brightness
Increase Volume
Components are implementations that allow HomeAssistant to support a specific platform
A component tells HomeAssistant how to translate the states to map real objects (or "things") to entities, and how to send commands, thus how to offer services
Architecturally speaking it follows the Adapter design pattern
The magic is that anyone can add components of their own
Adding support for new entities
Offer new services
Examples
Philips Hue component to offer support for Hue lights
Ikea tradfri component to offer support for Ikea lights
Weather component to represent the current weather as an entity
Also

Also

HomeAssistant can be seen as a big database holding the state of all `entities`


Entities, Services and Components
Every "thing" is represented by HomeAssistant as an entity
Can have state
Can be the target of service calls
Examples
A connected switch
A lightbulb
Whether or not the sun is up
Current weather
HomeAssistant offers an API to manipulate entities via Services
Service is an endpoint that can be called and which performs an action
Examples
Turn On / Off
Change Brightness
Increase Volume
Components are implementations that allow HomeAssistant to support a specific platform
A component tells HomeAssistant how to translate the states to map real objects (or "things") to entities, and how to send commands, thus how to offer services
Architecturally speaking it follows the Adapter design pattern
The magic is that anyone can add components of their own
Adding support for new entities
Offer new services
Examples
Philips Hue component to offer support for Hue lights
Ikea tradfri component to offer support for Ikea lights
Weather component to represent the current weather as an entity
Also
HomeAssistant can be seen as a big database holding the state of all entities

Questions about
HomeAssistant so far?

Everything is translated by **HomeAssistant**

- Tower of Babel
  - Many brands with many different protocols to implement essentially the same thing
  - They're obviously not compatible
  - For instance
    - • A Philips Hue Bulb
      • An Ikea LED Strip
    - Essentially the same thing: **A light**
      - On / off
      - Brightness
      - (Maybe) color temperature
    - Yet, 2 completely different **Apps** or **API** to control them
- Enters **HomeAssistant** ...
  - **HomeAssistant** would translate a light, *no matter the brand*, as ...
    - ... a `light` `entity`
    - and offer a standardised interface to access it
- `Entities`, `Services` and `Components`
  - Every "thing" is represented by **HomeAssistant** as an `entity`
    - Can have **state**
    - Can be the **target of service calls**
    - Examples
      - A connected switch
      - A lightbulb
      - Whether or not the sun is up
      - Current weather
  - **HomeAssistant** offers an API to manipulate entities via `Services`
    - `Service` is an endpoint that can be called and which performs an action
    - Examples
      - Turn On / Off
      - Change Brightness
      - Increase Volume
  - `Components` are implementations that allow **HomeAssistant** to support a specific platform
    - A `component` tells **HomeAssistant** how to translate the states to map *real objects (or "things")* to `entities`, and how to send *commands*, thus how to offer `services`
    - Architecturally speaking it follows the *Adapter design pattern*
    - The magic is that anyone can add `components` of their own
      - Adding support for new `entities`
      - Offer new `services`
    - Examples
      - *Philips Hue* `component` to offer support for *Hue* lights
      - *Ikea tradfri* `component` to offer support for *Ikea* lights
      - *Weather* `component` to represent the current weather as an `entity`
  - Also
    - **HomeAssistant** can be seen as a big database holding the state of all `entities`

Home Assistant & Appdaemon

- Architecture
  - Drawing on Board
- Everything is translated by **HomeAssistant**
- Everything is orchestrated by **Appdaemon**

Everything is orchestrated by **Appdaemon**

- **Appdaemon** is conceptually simpler
  - Mostly because it now deals with a unified interface
  - The beauty of good design, thank you **HomeAssistant** 😍
- **Appdaemon** uses the **HomeAssistant** *API* to orchestrate actions on the different `entities`
- Every automation is represented by an **Appdaemon** `App`

Appdaemon uses the **HomeAssistant** *API* to orchestrate actions on the different `entities`

It can listen to the *state* of `entities`, run at specific times and listen to `events` send by **HomeAssistant**

It then can trigger actions on `entities` by calling **HomeAssistant** `services`

Every automation is represented by an **Appdaemon** `App`

- An `App` is a `Python` class registered with **Appdaemon** as an `App`
- Every `App` has 2 phases
- Examples

Every `App` has 2 phases

Every `App` has 2 phases

Initialization

Register itself to listen to `entity` state changes, **HomeAssistant** `events`

Schedule function calls for later

Run

Every `App` has 2 phases

Initialization

Register itself to listen to `entity` state changes, **HomeAssistant** `events`

Schedule function calls for later

Run

Respond to `events` or `entity` state changes

Run scheduled function

Schedule new function calls

Every automation is represented by an **Appdaemon** `App`

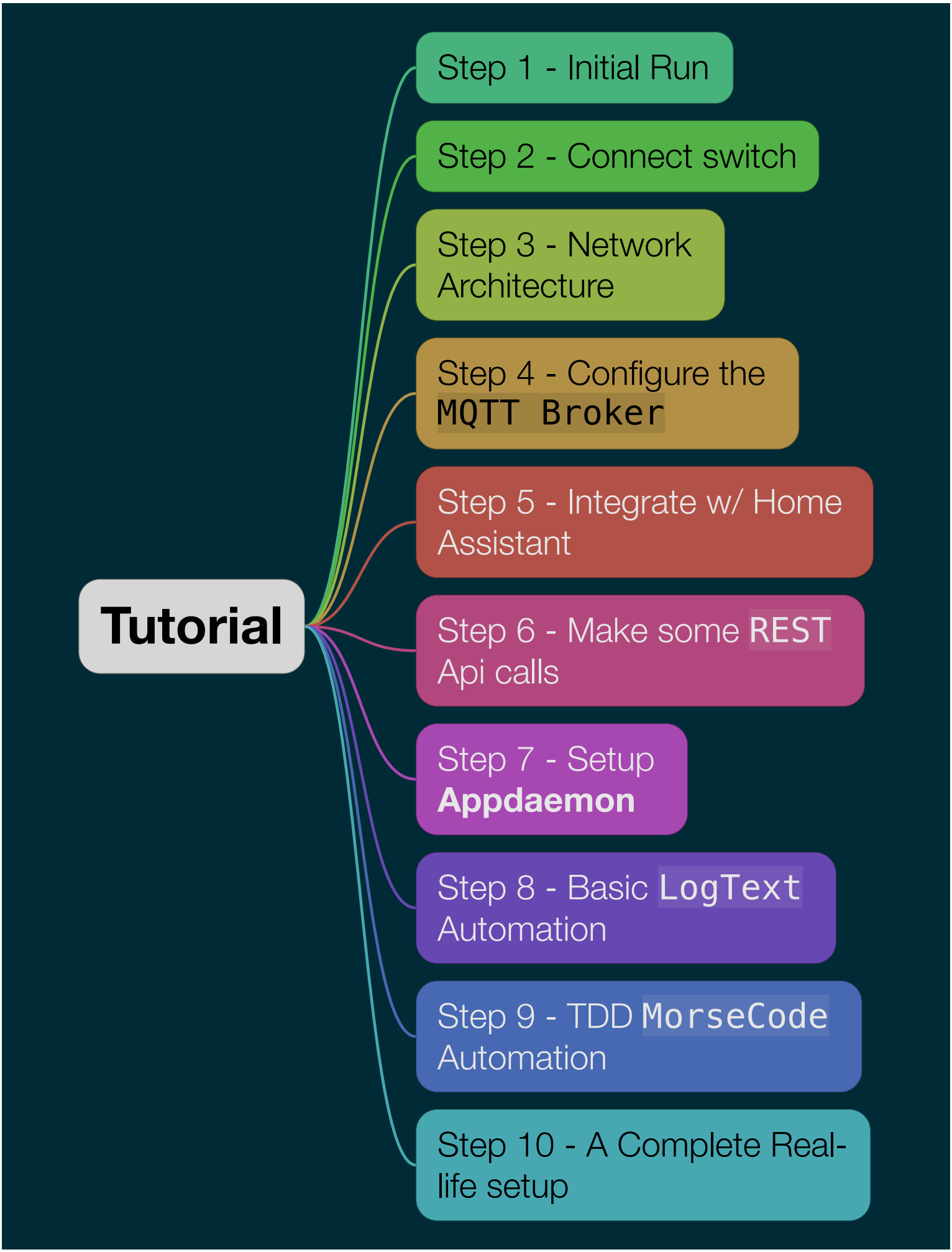
- An `App` is a `Python` class registered with **Appdaemon** as an `App`
- Every `App` has 2 phases
  - *Initialization*
    - Register itself to listen to `entity` state changes, **HomeAssistant** `events`
    - Schedule function calls for later
  - *Run*
    - Respond to `events` or `entity` state changes
    - Run scheduled function
    - Schedule new function calls
- Examples

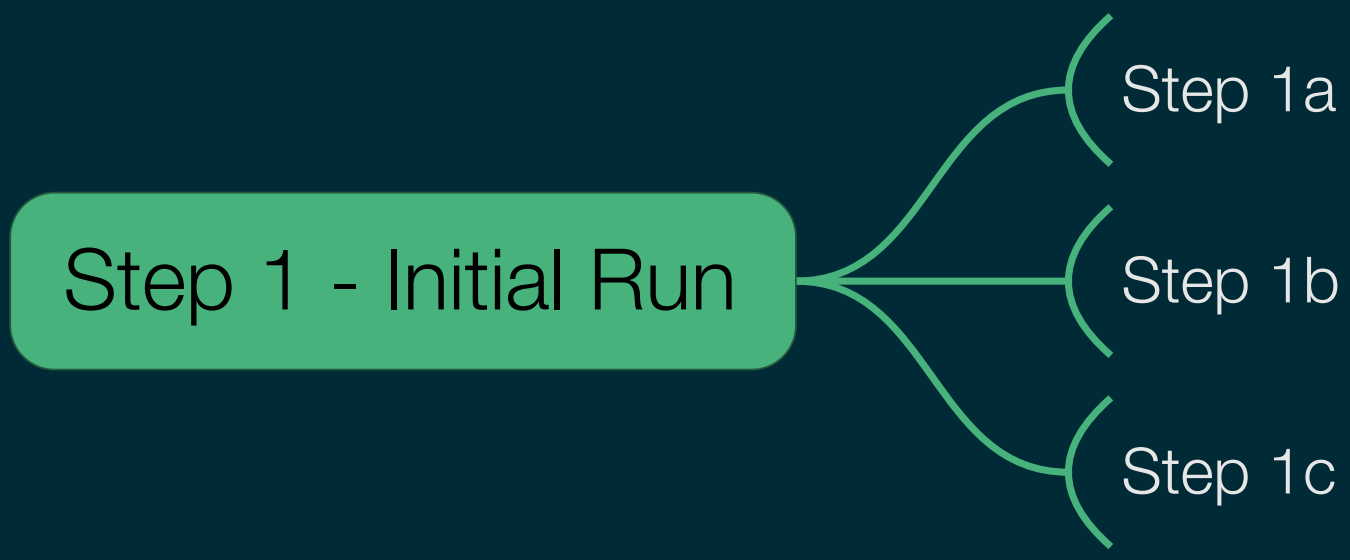
Every automation is represented by an **Appdaemon** `App`

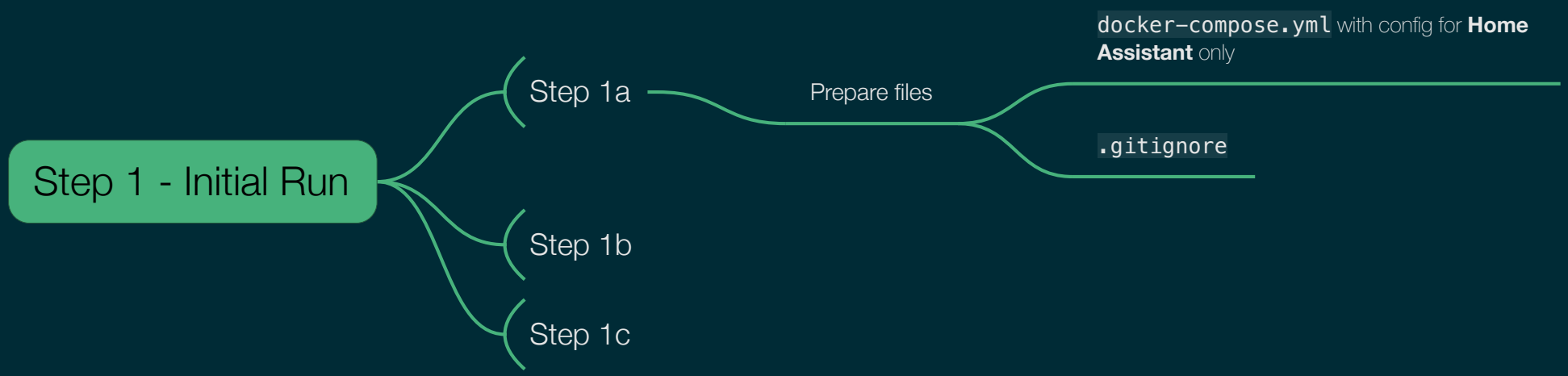
- An `App` is a `Python` class registered with **Appdaemon** as an `App`
- Every `App` has 2 phases
  - *Initialization*
    - Register itself to listen to `entity` state changes, **HomeAssistant** `events`
    - Schedule function calls for later
  - *Run*
    - Respond to `events` or `entity` state changes
    - Run scheduled function
    - Schedule new function calls
- Examples
  - Turn on the lights when I enter the room
  - Send me a message if I leave home with a window opened
  - Notify me with a message spoken through my all my speakers at home if a window is opened but it's about to rain
  - Turn on the heating when you see me leave work to come back home.
  - *(because* `Apps` *do not have to deal with the complexity of multiple platforms, it makes it easy to create really powerful automations)*

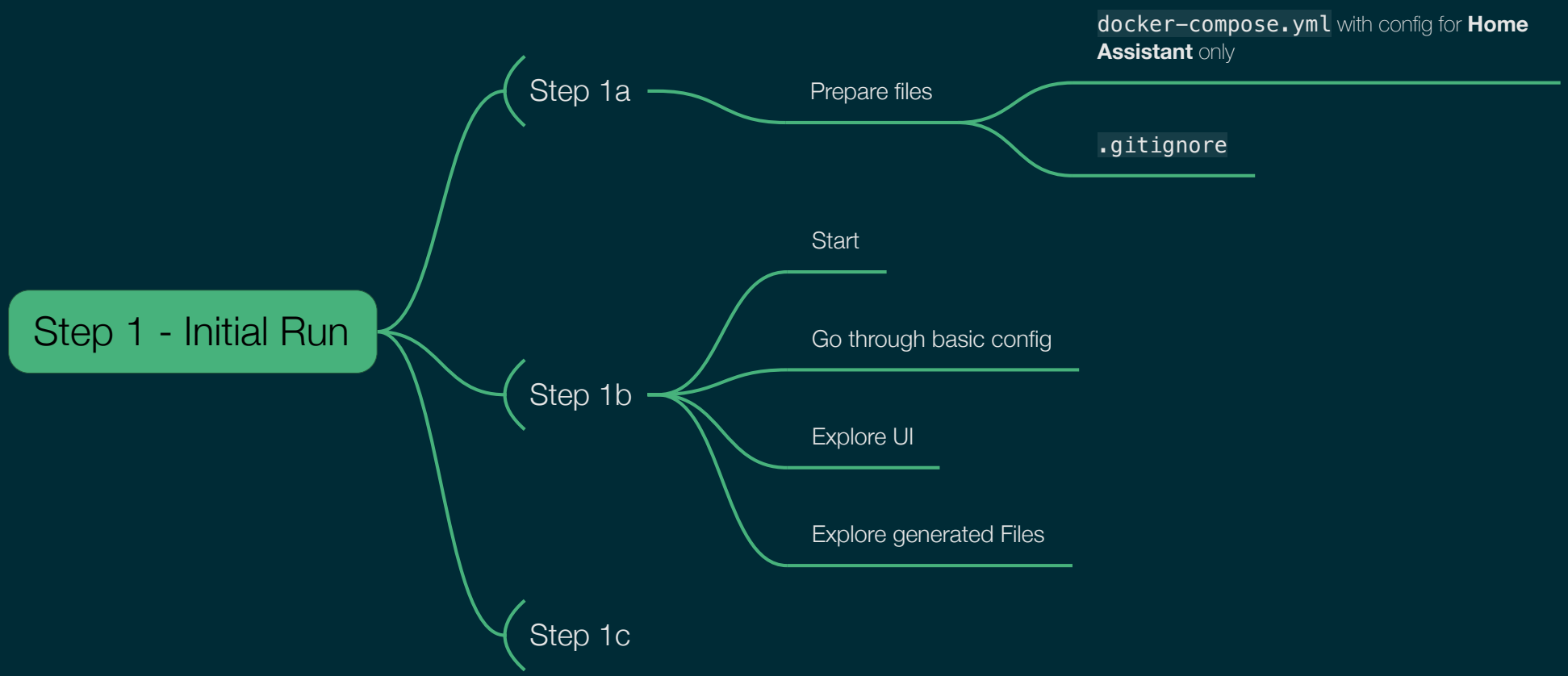
Home Assistant & Appdaemon

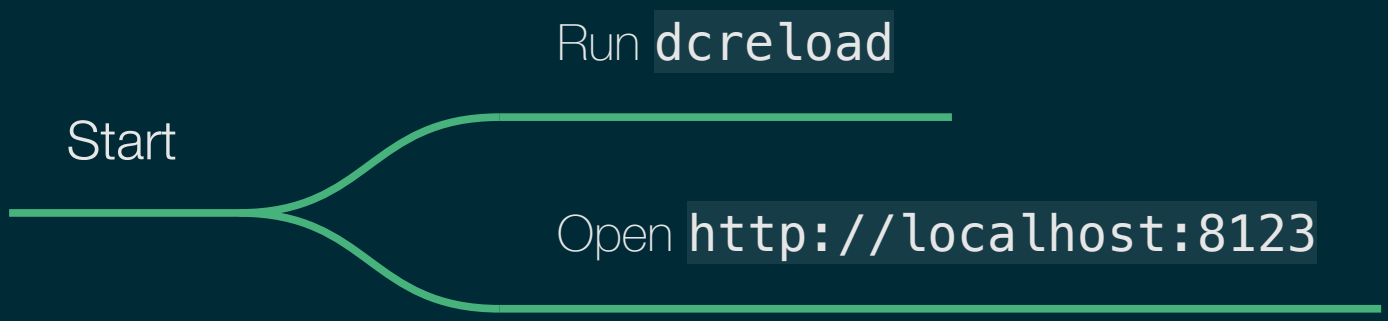
- Architecture
  - Drawing on Board
- Everything is translated by **HomeAssistant**
  - Tower of Babel
    - Many brands with many different protocols to implement essentially the same thing
    - They're obviously not compatible
    - For instance
      - • A Philips Hue Bulb
        • An Ikea LED Strip
      - Essentially the same thing: **A light**
        - On / off
        - Brightness
        - (Maybe) color temperature
      - Yet, 2 completely different **Apps** or **API** to control them
  - Enters **HomeAssistant** ...
    - **HomeAssistant** would translate a light, *no matter the brand*, as ...
      - ... a `light` `entity`
      - and offer a standardised interface to access it
  - `Entities`, `Services` and `Components`
    - Every "thing" is represented by **HomeAssistant** as an `entity`
      - Can have **state**
      - Can be the **target of service calls**
      - Examples
        - A connected switch — Entity
        - A lightbulb — Entity
        - Whether or not the sun is up — Boolean "sensor" — Entity
        - Current weather — "Thing" with state — Entity
    - **HomeAssistant** offers an API to manipulate entities via `Services`
      - `Service` is an endpoint that can be called and which performs an action
      - Examples
        - Turn On / Off
        - Change Brightness
        - Increase Volume
    - `Components` are implementations that allow **HomeAssistant** to support a specific platform
      - A `component` tells **HomeAssistant** how to translate the states to map *real objects (or "things")* to `entities`, and how to send *commands*, thus how to offer `services`
      - Architecturally speaking it follows the *Adapter design pattern*
      - The magic is that anyone can add `components` of their own
        - Adding support for new `entities`
        - Offer new `services`
      - Examples
        - *Philips Hue* `component` to offer support for *Hue* lights
        - *Ikea tradfri* `component` to offer support for *Ikea* lights
        - *Weather* `component` to represent the current weather as an `entity`
    - Also
      - **HomeAssistant** can be seen as a big database holding the state of all `entities`
- Everything is orchestrated by **Appdaemon**
  - **Appdaemon** is conceptually simpler
    - Mostly because it now deals with a unified interface
    - The beauty of good design, thank you **HomeAssistant** 😍
  - **Appdaemon** uses the **HomeAssistant** *API* to orchestrate actions on the different `entities`
    - It can listen to the *state* of `entities`, run at specific times and listen to `events` send by **HomeAssistant**
    - It then can trigger actions on `entities` by calling **HomeAssistant** `services`
  - Every automation is represented by an **Appdaemon** `App`
    - An `App` is a `Python` class registered with **Appdaemon** as an `App`
    - Every `App` has 2 phases
      - *Initialization*
        - Register itself to listen to `entity` state changes, **HomeAssistant** `events`
        - Schedule function calls for later
      - *Run*
        - Respond to `events` or `entity` state changes
        - Run scheduled function
        - Schedule new function calls
    - Examples
      - Turn on the lights when I enter the room
      - Send me a message if I leave home with a window opened
      - Notify me with a message spoken through my all my speakers at home if a window is opened but it's about to rain
      - Turn on the heating when you see me leave work to come back home.
      - *(because* `Apps` *do not have to deal with the complexity of multiple platforms, it makes it easy to create really powerful automations)*


Tutorial
Step 1 - Initial Run
Step 2 - Connect switch
Step 3 - Network Architecture
Step 4 - Configure the MQTT Broker
Step 5 - Integrate w/ Home Assistant
Step 6 - Make some REST Api calls
Step 7 - Setup Appdaemon
Step 8 - Basic LogText Automation
Step 9 - TDD MorseCode Automation
Step 10 - A Complete Real-life setup

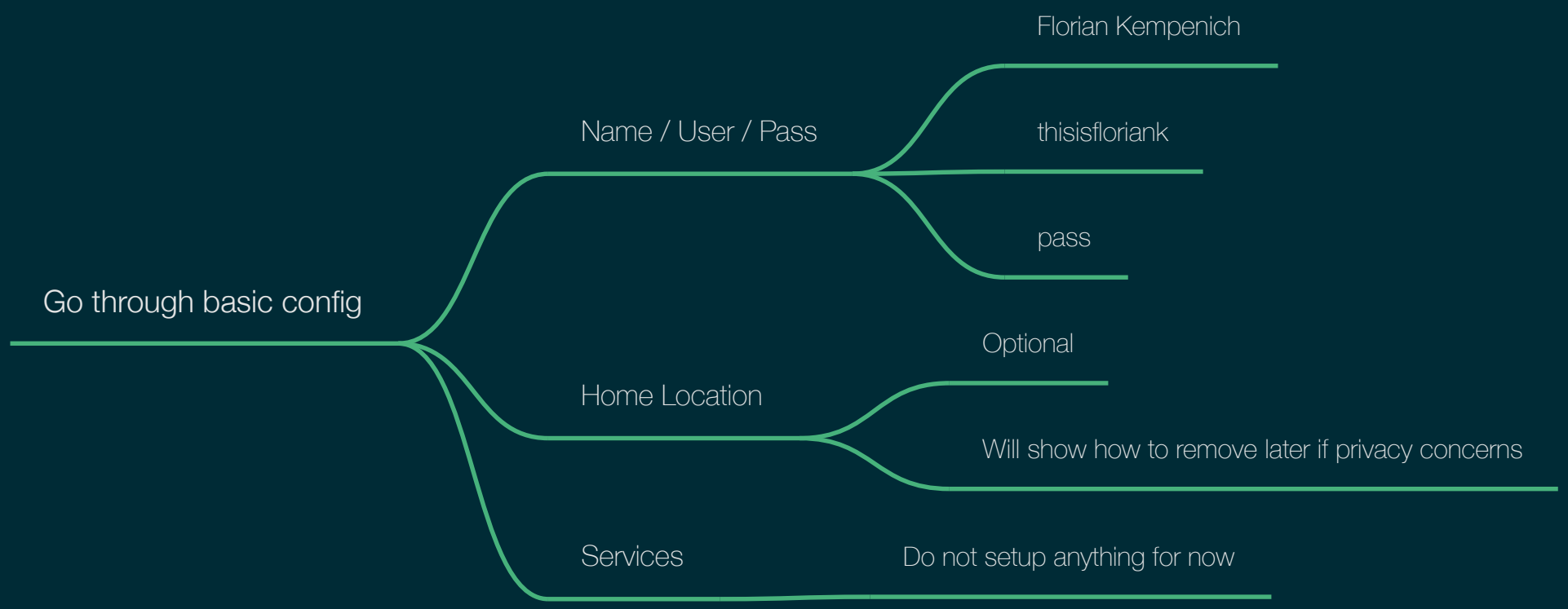

Step 1 - Initial Run
Step 1a
Step 1b
Step 1c


Step 1 - Initial Run
Step 1a
Step 1b
Step 1c
Prepare files
docker-compose.yml with config for Home Assistant only
.gitignore


Step 1 - Initial Run
Step 1a
Prepare files
docker-compose.yml with config for Home Assistant only
.gitignore
Step 1b
Start
Go through basic config
Explore UI
Explore generated Files
Step 1c


Start
Run dcreload
Open http://localhost:8123

Go through basic config


Go through basic config
Name / User / Pass
Florian Kempenich
thisisfloriank
pass
Home Location
Optional
Will show how to remove later if privacy concerns
Services
Do not setup anything for now

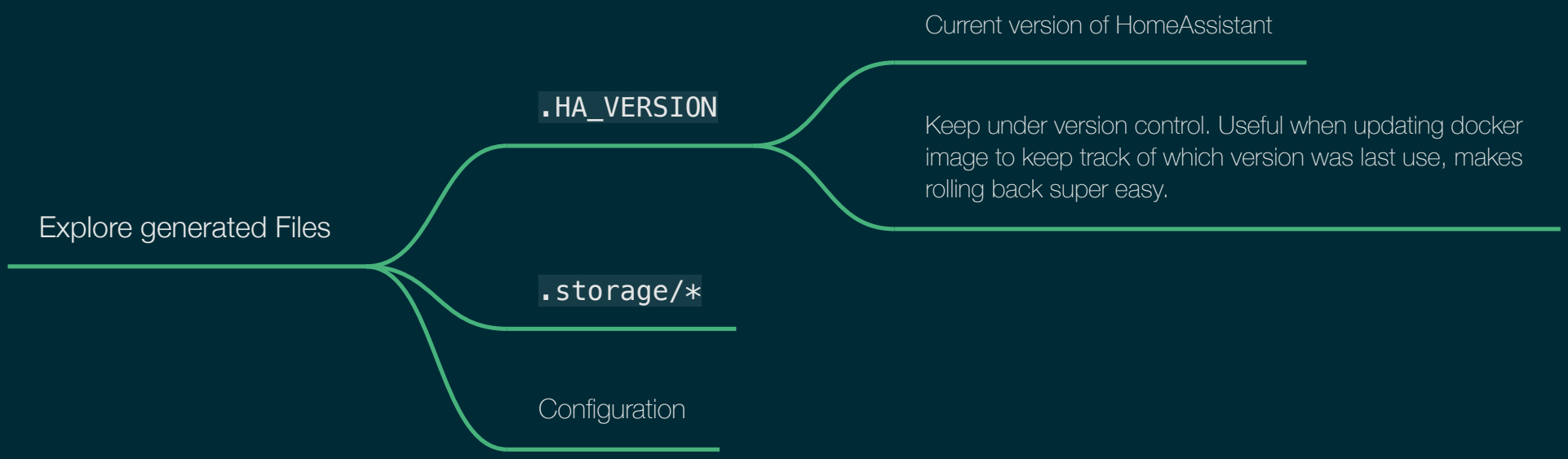
Explore UI

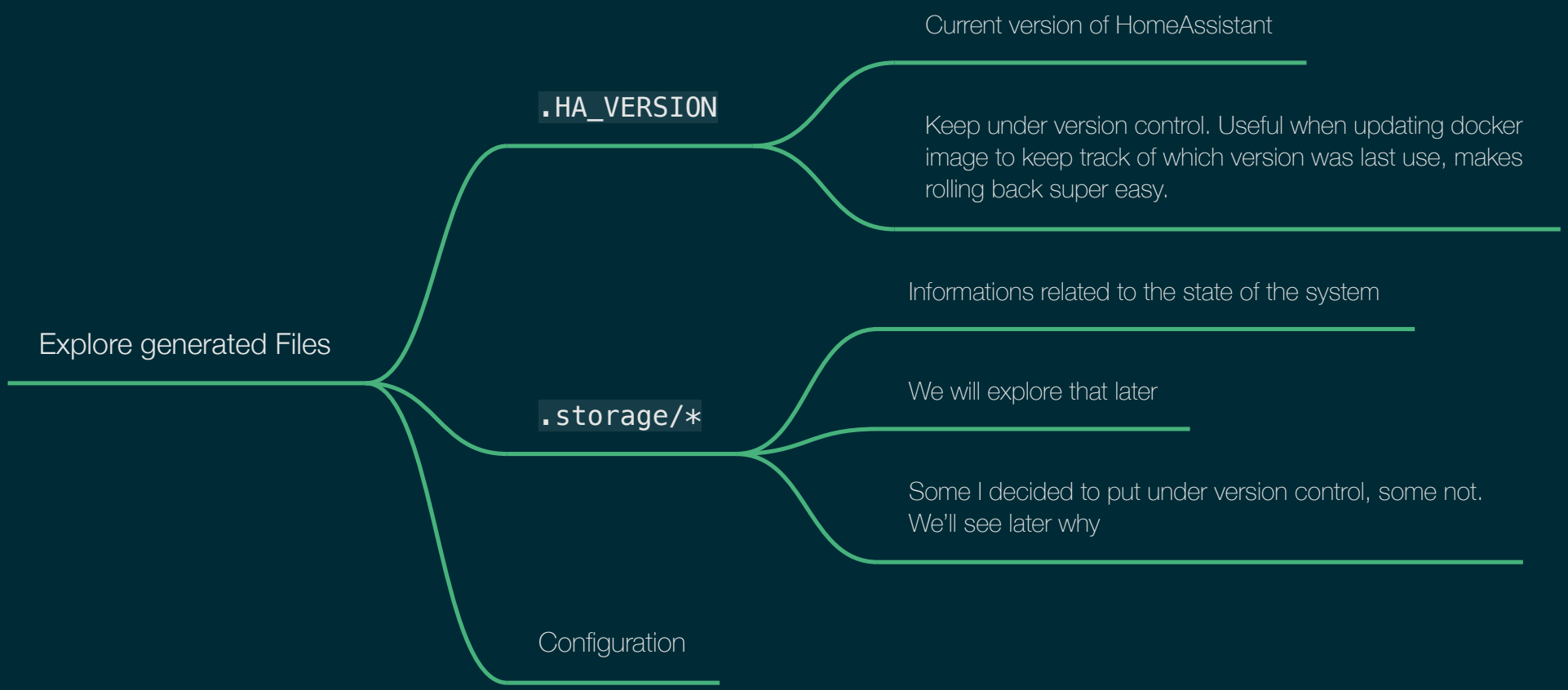

Save Login
Explore UI
Main UI

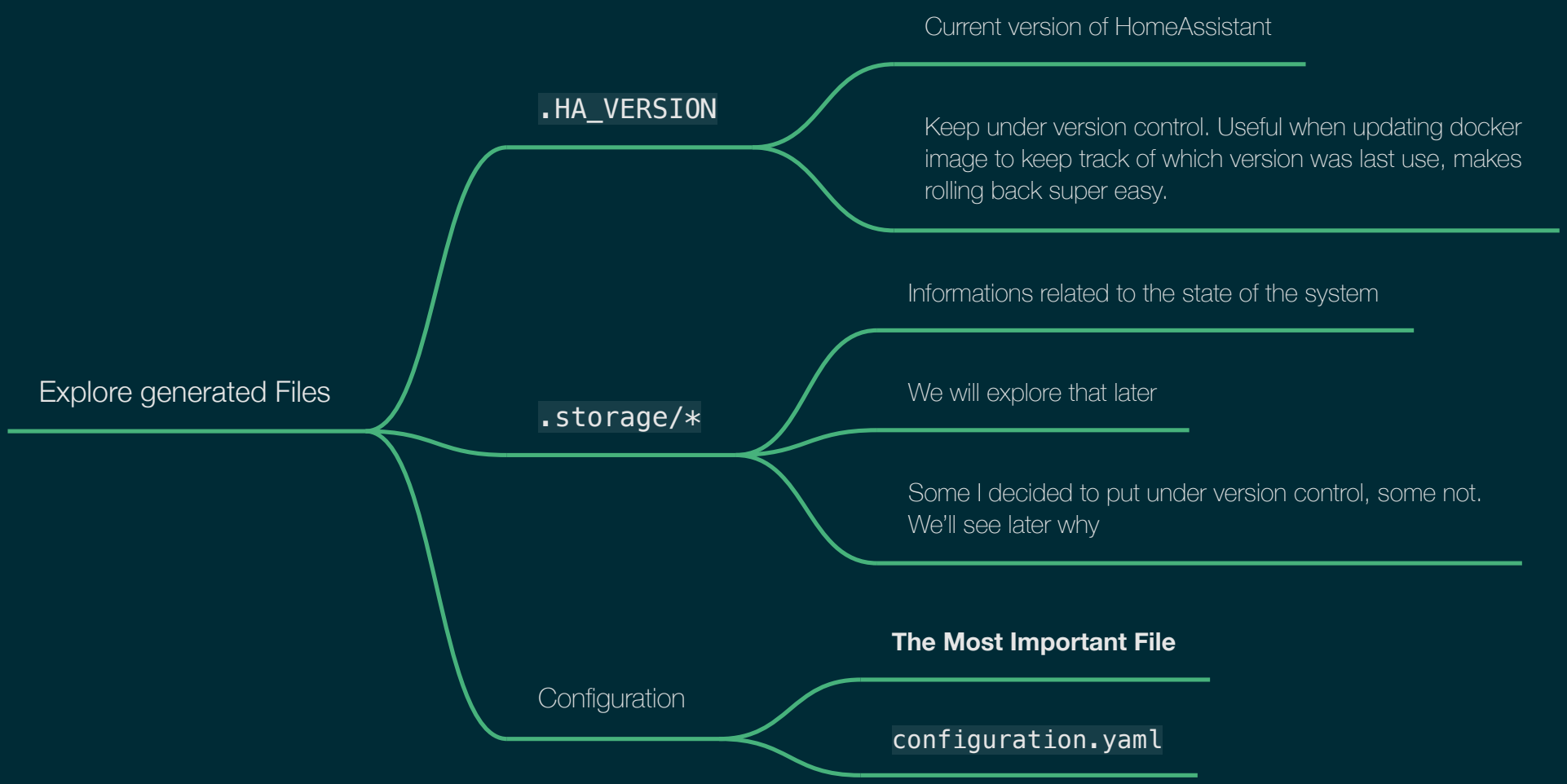
Explore generated Files

Explore generated Files

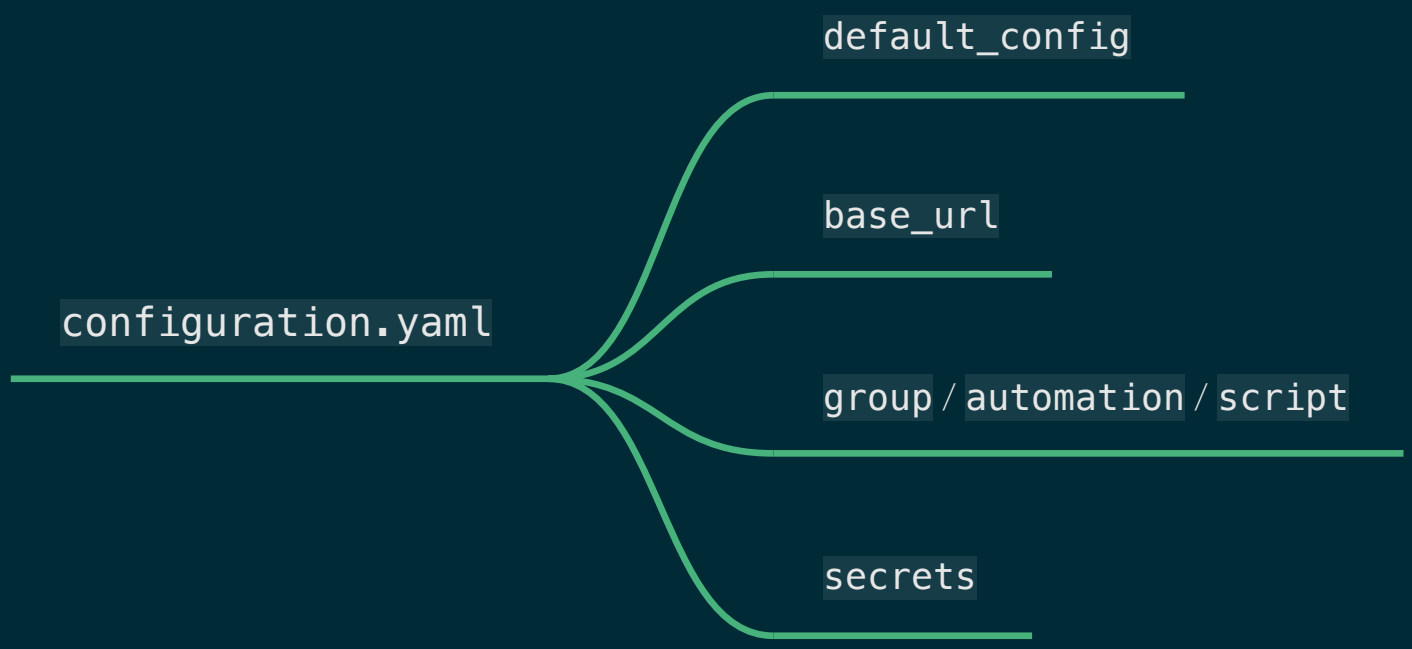
- `.HA_VERSION`
- `.storage/*`
- Configuration


Explore generated Files
.HA_VERSION
Current version of HomeAssistant
Keep under version control. Useful when updating docker image to keep track of which version was last use, makes rolling back super easy.
.storage/*
Configuration

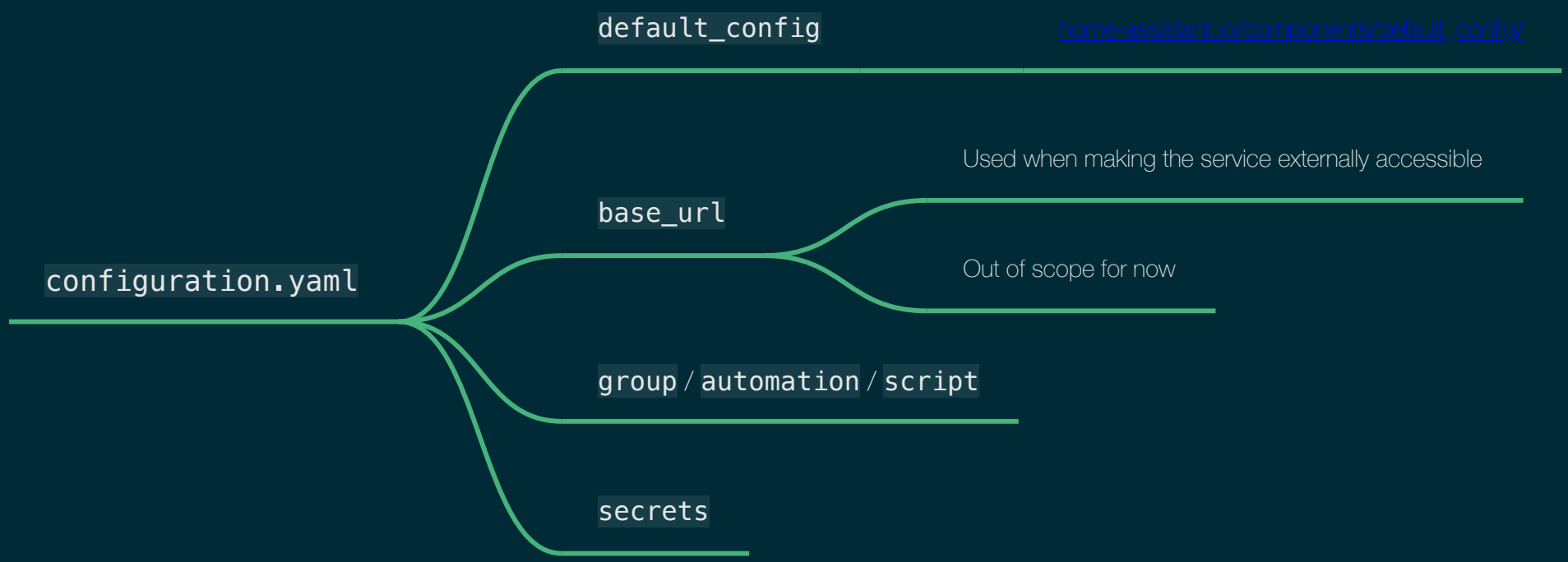

Explore generated Files
.HA_VERSION
Current version of HomeAssistant
Keep under version control. Useful when updating docker image to keep track of which version was last use, makes rolling back super easy.
.storage/*
Informations related to the state of the system
We will explore that later
Some I decided to put under version control, some not. We'll see later why
Configuration

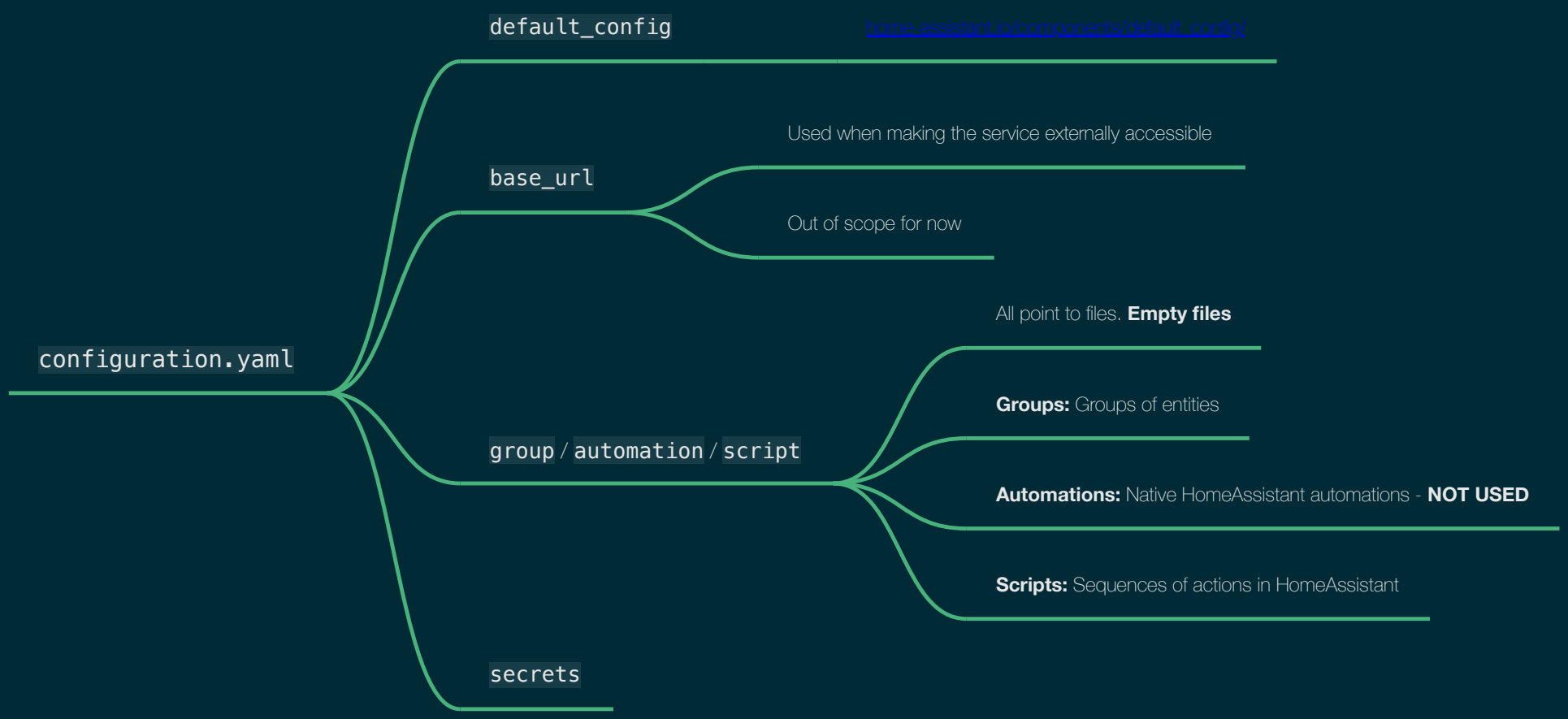

Explore generated Files
.HA_VERSION
Current version of HomeAssistant
Keep under version control. Useful when updating docker image to keep track of which version was last use, makes rolling back super easy.
.storage/*
Informations related to the state of the system
We will explore that later
Some I decided to put under version control, some not. We'll see later why
Configuration
The Most Important File
configuration.yaml

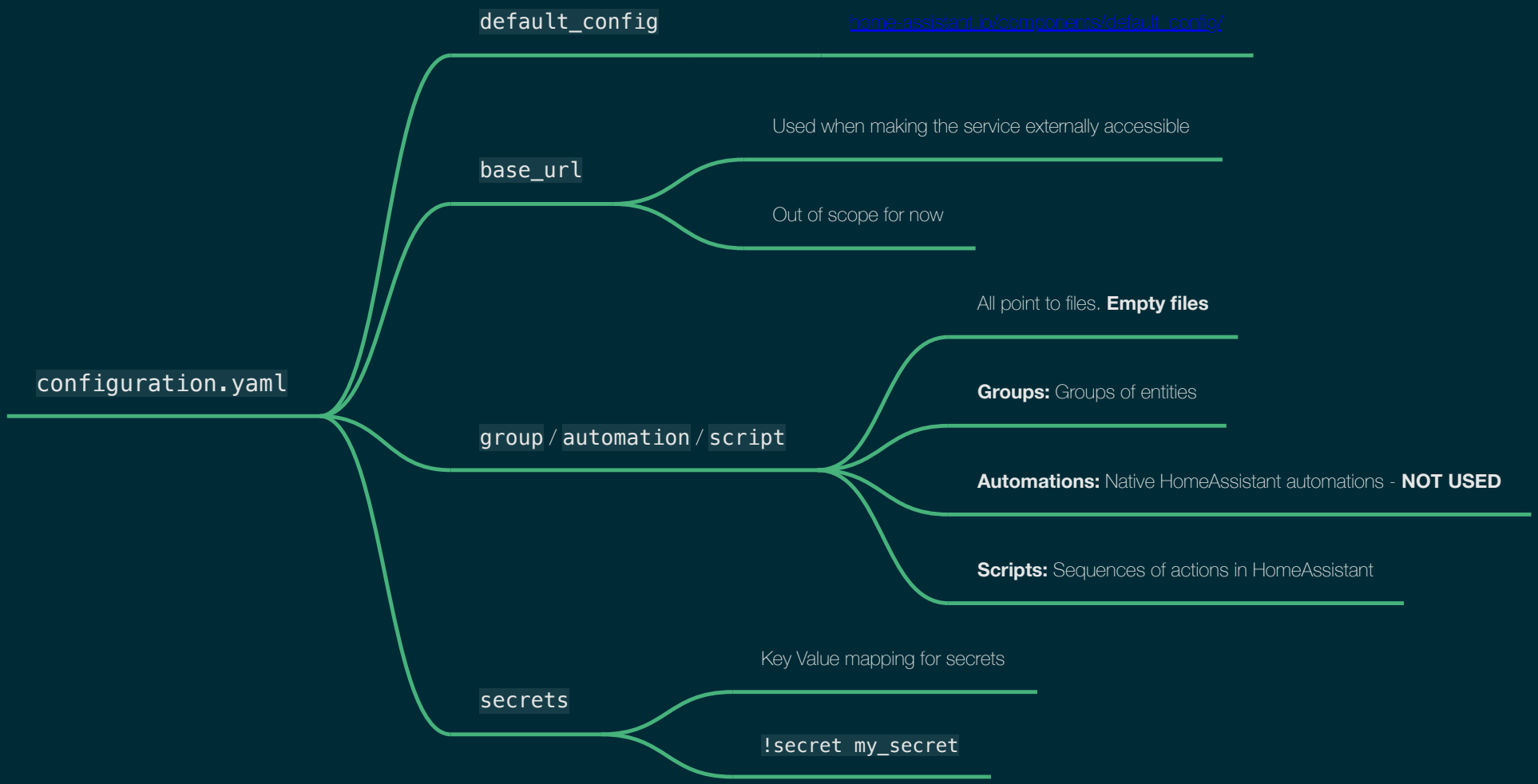
configuration.yaml

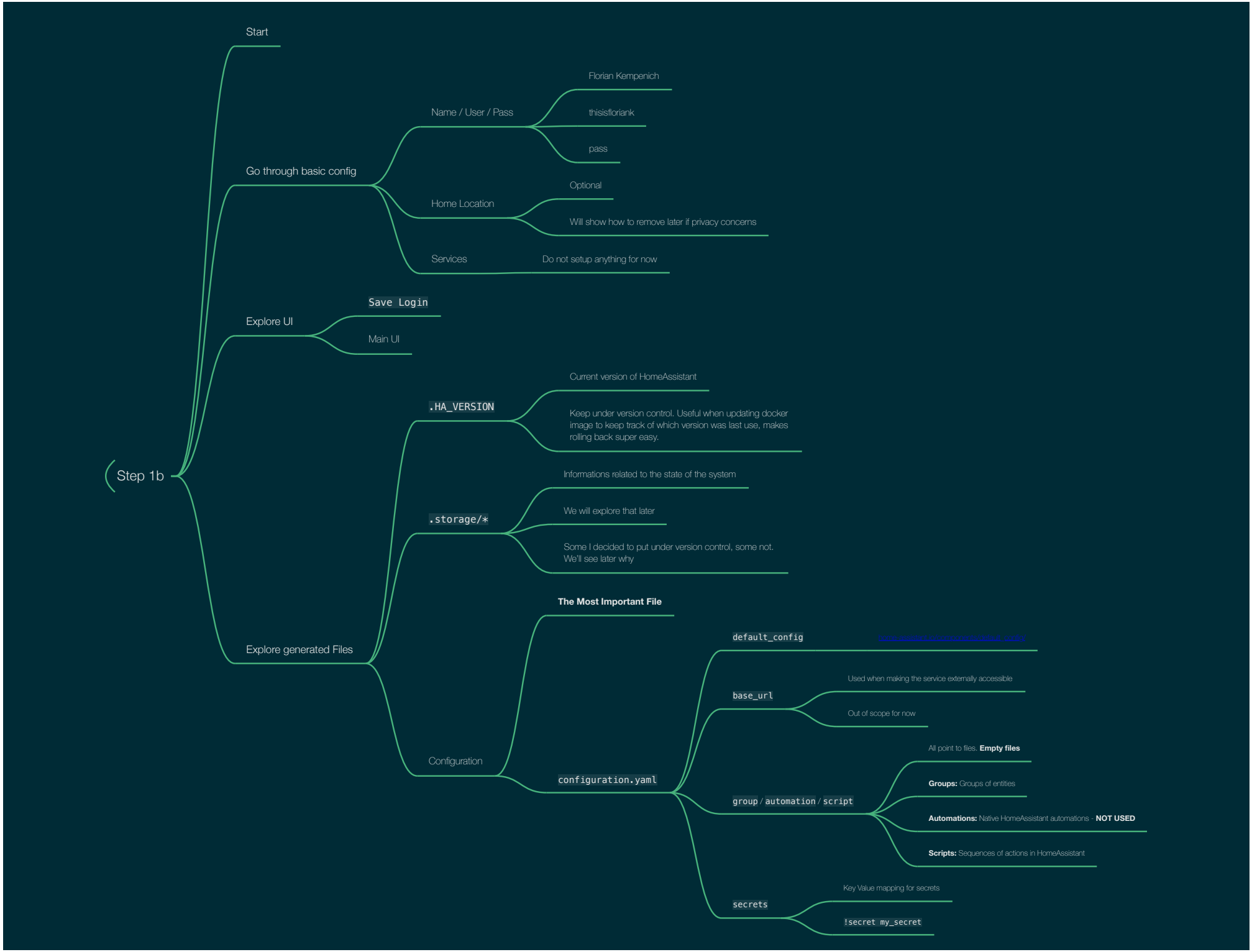

configuration.yaml
default_config
base_url
group / automation / script
secrets


configuration.yaml
default_config
home-assistant.io/components/default_config/
base_url
group / automation / script
secrets


configuration.yaml
default_config
home-assistant.io/components/default_config/
base_url
Used when making the service externally accessible
Out of scope for now
group / automation / script
secrets


configuration.yaml
default_config
home-assistant.io/components/default_config/
base_url
Used when making the service externally accessible
Out of scope for now
group / automation / script
All point to files. Empty files
Groups: Groups of entities
Automations: Native HomeAssistant automations - NOT USED
Scripts: Sequences of actions in HomeAssistant
secrets

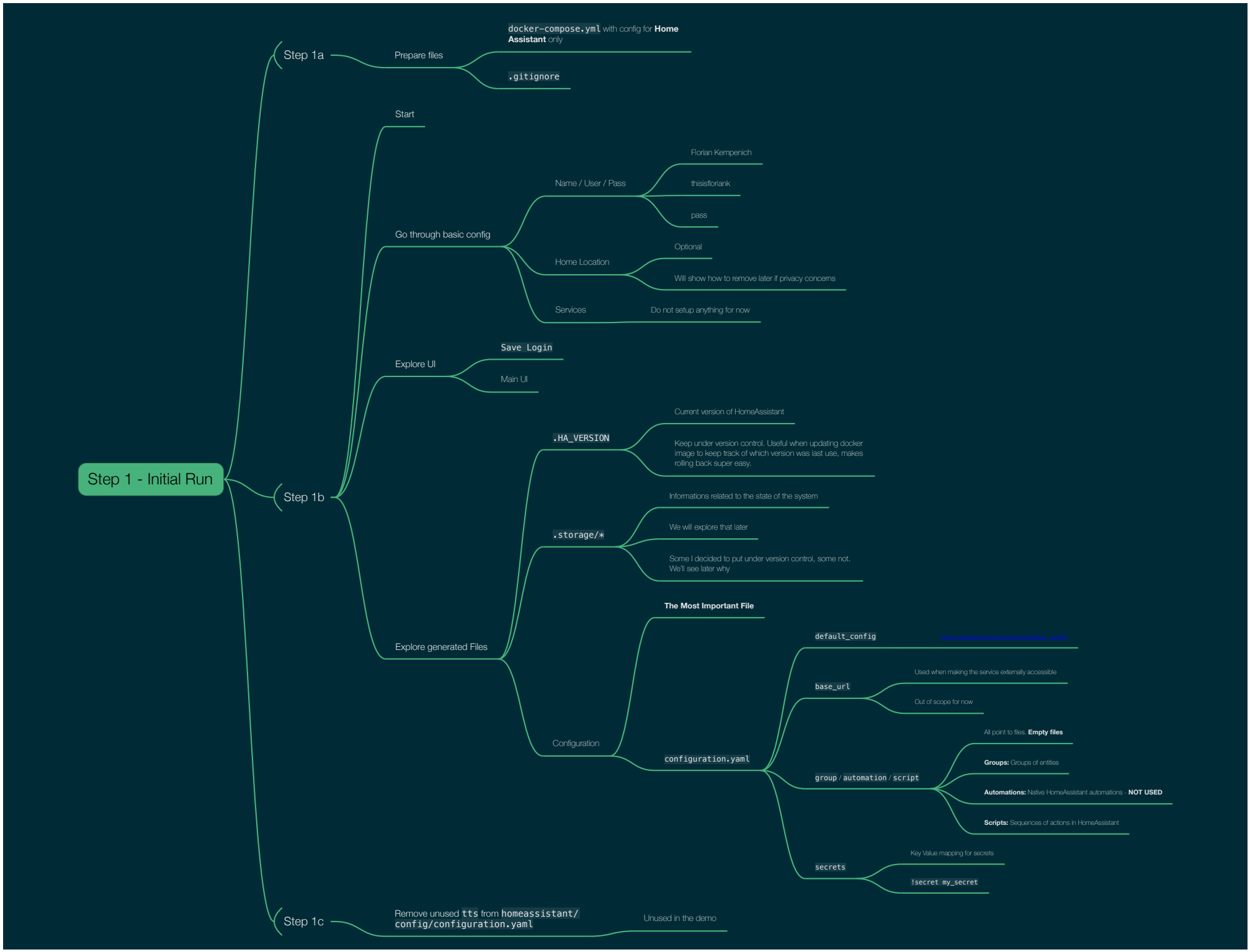

configuration.yaml
default_config
home-assistant.io/components/default_config/
base_url
Used when making the service externally accessible
Out of scope for now
group / automation / script
All point to files. **Empty files**
Groups: Groups of entities
Automations: Native HomeAssistant automations - **NOT USED**
Scripts: Sequences of actions in HomeAssistant
secrets
Key Value mapping for secrets
!secret my_secret


Step 1b
Start
Go through basic config
Name / User / Pass
Florian Kempenich
thisisfloriank
pass
Home Location
Optional
Will show how to remove later if privacy concerns
Services
Do not setup anything for now
Explore UI
Save Login
Main UI
Explore generated Files
.HA_VERSION
Current version of HomeAssistant
Keep under version control. Useful when updating docker image to keep track of which version was last use, makes rolling back super easy.
.storage/*
Informations related to the state of the system
We will explore that later
Some I decided to put under version control, some not. We'll see later why
Configuration
The Most Important File
configuration.yaml
default_config
home-assistant.io/components/default_config/
base_url
Used when making the service externally accessible
Out of scope for now
group / automation / script
All point to files. Empty files
Groups: Groups of entities
Automations: Native HomeAssistant automations - NOT USED
Scripts: Sequences of actions in HomeAssistant
secrets
Key Value mapping for secrets
!secret my_secret

Step 1c

Remove unused `tts` from `homeassistant/config/configuration.yaml`

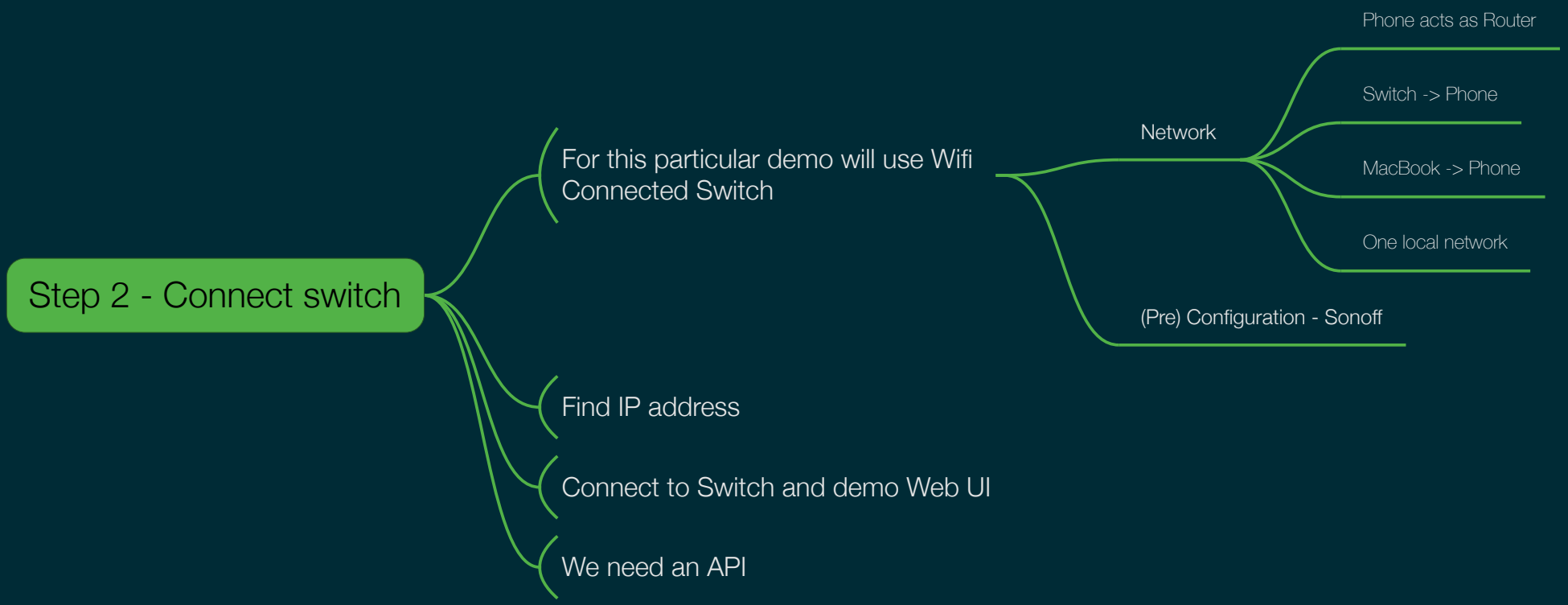
Unused in the demo


Step 1 - Initial Run
Step 1a
Prepare files
docker-compose.yml with config for Home Assistant only
.gitignore
Step 1b
Start
Go through basic config
Name / User / Pass
Florian Kempenich
thisisfloriank
pass
Home Location
Optional
Will show how to remove later if privacy concerns
Services
Do not setup anything for now
Explore UI
Save Login
Main UI
Explore generated Files
.HA_VERSION
Current version of HomeAssistant
Keep under version control. Useful when updating docker image to keep track of which version was last use, makes rolling back super easy.
.storage/*
Informations related to the state of the system
We will explore that later
Some I decided to put under version control, some not. We'll see later why
Configuration
The Most Important File
configuration.yaml
default_config
home-assistant.io/integrations/default_config/
base_url
Used when making the service externally accessible
Out of scope for now
group / automation / script
All point to files. Empty files
Groups: Groups of entities
Automations: Native HomeAssistant automations - NOT USED
Scripts: Sequences of actions in HomeAssistant
secrets
Key Value mapping for secrets
!secret my_secret
Step 1c
Remove unused tts from homeassistant/config/configuration.yaml
Unused in the demo

Step 2 - Connect switch

Step 2 - Connect switch

- For this particular demo will use Wifi Connected Switch
- Find IP address
- Connect to Switch and demo Web UI
- We need an API


Step 2 - Connect switch
For this particular demo will use Wifi Connected Switch
Network
Phone acts as Router
Switch -> Phone
MacBook -> Phone
One local network
(Pre) Configuration - Sonoff
Find IP address
Connect to Switch and demo Web UI
We need an API

Step 2 - Connect switch

- For this particular demo will use Wifi Connected Switch
  - Network
    - Phone acts as Router
    - Switch -> Phone
    - MacBook -> Phone
    - One local network
  - (Pre) Configuration - Sonoff
    - To turn a *Sonoff* switch into a connected switch usable with Home Assistant is quite a complex task and is totally out of scope
    - For more info:
      - Official Repo of the Project arendst/Sonoff-Tasmota
      - Step by Step Tutorial made by me
      (might change location in the future)
- Find IP address
- Connect to Switch and demo Web UI
- We need an API

Step 2 - Connect switch

- For this particular demo will use Wifi Connected Switch
  - Network
    - Phone acts as Router
    - Switch -> Phone
    - MacBook -> Phone
    - One local network
  - (Pre) Configuration - Sonoff
    - To turn a *Sonoff* switch into a connected switch usable with Home Assistant is quite a complex task and is totally out of scope
    - For more info:
      - Official Repo of the Project arendst/Sonoff-Tasmota
      - Step by Step Tutorial made by me
      (might change location in the future)
- Find IP address — `find-devices-on-local-network`
- Connect to Switch and demo Web UI
- We need an API

Step 2 - Connect switch

- For this particular demo will use Wifi Connected Switch
  - Network
    - Phone acts as Router
    - Switch -> Phone
    - MacBook -> Phone
    - One local network
  - (Pre) Configuration - Sonoff
    - To turn a *Sonoff* switch into a connected switch usable with Home Assistant is quite a complex task and is totally out of scope
    - For more info:
      - Official Repo of the Project arendst/Sonoff-Tasmota
      - Step by Step Tutorial made by me (might change location in the future)
- Find IP address
  - `find-devices-on-local-network`
- Connect to Switch and demo Web UI
  - This is the connected interface for the Switch
  - It has nothing to do with HomeAssistant at this point
  - Any connected device will have some way to access it
  - For this switch in particular, there is the Web UI. Other might have an app.
- We need an API

Step 2 - Connect switch

- For this particular demo will use Wifi Connected Switch
  - Network
    - Phone acts as Router
    - Switch -> Phone
    - MacBook -> Phone
    - One local network
  - (Pre) Configuration - Sonoff
    - To turn a *Sonoff* switch into a connected switch usable with Home Assistant is quite a complex task and is totally out of scope
    - For more info:
      - Official Repo of the Project arendst/Sonoff-Tasmota
      - Step by Step Tutorial made by me (might change location in the future)
- Find IP address
  - `find-devices-on-local-network`
- Connect to Switch and demo Web UI
  - This is the connected interface for the Switch
  - It has nothing to do with HomeAssistant at this point
  - Any connected device will have some way to access it
  - For this switch in particular, there is the Web UI. Other might have an app.
- We need an API
  - To integrate w/ HomeAssistant we need some way to programatically access the device
  - Most services services offer a REST api (ie, Philips Hue)
  - But this service allow to post/subscribe to a `MQTT` topic

Step 3 - Network Architecture

Step 3 - Network Architecture

- 3 types of architecture (mainly)
  - Do drawing on board
- We’re in the middle one

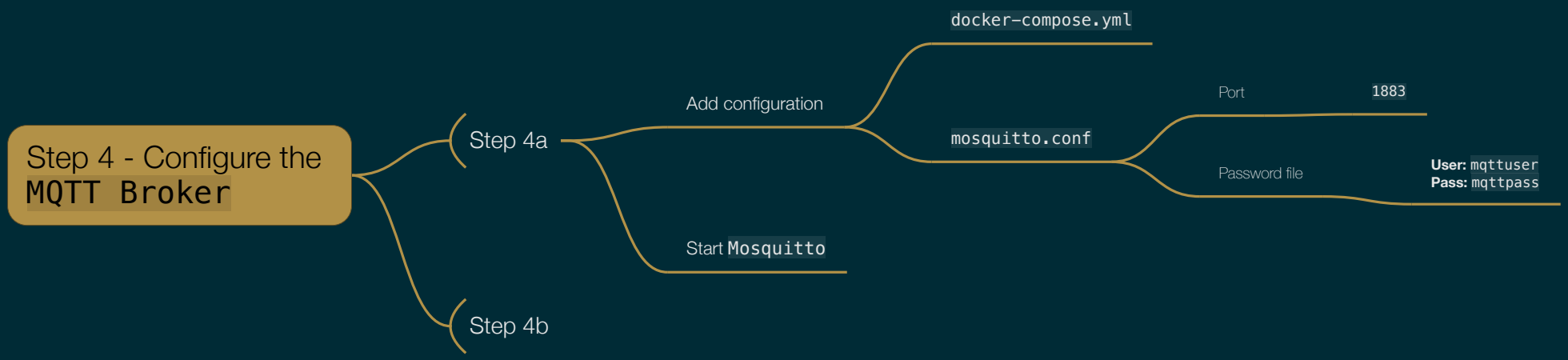
Step 3 - Network Architecture

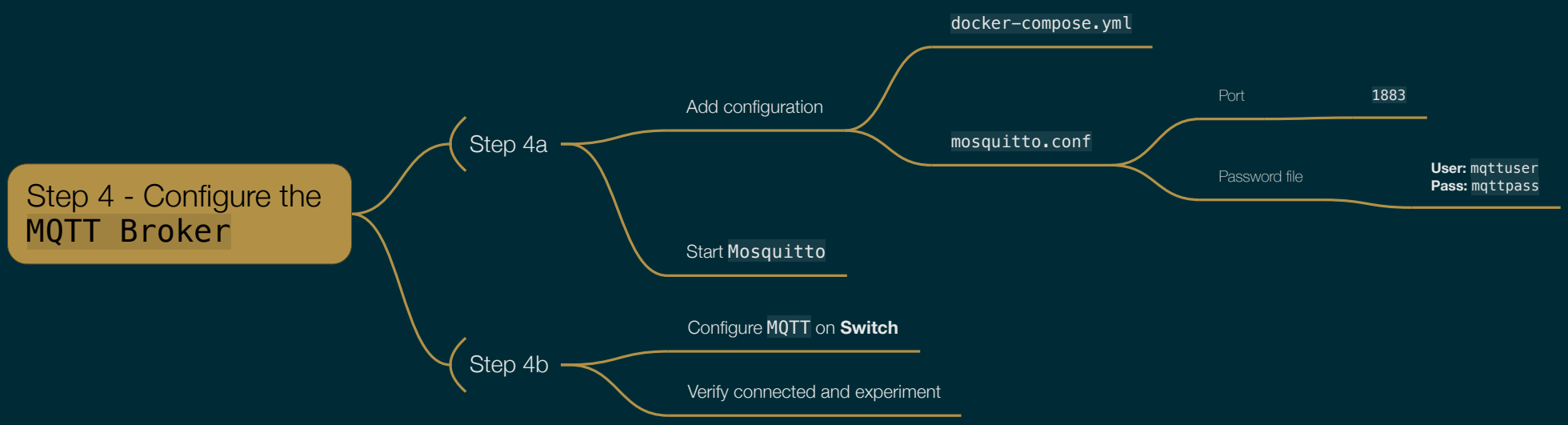
- 3 types of architecture (mainly)
  - Do drawing on board
- We’re in the middle one
  - Wifi device that needs a “Hub”, in our case a `MQTT Broker`

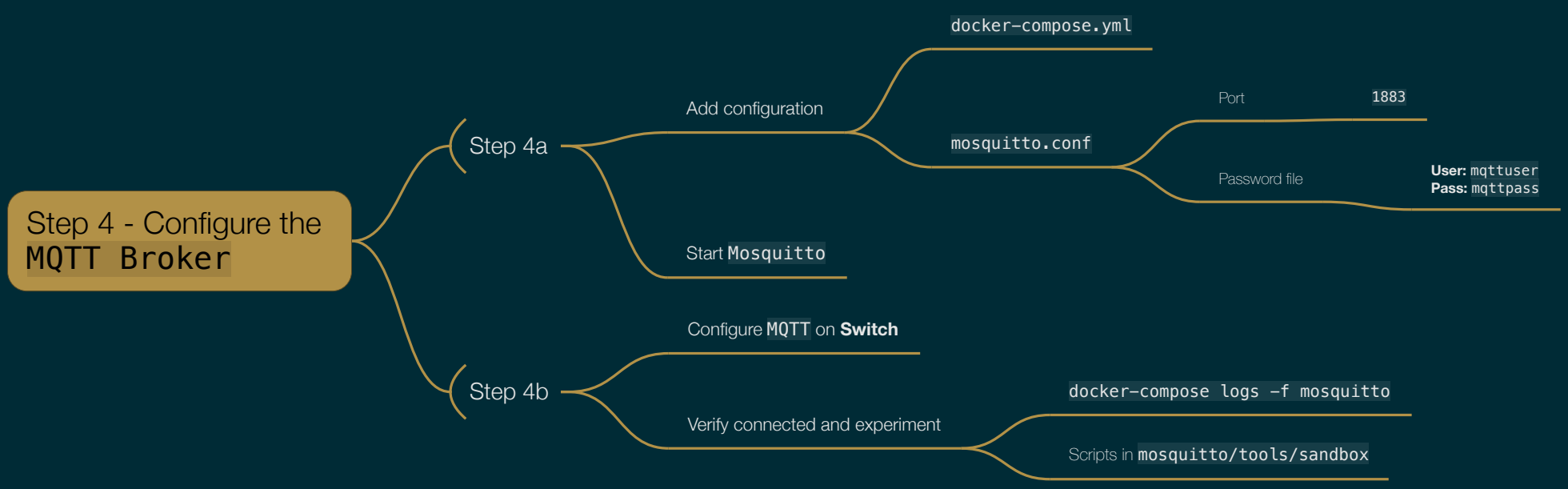
Step 4 - Configure the `MQTT Broker`

Step 4 - Configure the `MQTT Broker`

- Step 4a
  - Add configuration
  - Start `Mosquitto`
- Step 4b


Step 4 - Configure the MQTT Broker
Step 4a
Add configuration
docker-compose.yml
mosquitto.conf
Port
1883
Password file
User: mqttuser
Pass: mqttpass
Start Mosquitto
Step 4b


Step 4 - Configure the MQTT Broker
Step 4a
Add configuration
docker-compose.yml
mosquitto.conf
Port
1883
Password file
User: mqttuser
Pass: mqttpass
Start Mosquitto
Step 4b
Configure MQTT on Switch
Verify connected and experiment


Step 4 - Configure the MQTT Broker
Step 4a
Add configuration
docker-compose.yml
mosquitto.conf
Port
1883
Password file
User: mqttuser
Pass: mqttpass
Start Mosquitto
Step 4b
Configure MQTT on Switch
Verify connected and experiment
docker-compose logs -f mosquitto
Scripts in mosquitto/tools/sandbox

Step 5 - Integrate w/ Home Assistant

Step 5 - Integrate w/ Home Assistant

- Intro
  - We now have a 3rd party connected device to integrate
  - Every integration is different, but none are insurmountably complicated
    - Details for the `MQTT` integration are out of scope
    - `MQTT` integration is among the most complex ones
  - Look at the documentation to find the correct integration
- Step 5a - Manual
- Step 5b - Discovery

Step 5a - Manual

Step 5a - Manual

Using manual MQTT binding to explain quickly how it works under the hood

Hub Configuration:
We tell HomeAssistant how to access the `MQTT` Broker

Switch Configuration:
We tell **HomeAssistant** on which topic to listen to for the state of `DemoSwitch` and where to send commands to `DemoSwitch`

Experiment w/ `DemoSwitch` in **HomeAssistant** UI

Step 5a - Manual

Using manual MQTT binding to explain quickly how it works under the hood

Hub Configuration:
We tell HomeAssistant how to access the `MQTT` Broker

Switch Configuration:
We tell **HomeAssistant** on which topic to listen to for the state of `DemoSwitch` and where to send commands to `DemoSwitch`

Experiment w/ `DemoSwitch` in **HomeAssistant** UI

- Turn on/off
- Show the 2-way binding

Step 5b - Discovery

Step 5b - Discovery

In reality most services won't have you specify each and every device

There will most likely be some sort of discovery. Often implemented in the integration itself

For the `MQTT` integration, the option is simply called `discovery`

Step 5b - Discovery

In reality most services won't have you specify each and every device

There will most likely be some sort of discovery. Often implemented in the integration itself

For the `MQTT` integration, the option is simply called `discovery`

The device itself sends its own info on a special topic, and tells home assistant how to setup itself.

I already configured that ahead of times on the Switch

But since each platform is different, details are out of scope

Step 5 - Integrate w/ Home Assistant

- Intro
  - We now have a 3rd party connected device to integrate
  - Every integration is different, but none are insurmountably complicated
    - Details for the `MQTT` integration are out of scope
    - `MQTT` integration is among the most complex ones
  - Look at the documentation to find the correct integration
- Step 5a - Manual
  - Using manual MQTT binding to explain quickly how it works under the hood
  - **Hub Configuration:**
    We tell HomeAssistant how to access the `MQTT` Broker
  - **Switch Configuration:**
    We tell **HomeAssistant** on which topic to listen to for the state of `DemoSwitch` and where to send commands to `DemoSwitch`
  - Experiment w/ `DemoSwitch` in **HomeAssistant** UI
    - Turn on/off
    - Show the 2-way binding
- Step 5b - Discovery
  - In reality most services won't have you specify each and every device
    - There will most likely be some sort of discovery. Often implemented in the integration itself
  - For the `MQTT` integration, the option is simply called `discovery`
    - The device itself sends its own info on a special topic, and tells home assistant how to setup itself.
    - I already configured that ahead of times on the Switch
    - But since each platform is different, details are out of scope

Step 6 - Make some REST Api calls

Step 6 - Make some REST Api calls

- Generate Token
- Turn on and off

Step 6 - Make some REST Api calls

- Generate Token
  - Go to User profile
  - Long lived token
  - Paste in file `homeassistant/tools/sandbox/token`
- Turn on and off

Step 6 - Make some REST Api calls

- Generate Token
  - Go to User profile
  - Long lived token
  - Paste in file `homeassistant/tools/sandbox/token`
- Turn on and off
  - `homeassistant/tools/sandbox/turn_demoswitch_on.sh`
  - `homeassistant/tools/sandbox/turn_demoswitch_off.sh`

Step 7 - Setup **Appdaemon**


Step 7 - Setup
Appdaemon
Step 7a
Add config to docker-compose.yml
Step 7b
Step 7c

Step 7 - Setup **Appdaemon**

- Step 7a
  - Add config to `docker-compose.yml`
- Step 7b
  - Run a first time to generate **Appdaemon** config
- Step 7c

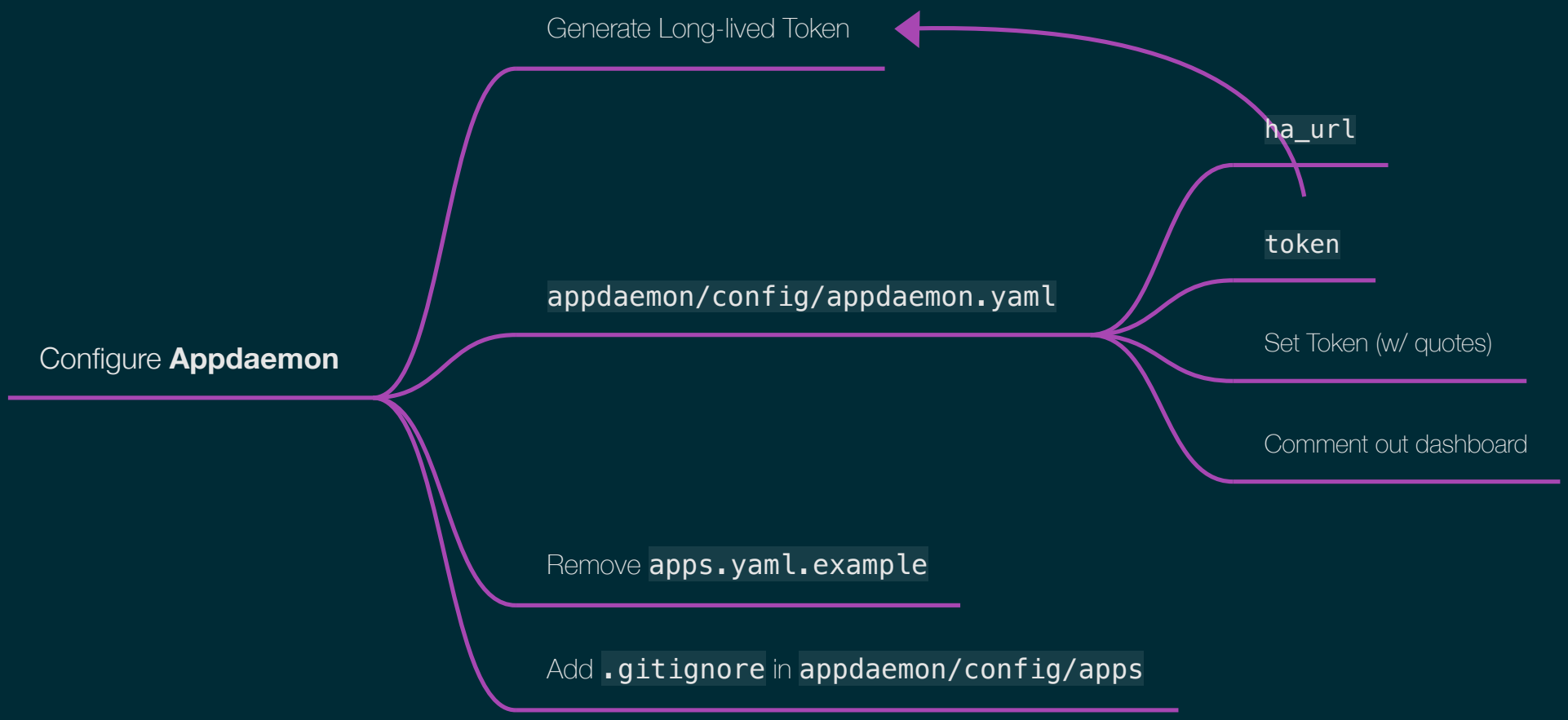
Step 7 - Setup **Appdaemon**

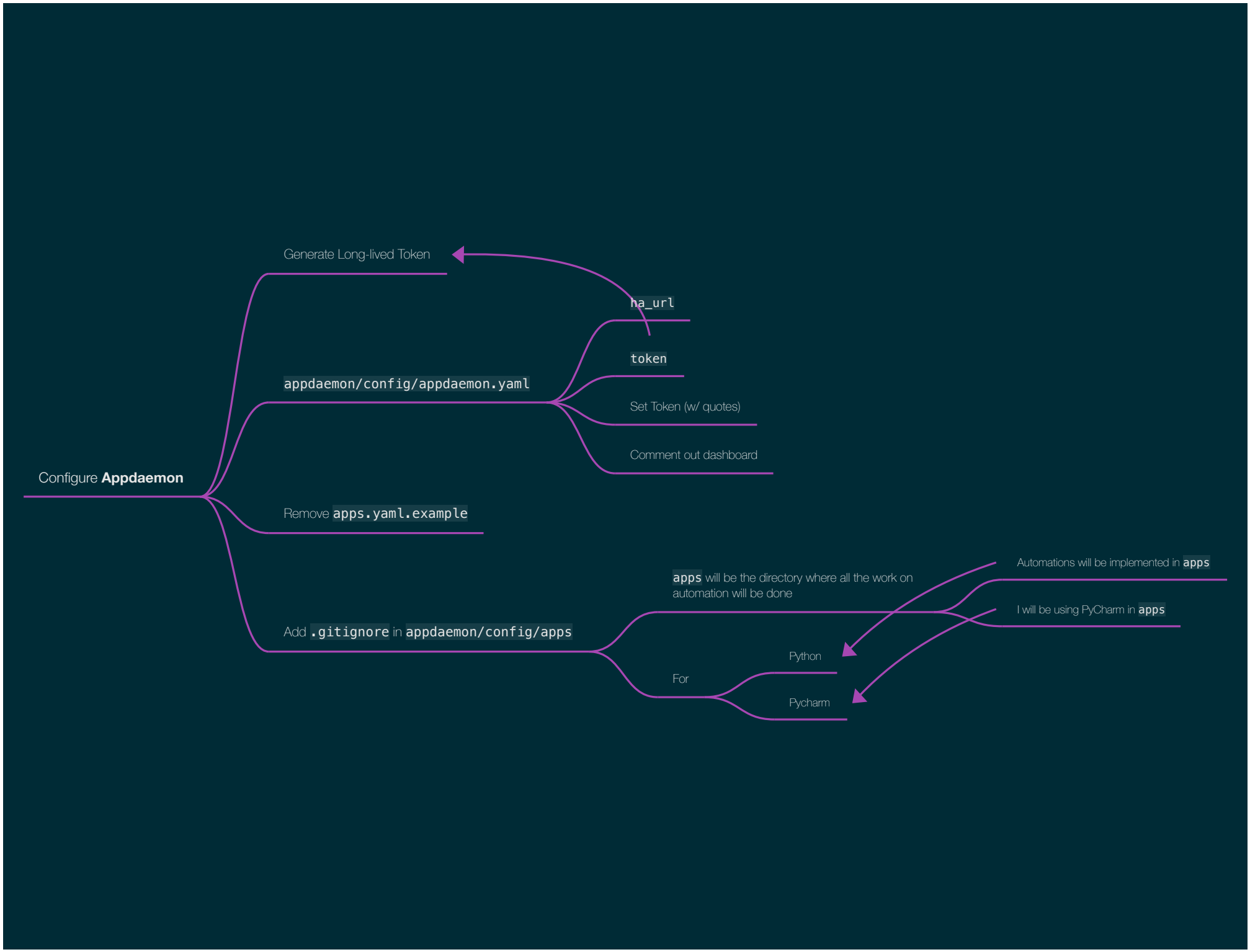
- Step 7a
  - Add config to `docker-compose.yml`
- Step 7b
  - Run a first time to generate **Appdaemon** config
- Step 7c
  - Configure **Appdaemon**
  - Explore `HelloWorld` automation in `apps`

Configure **Appdaemon**

Configure **Appdaemon**

- Generate Long-lived Token
- `appdaemon/config/appdaemon.yaml`
- Remove `apps.yaml.example`
- Add `.gitignore` in `appdaemon/config/apps`


Configure **Appdaemon**
Generate Long-lived Token
appdaemon/config/appdaemon.yaml
ha_url
token
Set Token (w/ quotes)
Comment out dashboard
Remove apps.yaml.example
Add .gitignore in appdaemon/config/apps

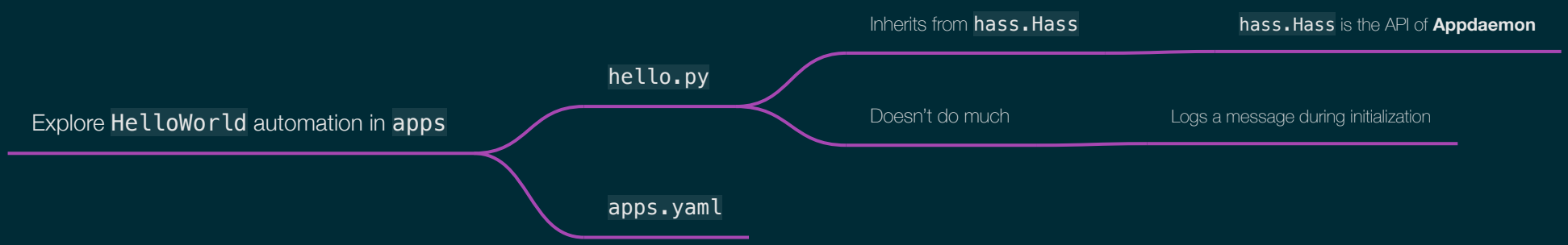

Configure **Appdaemon**
Generate Long-lived Token
`appdaemon/config/appdaemon.yaml`
`ha_url`
`token`
Set Token (w/ quotes)
Comment out dashboard
Remove `apps.yaml.example`
Add `.gitignore` in `appdaemon/config/apps`
`apps` will be the directory where all the work on automation will be done
Automations will be implemented in `apps`
I will be using PyCharm in `apps`
For
Python
Pycharm

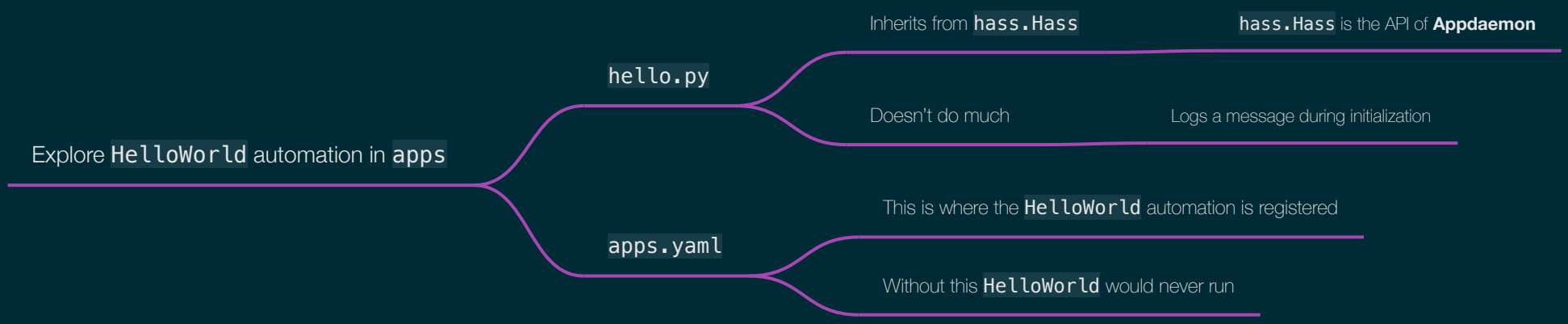
Explore `HelloWorld` automation in `apps`

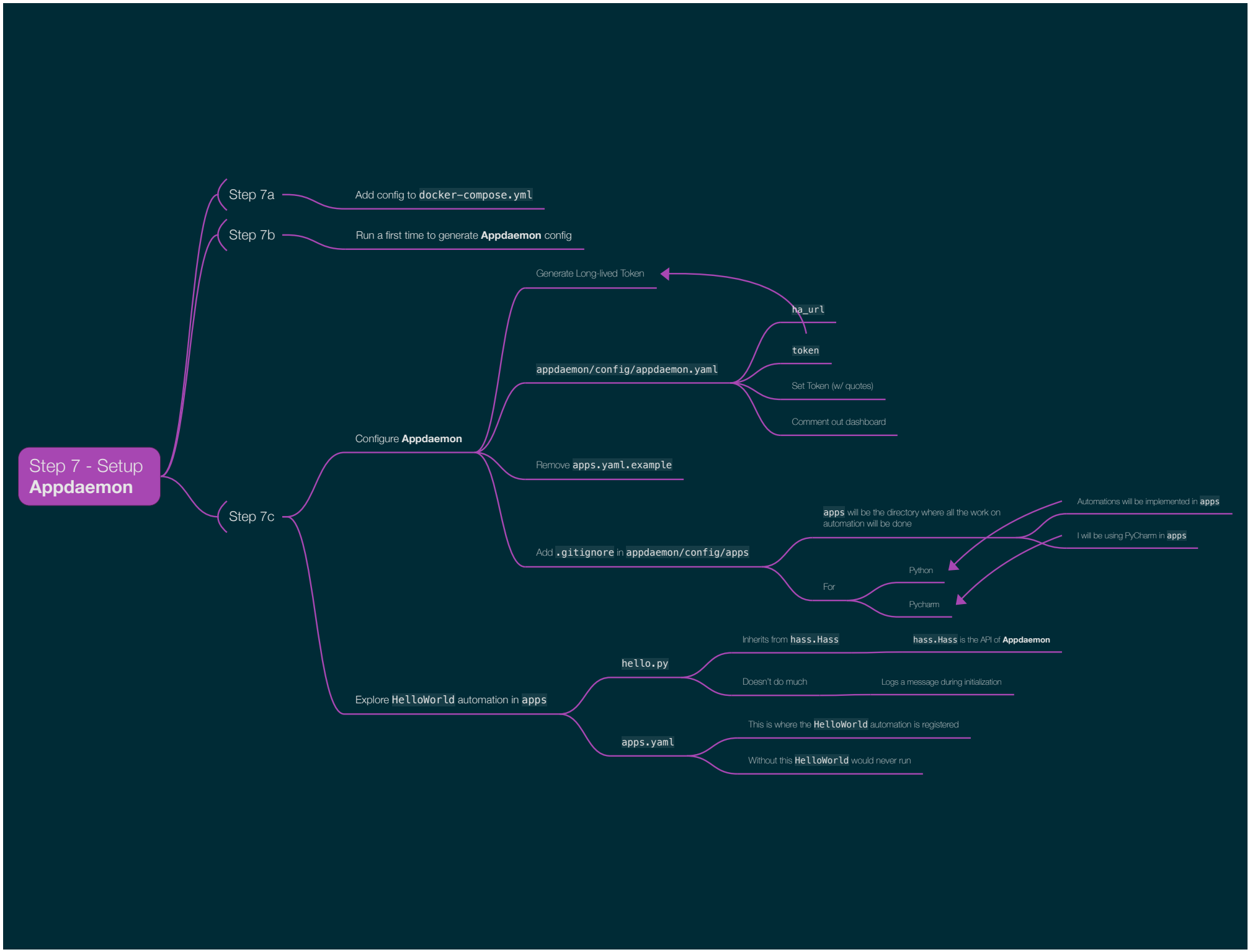
Explore `HelloWorld` automation in `apps`

`hello.py`

`apps.yaml`


Explore HelloWorld automation in apps
hello.py
apps.yaml
Inherits from hass.Hass
hass.Hass is the API of Appdaemon
Doesn't do much
Logs a message during initialization


Explore HelloWorld automation in apps
hello.py
Inherits from hass.Hass
hass.Hass is the API of Appdaemon
Doesn't do much
Logs a message during initialization
apps.yaml
This is where the HelloWorld automation is registered
Without this HelloWorld would never run


Step 7 - Setup Appdaemon
Step 7a
Add config to docker-compose.yml
Step 7b
Run a first time to generate Appdaemon config
Step 7c
Configure Appdaemon
Generate Long-lived Token
appdaemon/config/appdaemon.yaml
ha_url
token
Set Token (w/ quotes)
Comment out dashboard
Remove apps.yaml.example
Add .gitignore in appdaemon/config/apps
apps will be the directory where all the work on automation will be done
Automations will be implemented in apps
I will be using PyCharm in apps
For
Python
Pycharm
Explore HelloWorld automation in apps
hello.py
Inherits from hass.Hass
hass.Hass is the API of Appdaemon
Doesn't do much
Logs a message during initialization
apps.yaml
This is where the HelloWorld automation is registered
Without this HelloWorld would never run

Step 8 - Basic `LogText` Automation

Step 8 - Basic `LogText` Automation

- Goal
  - Simple automation that logs any text entered in a text field on the **HomeAssistant** Front-end
- Step 8a
- Step 8b
- Step 8c

Step 8 - Basic `LogText` Automation

- Goal
  - Simple automation that logs any text entered in a text field on the **HomeAssistant** Front-end
- Step 8a
  - Setup working environment
    - Install deps with `pipenv`
      - `appdaemon`
    - Create PyCharm project
- Step 8b
- Step 8c

Step 8 - Basic `LogText` Automation

- Goal
  - Simple automation that logs any text entered in a text field on the **HomeAssistant** Front-end
- Step 8a
  - Setup working environment
    - Install deps with `pipenv`
      - `appdaemon`
    - Create PyCharm project
- Step 8b
  - Create `input_text` in **HomeAssistant**
    - **HomeAssistant** does not only allow to represent *physical* entities, it also allow to create *software-only* entities as well
  - Show how **HomeAssistant** is a big database
    - Explore state of `input_text.my_text`
- Step 8c

Step 8 - Basic `LogText` Automation

- Goal
  - Simple automation that logs any text entered in a text field on the **HomeAssistant** Front-end
- Step 8a
  - Setup working environment
    - Install deps with `pipenv`
      - `appdaemon`
    - Create PyCharm project
- Step 8b
  - Create `input_text` in **HomeAssistant**
    - **HomeAssistant** does not only allow to represent *physical* entities, it also allow to create *software-only* entities as well
  - Show how **HomeAssistant** is a big database
    - Explore state of `input_text.my_text`
- Step 8c
  - Implement `LogText`

Implement `LogText`

Implement `LogText`

`listen_state`

appdaemon.readthedocs.io/en/latest/AD_API_REFERENCE.html#listen-state

Receives a *callback* that will be executed whenever the state of the watched **HomeAssistant** `entity` changes

Implement `LogText`

`listen_state`

appdaemon.readthedocs.io/en/latest/AD_API_REFERENCE.html#listen_state

Receives a *callback* that will be executed whenever the state of the watched **HomeAssistant** `entity` changes

The callback has a particular format

appdaemon.readthedocs.io/en/latest/APPGUIDE.html#state-callbacks

```
def my_callback(self, entity, attribute, old, new, kwargs):
  <do some useful work here>
```

Step 8 - Basic `LogText` Automation

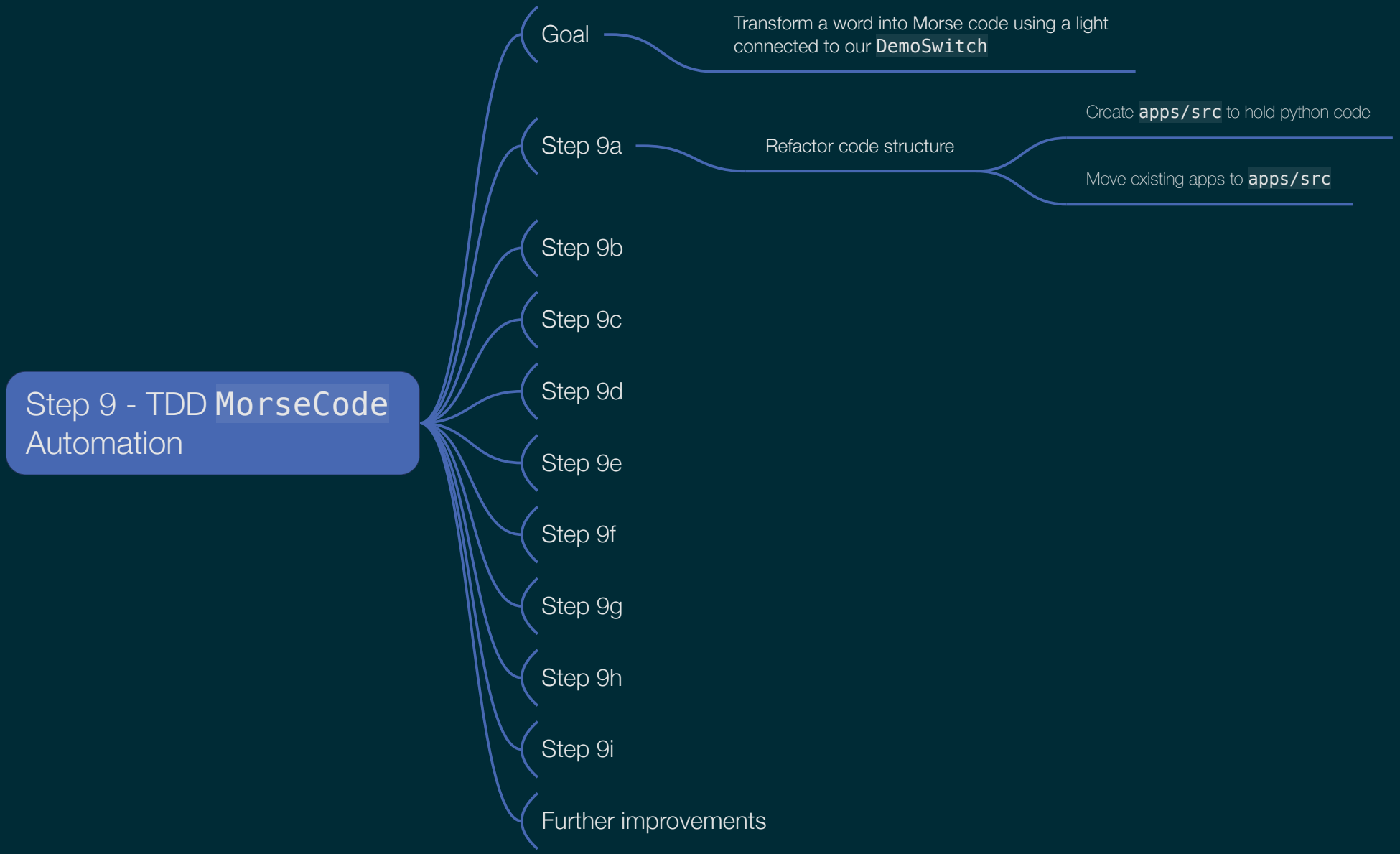
- Goal
  - Simple automation that logs any text entered in a text field on the **HomeAssistant** Front-end
- Step 8a
  - Setup working environment
    - Install deps with `pipenv`
      - `appdaemon`
    - Create PyCharm project
- Step 8b
  - Create `input_text` in **HomeAssistant**
    - **HomeAssistant** does not only allow to represent *physical* entities, it also allow to create *software-only* entities as well
  - Show how **HomeAssistant** is a big database
    - Explore state of `input_text.my_text`
- Step 8c
  - Implement `LogText`
    - `listen_state`
      - appdaemon.readthedocs.io/en/latest/AD_API_REFERENCE.html#listen_state
    - Receives a *callback* that will be executed whenever the state of the watched **HomeAssistant** `entity` changes
      - The callback has a particular format
        - appdaemon.readthedocs.io/en/latest/APIGUIDE.html#state-callbacks
      -

```
def my_callback(self, entity, attribute, old, new, kwargs):
  <do some useful work here>
```

Step 9 - TDD `MorseCode` Automation

Step 9 - TDD `MorseCode` Automation

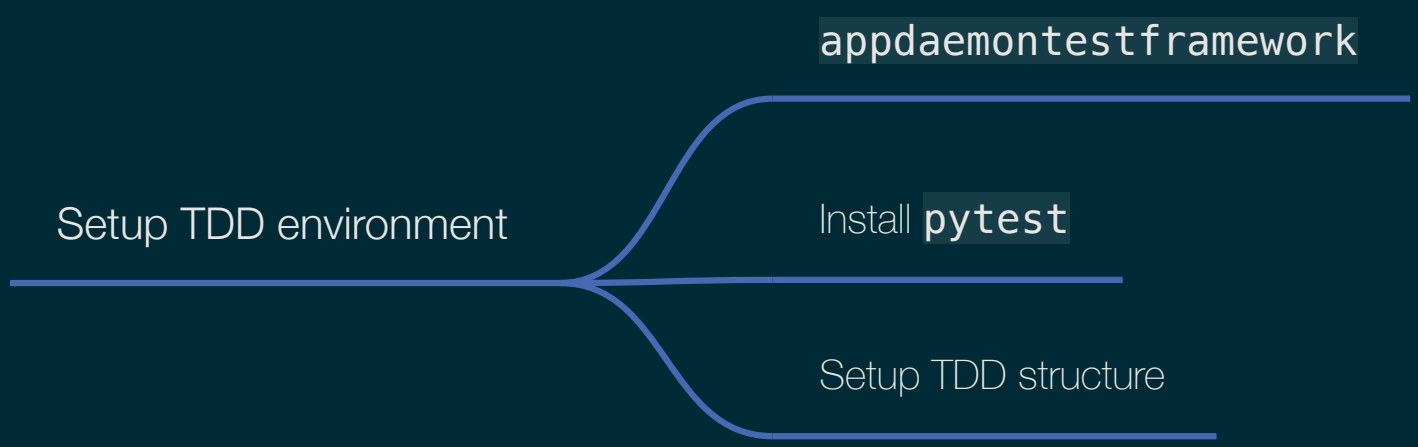
- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
- Step 9b
- Step 9c
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements


Step 9 - TDD MorseCode Automation
Goal
Transform a word into Morse code using a light connected to our DemoSwitch
Step 9a
Refactor code structure
Create apps/src to hold python code
Move existing apps to apps/src
Step 9b
Step 9c
Step 9d
Step 9e
Step 9f
Step 9g
Step 9h
Step 9i
Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
    - Create `apps/src` to hold python code
    - Move existing apps to `apps/src`
- Step 9b
  - Setup TDD environment
- Step 9c
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

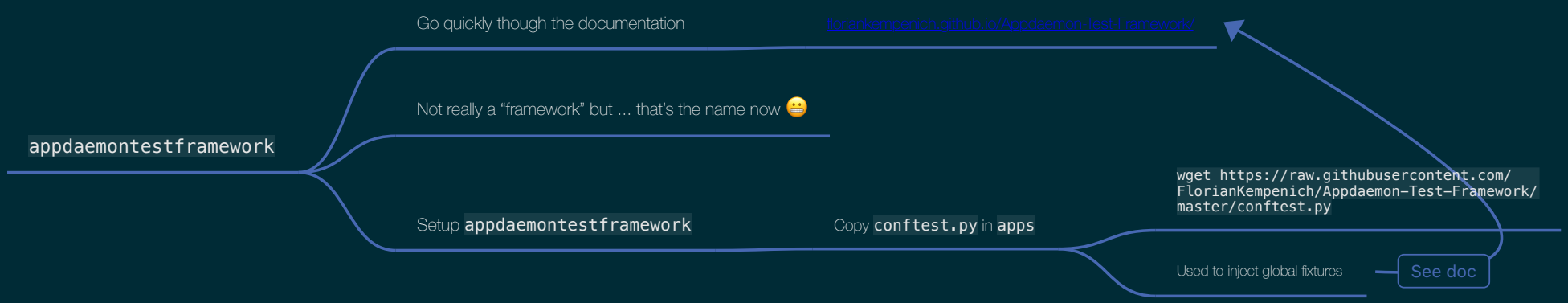
Setup TDD environment

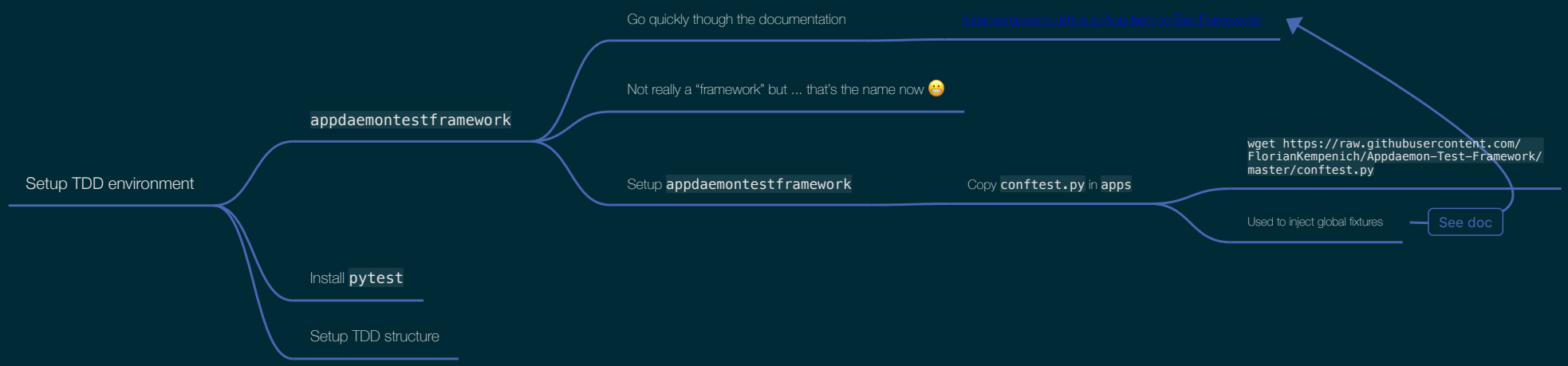

Setup TDD environment
appdaemontestframework
Install pytest
Setup TDD structure

appdaemontestframework

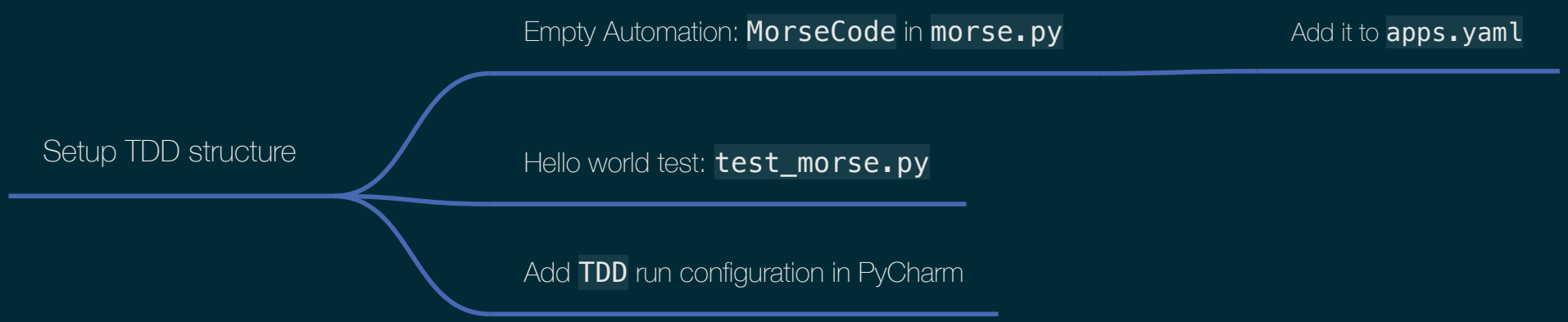
`appdaemontestframework`

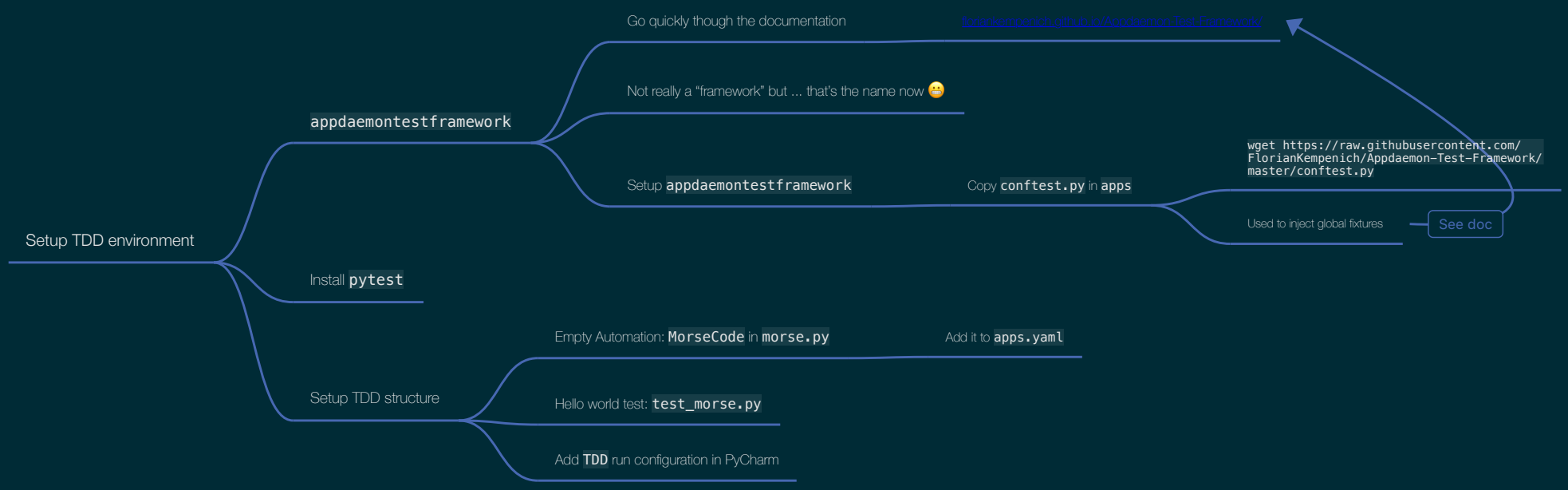
- Go quickly though the documentation
  - florianKempenich.github.io/Appdaemon-Test-Framework/
- Not really a “framework” but … that’s the name now 😬
- Setup `appdaemontestframework`

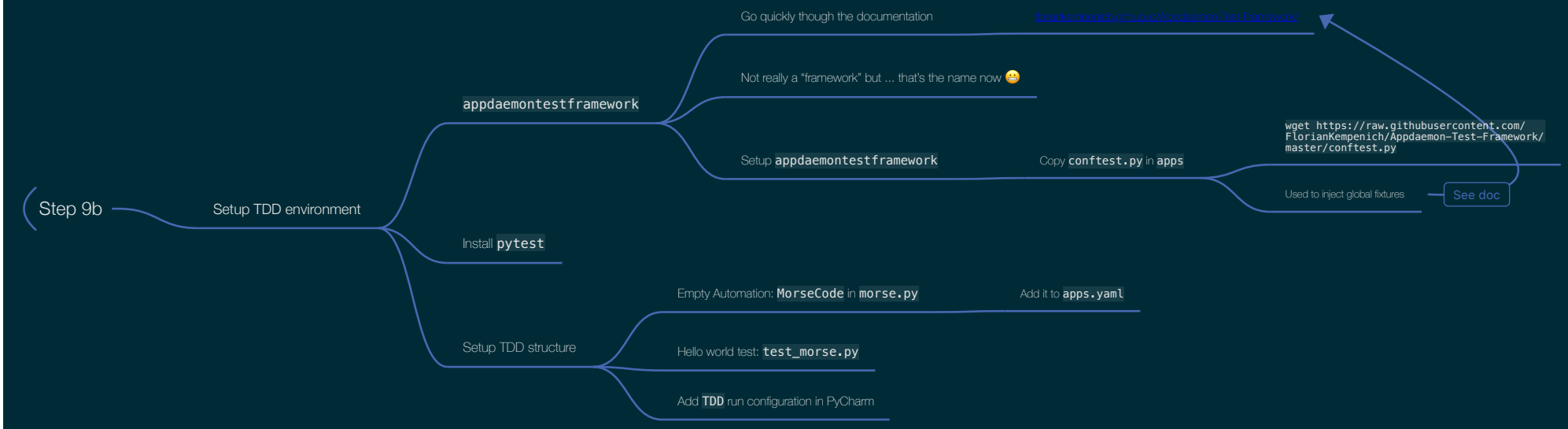

appdaemontestframework
Go quickly though the documentation
florankempenich.github.io/Appdaemon-Test-Framework/
Not really a “framework” but … that’s the name now 😐
Setup appdaemontestframework
Copy conftest.py in apps
wget https://raw.githubusercontent.com/FlorianKempenich/Appdaemon-Test-Framework/master/conftest.py
Used to inject global fixtures
See doc


Setup TDD environment
appdaemontestframework
Go quickly though the documentation
floriankempenich.github.io/Appdaemon-Test-Framework/
Not really a “framework” but ... that’s the name now 😬
Setup appdaemontestframework
Copy conftest.py in apps
wget https://raw.githubusercontent.com/FlorianKempenich/Appdaemon-Test-Framework/master/conftest.py
Used to inject global fixtures
See doc
Install pytest
Setup TDD structure

Setup TDD structure


Setup TDD structure
Empty Automation: MorseCode in morse.py
Add it to apps.yaml
Hello world test: test_morse.py
Add TDD run configuration in PyCharm


Setup TDD environment
appdaemontestframework
Go quickly though the documentation
florankempenich.github.io/Appdaemon-Test-Framework/
Not really a "framework" but ... that's the name now 😬
Setup appdaemontestframework
Copy conftest.py in apps
wget https://raw.githubusercontent.com/FlorianKempenich/Appdaemon-Test-Framework/master/conftest.py
Used to inject global fixtures
See doc
Install pytest
Setup TDD structure
Empty Automation: MorseCode in morse.py
Add it to apps.yaml
Hello world test: test_morse.py
Add TDD run configuration in PyCharm


Step 9b
Setup TDD environment
appdaemontestframework
Go quickly though the documentation
floriankempenich.github.io/Appdaemon-Test-Framework/
Not really a "framework" but ... that's the name now 😀
Setup appdaemontestframework
Copy conftest.py in apps
wget https://raw.githubusercontent.com/FlorianKempenich/Appdaemon-Test-Framework/master/conftest.py
Used to inject global fixtures
See doc
Install pytest
Setup TDD structure
Empty Automation: MorseCode in morse.py
Add it to apps.yaml
Hello world test: test_morse.py
Add TDD run configuration in PyCharm

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
    - Create `apps/src` to hold python code
    - Move existing apps to `apps/src`
- Step 9b
  - Setup TDD environment
    - `appdaemontestframework`
      - Go quickly though the documentation
        - https://floriankempenich.github.io/Appdaemon-Test-Framework/
      - Not really a "framework" but ... that's the name now 😕
      - Setup `appdaemontestframework`
        - Copy `conftest.py` in `apps`
          - `wget https://raw.githubusercontent.com/FlorianKempenich/Appdaemon-Test-Framework/master/conftest.py`
          - Used to inject global fixtures
            - See doc
    - Install `pytest`
    - Setup TDD structure
      - Empty Automation: `MorseCode` in `morse.py`
        - Add it to `apps.yaml`
      - Hello world test: `test_morse.py`
      - Add `TDD` run configuration in PyCharm
- Step 9c
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9c

Create multiple `input_number` to hold parameters for our `MorseCode` automation

Create `group.morse`

Step 9c

- Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - `input_number.morse_short`
  - `input_number.morse_long`
  - `input_number.morse_interval_symbols`
  - `input_number.morse_interval_letters`
  - Can not be below 1s, **Appdaemon** is only accurate to `± 1 second`
- Create `group.morse`

- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
    - `input_number.morse_short`
    - `input_number.morse_long`
    - `input_number.morse_interval_symbols`
      - Interval between 2 symbols in the same letter
    - `input_number.morse_interval_letters`
      - Interval between 2 letters
      - No parameter for Interval between 2 words because our `MorseCode` automation only supports 1 word
    - Can not be below 1s, **Appdaemon** is only accurate to `± 1 second`
  - Create `group.morse`

- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
    - `input_number.morse_short`
    - `input_number.morse_long`
    - `input_number.morse_interval_symbols`
      - Interval between 2 symbols in the same letter
    - `input_number.morse_interval_letters`
      - Interval between 2 letters
      - No parameter for Interval between 2 words because our `MorseCode` automation only supports 1 word
    - Can not be below 1s, **Appdaemon** is only accurate to `± 1 second`
      - Which is **perfectly fine** for 99.9999424% of Automations
      - I made up that number
  - Create `group.morse`

- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
    - `input_number.morse_short`
    - `input_number.morse_long`
    - `input_number.morse_interval_symbols`
      - Interval between 2 symbols in the same letter
    - `input_number.morse_interval_letters`
      - Interval between 2 letters
      - No parameter for Interval between 2 words because our `MorseCode` automation only supports 1 word
    - Can not be below 1s, **Appdaemon** is only accurate to `± 1 second`
      - Which is **perfectly fine** for 99.9999424% of Automations
      - I made up that number
  - Create `group.morse`
    - Holds all entities related to our `MorseCode` automation

Step 9c

Create multiple `input_number` to hold parameters for our `MorseCode` automation

Create `group.morse`

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9d

Ensure that we listen to the state

Step 9d

Ensure that we listen to the state

Explain `@automation_fixture` and `assert_that(...).listen_to...`

```
assert_that(morse_code)\
    .listens_to.state('input_text.morse')\
    .with_callback(morse_code.on_new_text)
```

From that point on we can call `morse_code.on_new_text` in our tests. We know it is correctly registered as a listener.

Step 9d

Ensure that we listen to the state

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9e

Test short letter

Step 9e

Test short letter

Explain `pytest` fixtures

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
    - The tests are starting to get a little verbose, so let's refactor a bit to prepare for further tests
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
- Step 9h
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
- Step 9h
  - Test composite letter
- Step 9i
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
- Step 9h
  - Test composite letter
- Step 9i
  - Test full word
- Further improvements

Test full word

Test full word

Bug in previous code detected and fixed

Test full word

Bug in previous code detected and fixed

Before we were adding a `interval_symbol` duration after every symbols, even the last one.

But that would stack with the `interval_letters` duration, and the morse code would end up with incorrect duration spacing between letters

Our test caught it, and it was fixed

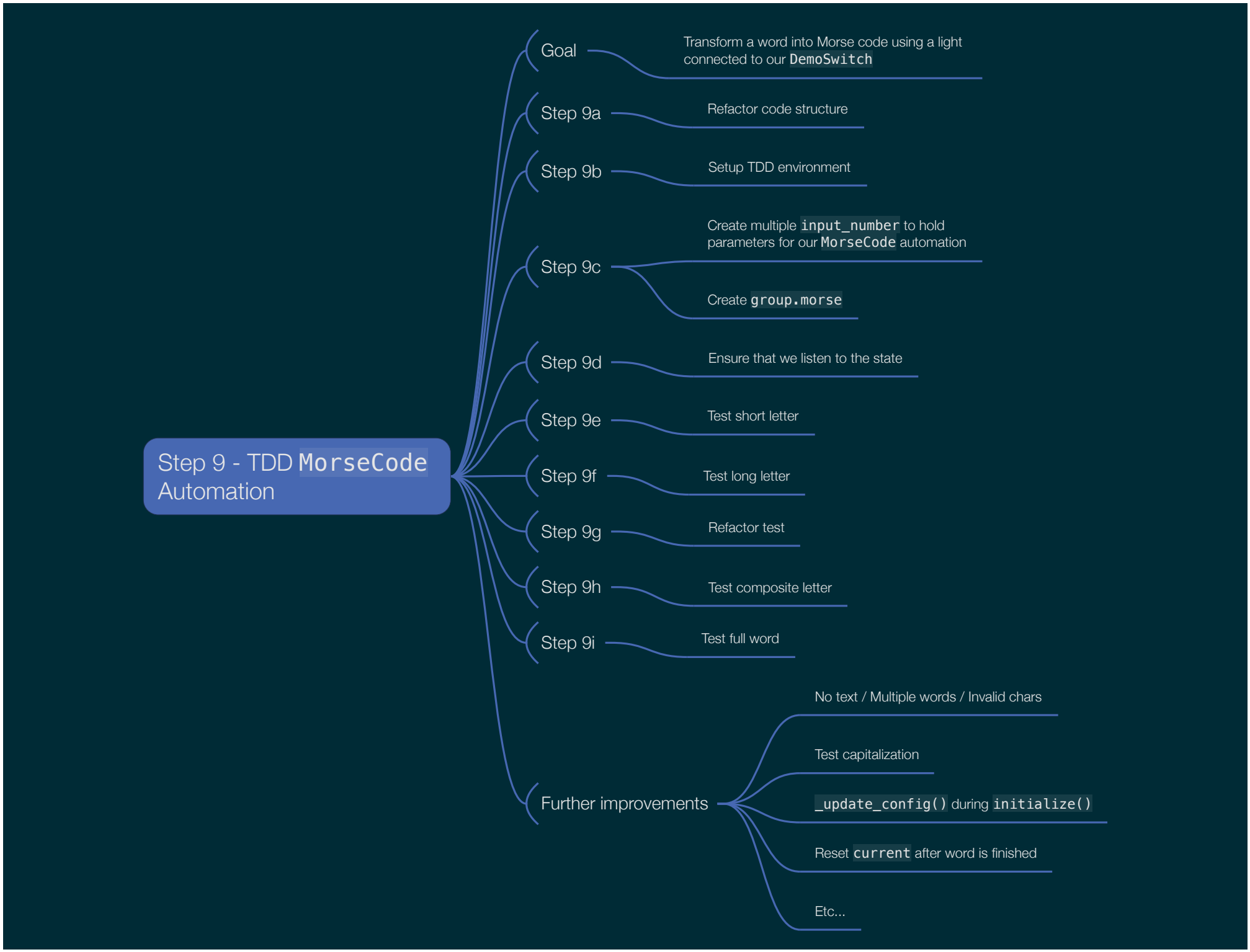
Thank you TDD 🙏😍

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
- Step 9h
  - Test composite letter
- Step 9i
  - Test full word
    - Bug in previous code detected and fixed
      - Before we were adding a `interval_symbol` duration after every symbols, even the last one.
      - But that would stack with the `interval_letters` duration, and the morse code would end up with incorrect duration spacing between letters
      - Our test caught it, and it was fixed
      - Thank you TDD 🙏😍
- Further improvements

Step 9 - TDD `MorseCode` Automation

- Goal
  - Transform a word into Morse code using a light connected to our `DemoSwitch`
- Step 9a
  - Refactor code structure
- Step 9b
  - Setup TDD environment
- Step 9c
  - Create multiple `input_number` to hold parameters for our `MorseCode` automation
  - Create `group.morse`
- Step 9d
  - Ensure that we listen to the state
- Step 9e
  - Test short letter
- Step 9f
  - Test long letter
- Step 9g
  - Refactor test
- Step 9h
  - Test composite letter
- Step 9i
  - Test full word
- Further improvements


Step 9 - TDD MorseCode Automation
Goal
Transform a word into Morse code using a light connected to our DemoSwitch
Step 9a
Refactor code structure
Step 9b
Setup TDD environment
Step 9c
Create multiple input_number to hold parameters for our MorseCode automation
Create group.morse
Step 9d
Ensure that we listen to the state
Step 9e
Test short letter
Step 9f
Test long letter
Step 9g
Refactor test
Step 9h
Test composite letter
Step 9i
Test full word
Further improvements
No text / Multiple words / Invalid chars
Test capitalization
_update_config() during initialize()
Reset current after word is finished
Etc...

Step 10 - A Complete Real-life setup

Step 10 - A Complete Real-life setup

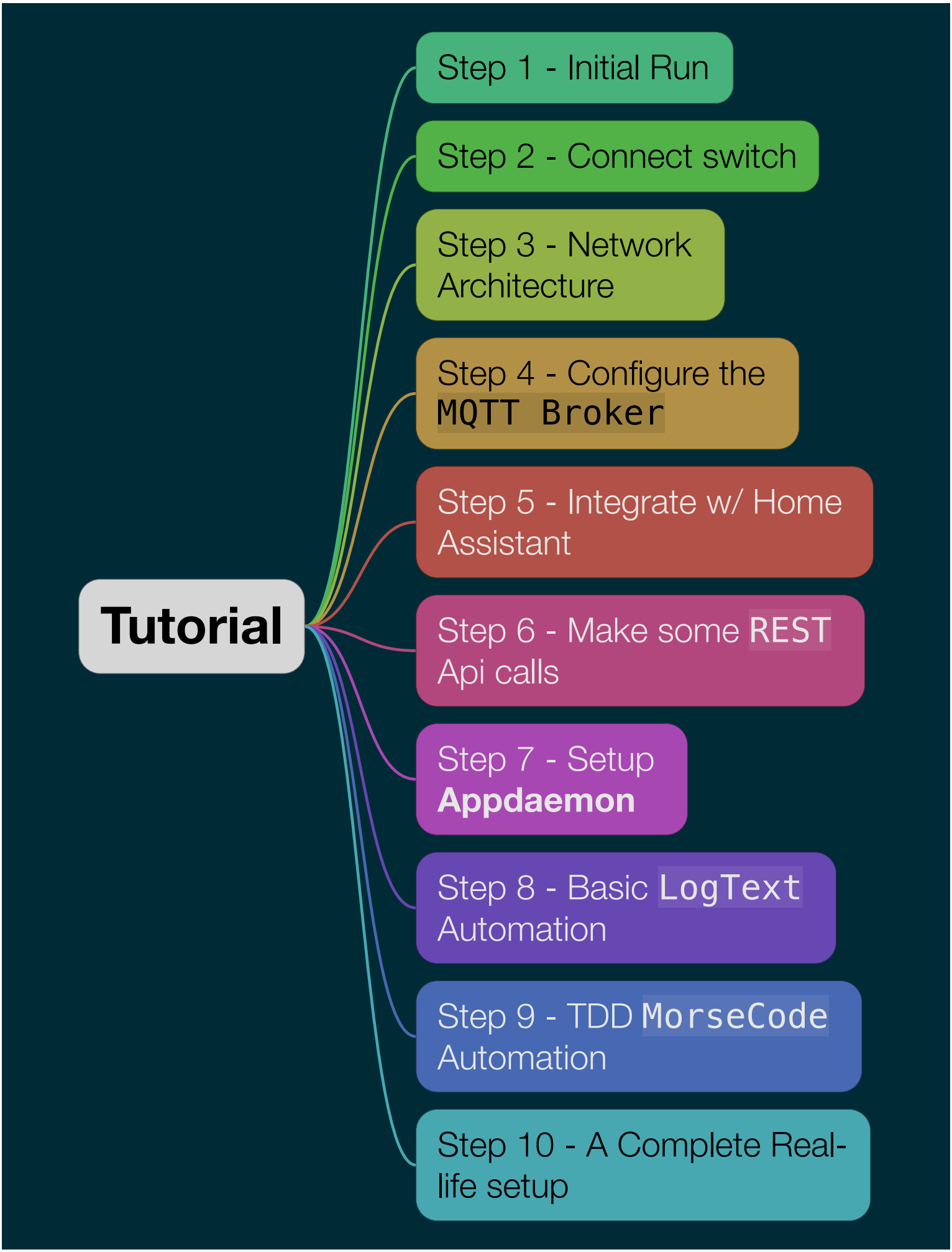
- Introduce Codebase
  - gitlab.com/FlorianKempenich/dadhome-public/tree/master
- Talk, and show, some of possible automations
- Give temporary visitor access to `DadHome`

Step 10 - A Complete Real-life setup

- Introduce Codebase
  - gitlab.com/FlorianKempenich/dadhome-public/tree/master
- Talk, and show, some of possible automations
  - Simple automated lights
  - Bedroom state machine
  - Tree light at sunset
  - Home Theater / Tree light
  - And more...
    - Window opened alert on smartphone when about to rain
    - Window opened alert on smartphone when leaving home
- Give temporary visitor access to `DadHome`

Tutorial

- Step 1 - Initial Run
  - Step 1a
    - Prepare files
      - `docker-compose.yml` with config for **Home Assistant** only
      - `.gitignore`
  - Step 1b
    - Start
    - Go through basic config
      - Name / Use / Pass
        - Finish framework
        - Installation
        - Lists
      - Home Location
        - Unit
        - Timezone
        - Will show how to exercise later (formatting / constants)
      - Discovery
        - Do not setup anything for now
    - Explore UI
      - `User logs`
      - Main UI
    - Explore generated Files
      - `.HA_VERSION`
        - Current version of HomeAssistant
        - Keep under version control. Useful when updating docker image to make sure of which version was just ran. Useful rolling back a error state
      - `.storage/`
        - Informations related to the state of the system
        - Do not replace that lists
        - Stores checked by our under version control, some not. Just ease dev only
      - Configuration
        - **The Most Important File**
        - `configuration.yaml`
          - `default_config`
            - https://github.com/home-assistant/core/tree/dev/homeassistant/components/default_config
          - `base_url`
            - Usefull when trying the access externally/remotely
            - Use at notify for **Empty files**
          - `group automation script`
            - **Groups**: Instance of entities
            - **Automations**: Rules to trigger actions and actions. **NOT LOGIC**
            - **Scripts**: Sequences of actions in HomeAssistant
          - `secrets`
            - Any file and mapping for secrets
            - `!secret my_secret`
  - Step 1c
    - Remove unused `ttl` from `homeassistant/configuration.yaml`
      - Unused in the demo
- Step 2 - Connect switch
  - For this particular demo will use Wifi Connected Switch
    - Network
      - Phone acts as Router
      - Switch + Phone
      - MacBook + Phone
      - One local network
    - (Pre) Configuration - Sonoff
      - To turn a Sonoff switch into a connected switch usable with Home Assistant is quite a complex task and is likely out of scope
      - For more info:
        - https://tasmota.github.io/docs/Quick-Installation-Guide-Part-1/#flashing-tasmota-for-the-first-time
        - https://tasmota.github.io/docs/Quick-Installation-Guide-Part-2/#configuration-of-tasmota
        - might change location in the future
  - Find IP address
    - `find-device-on-local-network`
  - Connect to Switch and demo Web UI
    - This is the connected interface for the Switch
    - It has nothing to do with HomeAssistant at this point
    - Any connected device will have some way to access it
    - Not a standard thing in particular. Every Wifi UI. Other might have an app
  - We need an API
    - To integrate w/ HomeAssistant we need some way to programmatically access the device
    - Most services services offer a REST api (ie. Philips Hue)
    - But this sensor does to publish/subscribe to a **MQTT** topic
- Step 3 - Network Architecture
  - 3 types of architecture (mainly)
    - Do drawing on board
  - We're in the middle one
    - Still device that needs a "hub", in our case a `MQTT Broker`
- Step 4 - Configure the `MQTT Broker`
  - Step 4a
    - Add configuration
      - `docker-compose.yml`
      - `mosquitto.conf`
        - Port
          - `1883`
        - Password file
          - **User** `mqttuser`
          - **Pass** `mqttpass`
    - Start `Mosquitto`
  - Step 4b
    - Configure `MQTT` on **Switch**
    - Verify connected and experienced
      - `docker-compose logs -f mosquitto`
      - Errors in `mosquitto/data/mosquitto.log`
- Step 5 - Integrate w/ Home Assistant
  - Intro
    - We now have a 3rd party connected device to integrate
    - Every integration is different, but none are unsurmountably complicated
      - Details for the `MQTT` integration are out of scope
      - `MQTT` integration is among the most complex ones
    - Look at the documentation to find the correct integration
  - Step 5a - Manual
    - Using manual MQTT binding to explain quickly how it works under the hood
    - **Hub Configuration:**
      We tell HomeAssistant how to access the `MQTT` Broker
    - **Switch Configuration:**
      We tell **HomeAssistant** on which topic to listen to for updates and on which topic it has to send commands to `Shelly1LD`
    - Experiment w/ `Shelly1LD` in **HomeAssistant UI**
      - Turn on/off
      - Show the 2-way binding
  - Step 5b - Discovery
    - In reality most remote won't have you specify each and every device
      - There will most likely be some sort of discovery. Often implemented in the integration itself
    - For the `MQTT` integration, the option is simply `mqtt:discovery`
      - The device itself sends its own info on a special topic, and lets home assistant hook to setup itself.
      - I already configured that ahead of time on the Switch
      - But since each platform is different, details are out of scope
- Step 6 - Make some `REST` Api calls
  - Generate Token
    - Go to User profile
    - Long lived token
    - Place in file
      - `homeassistant/tools/sandbox/token`
  - Turn on and off
    - `homeassistant/tools/sandbox/turn_demoswitch_on.sh`
    - `homeassistant/tools/sandbox/turn_demoswitch_off.sh`
- Step 7 - Setup **Appdaemon**
  - Step 7a
    - Add config to `docker-compose.yml`
  - Step 7b
    - Run a first time to generate **Appdaemon** config
  - Step 7c
    - Configure **Appdaemon**
      - `appdaemon/config/appdaemon.yaml`
        - Generate Long-lived Token
        - `secret`
          - `token`
        - Set Timezone not quotes
        - Comment out dashboard
      - Remove `apps.yaml.example`
      - Add `.gitignore` - `appdaemon/config/apps`
        - `apps` will be the directory where all the config and automation will live once
        - Automation will be implemented in `apps`
        - Call the config/ `Python` `apps`
        - Yes
          - Python
          - Pyscript
        - Check in the `hass.zkms` `hass.theory` file, the API - **Appdaemon**
      - Explore `HelloWorld` automation in `apps`
        - `hello.py`
          - Doesn't do much
          - Logs a message during initialization
        - `apps.yaml`
          - This is where the `HelloWorld` automation is registered
          - Without the `HelloWorld` would never run
- Step 8 - Basic `LogText` Automation
  - Goal
    - Simple automation that logs any text entered in a text field on the **HomeAssistant** front-end
  - Step 8a
    - Setup working environment
      - Install deps with `pipenv`
        - **appdaemon**
      - Create PyCharm project
  - Step 8b
    - Create `input_text` in **HomeAssistant**
      - **HomeAssistant** state not only allow to represent physical entities. It also allow to create software-only entities as well
    - Show how **HomeAssistant** is a big database
      - Explore state of `input_text.my_text`
  - Step 8c
    - Implement `LogText`
      - `listen_state`
        - https://www.home-assistant.io/docs/automation/trigger/#state-trigger
        - https://www.home-assistant.io/docs/automation/trigger/#state-trigger-examples
      - Receives a callback that will be executed whenever the state of the watched **HomeAssistant** entity changes
        - The callback has a particular format
        - https://www.home-assistant.io/docs/automation/trigger/#state-trigger-examples
        - `def my_callback(self, entity, attribute, old, new, kwargs):`
          `self.log("Hello! Switch state changed.")`
- Step 9 - TDD `MorseCode` Automation
  - Goal
    - Transform a word into Morse code using a light connected to our `Shelly1LD`
  - Step 9a
    - Refactor code structure
  - Step 9b
    - Setup TDD environment
  - Step 9c
    - Create multiple `input_number` to hold parameters for our `MorseCode` automation
    - Create `group.morse`
  - Step 9d
    - Ensure that we listen to the state
  - Step 9e
    - Test short letter
  - Step 9f
    - Test long letter
  - Step 9g
    - Refactor test
  - Step 9h
    - Test composite letter
  - Step 9i
    - Test full word
  - Further improvements
    - No text / Multiple words / Invalid chars
    - Test capitalization
    - `_update_config()` during `initialize()`
    - Reset `current` after word is finished
    - Etc...
- Step 10 - A Complete Real-life setup
  - Introduce Codebase
    - https://github.com/ivan-marcin/home-assistant-codebase
  - Talk, and show, some of possible automations
    - Simple autonomous lights
    - Bedroom state machine
    - Tree light at sunset
    - Home Theater / Tree light
    - And more...
      - Window opened alert on airpot/pure when about to rain
      - Window opened alert on airpot/pure when leaving home
  - Give temporary visitor access to **Dashfone**


Tutorial
Step 1 - Initial Run
Step 2 - Connect switch
Step 3 - Network Architecture
Step 4 - Configure the MQTT Broker
Step 5 - Integrate w/ Home Assistant
Step 6 - Make some REST Api calls
Step 7 - Setup Appdaemon
Step 8 - Basic LogText Automation
Step 9 - TDD MorseCode Automation
Step 10 - A Complete Real-life setup

Questions?

- Open to Questions at any point during the conference
  - If anyone missed part of the presentation or would like to know more they're welcome to approach me
- I'm on Twitter: @ThisIsFlorianK
- My Blog: Professional Beginner
  - Not much home automation content though
    - Actually none at all 🙃
- Direct contact: Flori@nKempenich.com