2.1.1 Installation HA Core Docker Compose

1. Docker Compose File

Installing Home Assistant via Docker compose is for the more advanced user. The simplest is people buying a home assistant box on which Home assistant OS is installed.

2. HA Reverse Proxy

Now, if you want your home assistant to be reachable via URL or domain name then we need to jump into the configuration.yaml file inside the container. Assuming your container is called "home-assistant" we type in the following command inside the docker vm to jump inside the container

```
docker exec -it home-assistant /bin/bash
```

This will drop you inside the container as *containter_number:/config#*. We will then enter the configuration.yaml file with

```
vi configuration.yaml
```

Then you need to make sure that change your configuration file to this

```
# Loads default set of integrations. Do not remove.
default_config:

http:
  use_x_forwarded_for: true
  trusted_proxies:
    - 10.56.4.1  #IP address of gateway or the VLAN network----> CHANGE THIS TO YOUR VALUES

# Load frontend themes from the themes folder
frontend:
  themes: !include_dir_merge_named themes

automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml
```

It is clear that the **http** part is the one that is added. Adjust the values to your configuration and save the file. To start editing in view one would click on the letter "i". Make the edit. Once you are done hit ESC and then type the command ":wq" to write to it and quit. After that is done you should restart Home Assistant.

2.1.1.3 HACS Installation

HACS is a custom integration that lets you manage and discover custom elements for Home Assistant from the UI. You can also publish your own custom elements, create shortcuts to repositories or issue trackers, and contribute to **HACS**.

Many integrations are not included in the official HA repositories but need to be integrated via community integrations. The piece that makes this possible is called HACS a.k.a Home Assistant Community Software. Hence, before we start deploying any integrations it is smart to deploy HACS on your HA instance already.

Now, there are three different installation methods depending on how you installed HA on your machine.

2.1.1.4 MQTT Broker

How It Works

Imagine your smart home devices (lights, thermostats, door locks, etc.) are family members who need to communicate. Instead of each device talking directly to every other device (which would be messy), they all use a central "message board" called an MQTT broker. When one device needs to send information or receive a command, it posts the message to this board, and other devices that care about that message can read it.

A Practical Example

Let's say you have:

- A smart light in your living room
- A motion sensor in your hallway
- Home Assistant (your smart home hub)

When the motion sensor detects movement, it doesn't directly tell the light to turn on. Instead, it posts a message saying "motion detected in hallway" to the central message board. Home Assistant reads this message, thinks "I have a rule that says turn on the living room light when motion is detected," and then posts a message saying "turn on living room light." The light reads this instruction and turns on.

Why It's Useful

- Devices don't need to know about each other — they just post and read messages
- It's lightweight — doesn't use much internet bandwidth or processing power
- It works reliably even if connections are slow — messages get delivered when devices reconnect
- You can automate things — Home Assistant acts as the "smart controller" that makes decisions based on the messages it receives

That's essentially what MQTT does in Home Assistant—it's the communication backbone that lets your devices talk to each other through a central hub.

Installation

Now, here we will assume that you have installed ha core via docker and not the HA OS. The HA OS mqtt installation is a lot simpler but you gain some and you loose some ;) So we will basically create another mqtt container that will serve as our mqtt broker.

Source: Setup MQTT broker in docker

The source from above explains everything pretty well but I want to add something to it that made the setup easier for me. That is simply setting the PUID and PGID to my user. By default the first user runs on PUID=PGID=1000 so we add this to the docker compose file.

But first assuming you use Dockge again we make the following folders and items before we deploy our compose files

```
mkdir -p /opt/stacks/mqtt/{config,data,log}
cd /opt/stacks/mqtt/
```

We will then create the docker compose file inside the mqtt folder

Compose file

```
services:
  mosquitto:
    container_name: mqtt
    image: eclipse-mosquitto:2
    environment:
      - PUID=1000
      - PGID=1000
    ports:
      # Standard MQTT protocol
      - 1883:1883
      # MQTT over WebSocket
      - 9001:9001
    volumes:
      # Mount configuration directory
      - ./config:/mosquitto/config
      # Persist message data
      - ./data:/mosquitto/data
      # Persist log files
      - ./log:/mosquitto/log
    restart: unless-stopped
    networks:
      mosquitto:

networks:
```

```
  mosquitto:
```

After that is done we will move inside config folder and create our mosquitto.conf file and add the following content to it

mosquitto.conf

```
# mosquitto-docker/config/mosquitto.conf - Basic Mosquitto configuration

# Listen on all interfaces for standard MQTT
listener 1883
protocol mqtt

# WebSocket listener on port 9001
listener 9001
protocol websockets

# Do not allow anonymous connections
allow_anonymous false
password_file /mosquitto/config/passwd

# Persistence settings - retain messages across restarts
persistence true
persistence_location /mosquitto/data/

# Log settings
log_dest file /mosquitto/log/mosquitto.log
log_dest stdout
log_type all
```

After that is done we will create the passwd file inside the config directory

```
touch passwd
```

Then before we deploy everything we need to make sure that all the files are owned by your user. To make sure of this we will use the following command

```
sudo chown user:user -R /opt
```

This recursively makes all items inside of /opt recursively ownership by PUID and GPID of "user". Finally make sure that the passwd has 700 as permissions. You can make sure of this by entering the config directory and use the command

```
chmod 700 passwd
```

If this is all done we can go back to our Dockge display and reload it. Now an mqtt container should be ready to be deployed. Just simply hit deploy and watch the logs.

If it is all up and running we need to make some users of the mosquitto broker. We can do this by entering the moquitto container with

```
docker exec -it mqtt sh
```

Then we use the following command to make the first user

```
# Create a password file with an initial user
mosquitto_passwd -c /mosquitto/config/passwd device1

# Add more users without the -c flag (which creates a new file)
mosquitto mosquitto_passwd /mosquitto/config/passwd device2
mosquitto mosquitto_passwd /mosquitto/config/passwd webapp
```

You can get this error message is your user is not root

> Warning: File /mosquitto/config/passwd owner is not root. Future versions will refuse to load this file.To fix this, use `chown root /mosquitto/config/passwd`

But this should be fine. Otherwise you can set it temporarily to root and change the ownership of passwd back to "user" after creation.

You will be prompted by setting up a password for the created users. Make sure you note them down since you will need it for authentication in HA and the apps that are using the MQTT broker.

Sweet!!! So now you finally have Mosquitto up and running.

2.1.1.5 HA Blue Prints

Source: Blue Prints

Blueprints are the easiest way to add an automation to your Home Assistant. They are ready-made automations shared by the Home Assistant community: someone else has already done the thinking and the writing, and you just plug in your own devices.

You can find blueprints for almost any common use case, from “turn the lights on when motion is detected” to “announce the weather every morning at 7” to “notify me when the laundry is done”. You install a blueprint once and can use it as many times as you like, with different devices and settings each time.

Check out the Blue Print Community exchange !!! For instance for our Frigate installation we used the frigate blue print from the HA blue print community exchange.

2.1.1.6 HA Companian App

Home Assistant also comes with a beautiful application. It is basically a must have !! This allows you to manage your home or building from a distance effectively without opening up any laptop or PC. The buttons created in HA are sweetly responsively translated to your phone. In this page I describe a couple of things that might be handy for users

Pinch to zoom

If you have pictures in one of your dashboard you might like the functionality pinch to zoom. This basically allows you to zoom into a picture inside your companian app. To do this go to

Settings -> Companian App -> Select Pinch to zoom

2.1.1.7 Notify Groups for Mobile Notifications

Notify groups are a great way to send a notifications to multiple devices at the same time. In this way if you have x devices that need a notification from an automation then you only need to make the automation once and refer to the group. This makes everything need, clear and easily scalable. So for this we assume that you already have more than or equal to two devices. For the example we use the devices that are registered with the Home Assistant companion app.

1. mobile_app_device1
2. mobile_app_device2

If that is established we will move into the GUI and create a Notify Group

1. Go to Settings > Devices & Services > Helpers
2. Here we choose the button in the right bottom corner + Create Helper
3. Then we select Group
4. Finally we select the type of group: Notify Group

Give the Notify Group a name e.g "GROUP_NAME" and add your devices to it. In this case "device1" and "device2". If they are recognized they must appear if you click on the members button.

Then save the group and your are 1/2 way there ;)))))) wohooooo!!1

What is next is going back into the terminal to access the configuration.yaml file that is located at /home/user/opt/stacks/home_assistant/config/.

```
nano configuration.yaml
```

Then we will add the following notification information to this file

```
notify:
  - platform: group
    name: GROUP_NAME
    services:
      - service: mobile_app_device1
      - service: mobile_app_device2
```

Then save the file and you are almost set! The only thing that is left is to restart your home assistant instance for the changes to take effect.

Then now, being officially registered and all --> you can choose the GROUP_NAME to receive notifications for all kinds of automation's. For instance Frigate Blueprint notifications or if devices are up or down and send a notifications to all the phones defined in the Notify Group.