Welcome to the HiveMQ Partner technical training

All resources for this training can be found here: https://github.com/hivemq/Orange_Workshop

Who is this workshop for ?

This session is aimed at our partners' technical teams. We take a closer look at the platform, its different bricks and how they work.

We will cover topics like enterprise broker deployment and security, extension configuration, DataHub policies and transformations, HiveMQ Edge and it's protocol adapters.

Prerequisite :
- Have a good understanding of HiveMQ Platform.
- Have basic knowledge on kubernetes and linux.

What is the goal of the training ?

The aim of this workshop is to train our partners' teams in the installation and advanced configuration of the HiveMQ platform.
After attending the workshop, you will be able to implement a production site with best practices on security and enterprise extensions configured.

We wish you a pleasant training

May 2025,

Anthony Olazabal
Kamiel Straatman

Preparations :

Have Docker, Docker Compose, Git, PgAdmin (optional) and the HiveMQ CLI installed:

https://docs.docker.com/engine/install/

https://docs.docker.com/compose/install/

HiveMQ CLI

Git CLI or GitDesktop

Useful additional software :

PgAdmin

MQTT explorer

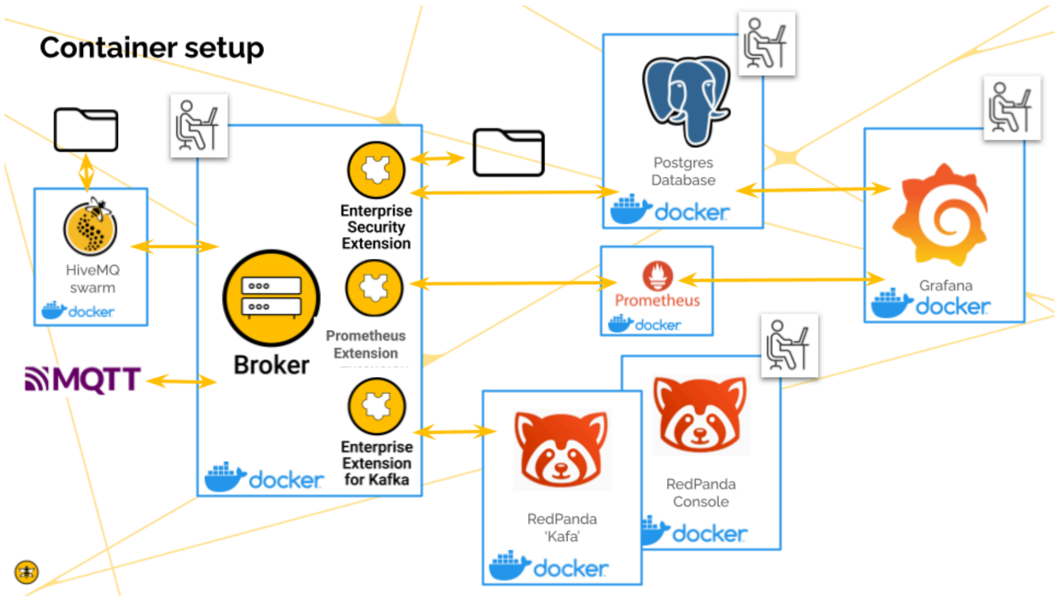
Introduction to the Container setup

This workshop is based on running the several needed software components, including the broker(s), in a multi-container environment. This is to create an equal and easy to manage sandbox infrastructure within a range of different laptop systems.

To get the resources please run the following command in your terminal within an appropriate local directory (eg. MyDocuments) :

```
Unset
git clone https://github.com/hivemq/Orange_Workshop.git
```

This clone will render you with infra-as-code for the following setup of containers:



Basically you have a broker container running that is able to interface with a Postgres database, a Redpanda Kafka system and a prometheus monitor database. The latter as well as the database can be queried by the Grafana visualisation system.

In the front-end a HiveMQ swarm container is foreseen that can generate MQTT traffic according to need.

In the `Dockerfile` you can selectively en/disable extensions:

```
# Default allow all extension, set this to false to disable it
ENV HIVEMQ_ALLOW_ALL_CLIENTS=true
# Enable ESE, set this to false to disable it
ENV HIVEMQ_ENABLE_ESE=false
# Enable Kafka extension, set this to false to disable it
ENV HIVEMQ_ENABLE_KAFKA=false
# Enable REST API default value
ENV HIVEMQ_REST_API_ENABLED=true
```

Please start initially with the above settings. Later in these sessions you can enable selective extensions one by one.

Use it / Start :

to start please `cd` into the cloned `Orange_Workshop` directory and run the following commands:

```
Unset
export HIVEMQ_VERSION=4.38.0
export REDPANDA_VERSION=24.2.7
export REDPANDA_CONSOLE_VERSION=2.7.2
./build.sh
docker-compose up -d  --build --force-recreate
```

and test it :

Send some MQTT data towards the Kafka propagation topic

```
Unset
mqtt pub -u superuser -pw supersecurepassword -t to-kafka/test -m kamiel
```

See in Redpanda console the MQTT sent message appear in the correct `kafka-topic` topic:

http://localhost:8090/topics/

Accessing HiveMQ and review directory structure

Connecting to the GUI

The HiveMQ GUI can be accessed for a local or dockerize broker on http://127.0.0.1 and by default on port 8080. The default credentials are **hivemq** and as password **admin** .

Please test: open your browser and go to page http://127.0.0.1. Familiarise yourself with the layout and don't forget to check out our new CC 2.0 interface.

Sending and receiving a MQTT message
A broker is not able to send MQTT packages by itself so we need to have some MQTT client software (or devices) to do so. A number of options to do this are available to your liking.

HiveMQ CLI both available for Mac and WIndows.
Downloadable at https://www.hivemq.com/blog/mqtt-cli/
We recommend this CLI because it provides a compact command line interface (CLI) for MQTT 3.1.1 and MQTT 5 clients that supports interactive command modes. It also supports access to the HiveMQ API in an easy to access way.

Good GUI-oriented, Mac and WIndows based alternatives are:
MQTT.fx (payed): https://www.softblade.de/
MQTT explorer (free) : https://mqtt-explorer.com/
Except for the HiveMQ specific API interaction these GUI based tools can be used during this course.

Please test your basic browser connectivity over mqtt by subscribing to, and subsequent publishing to a known topic. This can be your local broker but also try your neighbors IP address.

Example:

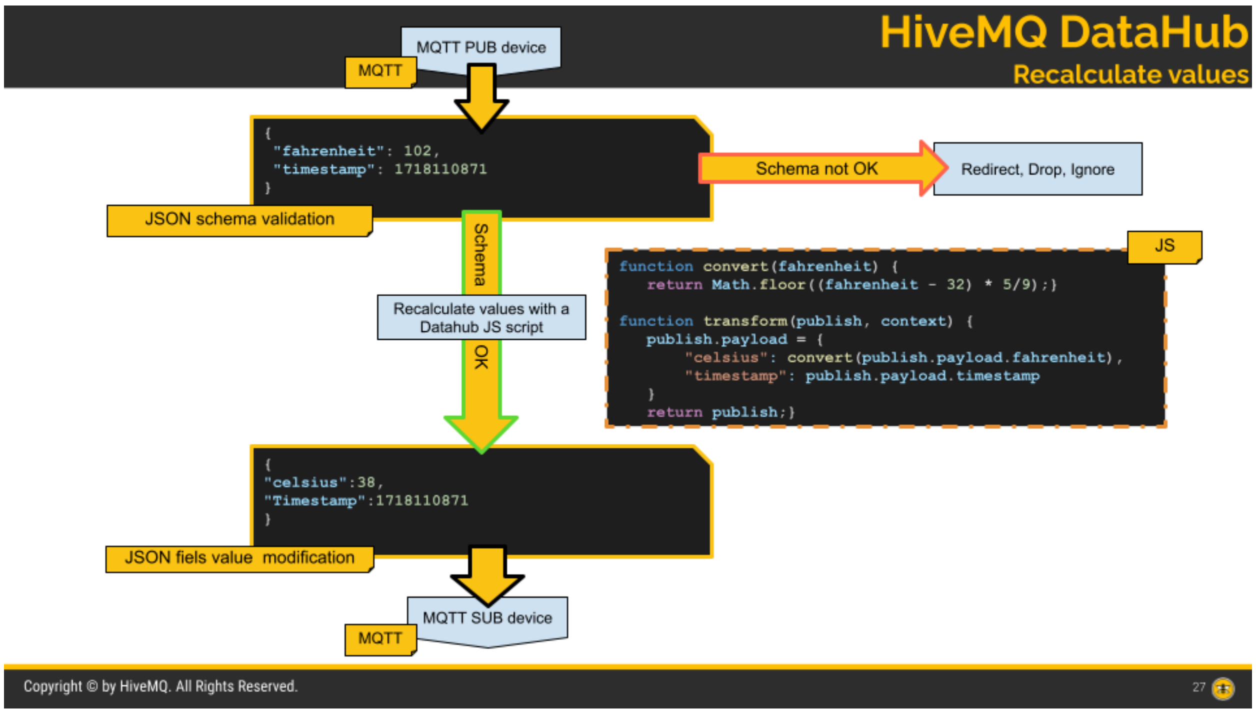
`mqtt sub -h 127.0.0.1 -t “#”` (this command will ‘wait’ for input, try the -J option as well)
And in another window:
`mqtt pub -h 127.0.0.1 -t “sensor/temp” -m “20.4 degrees”`

As an option use the IP address of your neighbour and replace the local 127.0.0.1 adres to send and receive messages from remote brokers.

Congratulations, you're able to send and receive MQTT messages.

Datahub

The HiveMQ Data Hub provides mechanisms to define how MQTT data and MQTT client behavior are handled in the HiveMQ broker. Data Validation in the HiveMQ Data Hub allows you to implement declarative policies that check whether your data sources are sending data in the data format you expect and provides the ability to even change it on the fly.



```
{
 "fahrenheit": 102,
 "timestamp": 1718110871
}
```

```
function convert(fahrenheit) {
   return Math.floor((fahrenheit - 32) * 5/9);}

function transform(publish, context) {
   publish.payload = {
       "celsius": convert(publish.payload.fahrenheit),
       "timestamp": publish.payload.timestamp
   }
   return publish;}
```

```
{
"celsius":38,
"Timestamp":1718110871
}
```

Open the GUI for the local broker (http://127.0.0.01 / hivemq/ admin) and select on the left menu 'Datahub/Schemas'. In the selected screen enable the 5 hour trail for datahub.

Now upload the two selected JSON schema definitions (schema-for.., schema-from..)

```
mqtt hivemq schema create --id schema-from-device --type json --file schema-from-device.json
mqtt hivemq schema create --id schema-for-fan --type json --file schema-for-fan.json
```

or copy paste their content in the schema definition fields. [1]

[1] This is an excellent json to schema convertor : https://www.liquid-technologies.com/online-json-to-schema-converter

Now do the same for the javascript:

```
mqtt hivemq script create --id=fahrenheit-to-celsius --file=script.js --type=transformation
```

or copy paste their content in the schema definition fields.

The policy can be manually created with the data policy entry on the left menu or can be uploaded over the API:

```
mqtt hivemq data-policy create --file policy.json
```

Or define it interactively in the GUI:
Matching Topic Filter : factory/#

Schema Validators
All Of - Schema with id *schema-from-device* and version *latest*

On Success Pipeline:

- operation- **Deserialize** Schema with id *schema-from-device*
- Operation- **Script** - fahrenheit-to-celsius - latest : Link to Scrip
- Operation- **Serdes** - Serialize id s*chema-for-fan*

On Failure Pipeline:
operation- Drop Message with Reason String: Your client ${clientId} sent invalid data according to the schema: ${validationResult}.

Now test :

```
mqtt pub -h 127.0.0.1 -t factory/test  -m "{\"fahrenheit\": 102,\"timestamp\": 1718110871}"
```

Output of `mqtt sub -h 127.0.0.1 -t “#”` should be like:

```
{"celsius":38,"timestamp":1718110871}
```