WELCOME TO

We will start the session shortly

WELCOME

Anja Helmbrecht-Schaar

linkedin.com/in/anjahelmbrechtschaar/

- **Senior Consultant @HiveMQ**

Georg Held

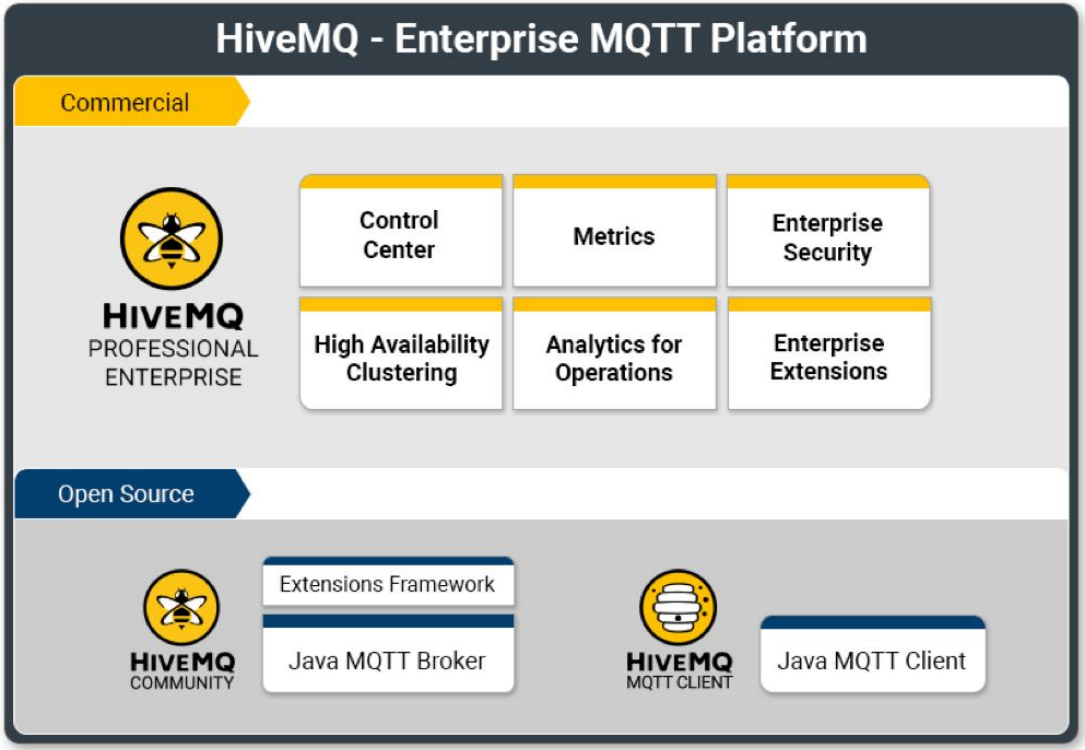
linkedin.com/in/sauroter/

github.com/sauroter

- **Software Developer @HiveMQ**

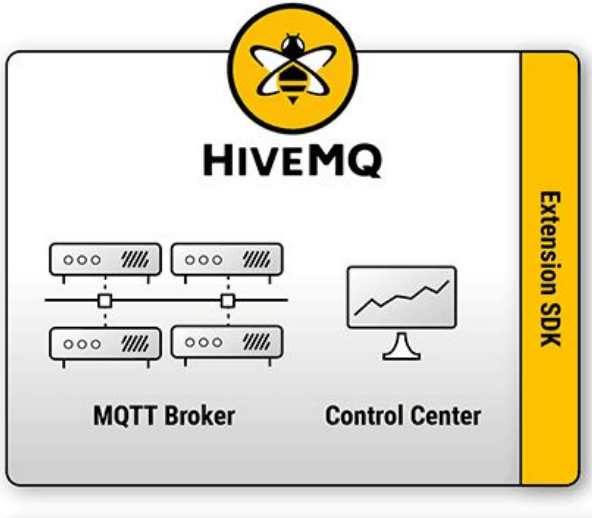
How to build your own HiveMQ Extension?

HiveMQ Ecosystem



HiveMQ Extension

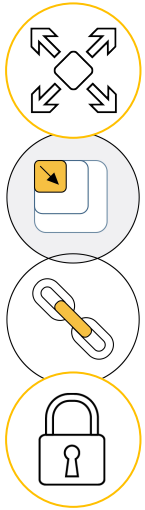
- An Extension is a simple program written in Java, openJDK >= 11, using the SDK to interact with the Broker.
- The Community Extension SDK allows you to seamlessly link your own business logic to events, messages and content that is processed by HiveMQ
- A comprehensive documentation and examples are available on our homepage



https://www.hivemq.com/docs/hivemq/latest/extensions

What Interactions Can Be Done?

- Use the Services API to interact with HiveMQ and the connected MQTT clients
- Register Callback Classes that are called by HiveMQ when a certain Event occurs
- Inspection and manipulation of MQTT related data, like sessions, retained messages, subscriptions and many more
- Implement fine grained Authentication and Authorization for MQTT clients
- Add a custom cluster discovery mechanism
- Add custom logic to HiveMQ



Use cases: write messages to a database, integrate service buses, collect statistics, add fine-grained security ...

Tools That Are Needed

https://www.hivemq.com/docs/latest/extensions-javadoc/index.htm

Setup Your Project

HANDS ON -
Creating the Project

Setup the HiveMQ Extension Gradle Project

We created a Gradle plugin that registers all the basic tasks you need from start to finish

1. Use the plugin in your **gradle settings file**

```
pluginManagement {
    plugins {
        id("com.hivemq.extension") version "1.0.0"
    }
}
rootProject.name = "database-extension"
```

Setup the HiveMQ Extension Gradle Project

2. And in your **gradle build file**

```
plugins {
        id("com.hivemq.extension")
}
```

3. Apply the HiveMQ extension plugin to your project.

Than you can configure all the important HiveMQ extension properties directly in your build file.

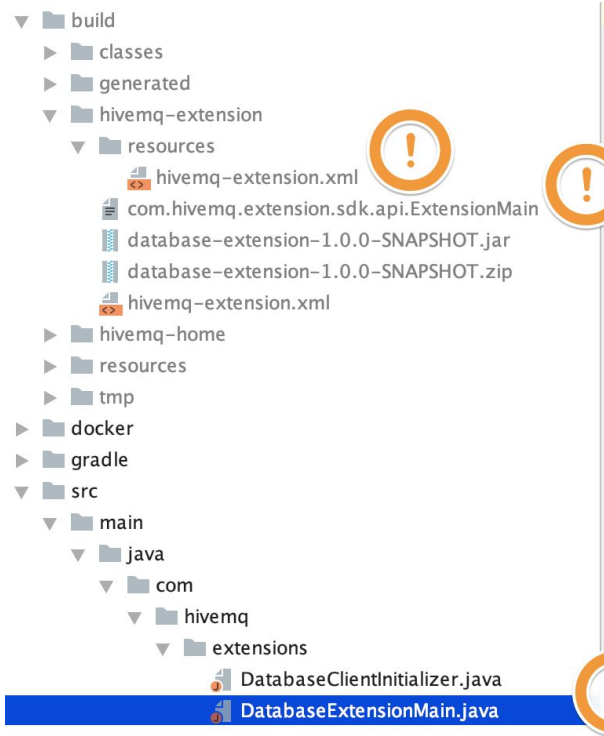
Setup the HiveMQ Extension Gradle Project

Gradle **build file**

```
5   group = "com.hivemq.extensions"
6   version = "1.0.0-SNAPSHOT"
7   description = "HiveMQ Database Extension"
8
9   hivemqExtension {
10      name = "HiveMQ Database Extension"          // Name of the HiveMQ extension, required
11      author = "HiveMQ"                           // Author of the HiveMQ extension, required
12      priority = 50                               // Priority of the HiveMQ extension, default: 0
13      startPriority = 1000                        // Start priority, default: 1000
14      mainClass = "$group.DatabaseExtensionMain"  // Main class of the HiveMQ extension, required
15      sdkVersion = "4.4.2"                        // Version of Extension SDK, default: latest
16  }
```

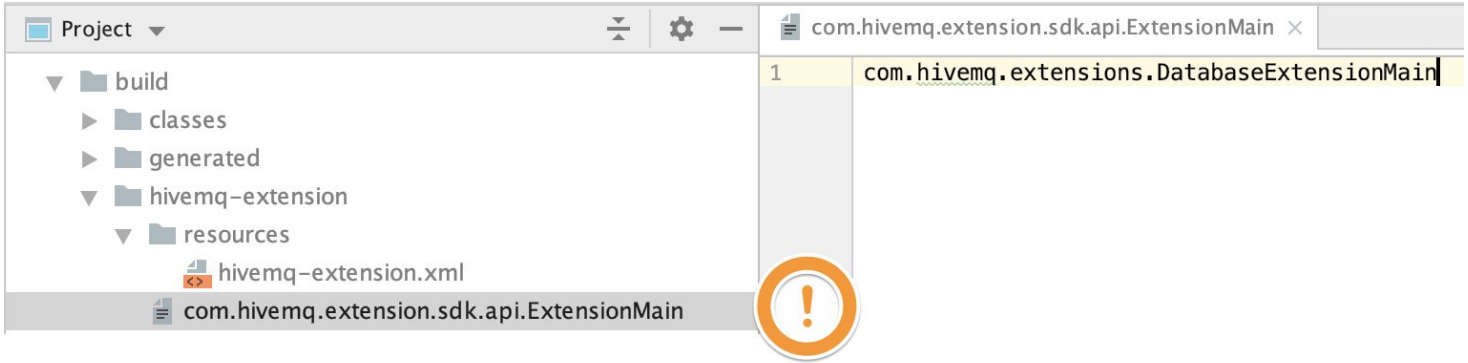
RUN simply: ./gradlew hivemqExtensionZip

Project Structure



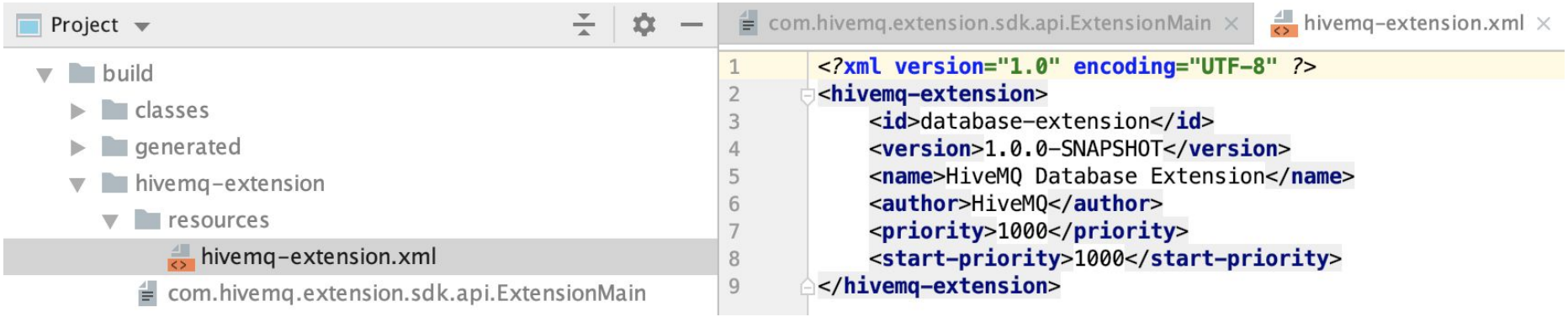

```java
package com.hivemq.extensions;

import ...

/**
 * @author Georg Held
 */
public class DatabaseExtensionMain implements ExtensionMain {

    private @NotNull HikariDataSource ds;

    @Override
    public void extensionStart(final @NotNull ExtensionStartInput e

    @Override
    public void extensionStop(final @NotNull ExtensionStopInput ext
        ds.close();
    }
}
```

Extension Information



The manifest File points to the right class name of the ‚Main' Class of the Extension
If this does not fit, the Extension will not be loaded

Extension Information



Utilized values from the build settings

Use Case

Our example extension dumps all incoming mqtt messages into a database

Major Steps:

1. Connect to our database during start of the extension
2. Catch any incoming MQTT publish and forward this to our external system

What Do We Need to Create the DB Connection?

Public class DatabaseExtensionMain implements ExtensionMain ...

```
@Override
public void extensionStart(final @NotNull ExtensionStartInput extensionStartInput,
                           final @NotNull ExtensionStartOutput extensionStartOutput) {
   ds = createDS();
   //do more
}

@Override
public void extensionStop(final @NotNull ExtensionStopInput extensionStopInput,
                          final @NotNull ExtensionStopOutput extensionStopOutput) {
   ds.close();
}
```

HANDS ON -
Set Up a DB Connection

First Try

1. Run ./gradlew hivemqExtensionZip
2. Put the extension zip into extension folder of your local hivemq instance
3. Unzip the extension.zip file
4. Take a look on the log files

```
2020-10-20 15:31:24,226 INFO  - HikariPool-1 - Start completed.
2020-10-20 15:31:24,228 INFO  - Extension "HiveMQ Database Extension" version 1.0.0-SNAPSHOT started successfully.
```

Extension Lifecycle

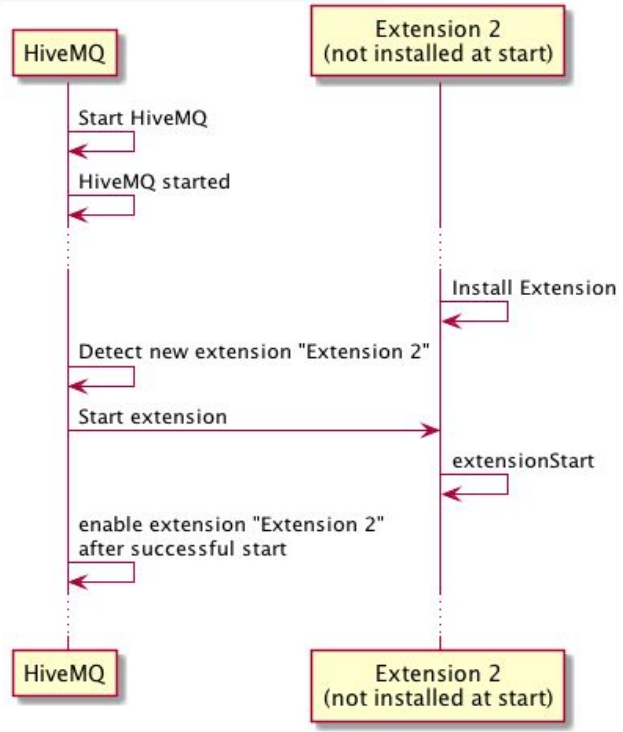
During running HiveMQ or at start:

The Extension will be automatically loaded

- When installed
- By removing the DISABLED flag if set

The Extension will be automatically stopped

- By adding the DISABLED flag
- If an error during Initialization phase occurs and preventFromStart is implemented



Run from your project with HiveMQ

1. Prepare your gradle setting

```
23  tasks.prepareHivemqHome {
24      hivemqFolder.set("/your/path/to/hivemq-4.4.2")
25  }
```

2. Simply run the `gradle task` **`runHivemqWithExtension`**

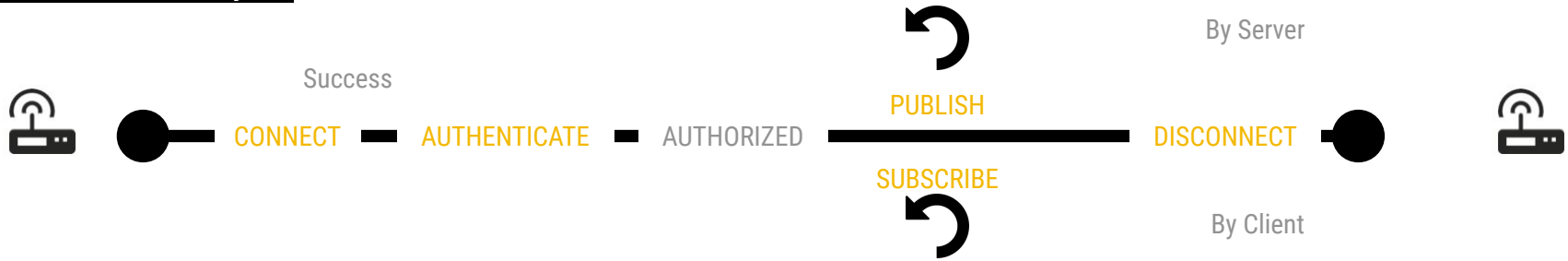
```
> Task :runHivemqWithExtension
2020-10-20 15:34:30,365 INFO  - Starting HiveMQ Enterprise Server
2020-10-20 15:34:30,368 INFO  - HiveMQ version: 4.4.2
2020-10-20 15:34:30,368 INFO  - HiveMQ home directory: /Users/ahelmbre/projects/workshop/webinar-database-extension/build/hivemq-home
2020-10-20 15:34:37,270 INFO  - Starting HiveMQ extension system.
2020-10-20 15:34:37,328 INFO  - HikariPool-1 - Starting...
2020-10-20 15:34:37,425 INFO  - HikariPool-1 - Start completed.
2020-10-20 15:34:37,428 INFO  - Extension "HiveMQ Database Extension" version 1.0.0-SNAPSHOT started successfully.
2020-10-20 15:34:39,810 INFO  - Started TCP Listener on address 0.0.0.0 and on port 1883
2020-10-20 15:34:39,810 INFO  - Started HiveMQ in 9452ms
<===========--> 90% EXECUTING [58s]
> :runHivemqWithExtension
```

What Do We Need to Forward MQTT Messages into DB?

- A way of interaction to be able to get each or specific incoming published message.
- Working with an external service in a non blocking way.
- Ideally have some metrics about these operations.
- …

Interactions

MQTT Client Lifecycle



Extension SDK supports

- Listen to Events: Connect, Disconnect
- Use Services for adding functionality to Authenticate, Authorization for Publish / Subscribe,
- Manipulate, process or persist Messages from Publishing
- Schedule async Processes during Client Lifecycle

SDK Provides Interceptor

- Available Interceptor
  - Connect Inbound, Connack Outbound
  - Publish Inbound, Publish Outbound
  - Disconnect
  - And many more
- Added to a Client/Global Initializer
- Registered via the Initializer Registry

Input Objects

An input object is an informational object.

Example Input Parameters:

ExtensionStartInput

- Extension information
- Enabled Extensions Map

ConnectionStartInput

- CONNECT Package
- Client information
- Connection Information

PublishInboundInput

- PUBLISH Package
- Client information
- Connection Information

The Publish Inbound Interceptor

- Performs interception at the moment of receiving an MQTT PUBLISH
- Can be used to **modify** inbound PUBLISH messages or **prevent** them
  - If delivery of a PUBLISH message is prevented, the message will be dropped.
- Multiple Interceptors are called sequentially and the Output Object will be updated after each interceptor
- The `PublishInboundOutput` object itself is blocking, but it can easily be used to create an asynchronous `PublishInboundOutput` object.

HANDS ON -
Intercept MQTT Publish

Gradle Build File Settings for Debug

- The HiveMQ Gradle plugin lets you run your extension with HiveMQ directly from your IDE.
- And let Gradle know, that we will debug.

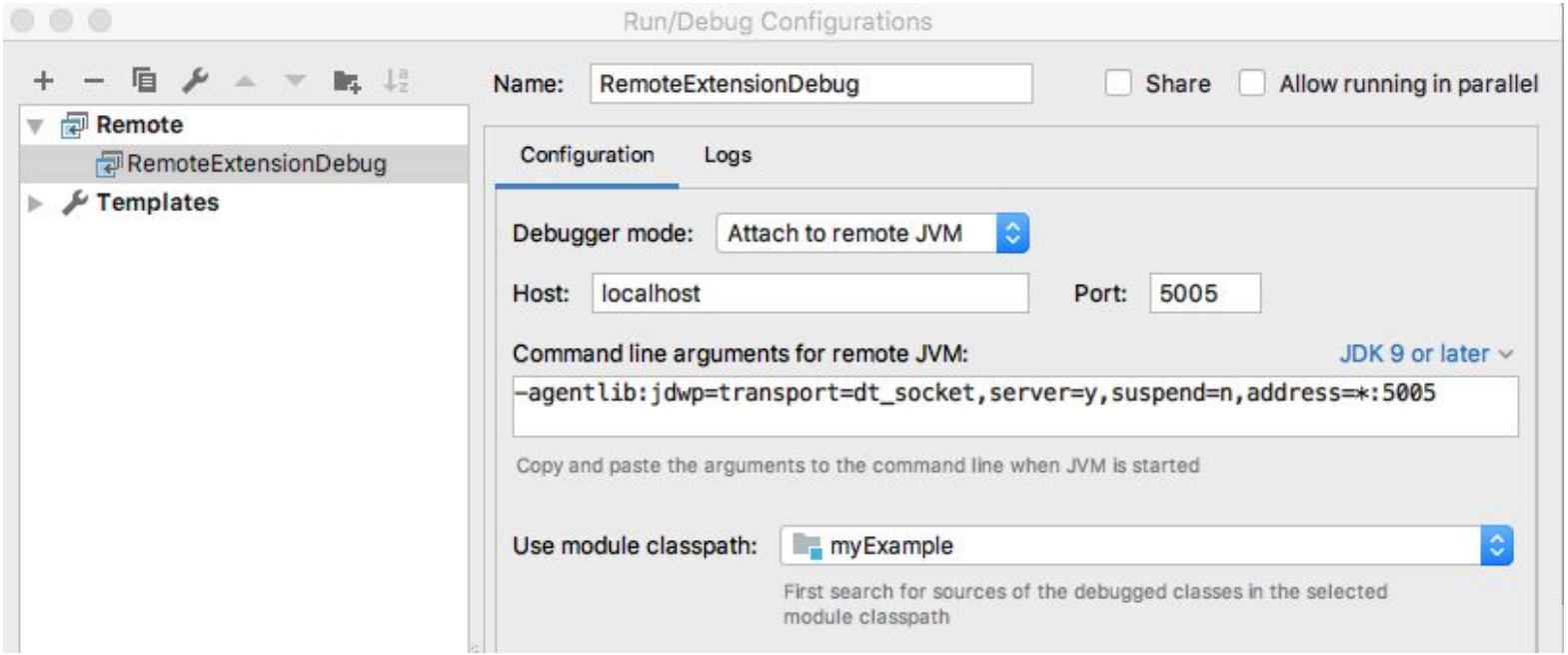
This allows you to run your extension together with other extensions

```
23  tasks.prepareHivemqHome {
24      hivemqFolder.set("/your/path/to/hivemq-4.4.2")
25      // You can add any files: configs, licenses, other extensions, etc.
26      from("src/test/resources/config.xml") { into("conf") }
27      from("src/test/resources/other-extension") { into("extensions") }
28  }
29
30  tasks.runHivemqWithExtension {
31      debugOptions {
32          enabled.set(true)
33      }
34  }
```

This enables you to attach a debugger to your HiveMQ instance

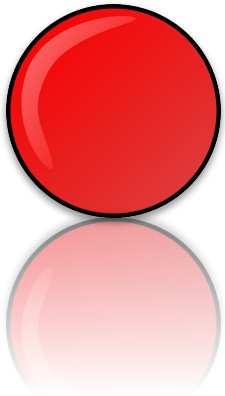
Prepare Debug

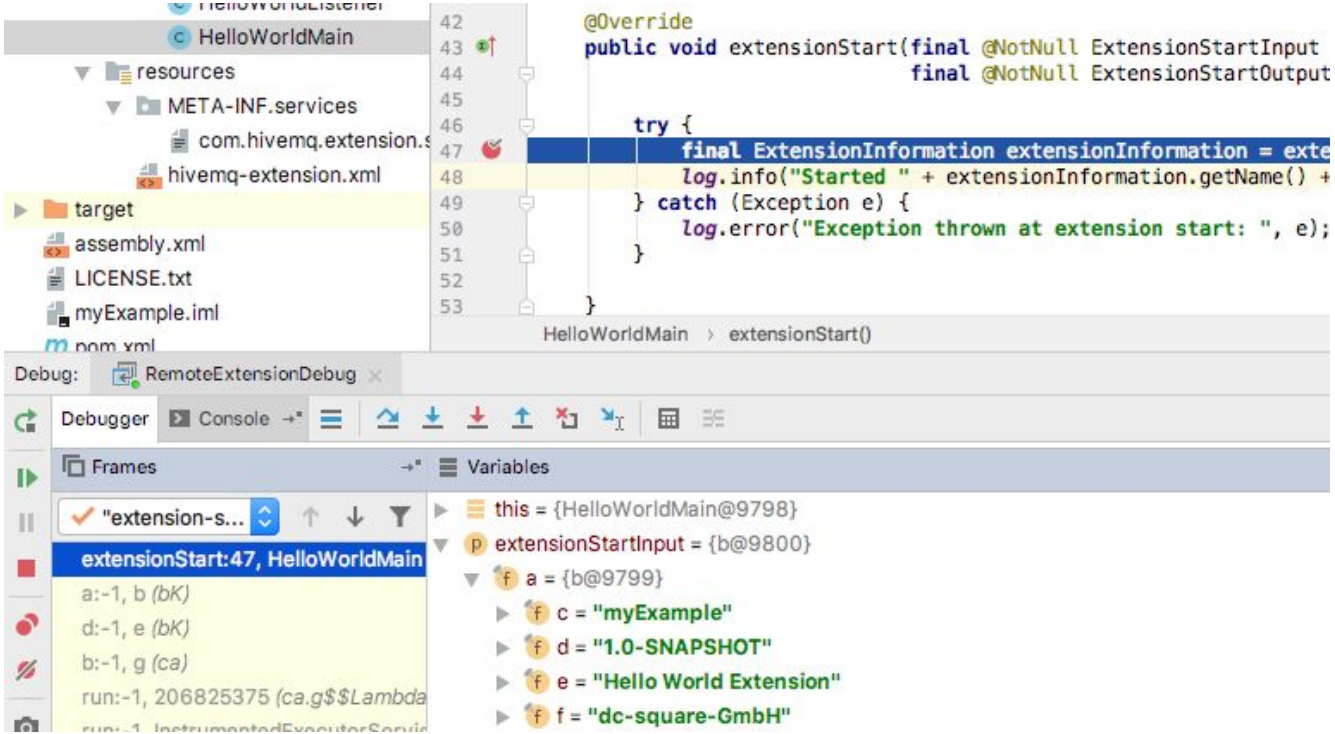
Create new run configuration for debugging: Add Configuration… → + → Remote → OK



Let's Debug

- Run your new Debug Configuration but
- Don't forget to set a Breakpoint





Test Your Own Extension

Use **HiveMQ Testcontainer** to do integration tests

https://github.com/hivemq/hivemq-testcontainer

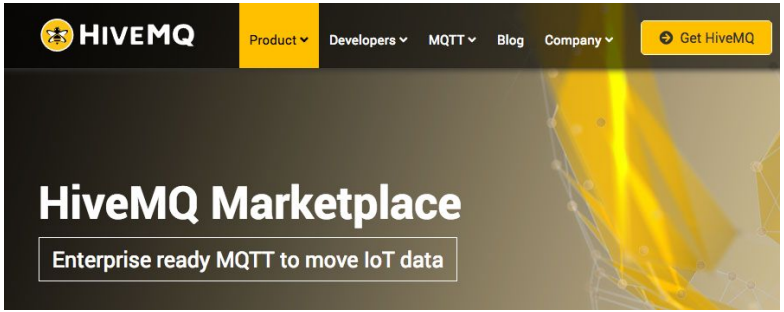
- Automatic starting HiveMQ docker containers for JUnit4 and JUnit5 tests.
- Enables integration testing of custom HiveMQ extensions
- Enables testing of MQTT client applications.

Summary

- What is the HiveMQ Extension SDK
- How powerful the SDK is
- How easy it is:
  - To set up
  - To run
  - And to debug an **own** extension
- But there is a marketplace ...

Pre-build Extensions

https://www.hivemq.com/extensions



https://github.com/hivemq

Catalog of pre-built extensions built by the HiveMQ team and the HiveMQ community.

Opensource Extensions

HiveMQ Extension - Deny Root Wildcard Subscriptions

Denies any subscription to the root wildcard topic.

Type: Security
License: Apache v2

Learn more

HiveMQ Extension - MQTT Message Log

The HiveMQ MQTT Message Log Extension provides the possibility to follow up on any clients communicating with the broker on the terminal.

Type: Logging
License: Apache V2

Learn more

HiveMQ Extension - File RBAC

HiveMQ File Role based Access Control Extension which adds client authentication.

Type: Security
License: Apache v2

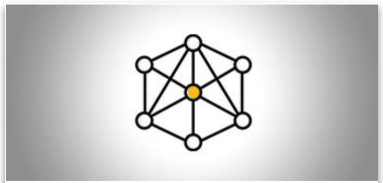
Learn more

HiveMQ Extension - InfluxDB

Allows HiveMQ to connect to an instance of InfluxDB for time series monitoring of the internal metrics.

Type: Monitoring
License: Apache v2

Learn more



HiveMQ Extension - DNS Discovery

Enables dynamic clustering for Cloud providers (AWS) or orchestration software (Kubernetes).

Type: Integration
License: Apache v2

Learn more

HiveMQ Extension - S3 Cluster Discovery

Allows HiveMQ cluster nodes to discover each other dynamically.

Type: Integration
License: Apache v2

Learn more

HiveMQ Extension - Heartbeat

Creates an HTTP Endpoint on broker start up that can be used for application layer load balancer health checks.

Type: Monitoring
License: Apache v2

Learn more

HiveMQ Extension - Prometheus Monitoring

Can be configured with the Prometheus Configuration Properties

Type: Monitoring
License: Apache v2

Learn more

Commercial Extensions

Commercial

HIVEMQ ENTERPRISE BRIDGE EXTENSION

HiveMQ Enterprise Bridge Extension

Makes it possible to stream MQTT data between different brokers and broker clusters.

Type: Integration
License: Commercial

Learn more

Commercial

HIVEMQ ENTERPRISE EXTENSION FOR KAFKA

HiveMQ Enterprise Extension for Kafka

Makes it possible to seamlessly integrate MQTT messages with Kafka clusters.

Type: Integration
License: Commercial

Learn more

Commercial

HiveMQ Enterprise Security Extension

The HiveMQ Enterprise Security Extension makes it possible to integrate existing enterprise security systems into your HiveMQ device authentication and authorization workflow.

Type: Security
License: Commercial

Learn more

Questions?

THANK YOU

For attending the webinar

We will upload the webinar on our YouTube Channel

Subscribe to our YouTube Channel: page.video/hivemq

Stay updated on upcoming webinars

Subscribe to our Newsletter: newsletter.social/hivemq

All unanswered questions will be answered on the **HiveMQ Community Forum**

To the HiveMQ Community Forum: community.hivemq.com

Register for our November Webinar:

"Freedom⁴ - Free Your Manufacturing Data with Apache PLC4X & MQTT

bit.ly/plc4x-webinar