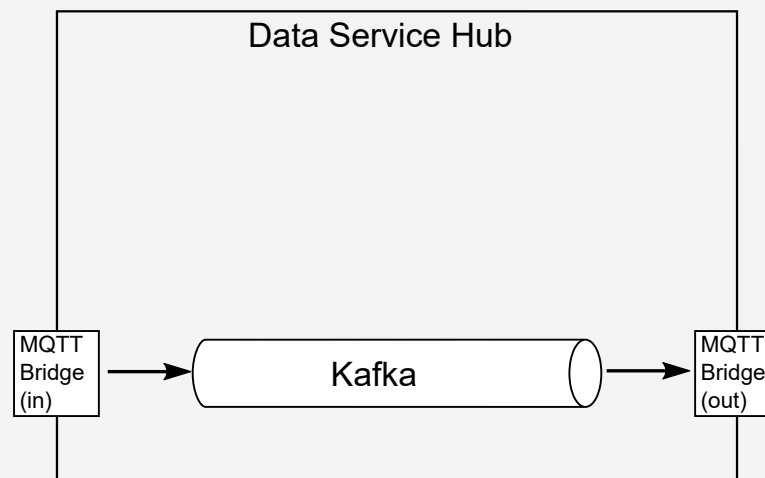# Streaming over Kafka

Kafka
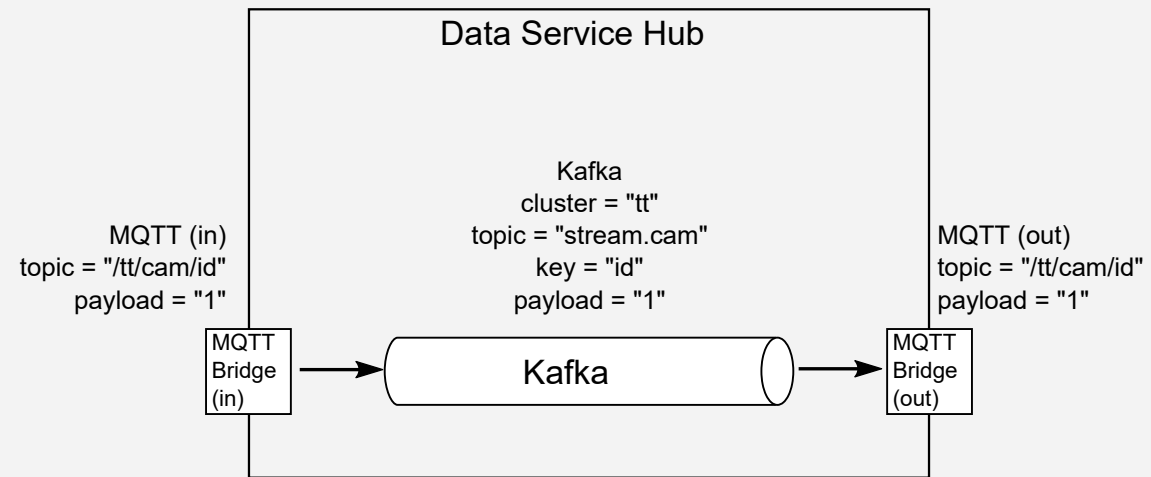
Kafka

Three Kafka stream-types

- *stream.* topic
- *internal.* topic
- *scratch.* topic
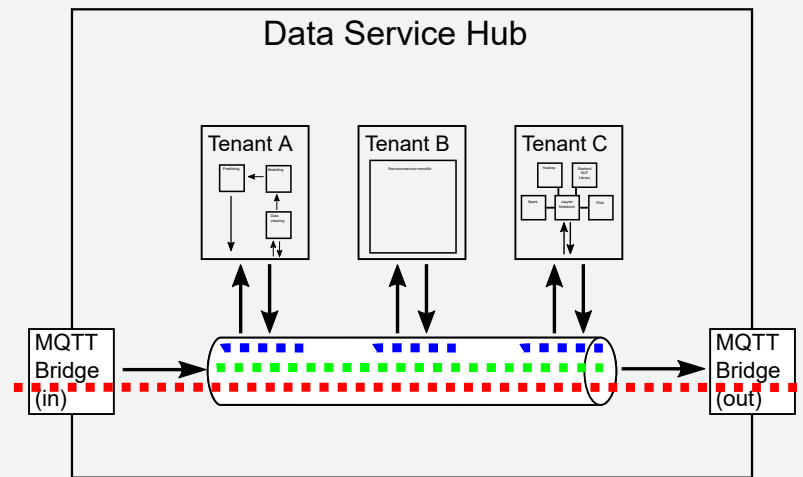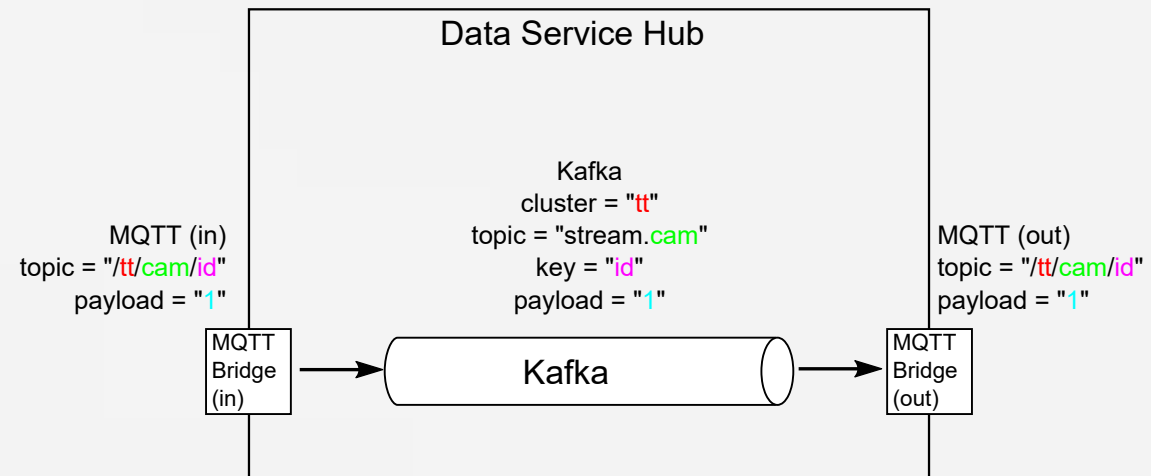
Data Service Hub

Tenant A

Tenant B

Tenant C

MQTT
Bridge
(in)

MQTT
Bridge
(out)

Data Service Hub

MQTT (in)
topic = "/tt/cam/id"
payload = "1"

Kafka
cluster = "tt"
topic = "stream.cam"
key = "id"
payload = "1"

MQTT (out)
topic = "/tt/cam/id"
payload = "1"

```
MQTT
Bridge
(in)
```
→
```
Kafka
```
→
```
MQTT
Bridge
(out)
```

Kafka

Data Service Hub

Kafka
cluster = "tt"
topic = "stream.cam"
key = "id"
payload = "1"

MQTT (in)
topic = "/tt/cam/id"
payload = "1"

MQTT (out)
topic = "/tt/cam/id"
payload = "1"

MQTT Bridge (in)

Kafka

MQTT Bridge (out)

Kafka and the MQTT Bridge

$$\begin{align} \text{MQTT topic prefix} &= \text{Kafka cluster name} \\ \text{MQTT topic infix} &= \text{Kafka topic name} \\ \text{MQTT topic suffix} &= \text{keys in Kafka} \\ \end{align}$$

Kafka and the MQTT Bridge

$$\begin{align} \text{MQTT topic prefix} &= \text{Kafka cluster name} \\ \text{MQTT topic infix} &= \text{Kafka topic name} \\ \text{MQTT topic suffix} &= \text{keys in Kafka} \\ \end{align}$$

```
MQTT(topic="/tt/cam/id", data="...")
```

$$=$$

```
Kafka(cluster="tt", topic="stream.cam.*", key="id", data="...
```

Goal

Learn to deploy an application on DSH that connects to DSH kafka.

Prerequisites

- *Installed:* Curl
- *Installed:* Mosquitto (MQTT-client)
- *Installed:* Docker-CE
- *Installed:* Git
- *Installed:* java SDK 8
- *Installed:* maven (mvn)
- *Installed:* DSH UMP-client
- *Available:* tenant UID
- *Available:* docker registry credentials
- *Available:* API-Key

Steps

1. Get tenant example
2. Build docker image
3. Push docker image to tenant's docker registry
4. Use UMP to deploy container on DSH
5. Use MQTT to contact container

Get tenant example

- Clone this repo from git: https://github.com/kpn-dsh/tenant-example
- It contains a fully working tenant example
  - in java
  - built with maven
  - builds docker image

Inspect

Dockerfile => UID

- Change your pwd (present working directory) to `tenant-example`
- Open the `Dockerfile` in your favorite text-editor
- Modify the `ENV id 1024` line according to the comment above it and save the modification

Inspect

pom.xml => tenant

- Open the `pom.xml` in your favorite text-editor
- Modify the tenant to your tenant

```
<tenant>dshdemo2</tenant>
```

- Modify version to something that contains your name: e.g.

```
<version>1.0.1-bruno-SNAPSHOT</version>
```

Build

```
mvn package
```

This will build the java binary and the docker image *locally*. Look in the output for the name of the docker image.

Push

Since every tenant has its own docker
registry this will be reflected in the
image tag name:

```
dataserviceshub-docker-$TENANT.jfrog.io/image:...
```

You need to be logged in to use this
registry:

```
docker login dataserviceshub-docker-$TENANT.jfrog.io
```

Now you can push your image to the tenant's docker registry:

```
docker push \
  dataserviceshub-docker-$TENANT.jfrog.io/tenant-example:...
```

Deploying

The docker image has now been built and safely stored in the docker registry. Next step: deploying a container on the DSH.

UMP

Download

Linux

Mac

Windows

Connect/Setup UMP

- Click + to add a new environment
- Fill in the requested values

Deploy application

- Click on the *Add new service*-button
- and name it `tenant-example-<your_name>`

DSH UMP   Edit

DEV-DSHDEMO

1 ▾

DSHDEMO1

⚙ SERVICES

cpu: 0.5 / 4

mem: 1 / 8 GB

SERVICES

NA⬛   CPU   MEM   STATUS

⊗

HOW WILL YOUR SERVICE BE CALLED?

tenant-example-rocketman

CREATE SERVICE DRAFT

✚ ADD NEW SERVICE
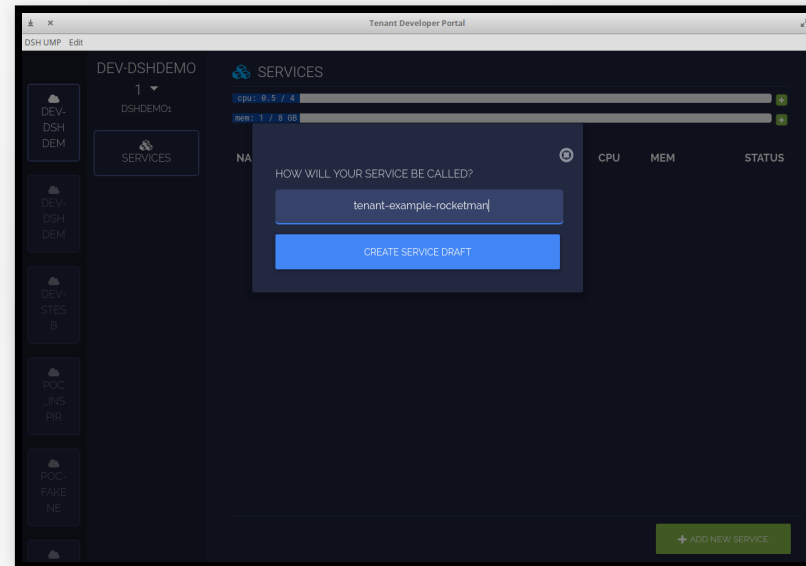
Deploy application

- create AMP definition (see `tenant-example.json`)
- modify the name of the image
- modify the user if needed
- and click *deploy*

DSH UMP   Edit

TENANT-EXAMPLE-
ROCKETMAN

☁
DEV-
DSH
DEM

☁
DEV-
DSH
DEM

☁
DEV-
STES
B

☁
POC
_INS
PIR

☁
POC-
FAKE
NE

☁

⚙ TASKS

⚙ CONFIGURATION

⚙ CONFIGURATION

```
1   {
2       "image": "dataserviceshub-docker-dshdemo1.jfrog.io/kpn-dsh-tenant-example-stesb:1.0.1-SNAPSHOT",
3       "cpus": 0.5,
4       "mem": 1024,
5       "env": {
6           "JAEGER_SAMPLER_TYPE": "const",
7           "JAEGER_SAMPLER_PARAM": "1",
8           "OUTPUT_STREAM": "stream.dshdemoshared",
9           "JAEGER_REPORTER_FLUSH_INTERVAL": "1000",
10          "JAEGER_REPORTER_LOG_SPANS": "false",
11          "JAEGER_SERVICE_NAME": "tenant-example",
12          "INPUT_STREAM": "stream.dshdemoshared",
13          "JAEGER_REPORTER_MAX_QUEUE_SIZE": "10000"
14      },
15      "instances": 1,
16      "singleInstance": false,
17      "needsToken": true,
18      "user": "1024:1024"
19  }
```

❮ SERVICES

🗑 DISCARD   ⬇ DOWNLOAD   ☁ DEPLOY

Test application

The kpn-tenant-example listens to the `dshdemoshared` topics on the `command` key. Two commands are supported:

- whoami
- restart

Responses to those commands are written on the `response` key.

Verify

You can set up an mqtt connection to verify:

```
mosquitto_sub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/response/#" \
--capath /etc/ssl/certs/ -d -P "`cat mqtt-token.txt`" \
-u $THING_ID -v
```

On macOS use

```
mosquitto_sub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/response/#"  \
--cafile /usr/local/etc/openssl/cert.pem -d \
-P "`cat mqtt-token.txt`" -u $THING_ID -v
```

Verify

```
mosquitto_pub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/command/" \
--capath /etc/ssl/certs/ -d -P "`cat mqtt-token.txt`" \
-u $THING_ID -l
```

On macOS use

```
mosquitto_pub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/command/" \
--cafile /usr/local/etc/openssl/cert.pem -d \
-P "`cat mqtt-token.txt`" -u $THING_ID -l
```

Type:

```
whoami
```

and see what's returned in the subscription.

Clean-up

- Remove your service after testing
- Select your service
- Select *configuration*
- Choose *destroy* and confirm

Thank you for participating.

End of Kafka Tutorial