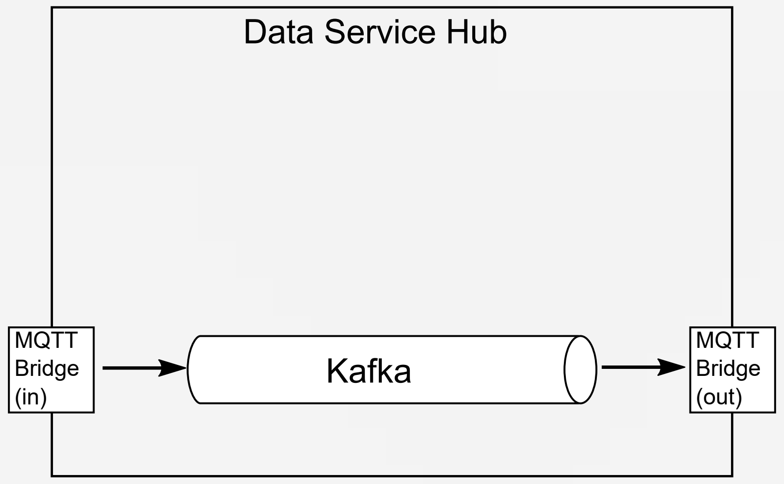
Streaming over Kafka

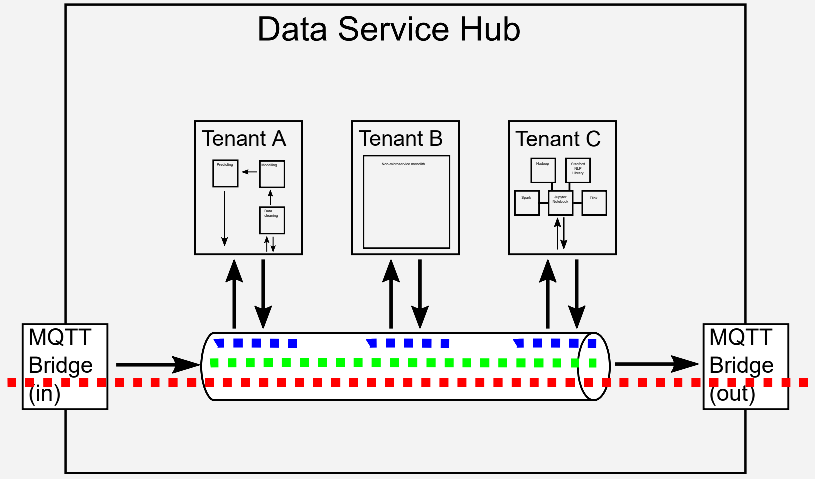
Kafka

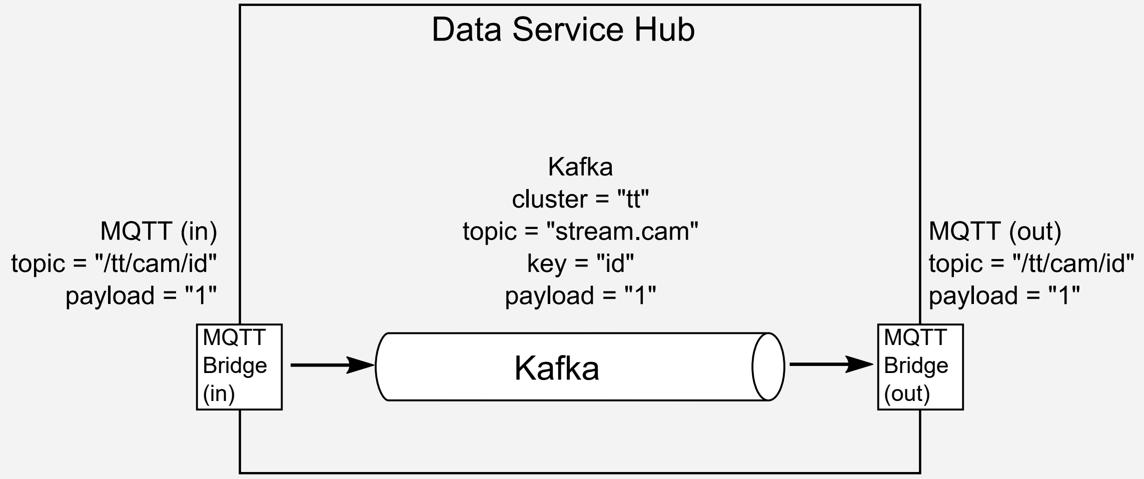


Kafka

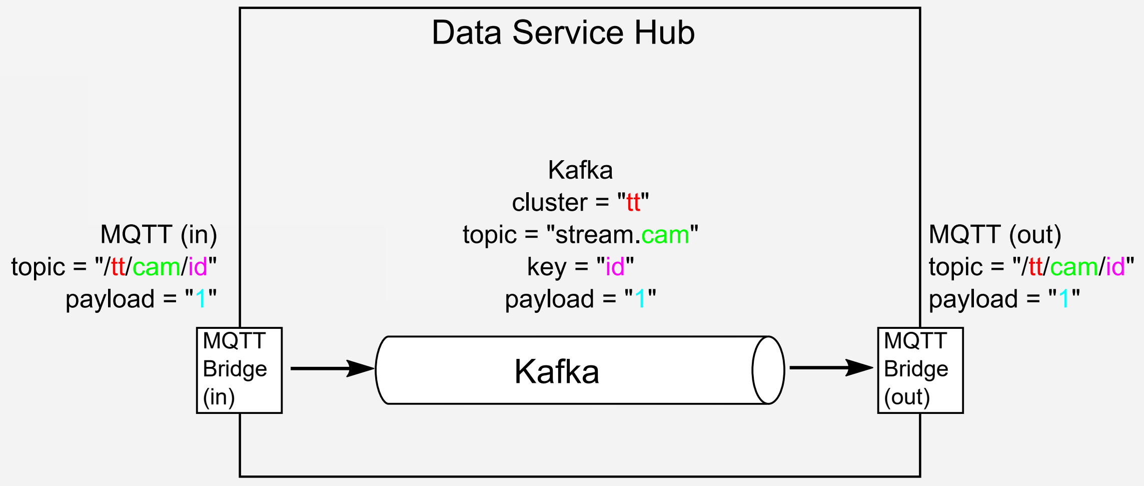
Three Kafka stream-types

- *stream.* topic
- *internal.* topic
- *scratch.* topic


Data Service Hub
Tenant A
Predicting
Modelling
Data cleaning
Tenant B
Non-microservice monolith
Tenant C
Hadoop
Stanford NLP Library
Spark
Jupyter Notebook
Flink
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
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 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 (when in doubt, use nano)
- 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>training</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 (username `training`, password: `ZWQ1ZWEzNDY2`):

```
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.

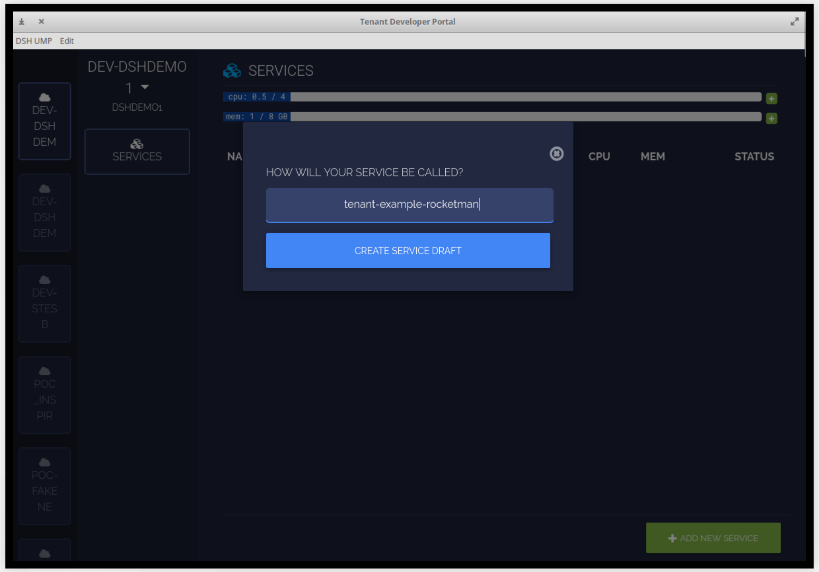
UMP

Connect/Setup UMP

- Click `+` to add a new environment
- Fill in the requested values
  - Environment name doesn't matter. `training` is good for now.
  - address is `https://api.poc.kpn-dsh.com`

Deploy application

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

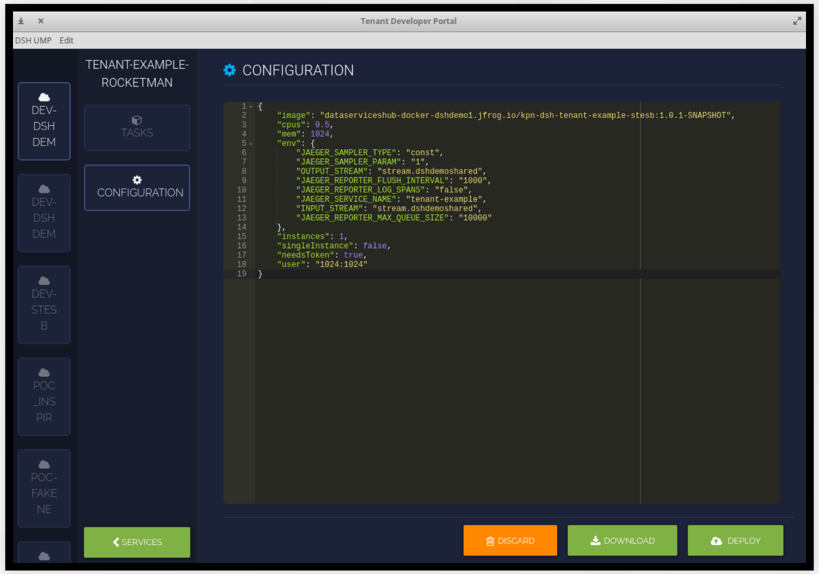


HOW WILL YOUR SERVICE BE CALLED?

tenant-example-rocketman

Deploy application

- create AMP definition (see `tenant-example.json`)
- modify the name of the image
- modify the user if needed (*verify* that the user is set to 1054:1054, and ask the trainer if this is still the case)
- and click *deploy*



```
{
    "image": "dataserviceshub-docker-dshdemo1.jfrog.io/kpn-dsh-tenant-example-stesb:1.0.1-SNAPSHOT",
    "cpus": 0.5,
    "mem": 1024,
    "env": {
        "JAEGER_SAMPLER_TYPE": "const",
        "JAEGER_SAMPLER_PARAM": "1",
        "OUTPUT_STREAM": "stream.dshdemoshared",
        "JAEGER_REPORTER_FLUSH_INTERVAL": "1000",
        "JAEGER_REPORTER_LOG_SPANS": "false",
        "JAEGER_SERVICE_NAME": "tenant-example",
        "INPUT_STREAM": "stream.dshdemoshared",
        "JAEGER_REPORTER_MAX_QUEUE_SIZE": "10000"
    },
    "instances": 1,
    "singleInstance": false,
    "needsToken": true,
    "user": "1024:1024"
}
```

Test application

The kpn-tenant-example listens to the `training` 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/training/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/training/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/training/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/training/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

Congratulations on completing the training!

Please do remember to fill in the evaluation questionnaire