redhat.

Open sourcing the IoT

Running EnMasse on Kubernetes

Paolo Patierno
Senior Software Engineer @ Red Hat
05/06/2017

Who am I ?

@ppatierno

- Senior Software Engineer @ Red Hat
  - Messaging & IoT team
- Lead/Committer @ Eclipse Foundation
  - Hono, Paho and Vert.x projects
- Microsoft MVP
- Technologies and protocols “globetrotter”
- Hacking low constrained devices in spare time
- Blogger and speaker about distributed systems, messaging, IoT and embedded “world”

Agenda

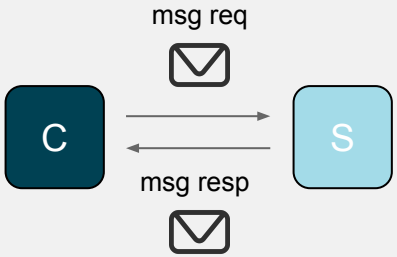
- Messaging … what ?
- Messaging … for IoT
- Messaging & IoT … in the cloud
- EnMasse : the open source MaaS !
- Running EnMasse …
  - Kubernetes
  - Azure Container Service
  - OpenShift

What is messaging?

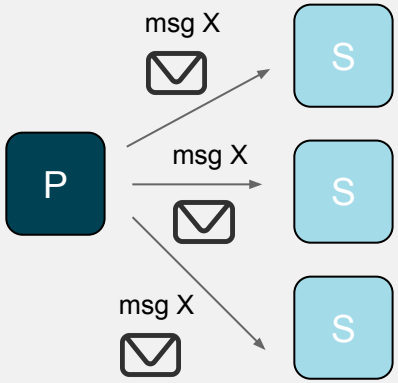
- It's about *messages* exchange
  - **Internally** in distributed systems
  - **Externally** between systems
- Communication at the ***application*** level
- Messages go from ***sender/producer/publisher*** to ***receiver/consumer/subscriber***
  - **Asynchronously**
  - Time **decoupling**
  - … or **directly** and **synchronously**

Messaging patterns

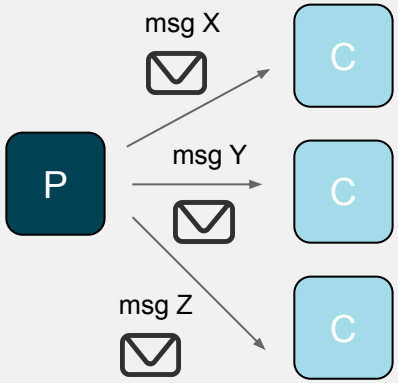
Request/Response



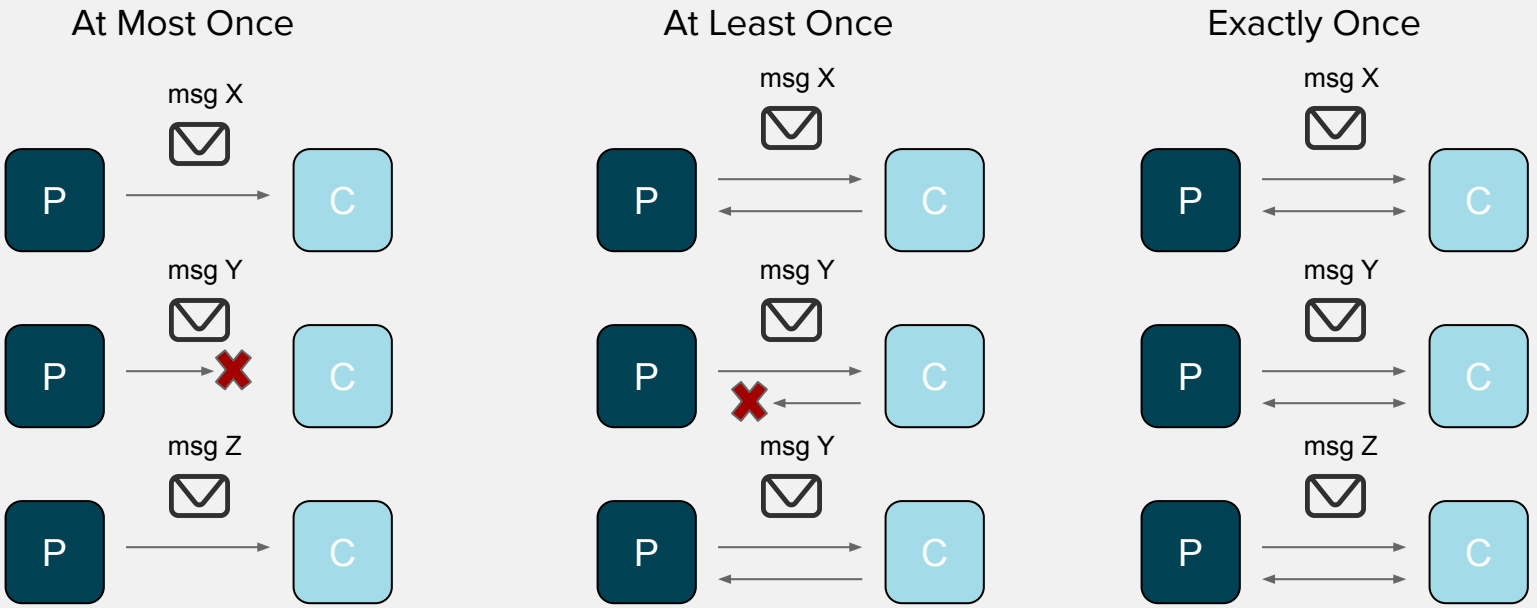
Publish/Subscribe



Competing Consumers



Quality of Service



IoT : messaging vengeance

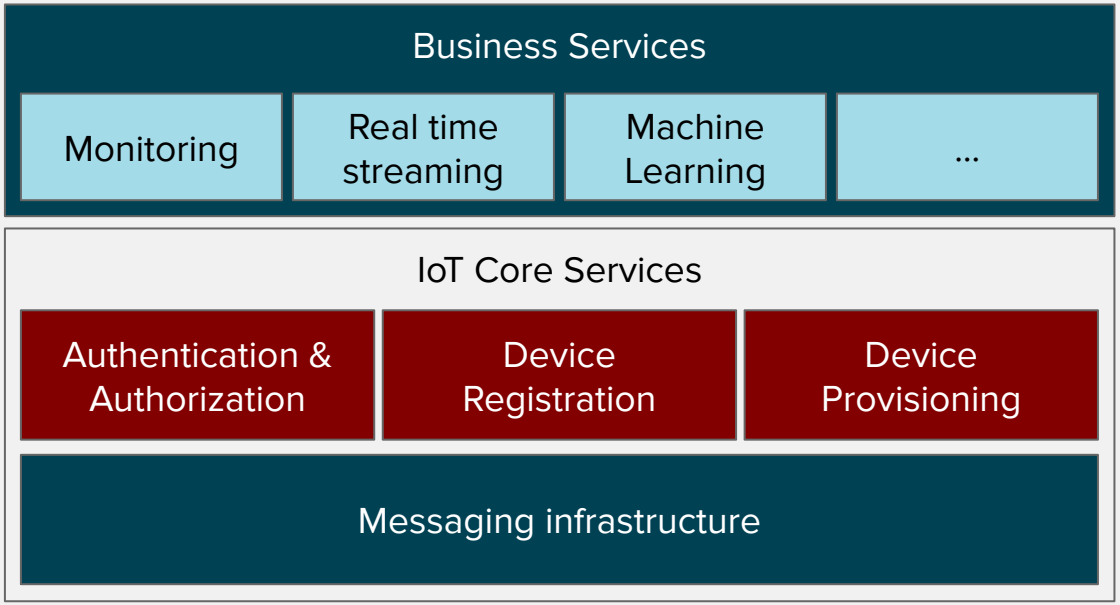
- ... maybe in the past ...
- ... **messaging** was not so cool for developers ...
- ... but today with **IoT** this is changed because ...
- ... **IoT is all about messaging** so ...

“Messaging vengeance” !

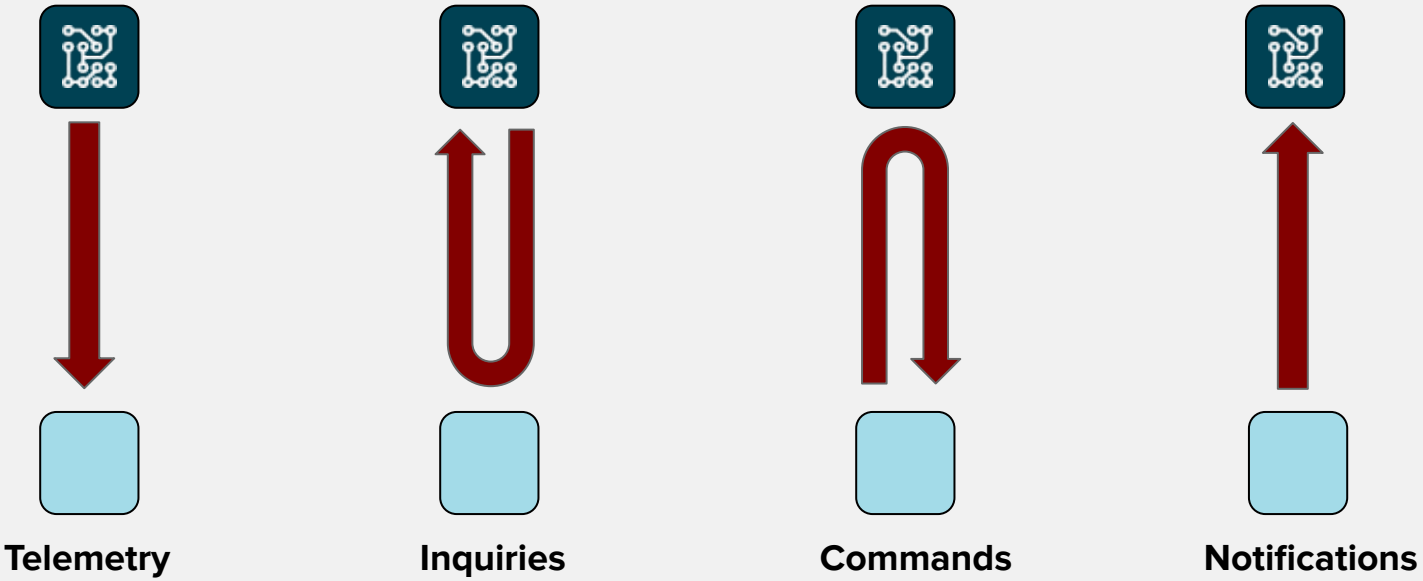
IoT : messaging as a “lever”

“give me a **scalable messaging platform**, and I shall move the **Internet of Things** world” (Archimedes)

What makes an IoT platform ?



IoT : communication patterns



IoT : communication patterns

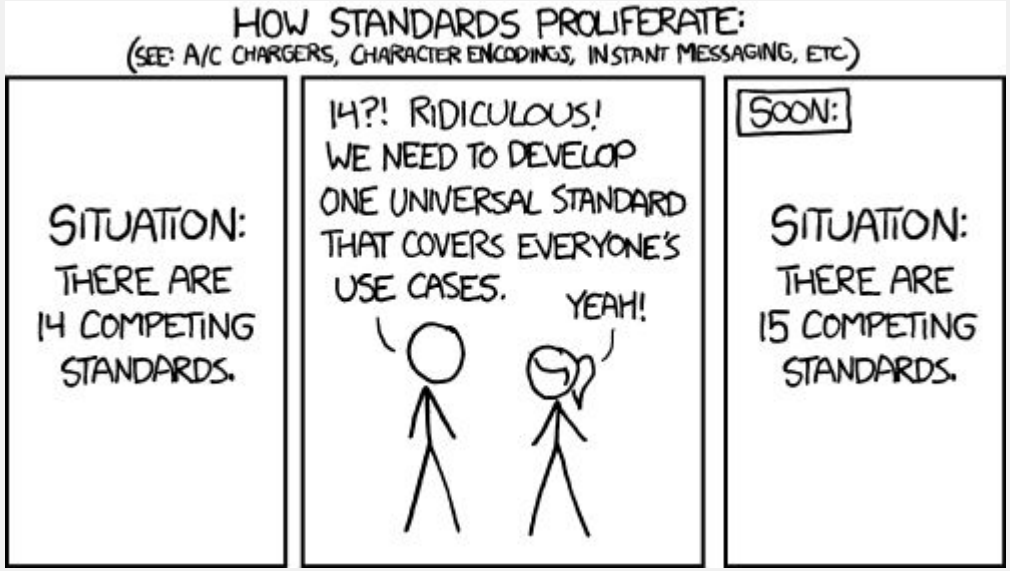
Messaging patterns & protocols

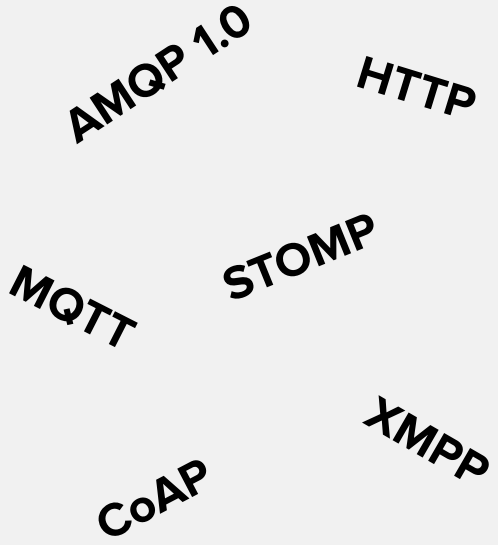
- **Telemetry** & **Notifications** are about ...
  - .... messaging **publish/subscribe**
- **Commands** & **Inquiries** are about ...
  - ... messaging **request/response**
- Different protocols (AMQP, MQTT, HTTP, ...) implement them in different way
  - As built-in support ...
  - ... or on top of it at application level
  - Read more on *“Strengths And Weaknesses Of IoT Communication Patterns”* *

* DZone IoT Guide : https://dzone.com/guides/iot-applications-protocols-and-best-practices

IoT : interoperability

Open standards





Messaging & IoT in the cloud

- Microsoft Azure
  - Service Bus + Event Hub
  - IoT Hub
- Amazon Web Services
  - Simple Queue Service (SQS)
  - AWS IoT
- Google
  - FireBase Cloud Messaging
  - IoT Core

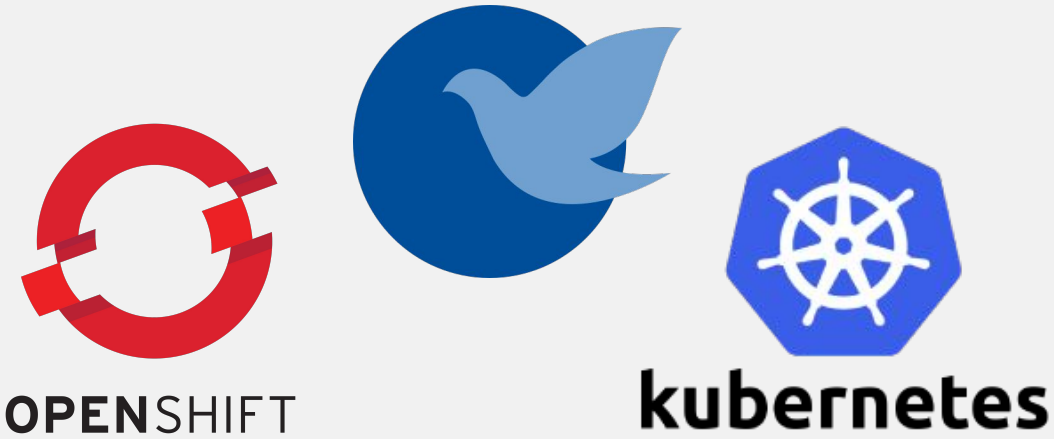
Cloud provider limitations

- They are not open source !
- Freedom of choice
  - On-premise or in the cloud
  - Ability to choose which cloud
  - Open Standards protocols allows users to choose client freely
- Migrating from one to the other can be complex

EnMasse

Messaging-as-a-Service

- Open source cloud messaging running on Kubernetes and OpenShift
- enmasse.io



EnMasse

Features

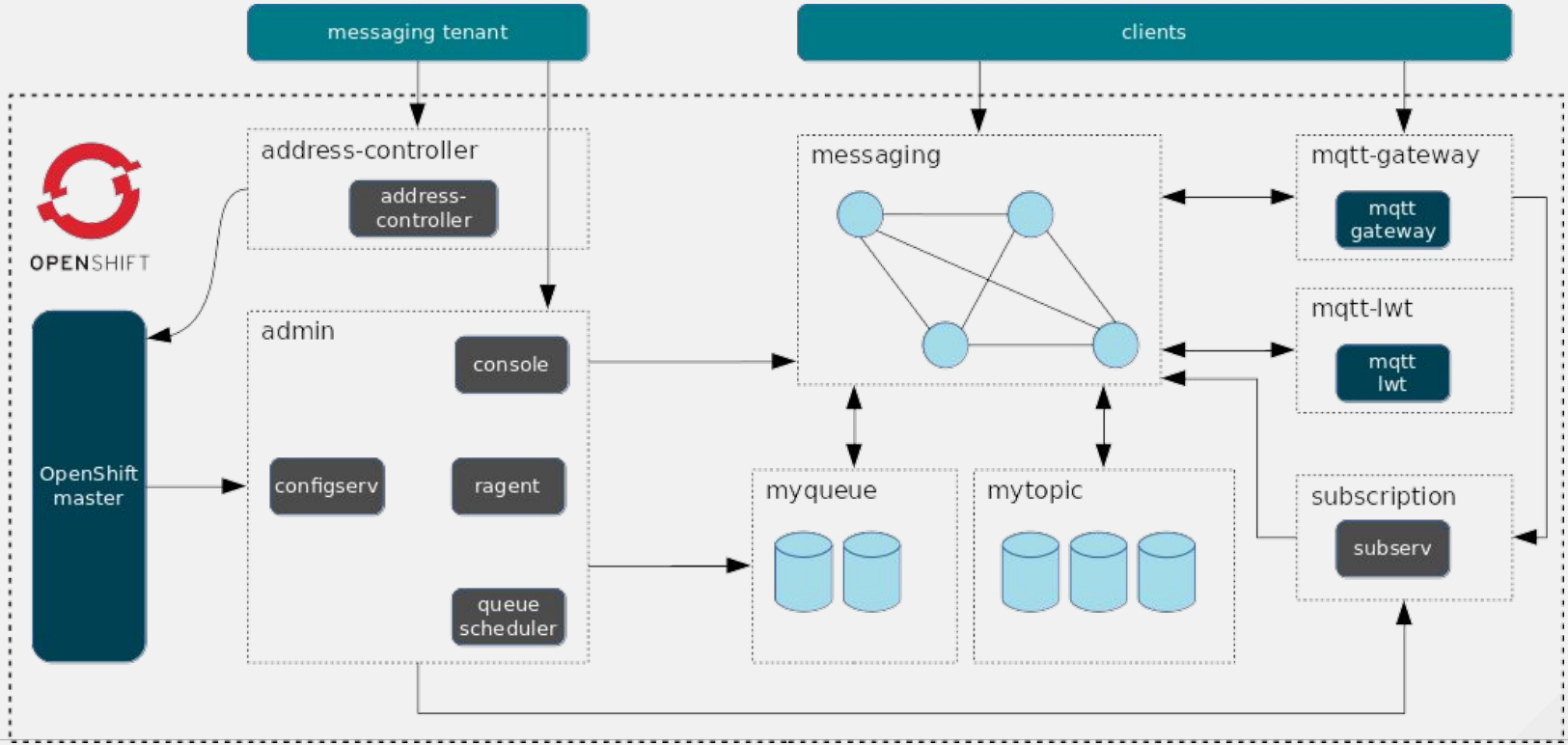
- Multiple communication patterns: **request/response**, **publish/subscribe** and **competing consumers**
- Support for **“store and forward”** and **direct** messaging mechanisms
- **Scale** and **elasticity** of message brokers
- **AMQP 1.0** and **MQTT** support
- Simple **setup**, **management** and **monitoring**
- **Multitenancy**: manage multiple independent instances
- Deploy **“on premise”** or in the **cloud**

EnMasse

Coming features

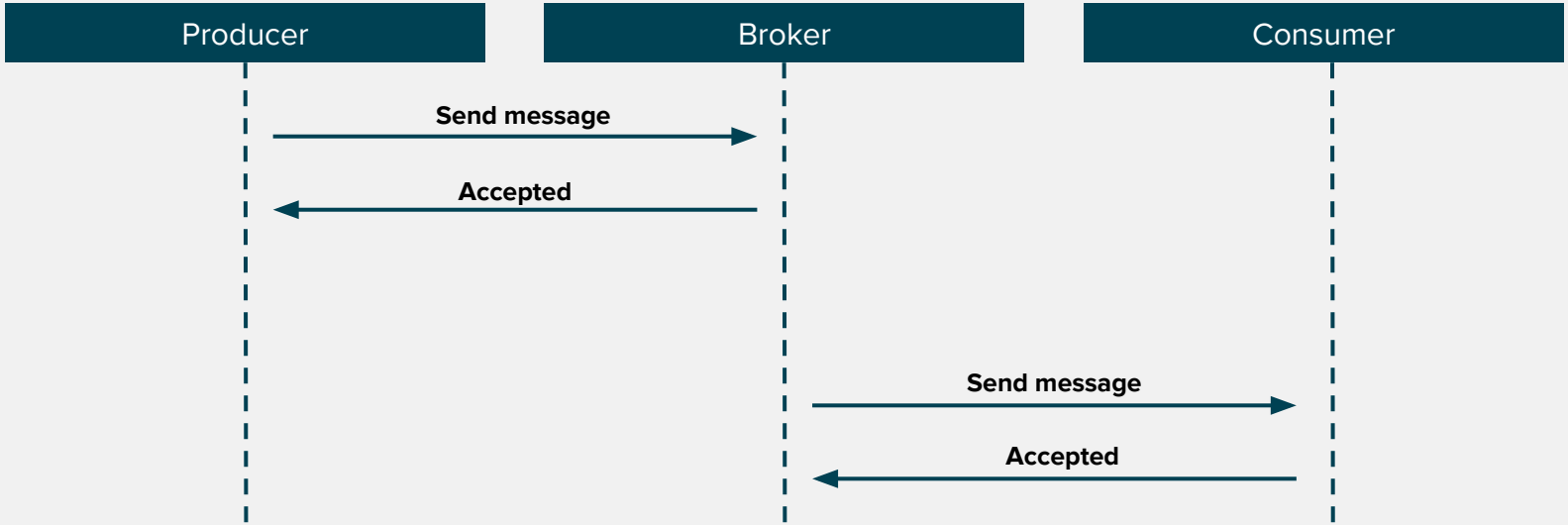
- Authentication and authorization
- Service broker API
- HTTP(S)
- Message grouping
- Distributed transactions
- Message ordering
- Multiple flavors
  - Apache Kafka
- ...

Architecture



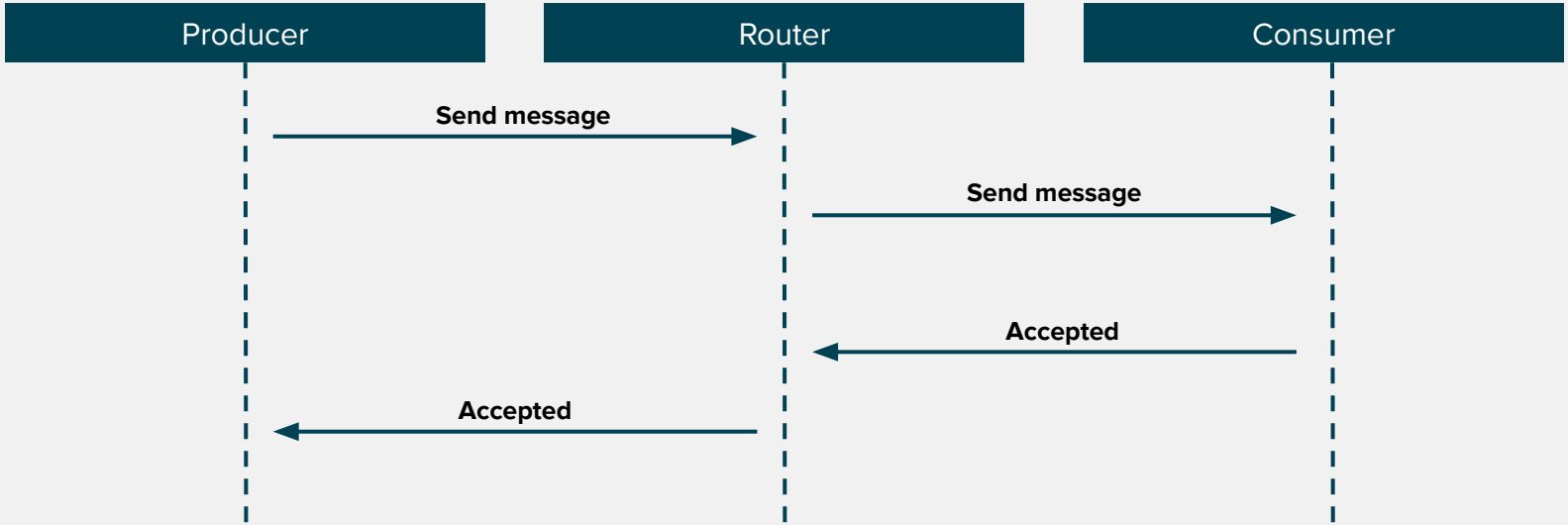
Routing vs “Broking”

Broker



Routing vs “Broking”

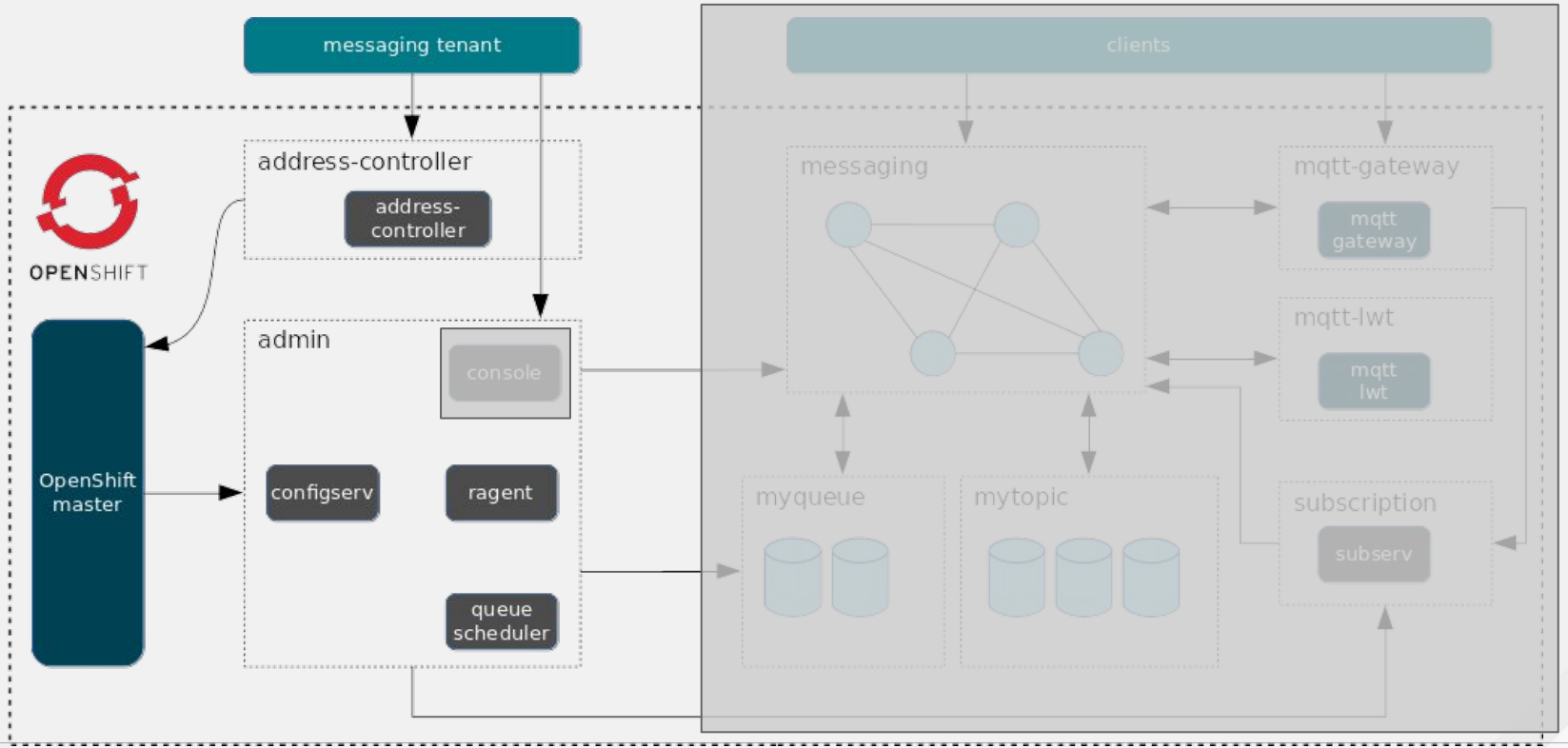
Router



MQTT over AMQP

- **MQTT gateway**
  - Handles connections with remote MQTT clients
  - Bridges MQTT - AMQP protocols
- **MQTT lwt**
  - Provides the “will testament” feature
  - In charge to recover & send the “will” if client dies
- It brings **MQTT features over AMQP** so ...
  - ... “will testament” works for AMQP clients as well

Configuration distribution



Configuration interface

```
{
    "apiVersion": "v3",
    "kind": "Address",
    "metadata": {
        "name": "myqueue"
    },
    "spec": {
        "store_and_forward": true,
        "multicast": false,
        "flavor": "vanilla-queue"
    }
}
```

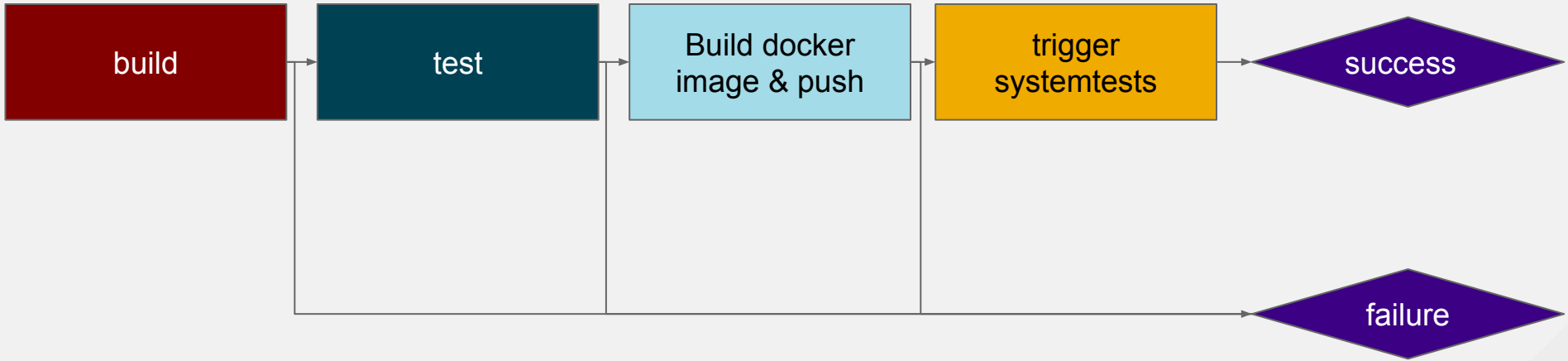
Configuration interface

```
{
    "apiVersion": "v3",
    "kind": "Flavor",
    "metadata": {
        "name": "vanilla-queue"
    },
    "spec": {
        "type": "queue",
        "Description": "Simple in-memory queue",
        "templateName": "queue-inmemory",
        "templateParameters": {}
    }
}
```

Continuous integration

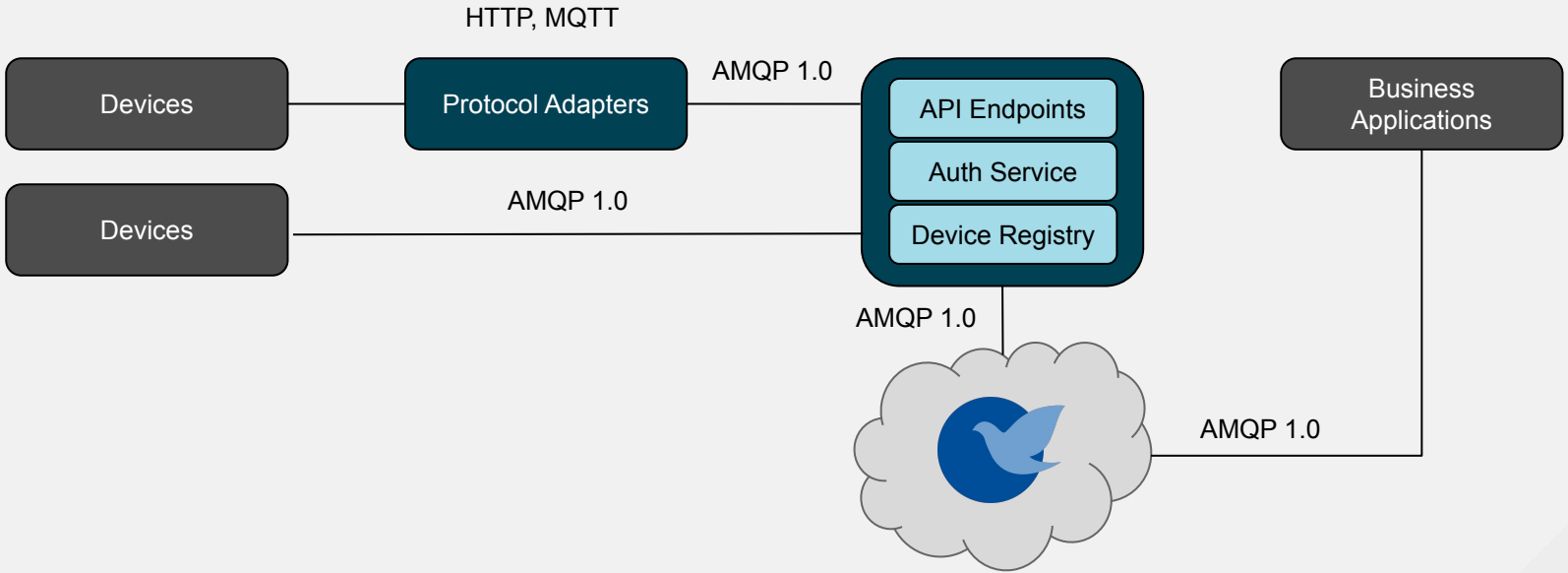
Continuous integration

Component build pipeline



Eclipse Hono

Connect. Command. Control



Eclipse Hono

IoT API

- Telemetry
  - used by devices to **send data downstream**
  - leverages on **“direct messaging”**
- Device Registration
  - used to make Hono **aware of devices** that will connect to the service
  - register, deregister, get information …
- Event
  - used by devices to **send event downstream**
  - differ from Telemetry on using **“store and forward”** (with TTL)
- Command & Control
  - used by applications to **send commands to devices**
  - command execution can be “just in time” or “deferred”

IoT : how to deploy ?

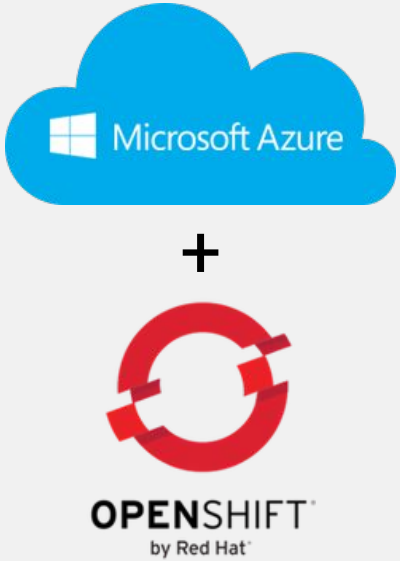
- “On premise” …
  - … maybe for a not so big solution
  - … ingesting few data and handling few devices
- “Cloud” …
  - … needs for more scalability
  - … don’t want to manage the infrastructure
- “Hybrid” …
  - … needs for processing at the edge
  - … needs for not making sensible data public

Azure Container Service

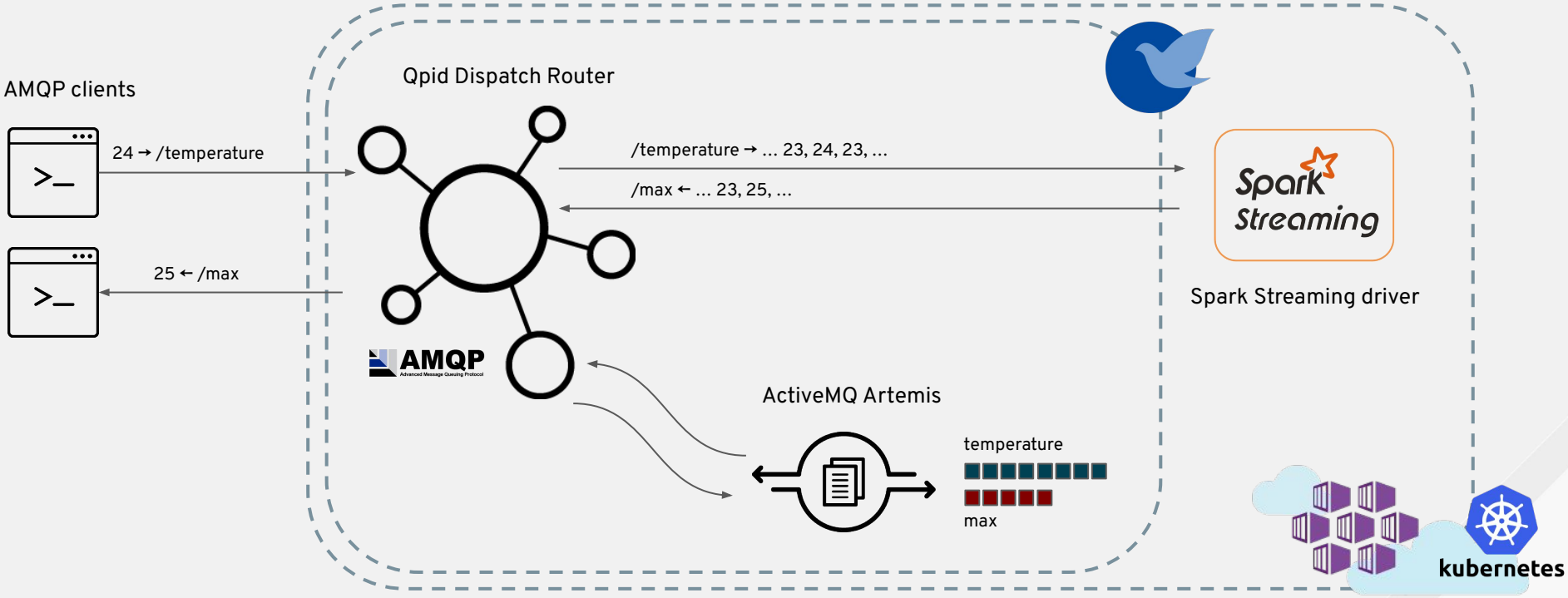
- A containers hosting solution
- Scale and orchestrate using …
  - Kubernetes
  - Docker Swarm
  - DC/OS
- Deploying a cluster using Azure CLI / portal
  - Resource group with VMs, load balancer, …
- Managing directly your preferred “orchestrator”
  - ACS provides you “only” the infrastructure

Azure & OpenShift

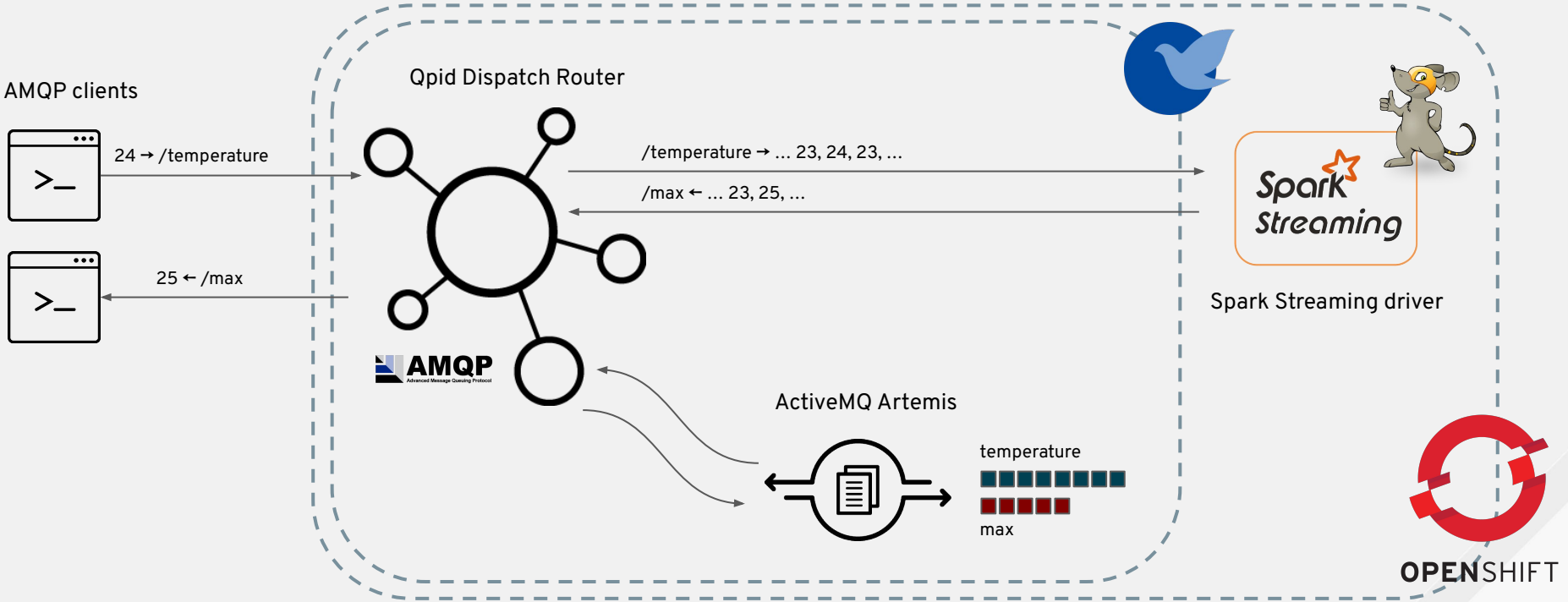
- OpenShift Origin
  - the upstream open source project
- OpenShift Container Platform
  - the Red Hat productized version
  - enterprise grade container platform



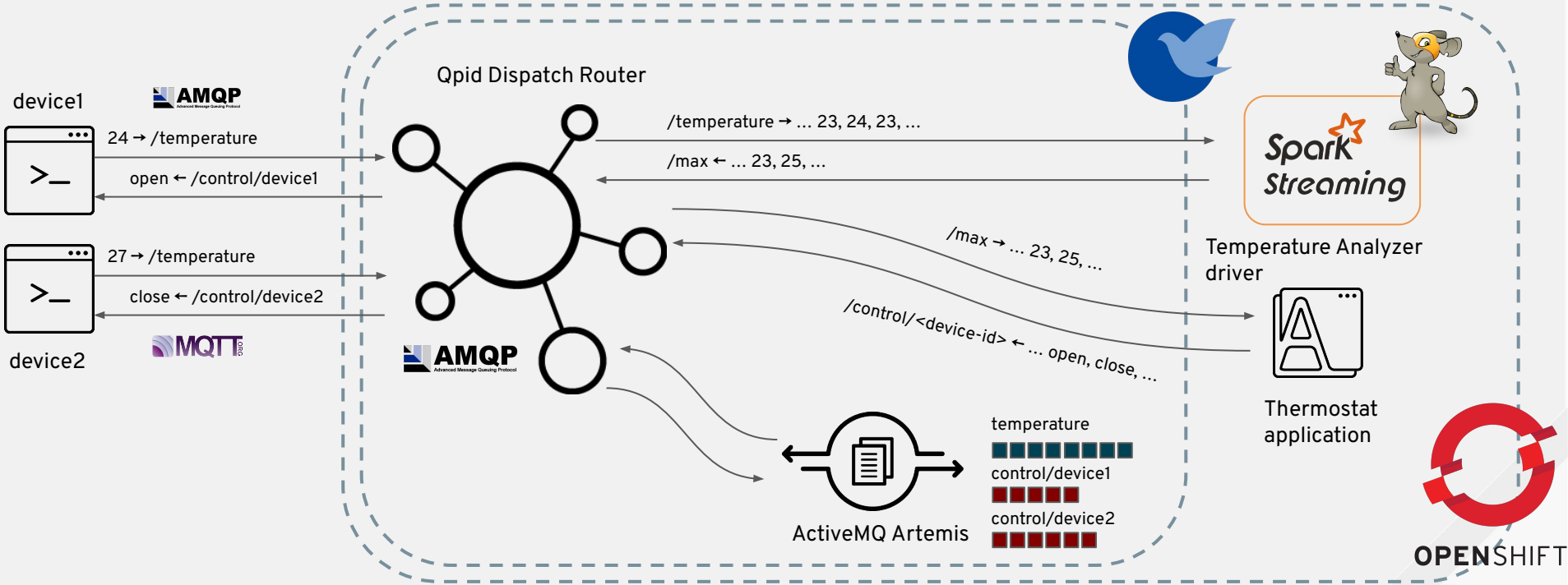
Demo : the deployment on Kubernetes with Spark



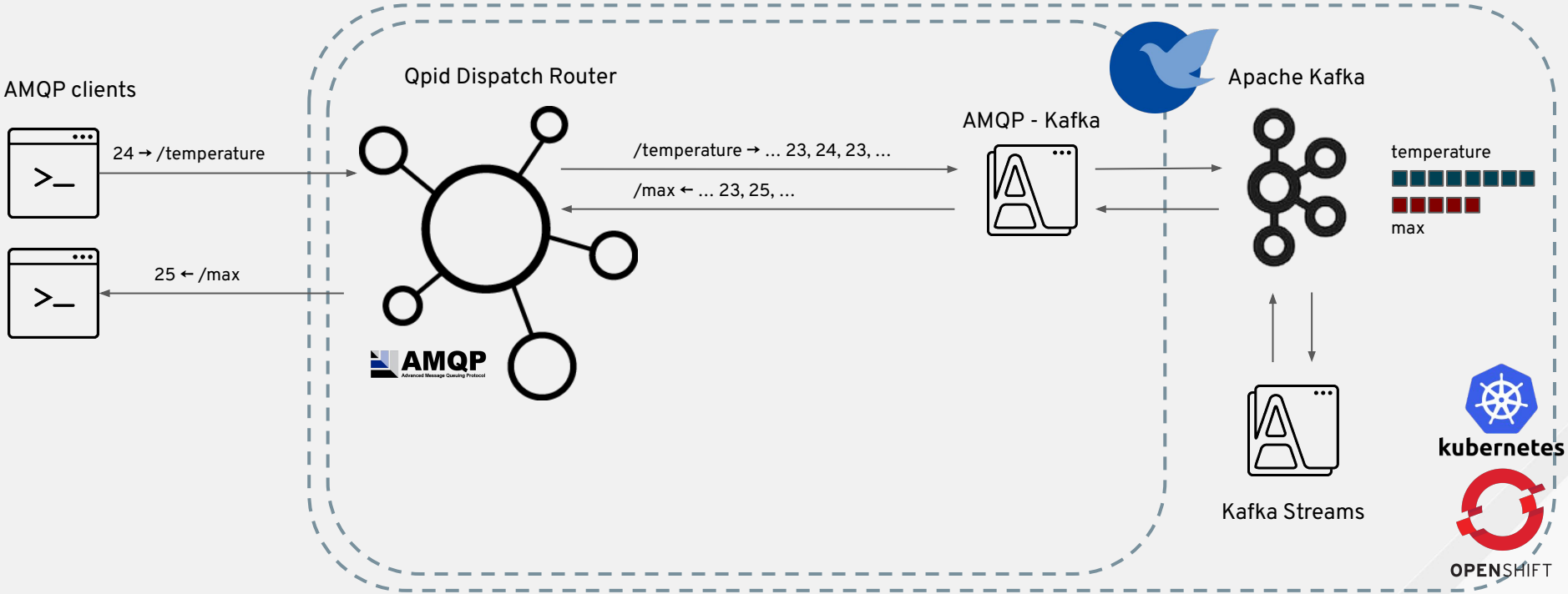
Demo : the deployment on OpenShift with Spark



Demo : the deployment on OpenShift with Spark



Demo : the deployment with Kafka



DEMO

Resources

- **EnMasse** : https://enmasseproject.github.io/
- **Qpid Dispatch Router** : http://qpid.apache.org/components/dispatch-router/
- **ActiveMQ Artemis** : https://activemq.apache.org/artemis/
- **Azure Container Service** : https://azure.microsoft.com/en-us/services/container-service/
- **OpenShift on Azure** : http://aka.ms/openshift
- **Eclipse Hono** : https://www.eclipse.org/hono/
- **Demo** : https://github.com/ppatierno/enmasse-spark-demo
- **My blog** : https://paolopatierno.wordpress.com/

redhat.

Thank you ! Questions ?

@ppatierno