Service Blueprint and Deployment for an IoT Cloud Integration Platform
Ludwigsburg – October 24th 2017

Stefano Morson – Senior Software Manager at Eurotech
Jens Reimann – Senior Software Engineer at Red Hat
Eclipse Kapua

Agenda

• Kapua Platform Overview
• Core IoT Services
• Modular Architecture
• Deployment Options
• Roadmap
Eclipse Kapua
Platform Overview

- Modular Integration Platform for End-to-End IoT Solutions
- Provides Multi Tenant Environment
- Focuses on Providing Comprehensive Management of Edge Devices
- Focuses on Data Collection and Routing for Real Time Data Processing and Storage
Eclipse Kapua
Platform Overview

Application Integration

Data Management
- Data Registry
- Data Store

Device Management
- Device Configuration
- Device Package Update
- Device Management Protocols
  Kura, LWM2M

Device Registry

Security
- Account Management
- Access Control

Device Connectivity
- AMQP
- CoAP
- MQTT

Message Routing

Administration

REST API

Device Mgmt Console

Eclipse Kapua
Platform Overview
Eclipse Kapua Architecture
Core IoT Services

- Account
- User
- Authentication & Authorization
- Device Registry
- Device Management
- Datastore
- Scheduler
- Device Jobs

- Identity Validation
- Credentials
- Device Registry
- Permissions
- Device Lifecycle
- Permission Check
- Device Connections
- Role \\
- Device Registry
- Remote Command Execution
- Channel Registry
- Asset/Channel Read/Write
- Metric Registry
- Package Install
- Message Store
- Message Write
- Generic Request/Response
- Bundle Start/Stop
- Command Execution
- Configuration Write
- Package Install/Uninstall
A service defines a bounded context
All interactions between services are handled through the Kapua Service interfaces
Can manage resources
No resource sharing between modules
A Kapua Service exposes a interface and a object model that conforms to a service blueprint
Serializable Model Objects (JSON and XML formats supported)
Kapua Services can handle persistent entities
Kapua Services can raise/handle events for choreography interactions
Support per tenant “scoped” configurations
Service implementations may optionally leverage Kapua Commons Library to reduce the burden of implementation
Eclipse Kapua Deployment

- Applications provided:
  - Web Console
  - RESTful API
  - Multiprotocol Messaging
  - Service Event Bus[1]

- Provide extension points for:
  - Web Console
  - RESTful API

- Data tier:
  - NoSQL Schema Less Engine (Elasticsearch)
  - Multiple SQL DB Support
  - Support for Database Migrations

- Provides demo deployments for:
  - Plain Docker Containers
  - Orchestrated Dockers (OpenShift Origin)
  - Virtual Machines (Vagrant)

• Note:
  1. Available with next release (1.0.0)
Eclipse Kapua Deployment
Default Deployment

Frontend Services
- Administration Console
  - Kapua Locator
  - ACCOUNT SERVICE
  - USER SERVICE
  - AUTH SERVICE
  - TAG SERVICE
  - DEVICE REGISTRY SERVICE
  - DEVICE MANAGEMENT SERVICES
  - DATASTORE SERVICE
  - SCHEDULER SERVICE
  - JOB SERVICE
  - Jetty
  - Docker

API Gateway
- Kapua Locator
- ACCOUNT SERVICE
- USER SERVICE
- AUTH SERVICE
- TAG SERVICE
- DEVICE REGISTRY SERVICE
- DEVICE MANAGEMENT SERVICES
- DATASTORE SERVICE
- Jetty
- Docker

Device Broker
- Kapua Locator
- ACCOUNT SERVICE
- USER SERVICE
- AUTH SERVICE
- TAG SERVICE
- DEVICE REGISTRY SERVICE
- DEVICE MANAGEMENT SERVICES
- DATASTORE SERVICE
- ActiveMQ
- Docker

Service Events Broker
- Artemis
- Docker

- NoSQL Datastore (Elasticsearch)
- SQL Persistency (Default H2)
Eclipse Kapua Extensibility
Application Integration

- The REST API allow programmatic access to the platform. It exposes all the platform functionality, including, tenant management, device management and data management.
- The Message Routes allow routing of telemetry data to a wide range of destinations.
- The Event Bus allow both backend services and external applications to react to relevant events triggered.
- Kapua supports authentication via OpenID. OpenID service providers can be used out of the box to provide authentication services.
Conterization
Plain old hardware
Conterization

Scaling up
Conterization

Virtualization
Conterization
Containers
Conterization
Building containers

Layer 1 + Layer 2 = Application Environment

Instances
Containerization
Orchestration – Kubernetes/OpenShift

- Node 1
- Node 2
- Node n
- Router

Schedule

Master

Environment
Application
Environment
Application
Environment
Application
Environment
...
Micro services
Kapua 0.3.0 / 1.0.0

Node 1
- Storage
- Broker
- Web Console
- API Server

Node 2
- Web Console
- API Server

Node n
- API Server

Router

MQTT
HTTP

Web Console
API Server
Micro services
Kapua beyond 1.0.0

Node 1
- Storage
- Device
- User / Groups
- AuthZ / AuthN
- ...
- Web Console
- API Server

Node 2
- Device
- Device
- Web Console

Node n
- API Server

Router
- MQTT
- HTTP
Eclipse Kapua Roadmap

Microservices

- **Frontend Services**
  - Administration Console
    - Kapua Locator
    - Jetty
    - Docker
  - API Gateway
    - Kapua Locator
    - Jetty
    - Docker
  - Device Broker
    - Kapua Locator
    - ActiveMQ
    - Docker
  - Service Events Broker
    - Artemis
    - Docker

- **Service Discovery**

- **µService Endpoint**
  - ACCOUNT SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - USER SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - AUTH SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - TAG SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - DEVICE REGISTRY SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - DATASTORE SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - SCHEDULER SERVICE
    - Kapua Locator
    - JVM
    - Docker
  - JOB SERVICE
    - Kapua Locator
    - JVM
    - Docker
• Services advertise themselves to the service discovery
• Service discovery is delegated to the Locator
• Client code is agnostic of the location of the services it uses
• Client code is agnostic of the protocol used to communicate with the microservice
• Should have minimal assumptions on the container model
Eclipse Kapua
Service Events

- Backend services should be able to exchange Events to react to changes in the global status of the backend.
- Service Events are raised upon the execution of a service method.
- Event management guarantees that the global status of the system is eventually consistent.
Eclipse Kapua Roadmap
Integration with Other IoT Projects

• Messaging: Eclipse Hono
  – https://www.eclipse.org/hono/

• Device Protocols:
  – Eclipse Leshan
    • https://www.eclipse.org/leshan/
  – Eclipse Kura
    • https://www.eclipse.org/kura/

• Software Updates: Eclipse hawkBit
  – https://projects.eclipse.org/projects/iot.hawkbit

• Digital Twins: Eclipse Ditto
  – https://projects.eclipse.org/projects/iot.ditto
Eclipse Kapua
Open Discussion

Interested to Discuss More?

Join Us this Evening
at
the Birds of Feather (BoF)
Thank You!

www.eurotech.com
www.redhat.com

Want to Learn More?
Meet Eurotech at Booth #2
Evaluate the Sessions

Sign in and vote at eclipsecon.org

-1 0 +1