From cloud to edge:
Why cloud-native application development matters in supporting IoT

David Bericat
Chief Architect Data Analytics (AI/ML) • Edge • IoT
Global Partner Solutions
Red Hat

Erik Jacobs
Principal Technical Marketing Manager, OpenShift Cloud Platforms BU
Red Hat

Thursday May 9th
THE INTERNET IS ALL THINGS
YOU SIMPLY DON’T REALIZE IT YET...
AGENDA

● What has Red Hat been doing in IoT

● From IoT to end-to-end Analytics (AI/ML) to Edge computing

● Red Hat’s next generation cloud-computing development

● Dreaming exercise: serverless & FaaS at the “edge”
Customers use our parts of our portfolio in a variety of ways to address IoT challenges on a hybrid cloud infrastructure.

We do not claim to provide an IoT platform. We work with our commercial partner ecosystem to define more complete solutions and bring them to market.

We take a leadership role in open source communities to build IoT technology.
KEY FUNCTIONALITY
FOR AN END-TO-END IoT ARCHITECTURE

Device Management & Connectivity
Securely connect, authenticate and manage disparate connected devices that speak different protocols

Intelligent Edge Processing & Analytics
Apply analytics at the edge with machine learning and business rules to enable local, low-latency decision making

Advanced Analytics & Machine Learning
Centralize IoT data processing, analytics and machine learning to enable deep business insights and actionable intelligence

Business & Application Integration
Enable integration with enterprise and business applications to bridge the gap between OT and IT and reduce complexity

End-to-End Security & Compliance
Tools to enable end-to-end data security, compliance, authorization and authentication
OPEN END-TO-END IoT ARCHITECTURE
INTEGRATING OT, IT, DATA MANAGEMENT, ANALYTICS & APPLICATIONS

- Modular, secure, end-to-end architecture
- Streaming analytics and machine learning
- Open, interoperable on hybrid cloud
- Modern application development and agile integration

APPLICATION DEVELOPMENT, DELIVERY, & INTEGRATION

DATA MANAGEMENT & ANALYTICS PLATFORM

Cloud-native apps
Traditional apps

public, private, hybrid cloud
END-TO-END ANALYTICS
PUSHING AI/ML FROM CORE TO EDGE

Protocol Translation
Intelligent Filtering
Aggregation
Routing

CONNECTED “THINGS” IoT EDGE IoT INTEGRATION HUB

Application Integration
Telemetry Data
ML Model
ML Model

Actions
Prediction / Alert

APPLICATION DEVELOPMENT, DELIVERY, & INTEGRATION
Cloud-native apps
Traditional apps

Deep data analysis & insights

DATA MANAGEMENT & ANALYTICS PLATFORM

Data Ingest
Real-Time Processing
Data Storage

Real-Time Analytics
Data Security

Machine Learning
HOW RED HAT PORTFOLIO CAN HELP

CONNECTED “THINGS”

IoT EDGE

- RED HAT DECISION MANAGER
- RED HAT AMQ
- RED HAT FUSE
- LINUX
- RED HAT ENTERPRISE LINUX
- Apps
- Edge analytics
- Machine learning

Telemetry

IoT INTEGRATION HUB

- RED HAT DECISION MANAGER
- RED HAT AMQ
- RED HAT FUSE

Telemetry

APPLICATION DEVELOPMENT, DELIVERY, & INTEGRATION

- RED HAT SCALE
- RED HAT FUSE
- RED HAT JBOSS MIDDLEWARE

OPENSHIFT

by Red Hat

DATA MANAGEMENT & ANALYTICS

Machine learning model

#redhat #rhsummit
AGILE INTEGRATION
STANDARDIZING THE “WILD, WILD WEST”
PORTABILITY ACROSS ANY INFRASTRUCTURE

CONNECTED “THINGS”

IoT EDGE

IoT INTEGRATION HUB

DATA MANAGEMENT & ANALYTICS PLATFORM

- Linux containers
- Kubernetes-based orchestration
- Deployment flexibility
- Automation
- Scalability
- Persistent storage

**OPENSHIFT by Red Hat**

- Telemetry
- Management

**IoT INTEGRATION HUB**

- Container
- IoT Services
- OS dependencies

- Container
- IoT Services
- OS dependencies

- Container
- IoT Services
- OS dependencies

- Container
- IoT Services
- OS dependencies

- Container
- IoT Services
- OS dependencies

- Container
- IoT Services
- OS dependencies

**CONNECTED “THINGS”**

- Laptop
- Bare Metal
- Virtual Machine
- Private Cloud
- Public Cloud

**#redhat #rhsummit**
APPDEV & CONTINUOUS DELIVERY

APPLICATION DEVELOPMENT, DELIVERY, & INTEGRATION

- Cloud-native applications
  - Developer services
  - Orchestration
  - DevOps

- Traditional applications

- Application development & management
- Polyglot, multi-language support
- Self-service provisioning
- Linux containers
- Container portability
- Deployment flexibility
- Kubernetes-based orchestration
- Automation
- Scalability
- Persistent storage

Self-service Provisioning
Consistent environments
Automated build & deploy
CI/CD pipelines
Configuration management
App logs & metrics

SPRING & JAVA EE
MICROSERVICES
FUNCTIONS
LANGUAGES
DATABASES
APPLICATION SERVICES
LINUX
WINDOWS*

* coming soon

#redhat #rhsummit
APPDEV WITH CODEREADY WORKSPACES
PLANNING, CREATING, AND DEPLOYING HYBRID CLOUD SERVICES

The collaborative OpenShift-Native IDE. Free for any customer of OpenShift Dedicated or OpenShift Container Platform.

- **Container Workspaces**: Workspace replicas to end “works on my machine” and enable team collaboration.
- **DevOps Integrations**: Reference developer workspaces from any issue, failed build, or git notification.
- **Protect Source Code**: Full access to source code without any of it landing on hard-to-secure laptops.

**Use It To**: Replace VDI for devs, and enable true container-based DevOps.

Based on the open Eclipse Che project
Red Hat Linux and Application Infrastructure
Plugin model for extensibility
Serverless support (coming later)
EDGE CONTINUOUS DEPLOYMENTS

1. Create App / microservice / function
   Who: Developers
   Where: CodeReady Workspaces

2. Push to the edge

3. Execute at the edge
   App / Microservice / Function
   Developer → Git Server → Artifact Repository → Release Manager
   → OpenShift CICD Pipeline (Jenkins)
   → OpenShift Image Registry
   → OpenShift Cluster
   → Non-Prod (Dev, Test, UAT) → Prod
WHAT IS THE EDGE ANYWAYS?

**EDGE**
many small sites (scale-out)

- **better economies-of-scale and resource sharing efficiency**
- Better bandwidth, latency, resiliency, data sovereignty

**CORE**
few, large sites (scale-up)

End-User Premise Edge

- *Device Edge*
- *Infrastructure Edge*

“last mile”

Provider Edge

- **Provider Device Edge**
- **Provider Access Edge**
- **Provider Aggregation Edge**

Provider/Enterprise Core

- **Regional Data Center**
- **Core Data Center**

**What is Edge Computing?**

**Why choose Red Hat for Edge computing?**
## RED HAT EDGE COMPUTING PoV

<table>
<thead>
<tr>
<th>Provider/Enterprise core</th>
<th>Traditional “non-edge” tiers, owned and operated by public cloud providers, telco service providers, or large enterprises.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core datacenter</td>
<td>Larger centralized presence.</td>
</tr>
<tr>
<td>Regional datacenter</td>
<td>Smaller regional presences.</td>
</tr>
<tr>
<td>(Service) Provider edge</td>
<td>Edge tiers between the core/regional data centers and the last mile access, commonly owned and operated by a telco or internet service provider and from which this provider serves multiple customers.</td>
</tr>
<tr>
<td>Provider aggregation edge</td>
<td>Higher tiers aggregating multiple smaller presences.</td>
</tr>
<tr>
<td>Provider access edge</td>
<td>For sites terminating the access link.</td>
</tr>
<tr>
<td>Provider device edge</td>
<td>Standalone non-clustered devices.</td>
</tr>
</tbody>
</table>

### End-user premises edge
- Edge tiers on the end-user (or customer) side of the last mile access.
  - Enterprise edge (e.g. a retail store, a factory, a train).
  - End-user premises edge - indicating ownership/control (e.g. a residential household, a car).

### Infrastructure edge
- 1-to-N nodes of generic, clustered compute/storage infrastructure accessible in a cloud-like manner.

### Device edge
- Standalone (non-clustered) devices that directly connect sensors/actuators via non-internet protocols.

*** Minimum of 3 nodes required today
BEST EDGE COMPUTING FOUNDATION?

EDGE
many small sites
(scale-out)

CORE
few, large sites
(scale-up)

better economies-of-scale and
resource sharing efficiency

Better bandwidth, latency,
resiliency, data sovereignty

End-User Premise Edge

Device Edge

Infrastructure Edge

Provider Edge

Provider Device Edge

Provider Access Edge

Provider Aggregation Edge

Provider/Enterprise Core

Regional Data Center

Core Data Center

#redhat #rhsummit
KNATIVE
SERVERLESS BUILDING BLOCKS

"...an extension to Kubernetes exposing building blocks to build modern, source-centric, and container-based applications that can run anywhere".

**Build**
A pluggable model for building artifacts, like jar files, zips or containers from source code.

**Serving**
An event-driven model that serves the container with your application and can "scale to zero".

**Eventing**
Common infrastructure for consuming and producing events that will stimulate applications.
KNATIVE
EVENTING FLOW

Data plane (implementation) forwarding

receive adapter

channel receiver

channel subscriber
delivery sidecar

Service

Kafka

channel subscriber
delivery sidecar

Service

channel receiver

CloudEvents over HTTP POST

Implementation specific protocol

Pod boundary
SERVERLESS ADOPTION
OK, WE HAVE THE BUILDING BLOCKS, NOW WHAT?

- 62.9% Node.js
- 20.8% Python
- 6.4% Go
- 6.1% Java
- 3.8% C#

QUARKUS
SUPERSONIC SUBATOMIC JAVA

A Kubernetes Native Java stack tailored for GraalVM & OpenJDK HotSpot, crafted from the best of breed Java libraries and standards

100x faster startup, 10% of the memory in native mode
10x faster and ½ the of the memory on Hotspot/OpenJDK

Quarkus with GraalVM
Quarkus with OpenJDK
Traditional cloud-native stack

Memory (RSS) in Megabytes

REST
13 MB 74 MB 140 MB

REST + CRUD
35 MB 130 MB 218 MB

Boot + First Response Time in Seconds

REST
.014 sec .75 sec

REST + CRUD
.055 sec 2.5 sec

MEMORY & BOOT + FIRST RESPONSE TIME

Minimal footprint Java applications -> Native vs Quarkus+OpenJDK JVM vs Traditional JVM
Quarkus enables Java developers to easily use the most popular frameworks and standards directly on SubstrateVM without any hassle.
APACHE CAMEL K

SERVERLESS INTEGRATION

- A lightweight platform for directly running Camel integration DSL in a cloud-native way
- Based on (ex Core OS) operator-sdk
- Works on OpenShift and Kubernetes
- Create Event Sources, use EIPs, define Integration Functions
CAMEL K
OVERVIEW

Time to run an integration using different strategies (in seconds)

File routes.groovy

```groovy
from("telegram:bots/bot-id")
    .transform()...
    .to("kafka:topic");

from("kafka:topic")
    .to("http://my-host/api/path");
```

```bash
$ kamel run routes.groovy
```

Lower is better :)
DEVICE EDGE
WHAT CAN WE DO WITHOUT KNATIVE/K8s?

- Run native functions in containers at the edge
- Benefit from low footprint and fast startup
CALL TO ACTION

- https://try.openshift.com/
- https://developers.redhat.com/
- https://knative.dev/docs/
- https://quarkus.io/
- https://github.com/apache/camel-k

LET'S BUILD CLOUD NATIVE APPS
THANK YOU

linkedin.com/company/Red-Hat
youtube.com/user/RedHatVideos
facebook.com/RedHatInc
twitter.com/RedHat
CASE STUDY - Smart Manufacturing - Deep Dive

Integrating open source IoT operating technology, data management, analytics, and applications

Field Protocol

- Device connectivity
- Data transformation
- Intelligent routing
- Business logic
- Edge analytics & real-time decisions

TCP/IP

Kafka

Data Science Workbench

Model Deployment

Unified Services Layer

Sentry

YARN

TCP/IP

TCP/IP

Kafka

Spark

Kudu

Spark

Impala

- Data ingest
- Stream / batch processing
- Persistent data storage
- Machine learning and real-time analytics

TCP/IP

- Device management, security, and access control
- Data aggregation
- Event processing
- Integration services (API's)
Knative Overview - Serving

- **Configurations** represent the ‘floating HEAD’ of a history of **Revisions**

- **Revisions** represent immutable snapshot of code and configuration

- **Routes** configure ingress over a collection of Revisions and/or Configurations

- **Services** (nope, not K8s services) are top-level controllers that manage a set of Routes and Configurations to implement a network service
SERVERLESS ADOPTION

- 62.9% Node.js
- 20.8% Python
- 6.4% Go
- 6.1% Java
- 3.8% C#

ENTERPRISE JAVA WAS DESIGNED FOR 3-TIER ARCHITECTURE

- **Presentation Logic**
  - HTML
  - Javascript
  - Web

- **Business Logic**
  - Inventory
  - Catalog
  - Cart
  - Promo
  - Ratings
  - Orders

- **Data Access Logic**
  - Data Access

**APPLICATION SERVER / JVM**

**OPERATING SYSTEM**
HOW APPLICATION SERVERS DEPLOYED JAVA APPS

APPLICATION SERVER / JVM

OPERATING SYSTEM
MICROSERVICES CHANGED HOW WE DEPLOY APPS

CONTAINER ORCHESTRATION
THE HIDDEN TRUTH ABOUT JAVA IN CONTAINERS

CONTAINER ORCHESTRATION

https://developers.redhat.com/blog/2017/03/14/java-inside-docker/
Function as a Service
QUARKUS NATIVE DEPLOYMENT

Traditional Cloud-Native Java Stack

Quarkus

Container Orchestration
HOW IS THIS POSSIBLE?
GraalVM™
Polyglot, Native or JVM, Embeddable
DEVELOPING NATIVE APPLICATIONS FOR SUBSTRATEVM IS PAINFUL

Quarkus enables Java developers to easily use the most popular frameworks and standards directly on SubstrateVM without any hassle.
FRAMEWORK OPTIMIZATIONS

- Moved as much as possible to build phase
- Minimized runtime dependencies
- Maximize dead code elimination
- Introduced clear metadata contracts
- Spectrum of optimization levels
  (all -> some -> no runtime reflection)

Optimizations benefit both GraalVM (SVM) and HotSpot
bwm - https://www.flickr.com/photos/pinemikey/8850948516
android - https://www.flickr.com/photos/microsiervos/15350193299
nest - https://www.flickr.com/photos/161156731@N05/45096474951
SECTION BREAK
SECTION BREAK