Death of the ESB: 10 Pitfalls in the transition to Cloud-Native Integration

Rahul Kamdar, 13th Sept 2017 – Open Source Summit 2017 (North America)
What are we going to talk about today?

- **ESBs ??**
  A quick recap on what are ESBs

- **Cloud & Edge Native**
  What has changed? And why does it matter?

- **Pitfalls & Lessons**
  Common mistakes and how do we address them

- **About Me/Us**
  Who are we and how can we help you or where can you send feedback
Evolution of Integration Patterns

**WOA**
- **What?** Microservices, Web APIs
- **How?** In-Memory, Multicore, REST, JSON
- **Why?** Mobile, Cloud, API Economy, IoT

**SOA**
- **What?** ESB, Real-time
- **How?** XML, SOAP, WS-* Process Modeling
- **Why?** E-Commerce, BPM

**EAI**
- **What?** Adapters, Real-time
- **How?** Client-Server, Messaging Middleware
- **Why?** ERP, Analytics

**ETL**
- **What?** Batch Jobs, Non-realtime
- **How?** Mainframe, ETL, Databases
- **Why?** Data Processing, MIS

“Application integration can be described as making independently designed applications work together.”
Enterprise Service Bus (ESB)

- Architecture for application integration
- Enables discrete heterogeneous apps & services to communicate with each other and exchange data
- Central (Messaging + App) Bus + tech-specific “Adapters”
- Leverages SOA (Service Oriented Architecture)
- Performs a set of common functions, such as
  - Data transformation and manipulation
  - Supports Message Exchange Patterns (MEP)
  - Mediation & Protocol Conversion
  - Service Orchestration
Cloud & Edge Native Integration

- Follows a set of specific patterns
  - Small, independent, granular services
  - Built around business capabilities/reqs
  - Modern-infra friendly – containers, modularized build/deploy/manage, automation ready, easy to scale up/down based on SLAs/rules
  - Clean separation of state, config data
- Consistently deploy/communicate from anywhere – no dependency on server/infra/language
  - NOT JUST RETROFITTING EXISTING TECH!
#1 – Reuse SOAPful APIs & Server-side conversations as-is

**Strangler Pattern**

**Compose APIs**

- Adapters
- SOAP Services
- Wrap
- Proxy

(referencing Martin Fowler’s work in 2004)
#2 – Key and certificate management are not part of the things that change

- Introspect and define types of service communication
- Map out existing process for certificate request, approval, deployment and storing of secrets
- Group or classify services – remember IPs/locations are ephemeral and there are no fixed boundaries any longer
- Define policies for groups or clusters and ideally automate enforcement as part of CD process
- Encrypt data in transit and at rest
- Enforce: Testing + Regular Library Updates
#3 – 1 public-facing API = 1 App

• External or Business APIs are often composed by choreography between smaller and finer internal APIs (and their respective app implementations)

• Each individual microservice/API has it’s own independent lifecycle

• Final API lifecycle reflects changes down the chain through versioning and managed publish to external consumers

• **Approach**: ✔ API products & App Platforms VS. ☒ API programs and App Projects
#4 - LIFT-&-SHIFT 😞

- Qualities of tall apps
  - Horizontally Scalable
  - Fast startup times
  - Resiliency to servers going and coming at will

**Pros:** Speed, Easier in terms of effort (short-term)

**Cons:** Operational overheads and related risk, costs and potential rework
#5 – Can put my current tools in a Docker container, sure it will work, right?

- Can you externalize and manage state?
- Can you break down the runtime dependencies and run them independently?
- Can the config and env variables be defined externally?
- Are you creating bloated containers or these are optimized for your app workloads?
- Scaling and elasticity can be leveraged without any effort?
- No hard dependencies on OS or file-system

© Copyright 2000-2017 TIBCO Software Inc.
#6 – all systems I integrate to have to be co-located always

Data spans across users, patterns, deployments and connected endpoints

**Integration Patterns**

- **B2B**
  - Application
  - API
  - Process
  - MFT

Deployment Model

- **Personas**
  - Integration Specialist
  - App Developers
  - Business Users

- **Data & Systems**
  - On-Premises
  - Cloud
  - Things

- **Integrations**
  - B2B
  - Process
  - MFT

- **End-Points**
  - On-Premises
  - Mobile
  - Things

© Copyright 2000-2017 TIBCO Software Inc.
#7 – dev = code, qa = test, ops = build and manage

- Break silos
- Enforce mandatory automation for entire lifecycle
- Integration testing – not as an after-thought
- Collective responsibility for each service
#8 – Bandwidth is always aplenty and resiliency comes from cloud

- Bandwidth is expensive and unreliable – especially on devices and IoT scenarios
- Build to minimize network/wire chatter
- Cloud could imply ephemeral servers and containers – build for rapid start/recovery
- Design to account for fragile/legacy back-ends (circuit breakers)
#9 – footprints don’t matter as much

On-Premises Infra
“My Pet”

Cloud Infra
“Herd of Cattle”

IoT Edge Infra
“Swarm of Bees”
#10 – Once the tech and tooling is addressed, the problem is resolved

Culture & Organization
Central bus arch → Distributed apps & endpoints

Leverage the wide spectrum of OS frameworks and best practices available but customize the journey to your use-case and app landscape.
About TIBCO

TIBCO fuels digital business by enabling better decisions and faster, smarter actions through the TIBCO Connected Intelligence Cloud. From APIs and systems to devices and people, we interconnect everything, capture data in real time wherever it is, and augment the intelligence of your business through analytical insights. Thousands of customers around the globe rely on us to build compelling experiences, energize operations, and propel innovation. Learn how TIBCO makes digital smarter at www.tibco.com.
What we have been doing for 20+ years

API Management
APIs
Events
Microservices / Flows
Messaging
Connectors

Deploy anywhere
Cloud
PaaS
Hybrid
On-Prem
Edge devices

Connect to anything
100’s of connectors
Build your own

Systems & Data
IoT Devices
Project Flogo

Open Source ultra-light edge microservices for

- IoT Integration
- Edge-based Machine Learning
- Conversational Engine for UX
- Cloud-native Microservices + Serverless Computing*

---

Inception

2016-03

0.2.1
1st GitHub OSS Release

2016-03

0.3.1
1st Public Docker Hub Release

2017-01

0.3.2
App Model support for WI

2017-03

0.3.3
Trigger reuse Support for WI
Support for GW

2017-04

0.4.0
Native lambda support*
Adaptive triggers for IoT*
Support for GW

2017-06

© Copyright 2000-2017 TIBCO Software Inc.
what can you do next?

Stay back and grab a beer with us? 😊

Docker Pull? Go to flogo.io

Check out the code – github.com/tibcosoftware/flogo
@rahulkamdar
rkamdar@tibco.com