Integration ...
The missing link in your cloud adoption strategy

Keith Babo
Consulting Product Manager
May 7, 2018
A TALK IN THREE PARTS

What is this ‘Integration’ you speak of?

Where Integration fits in one’s cloud adoption strategy

Patterns and best practices for cloud-native integration
INTEGRATION (?)
CLOUD ADOPTION - THE INTEGRATION GAP
THE JOY OF CLOUD ADOPTION
CLOUD ADOPTION - STAGE 1

Container Platform

NEW!!

0 NO HURT
2 HURTS LITTLE BIT
4 HURTS LITTLE MORE
6 HURTS EVEN MORE
8 HURTS WHOLE LOT
10 HURTS WORST
CLOUD ADOPTION - STAGE 2

Container Platform
CLOUD ADOPTION - STAGE 3

Container Platform

Existing IT

0
NO HURT

2
HURTS LITTLE BIT

4
HURTS LITTLE MORE

6
HURTS EVEN MORE

8
HURTS WHOLE LOT

10
HURTS WORST
CLOUD ADOPTION - STAGE 6

Container Platform

Existing IT

0
NO HURT

2
HURTS LITTLE BIT

4
HURTS LITTLE MORE

6
HURTS EVEN MORE

8
HURTS WHOLE LOT

10
HURTS WORST
MICROSERVICES
MICROSERVICES MADE EASY

Crusty Monolith

Awesome Microservices
REALITY CHECK - THIS IS EASY

Crusty Monolith

Scaling
Deployment
Shared Services
Locality

Monitoring
Security
Governance
Transactions

#redhat #rhsummit
WHOOPS!

(NOT SO) Awesome Microservices
INNER vs. OUTER ARCHITECTURE

https://blogs.gartner.com/gary-olliffe/2015/01/30/microservices-guts-on-the-outside/
JUST WHEN YOU THOUGHT IT WAS SAFE TO MOVE TO THE CLOUD ...
ARE YOU DOING MULTICLOUD?

Multicloud is a cloud approach made up of more than 1 cloud service, from more than 1 cloud vendor—public or private.

https://www.redhat.com/en/topics/cloud-computing/what-is-multicloud
CLOUD-NATIVE INTEGRATION PATTERNS & BEST PRACTICES
DON’T WING IT
THREE PILLARS OF AGILE INTEGRATION

Key foundational capabilities needed by today’s enterprises

DISTRIBUTED INTEGRATION
- Lightweight
- Pattern Based
- Event Oriented
- Community Sourced

MICROSERVICES CONTAINERS
- Cloud Native Solutions
- Lean Artifacts
- Individually Deployable
- Container Based Scaling and High Availability

APIs
- Well Defined
- Reusable
- Well Managed End-points
- Ecosystem Leverage

FLEXIBILITY

SCALABILITY

RE-USABILITY

#redhat #rhsummit
AGILE INTEGRATION ARCHITECTURE

Containers

Application Network Layer
- Policies
- Access Control
- Proxy Routing

Composite Layer
- Enterprise Integration Patterns
- Service Composition
- Service Interactions
- Events
- Mesh
- Anti Corruption Layer

Core Layer
- Cloud Native Service (Runtime 1)
- Cloud Native App (Runtime 2)
- Containerized App (Lift and Shift EAP)
- Traditional App (EAP)

DevOps Automation / Continuous Integration / Continuous Delivery (Ansible)
CORE SERVICES LAYER

* Brownfield and greenfield services
* Microservices and monoliths
* Delivered independently

* Independent data contexts
* Mixed connectivity
COMPOSITE LAYER

* Service composition
* Routing and orchestration
* Data transformation

* Connectivity
* API ←→ Event Bridging
* Legacy facade (ACL)
APPLICATION NETWORK LAYER

* Gateway layer for services
* Access and policy control
* Developer onboarding
* Analytics
* Monetization
* Scales independent of other layers
AGILE != ANARCHY
API-CENTRIC INTEGRATION LIFECYCLE

Strategy

Design  Implement  Deploy  Manage
API STRATEGY

AUDIENCE
- Internal API clients
- External API clients

GOVERNANCE
- Security
- Lifecycle
- Automation

SCOPE
- Single API
- Multiple microservices
- External APIs

ENVIRONMENT
- Standalone
- Private Cloud
- Public Cloud
- Hybrid Cloud

BUSINESS STRATEGY
CLIENT-FOCUSED
- Design with the API client in mind
- Design with tooling fit for purpose
- Collaborate ASAP

VALIDATE EARLY
- Use API mocking for early feedback
- Skeleton implementation can be just as good as a mock

FAVOR INTEROPERABILITY
- Create API definitions based on standards in open communities
- Maximize tool portability and client generation
| HONOR THE TRUTH | API Definition is the source of truth  
|                | Favor generation over translation |
| NOT ALL APIs ARE THE SAME | Standalone  
|                       | Data API  
|                       | Orchestration  
|                       | Event Bridge  
|                       | Legacy Facade |
| WHICH PERSONA?      | Developer  
|                     | Non-developer |
| CONTAINERS                          | - Best way to develop services (polyglot, portability, availability, service wiring, advanced deployment, …)  
|                                   | - Maximize inner vs. outer architecture pattern |
| HYBRID ENVIRONMENT                | - Support integration and management of APIs living outside containerized environment  
|                                   | - Consistent architecture across private, public, and managed cloud |
| AUTOMATE                          | - API-driven infrastructure services  
|                                   | - Ability to automate application and infrastructure services in a single pipeline |
CONTROL
- Securing APIs
- Traffic flow control via policy
- Policy extensibility

VISIBILITY
- Developer onboarding and engagement
- Traffic and policy alerts
- Use analytics to understand how APIs are tracking against business objectives

FLEXIBILITY
- Centralized management and distributed enforcement
- API management architecture must span multiple environments
HYBRID CLOUD > MULTICLOUD
HYBRID SERVICE PLANE

Security    Connectivity    Composition    Discovery    Analytics

Bare Metal / VMs    Private Cloud    Public Cloud    SaaS
AMQ Interconnect Event Mesh
- Overlay messaging network spanning private and public cloud for seamless inter-service communication
- Secure
  - Mutual TLS between routers with dedicated CA
  - No inbound TCP connections to private cloud
- Logical addressing, not hosts & ports

Use-cases
- Expose data securely from private to public cloud (APIs and Events)
- Topology-aware request routing
- Cloud bursting
- Fault-tolerance
DOUBLE ROADMAP!!

PART I
APIs, events, and data—your roadmap for agile integration with Red Hat

Wednesday @ 10:30am
Room 160A

PART II
Best practices for developing modern applications with Red Hat Integration

Wednesday @ 11:30am
Room 157B
THREE THINGS TO REMEMBER

1. You *are* doing integration today
2. The move to cloud *increases the need* for integration
3. Integration is a core component of your *cloud adoption strategy*