An Integration Fabric For Microservices

Vita Bortnikov

IBM

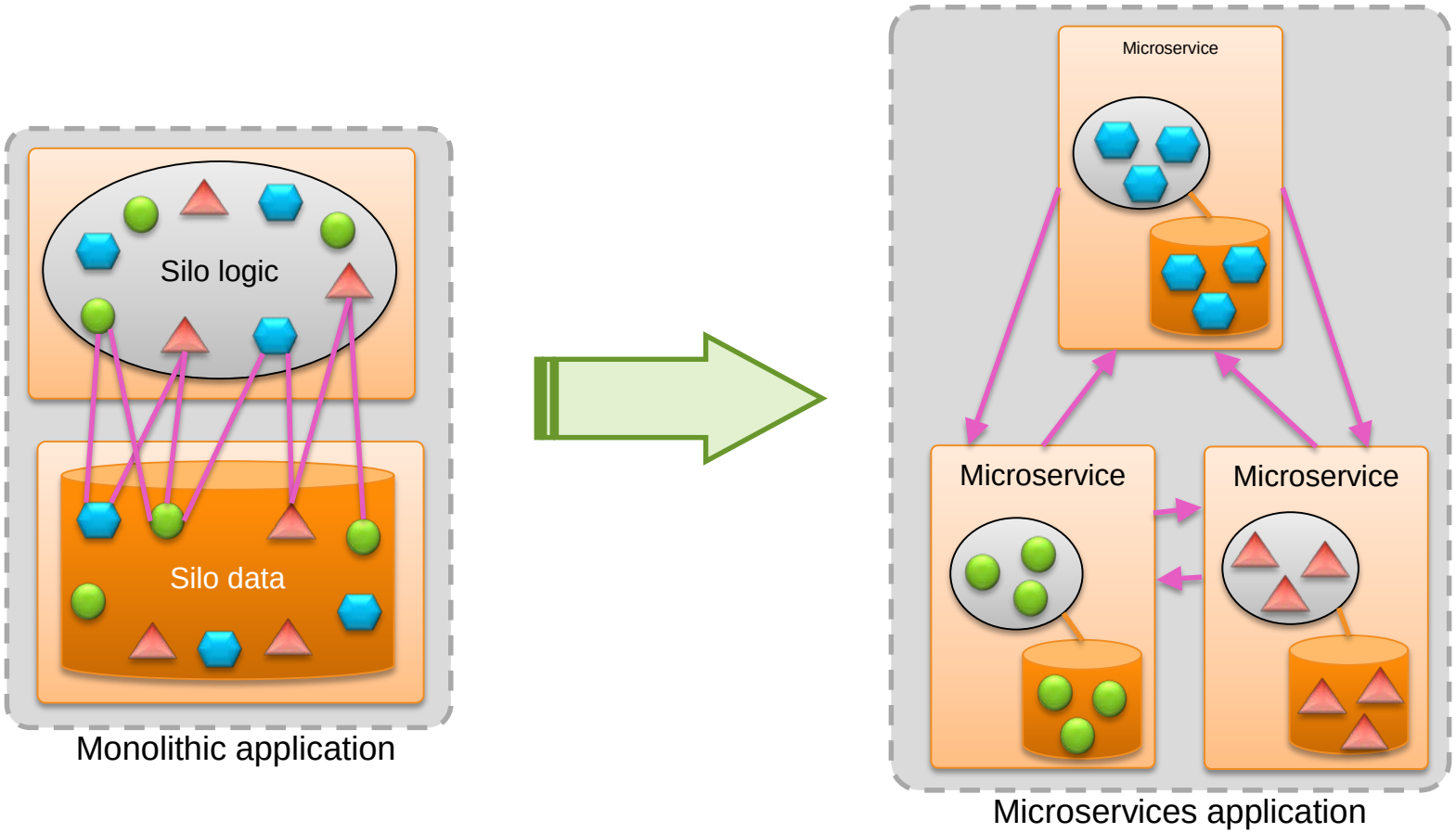
Microservices

An **engineering approach** focused on **decomposing** an application into **single-function** modules with **well defined interfaces** which are **independently** deployed and operated by a **small team** who owns the **entire lifecycle** of the service.

Microservices accelerate delivery by **minimizing communication** and coordination between people while **reducing the scope** and risk of change.

Jason McGee, IBM Fellow and VP, CTO Cloud Pla

Microservices

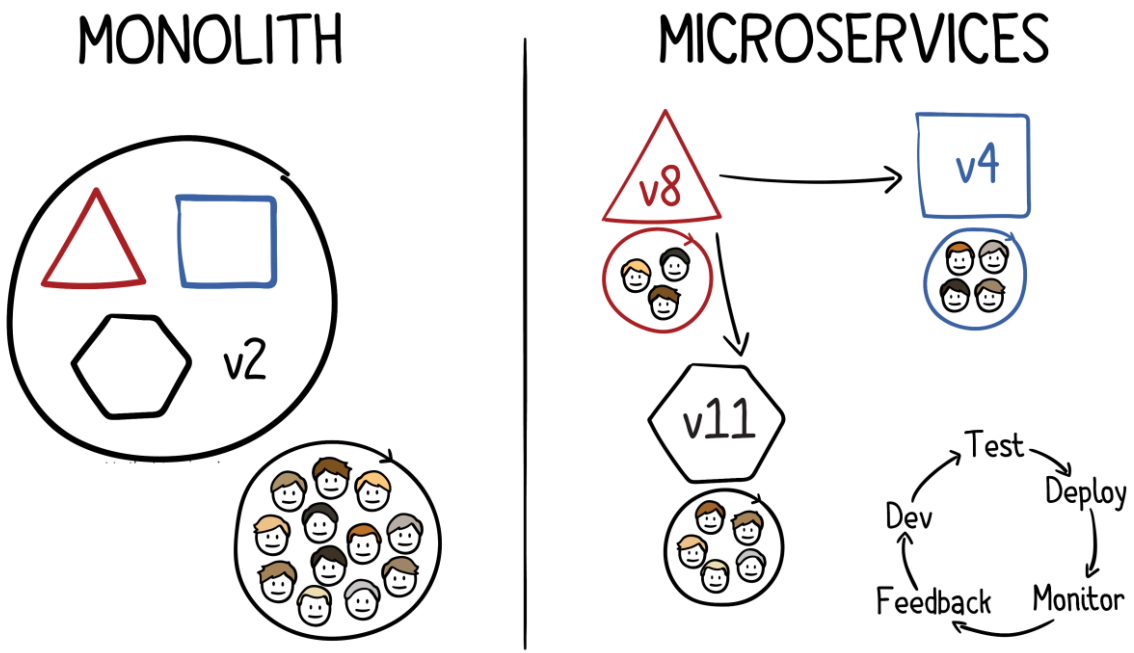


Resilience → Agility → Scale

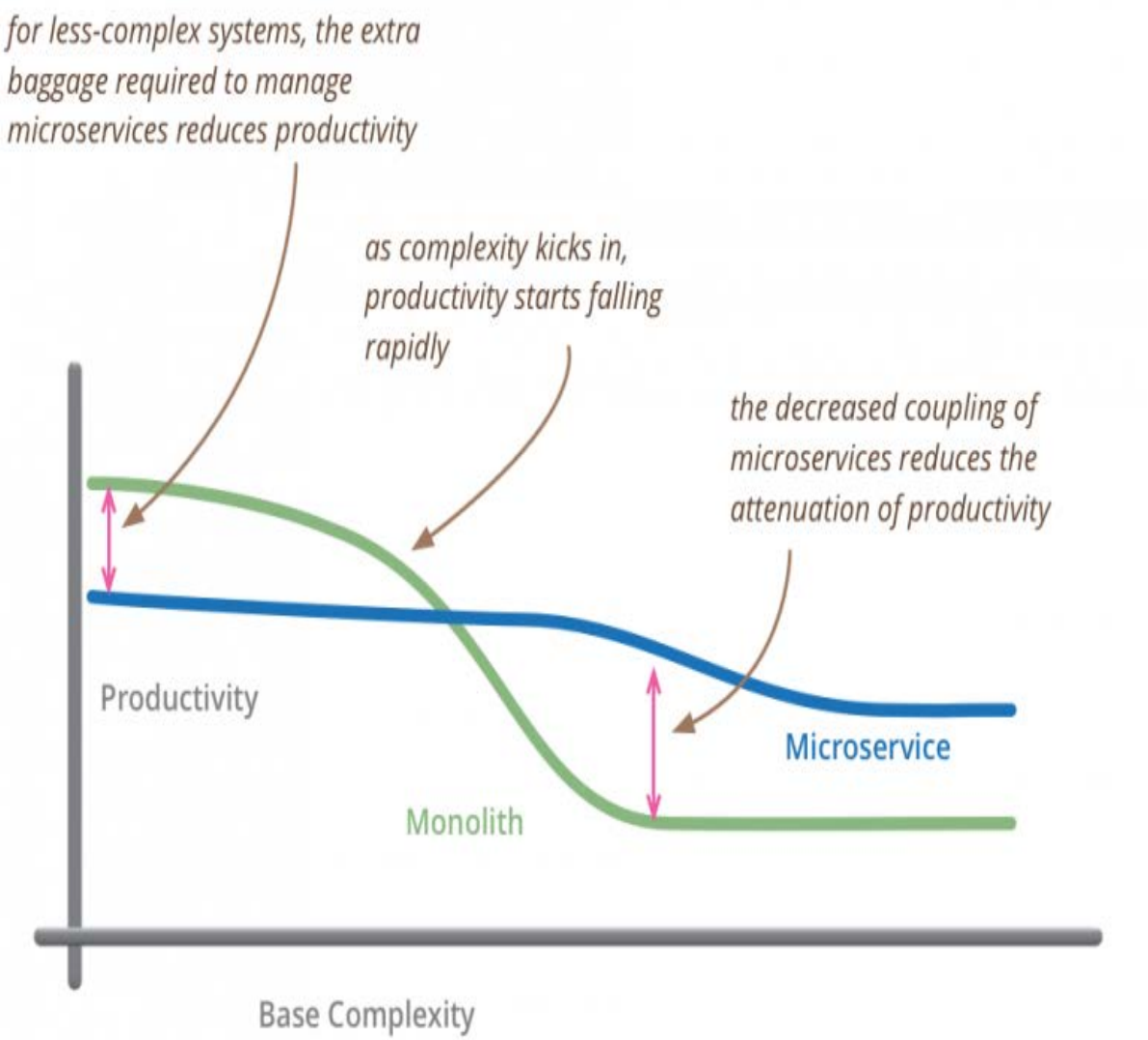
Culture

“Any organization that designs a system (defined more broadly here than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure.”

— Melvyn Conway, 1968



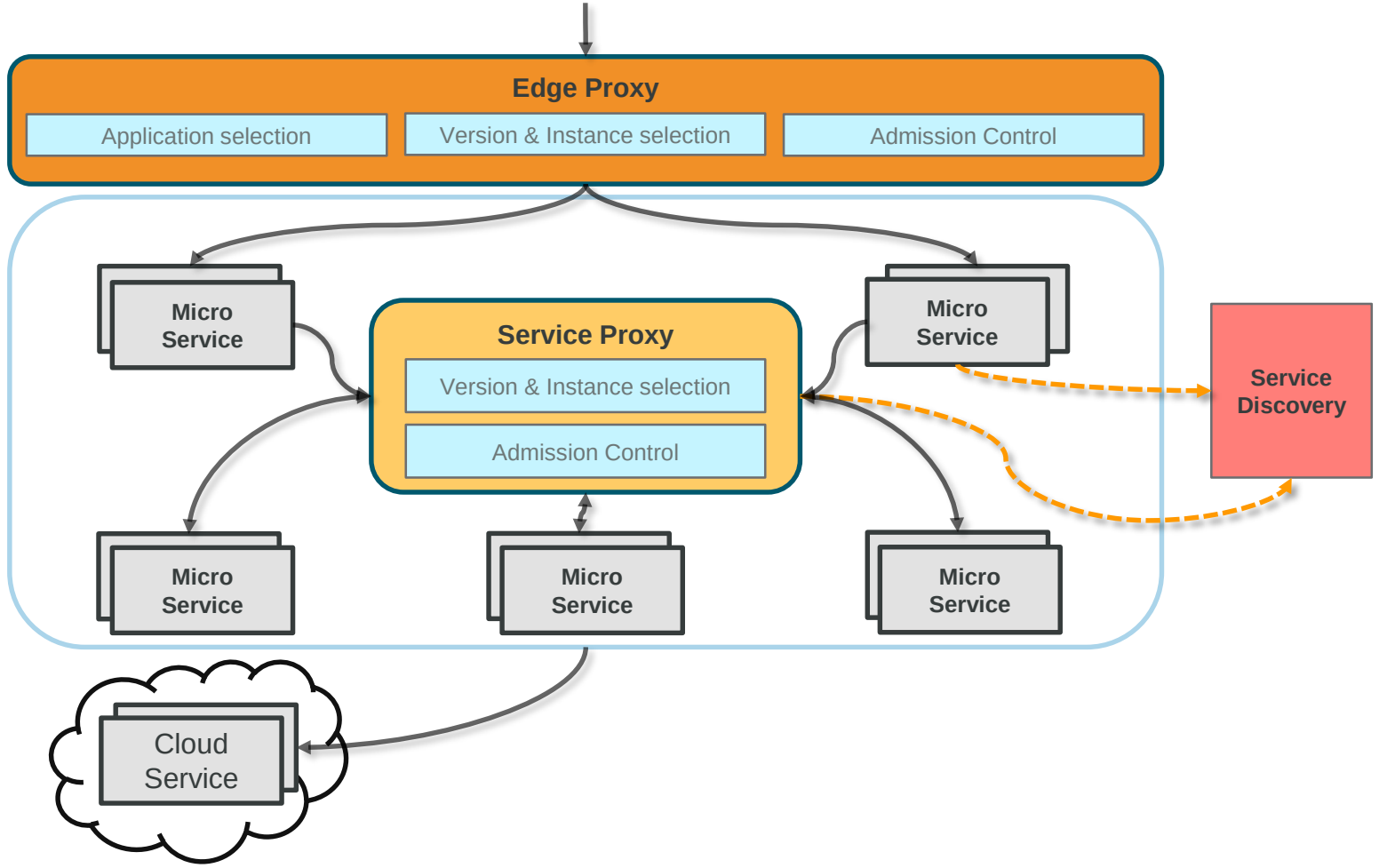
Microservices add complexity



http://martinfowler.com/bliki/MicroservicePremium.html

but remember the skill of the team will outweigh any monolith/microservice choice

Microservices Fabric



Our Journey

Use Netflix OSS components: Eureka, Ribon etc

Proprietary Fabric on IBM Bluemix

Open source fabric for Polyglot Microservices


Amalgam8

Compose and Orchestrate your Polyglot Microservices with Amalgam8

Amalgam8

Platform & Runtime Independent

Multi-Tenancy Supported

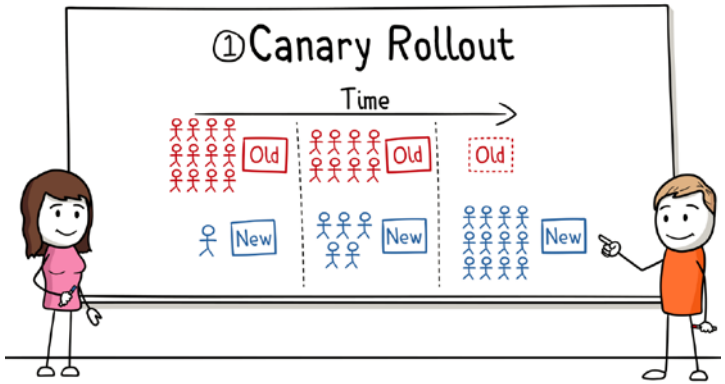
Shortens Development Cycles

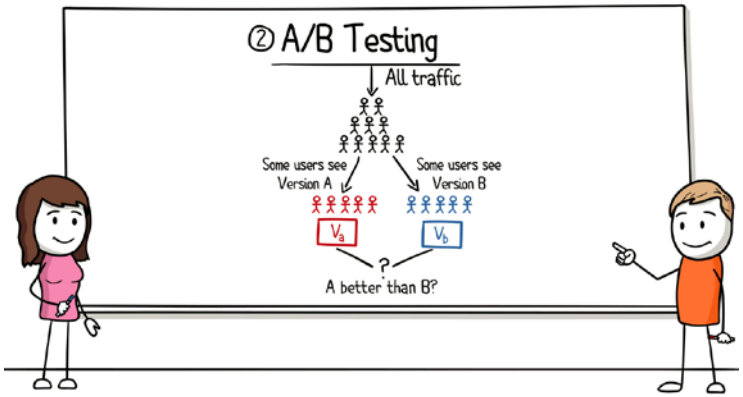
- Simplified Service Discovery & Load Balancing
- Red/Black Deployment & Canary Testing

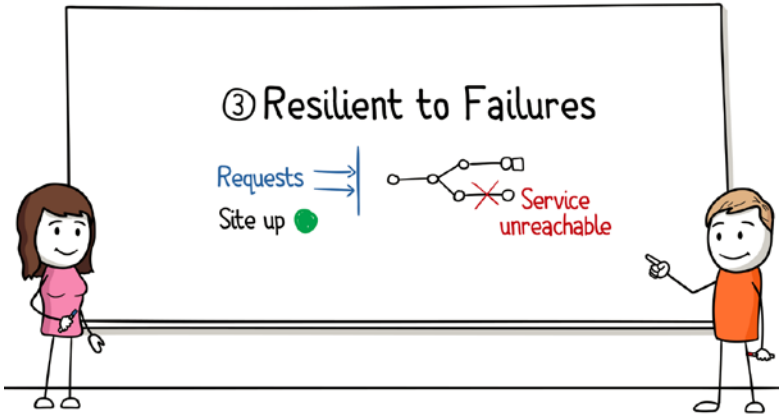
Target Capabilities

Runtime functions Core services needed to build microservices apps

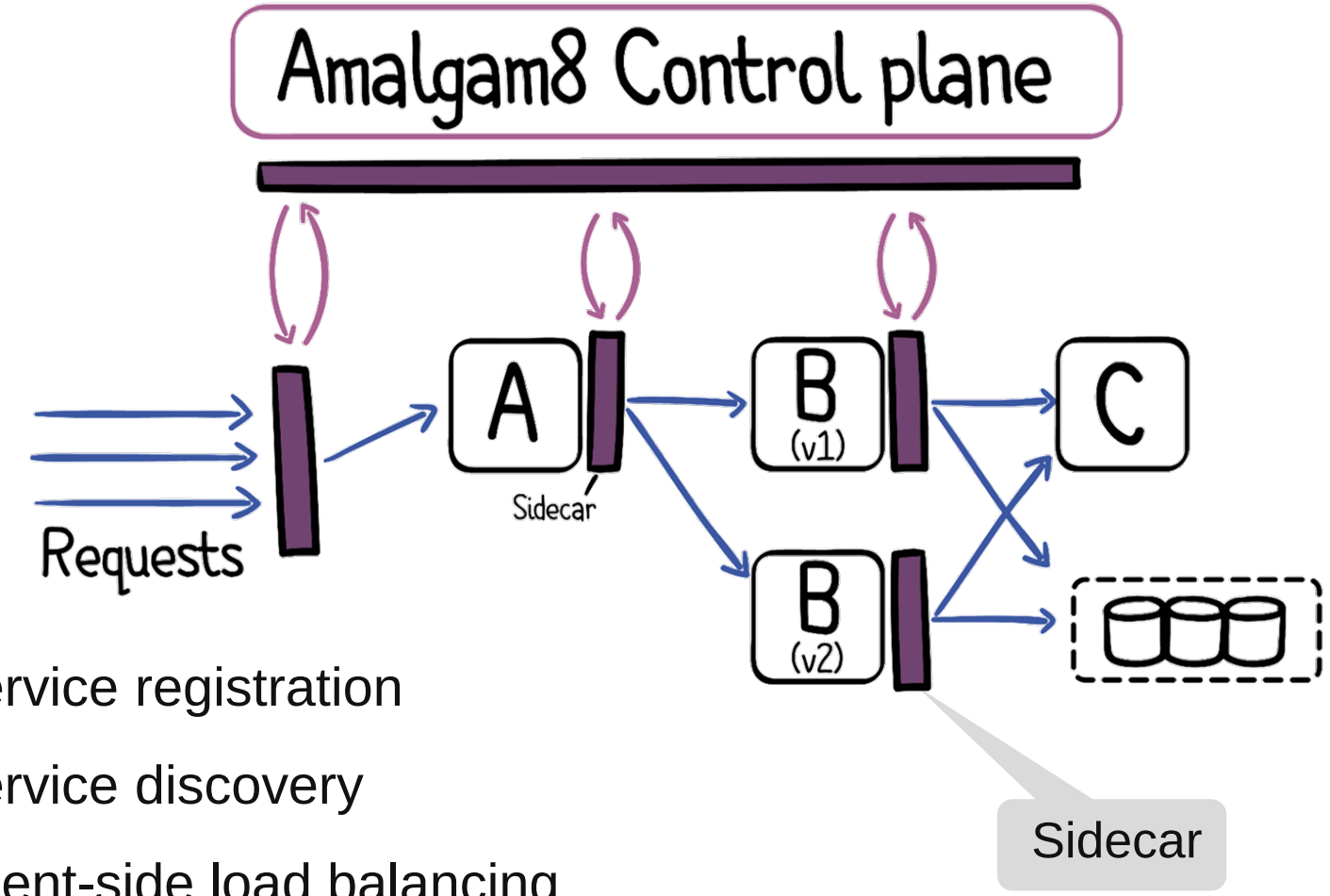
Lifecycle services Services needed to enable DevOps practices







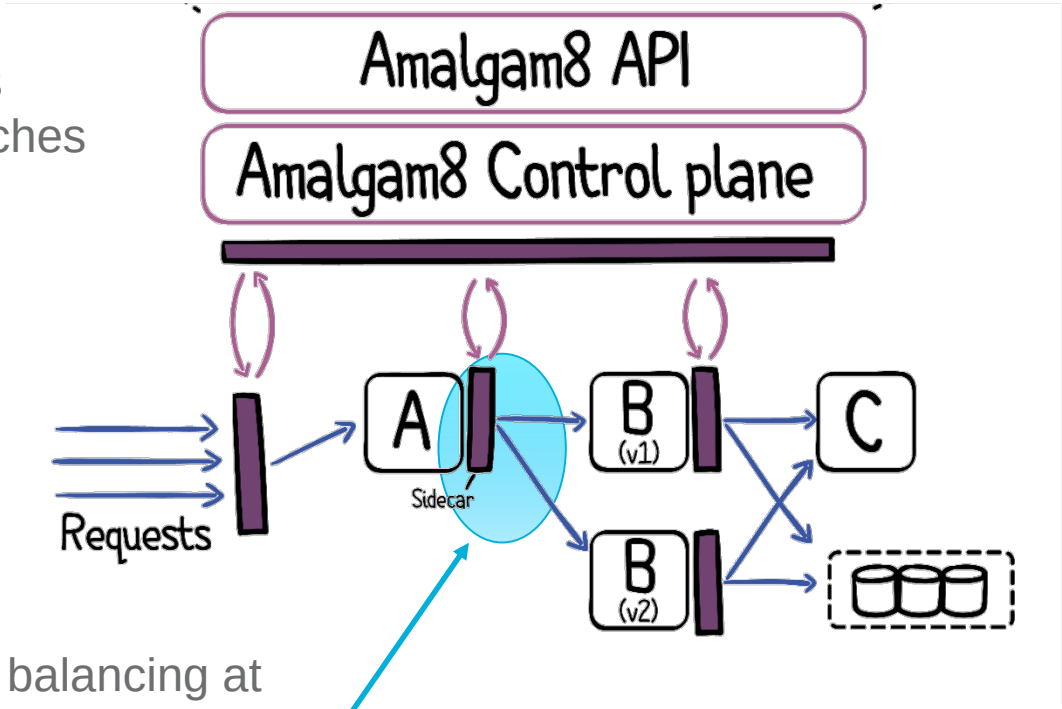
Amalgam8



- Service registration
- Service discovery
- Client-side load balancing

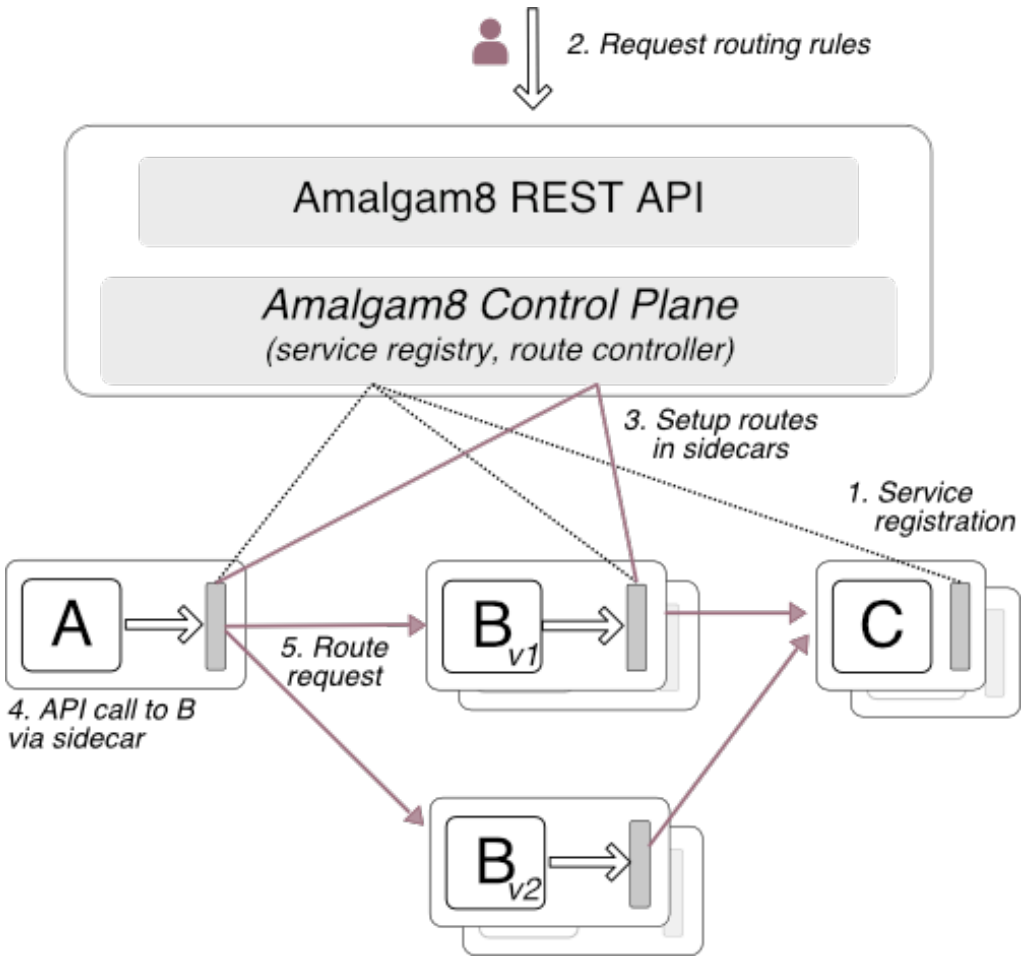
Amalgam8

- Continuous delivery tasks
  - Internal releases/dark launches
  - Canary deployments
  - Red/Black deployments
  - **Testing in production**

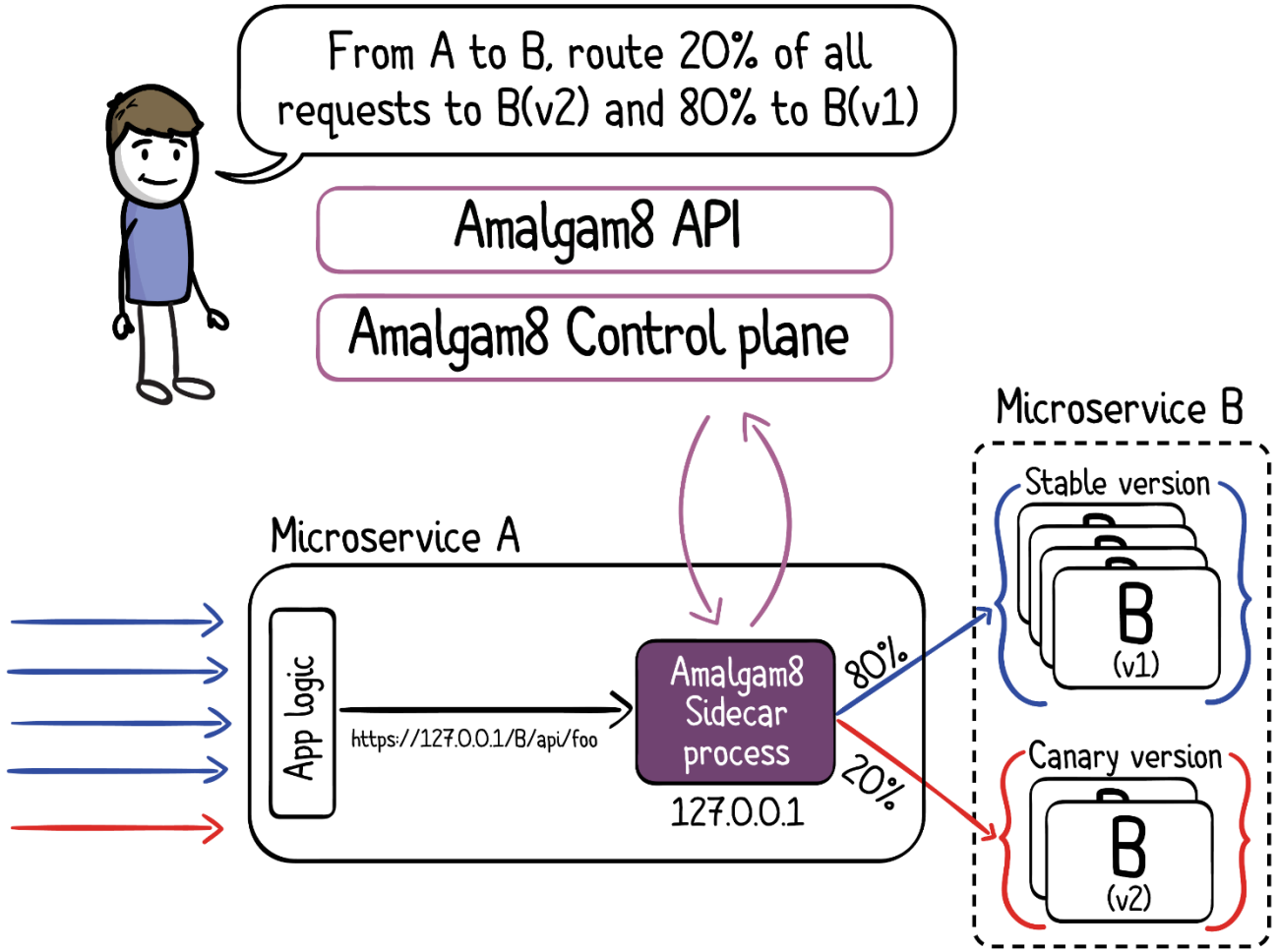


- How
  - Dynamically control of load balancing at sidecars
    - Setup routing rules using Control Plane APIs. Optimized rules are percolated to sidecars.
    - Rules determine how requests are load balanced across versions
    - Program fault injection rules to simulate failures of dependencies

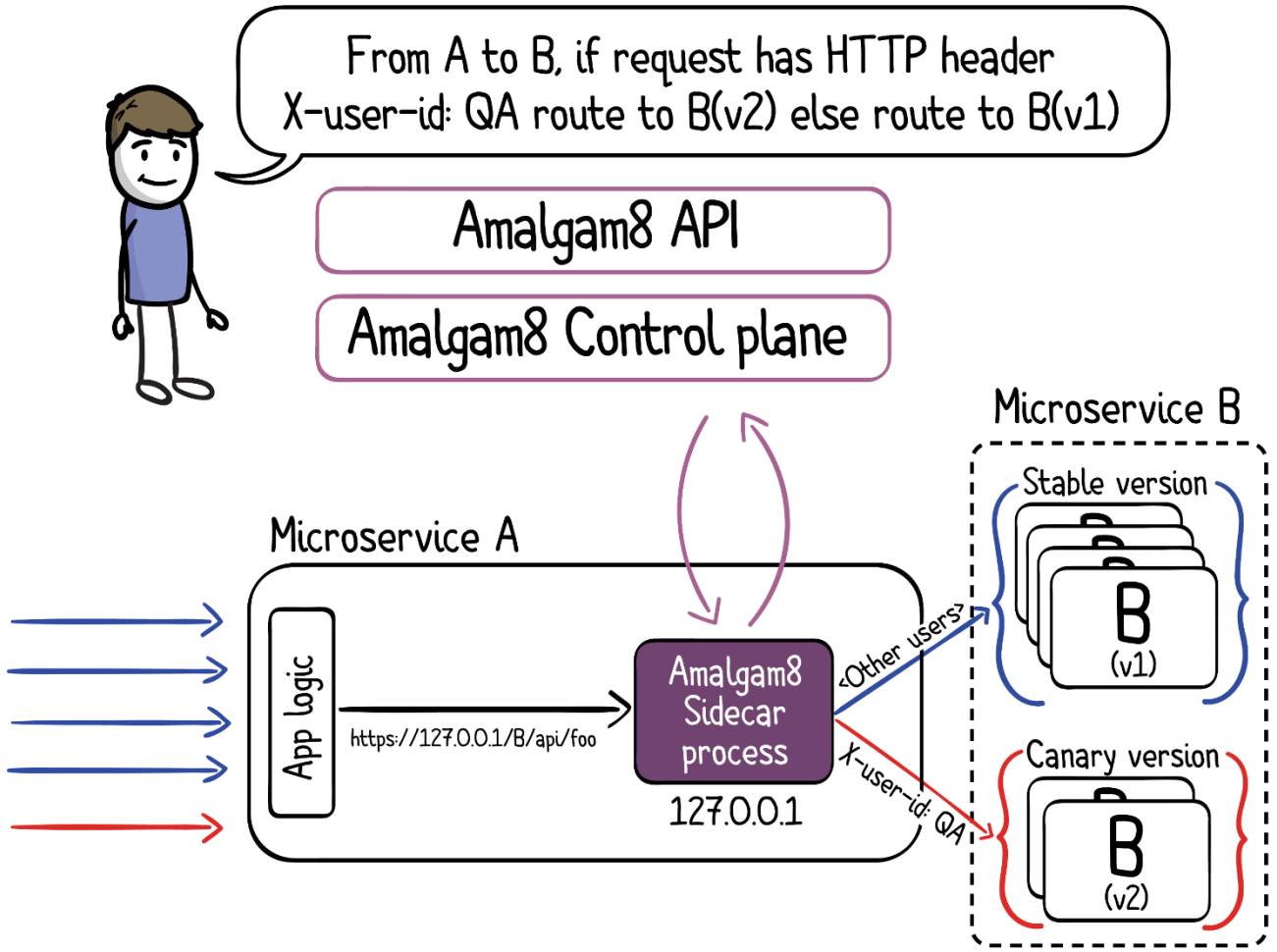
Amalgam8


2. Request routing rules
Amalgam8 REST API
Amalgam8 Control Plane
(service registry, route controller)
3. Setup routes
in sidecars
1. Service
registration
A
B v1
C
5. Route
request
4. API call to B
via sidecar
B v2

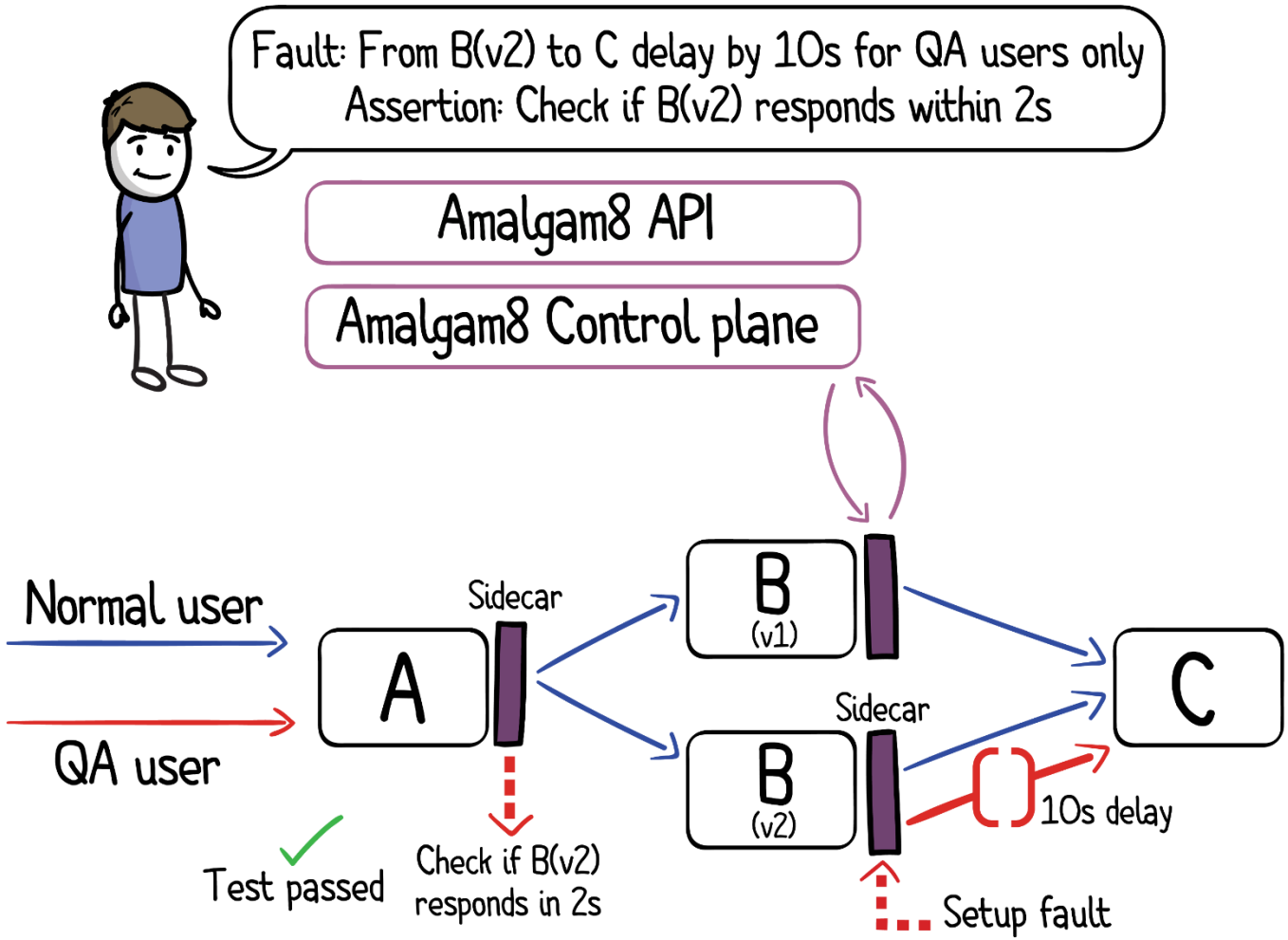
Version-based routing



Content-based routing



Systematic Resiliency Testing

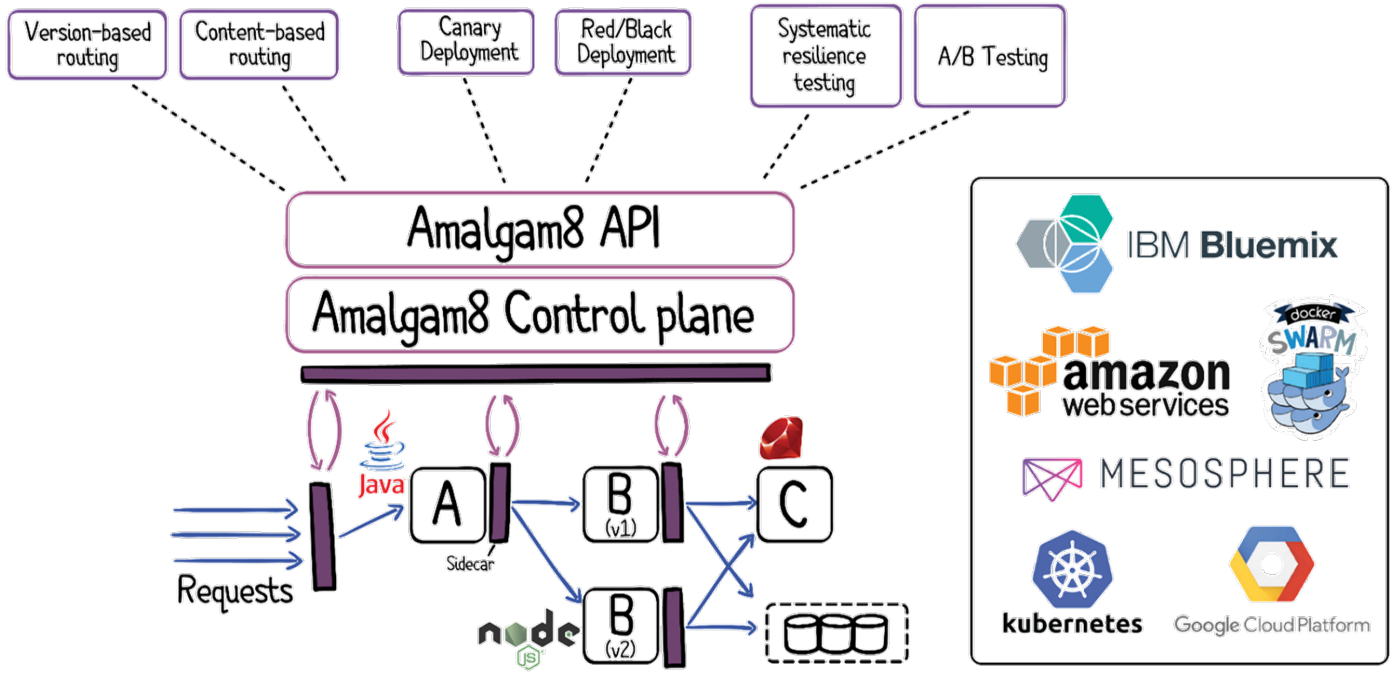


Amalgam8

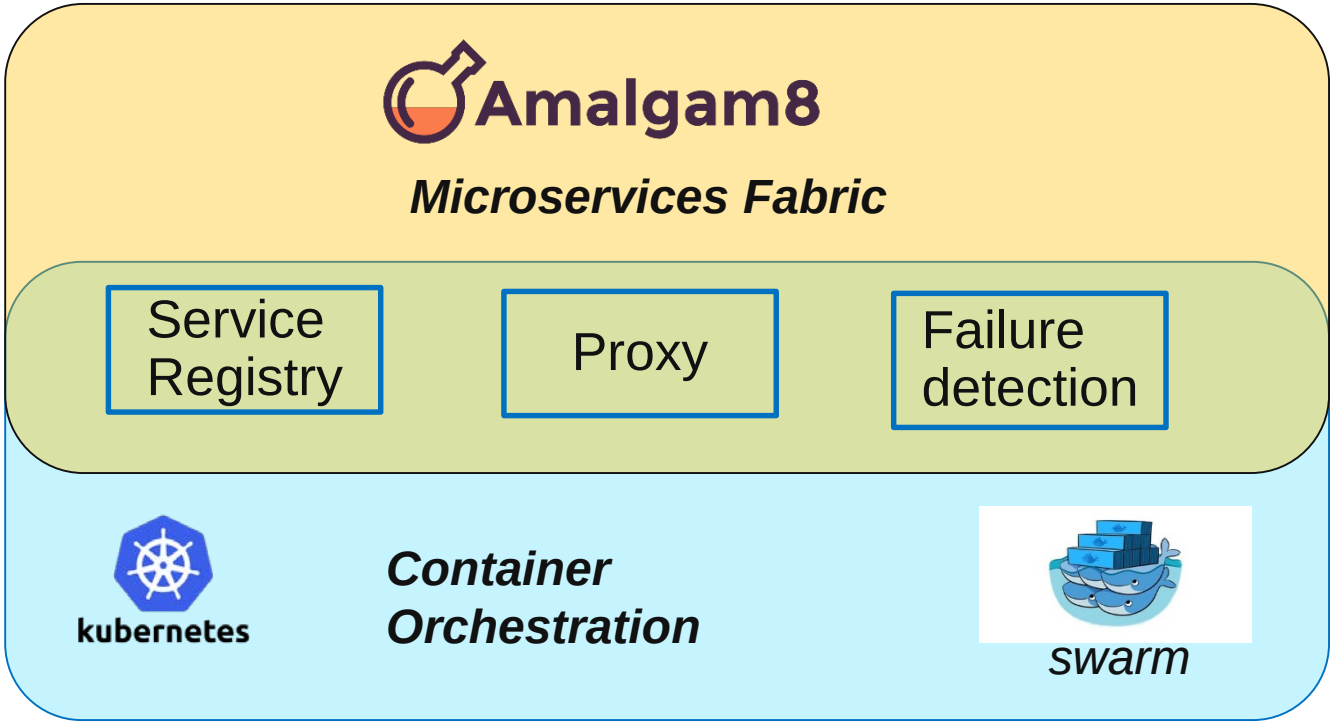
Easy to build rich toolset for continuous delivery, experimentation and testing

Programmable

Polyglot



What's next?



Orchestrator API becomes the platform

Microservices runtime value-add services => be integrated into the platform

References

- Amalgam8
  - Webpage: amalgam8.io
  - Docs: amalgam8.io/docs
  - Code: github.com/amalgam8
  - Slack: self invite via amalgam8-slack-invite.slack.com