

Microservices and Service Mesh

Oxford University
Software Engineering
Programme
December 2019



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Agenda

- Why?
- History and evolution
- Architecture
- Pros and Cons
- More resources



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Microservices

- Building a single app from multiple services
 - Each service in its own process
 - Lightweight communications between each other
 - HTTP REST or Async models (ZeroMQ, Kafka, RabbitMQ, etc)



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Microservices Characteristics (Martin Fowler)

- **Componentization**
 - Replacability
- **Organisation**
 - around business capabilities instead of around technology.
- **Smart endpoints and dumb pipes**
 - explicitly avoiding the use of an Enterprise Service Bus (ESB)
- **Decentralised data management**
 - with one database for each service instead of one database for a whole company.
- **Infrastructure automation**
 - with continuous delivery being mandatory.

<http://martinfowler.com/articles/microservices.html>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

You build it you run it

Amazon story

2001

Exactly equal to Microservices!

(this isn't new!)



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Benefits of Microservices

- Independent organization makes it easier for developers
 - Separation of concern
 - Simple code
 - Simple test cases
 - Simple scaling
 - Faster to build, deploy and test





Stuart Charlton @svrc · 16h

the next time i see a team of 4 wanting to create 63 microservices for their app, i want this posted in big neon letters to their team room

Andrew Clay Shafer @littleidea

The point of microservices is to unblock independent queues of work. Both in the system of services, and the system of people.

◀ 2

↑↓ 46

♥ 46

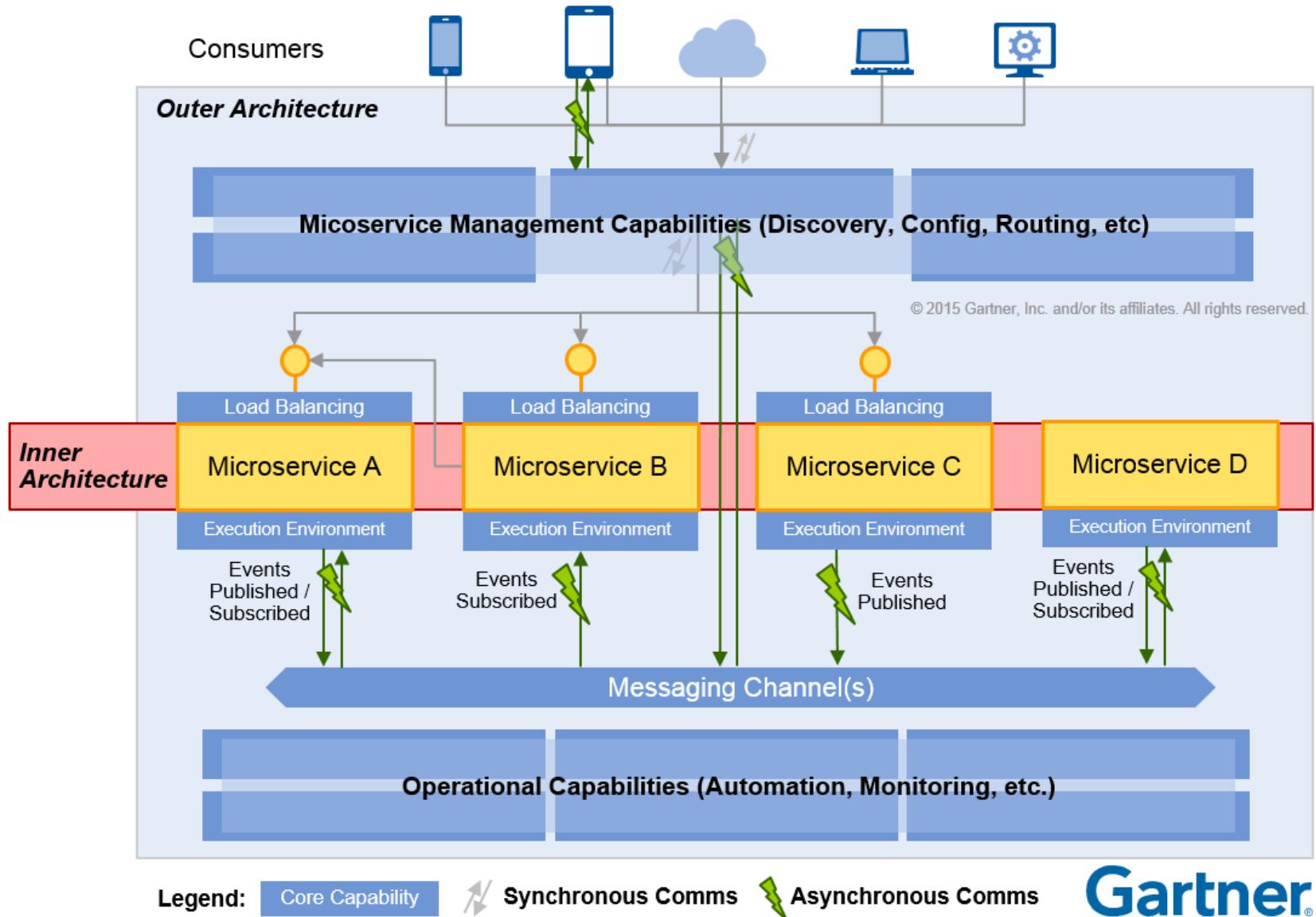


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Microservices deployment model

- Increasingly fitting with “containerisation”
 - Docker
 - CoreOS
 - Kubernetes
 - Etc
- Container model is lightweight virtualization with each “VM” running a single process

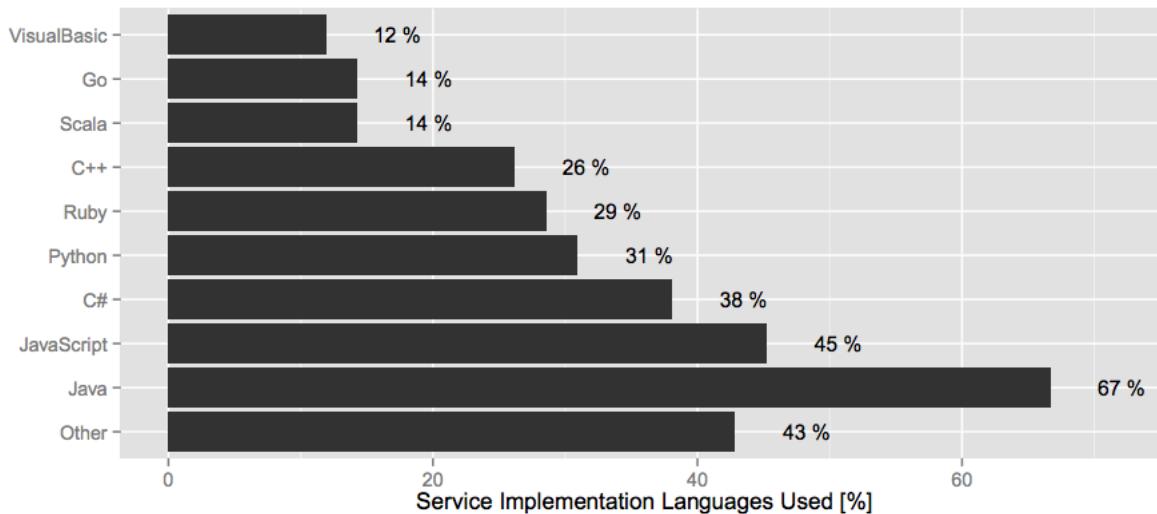




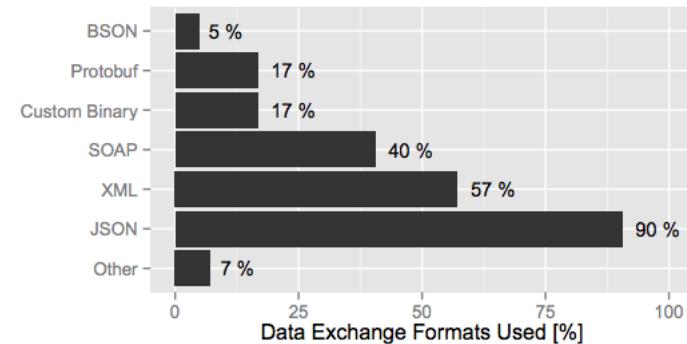
Traditional SOA versus microservices

	Traditional SOA	Microservices
Messaging type	Smart, but dependency-laden ESB	Dumb, fast messaging (as with Apache Kafka)
Programming style	Imperative model	Reactive actor programming model that echoes agent-based systems
Lines of code per service	Hundreds or thousands of lines of code	100 or fewer lines of code
State	Stateful	Stateless
Messaging type	Synchronous: wait to connect	Asynchronous: publish and subscribe
Databases	Large relational databases	NoSQL or micro-SQL databases blended with conventional databases
Code type	Procedural	Functional
Means of evolution	Each big service evolves	Each small service is immutable and can be abandoned or ignored
Means of systemic change	Modify the monolith	Create a new service
Means of scaling	Optimize the monolith	Add more powerful services and cluster by activity
System-level awareness	Less aware and event driven	More aware and event driven

What are services like in reality?



Most services (51%) were 1,000-10,000 LoC
Only 3% of services in the survey were <100 LoC
43% 100-1,000 LoC



All the Services Large and Micro: Revisiting Industrial Practice in Services Computing
<https://peerj.com/preprints/1291.pdf>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Real world examples

- The previous case studies are in many cases microservices
 - eBay, Netflix, Amazon
 - Many more out there and growing rapidly



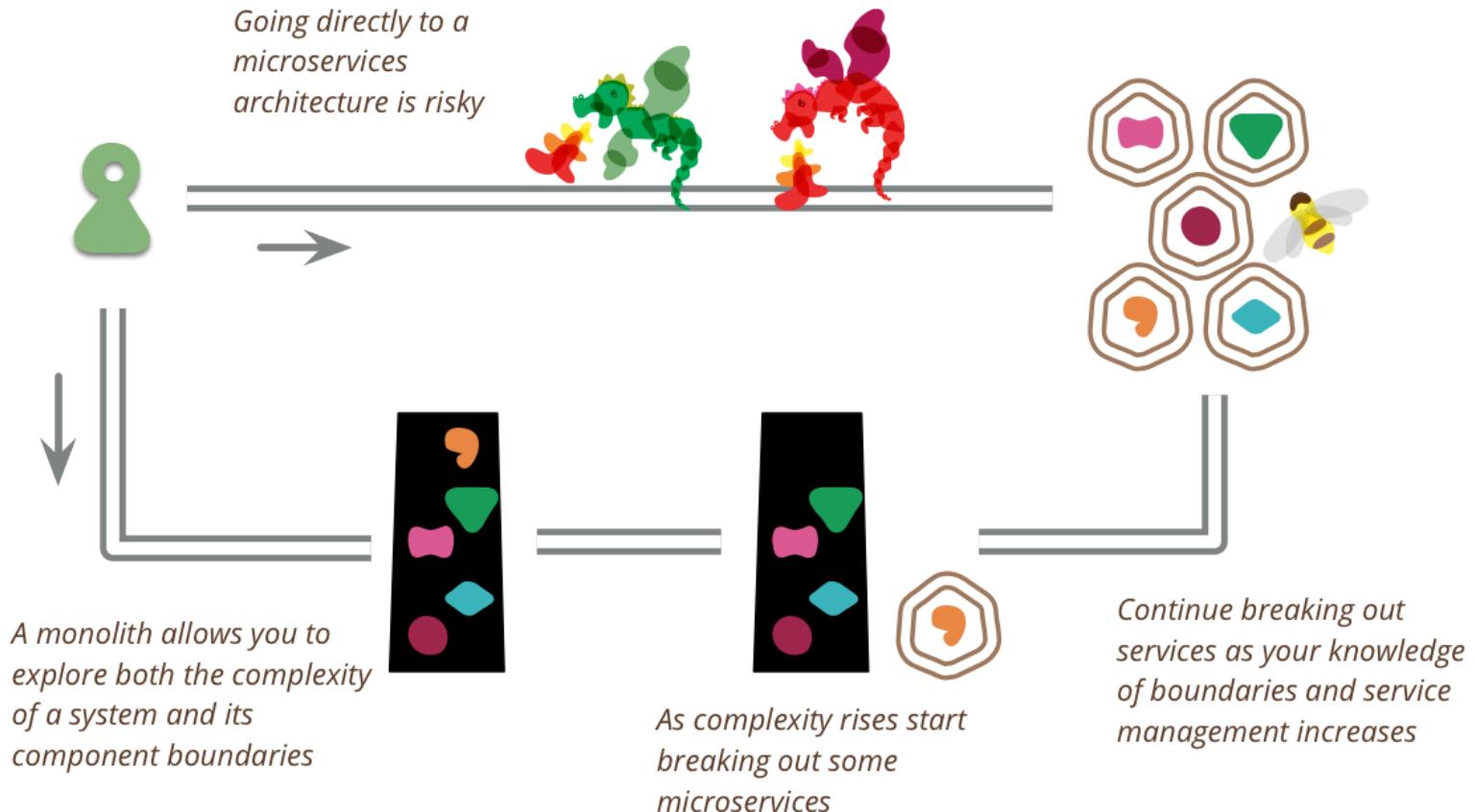
© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Polyglot

- Microservices can be built in multiple languages
 - Hackathon last year I built a simple app
 - Node, Python and MQTT
 - One day's effort
 - Microservice architecture
 - <http://pzf.fremantle.org/2013/12/commshack.html>



Start with a Monolith?



<http://martinfowler.com/bliki/MonolithFirst.html>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Cons!

- Debugging
- Deployment and devops
- Operations overhead
- Implicit interfaces and contracts
- Latency
- Transactions
- Etc.!

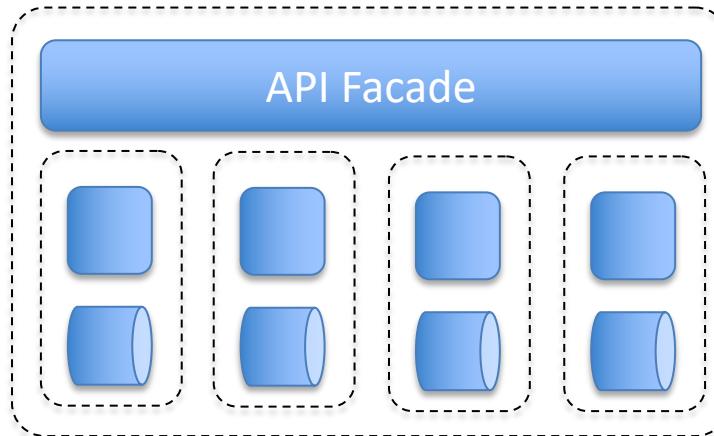
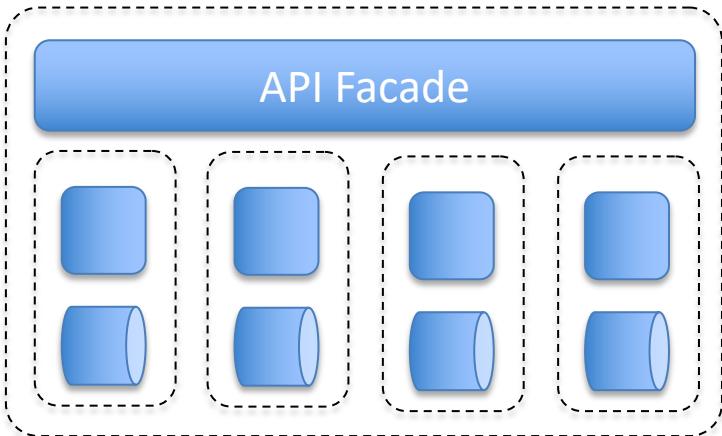
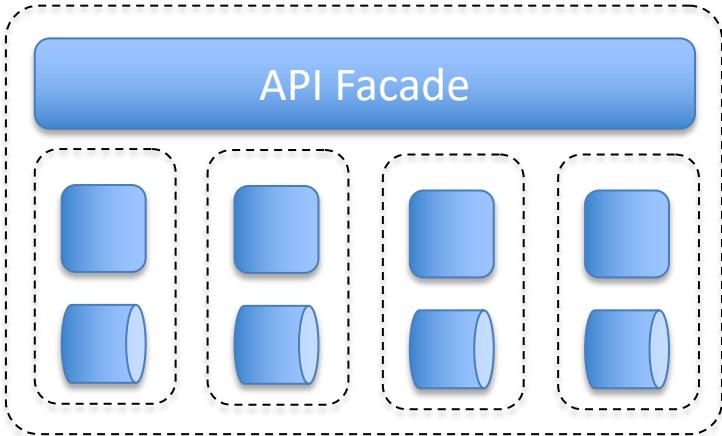


Smart endpoints and dumb pipes

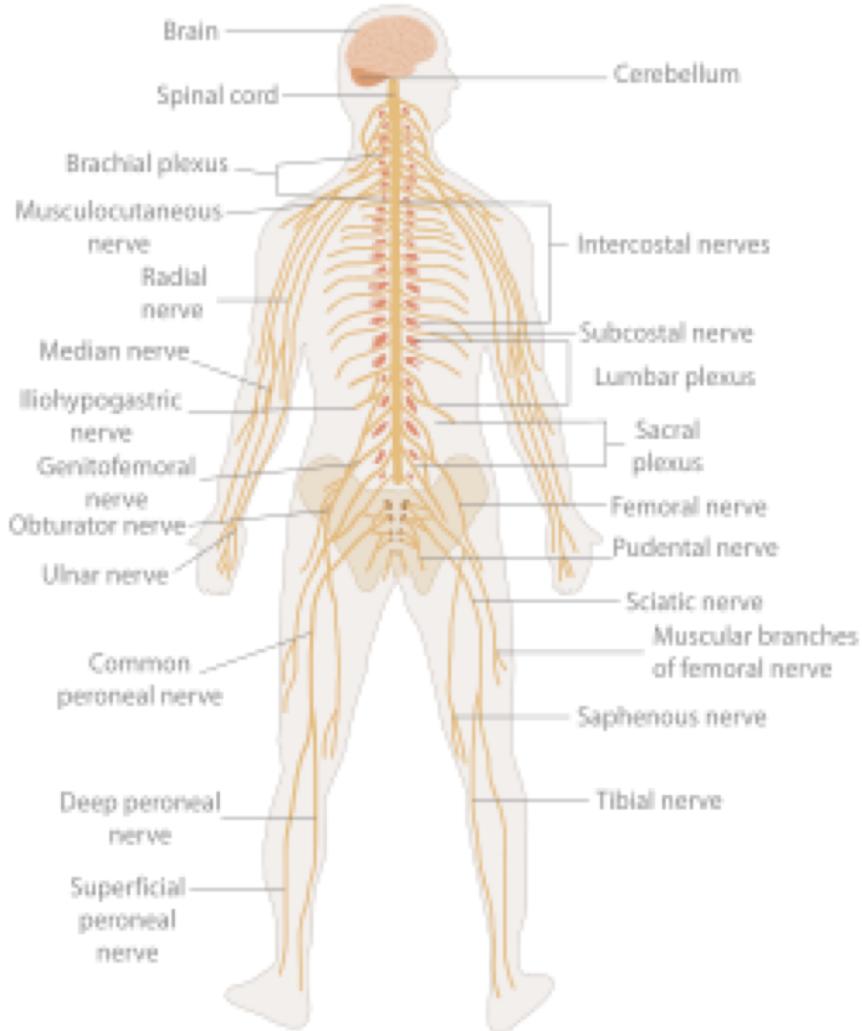
- Microservices are based on the idea of simple RESTful APIs directly implemented
- Need to manage contracts cleanly and carefully
- ESB is not part of this architecture
 - But an API Gateway might be
 - Don't confuse the application architecture with the Enterprise architecture



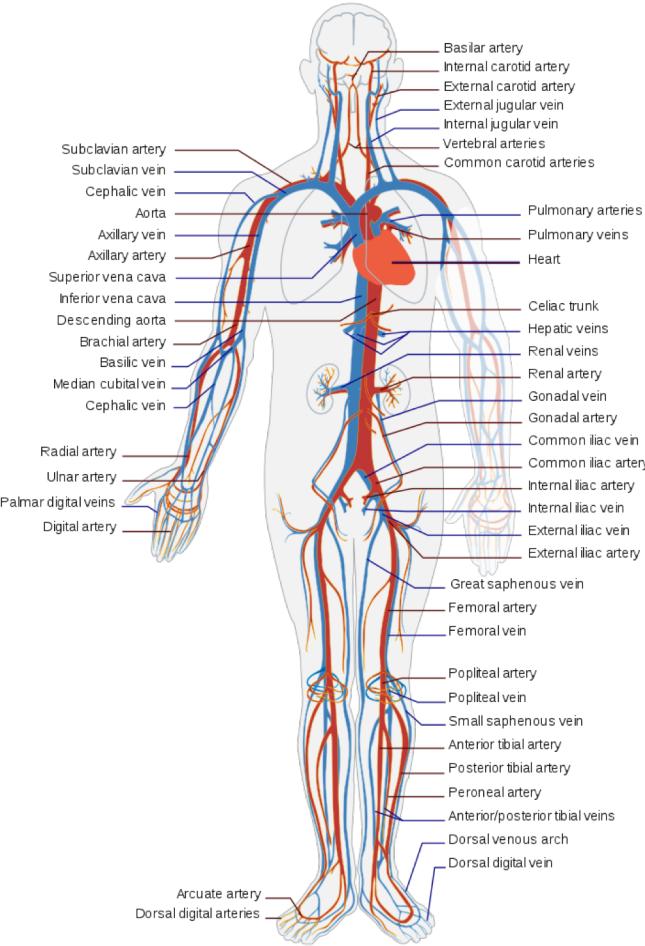
Micro and Macro Services



Control Plane



Data Plane



API Gateway and Microservices

- Versioning
- Single URI structure out of many independent backends
- Contracts and documentation
- More discussion later

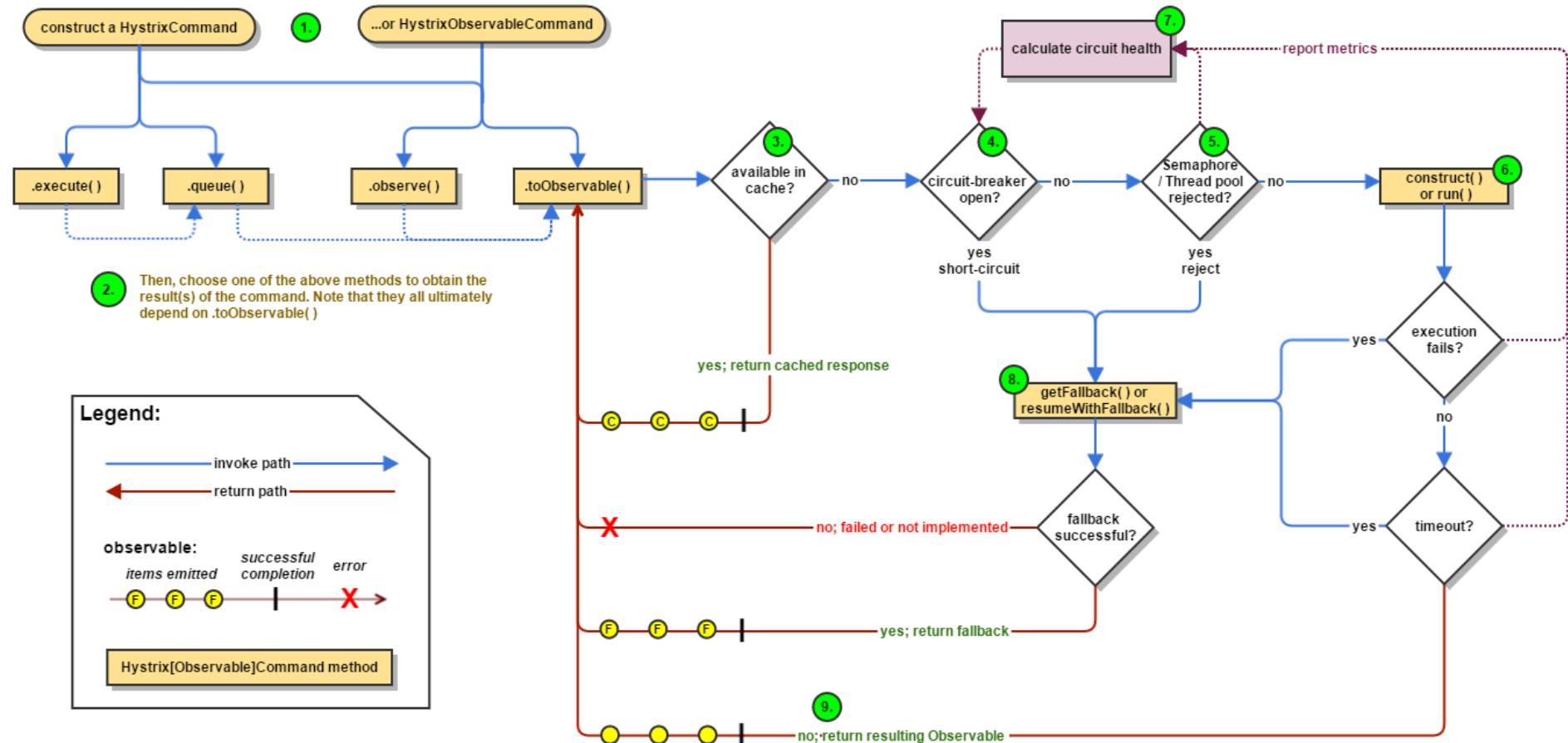


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Service Mesh

- How do I guarantee resilience in a microservices environment?
- Load-balancing, failover, discovery
- Building it into code is complex
 - Libraries such as Hystrix help
- But inhibits polyglot
 - Each language needs its own approach



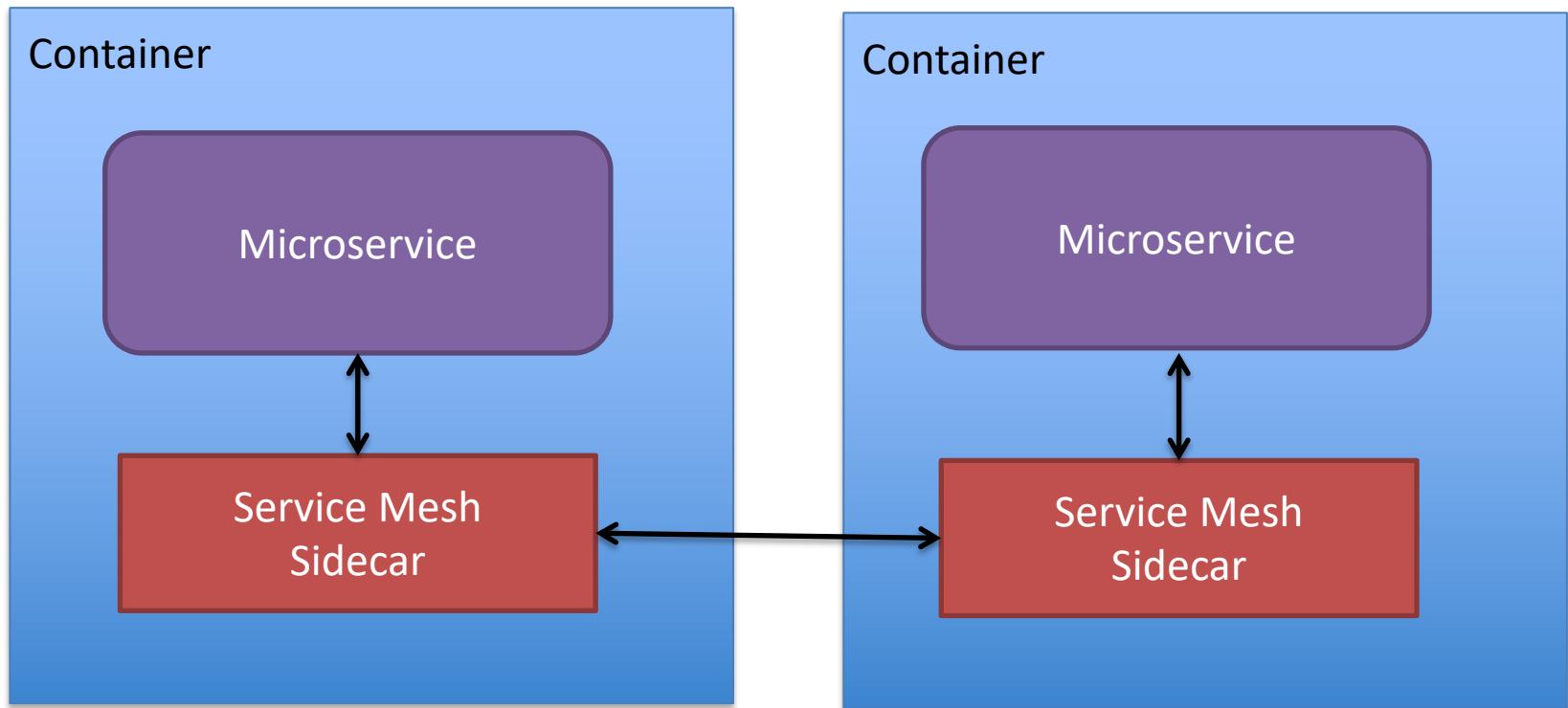


<https://github.com/Netflix/Hystrix/wiki/How-it-Works>



Service Mesh

Sidecar architecture



Service Mesh

- Load-balancing, circuit breakers
- Discovery / Naming
- Traffic control
- Per-request/content-based routing
- Timeouts and deadlines
- Mutual TLS
- Observability



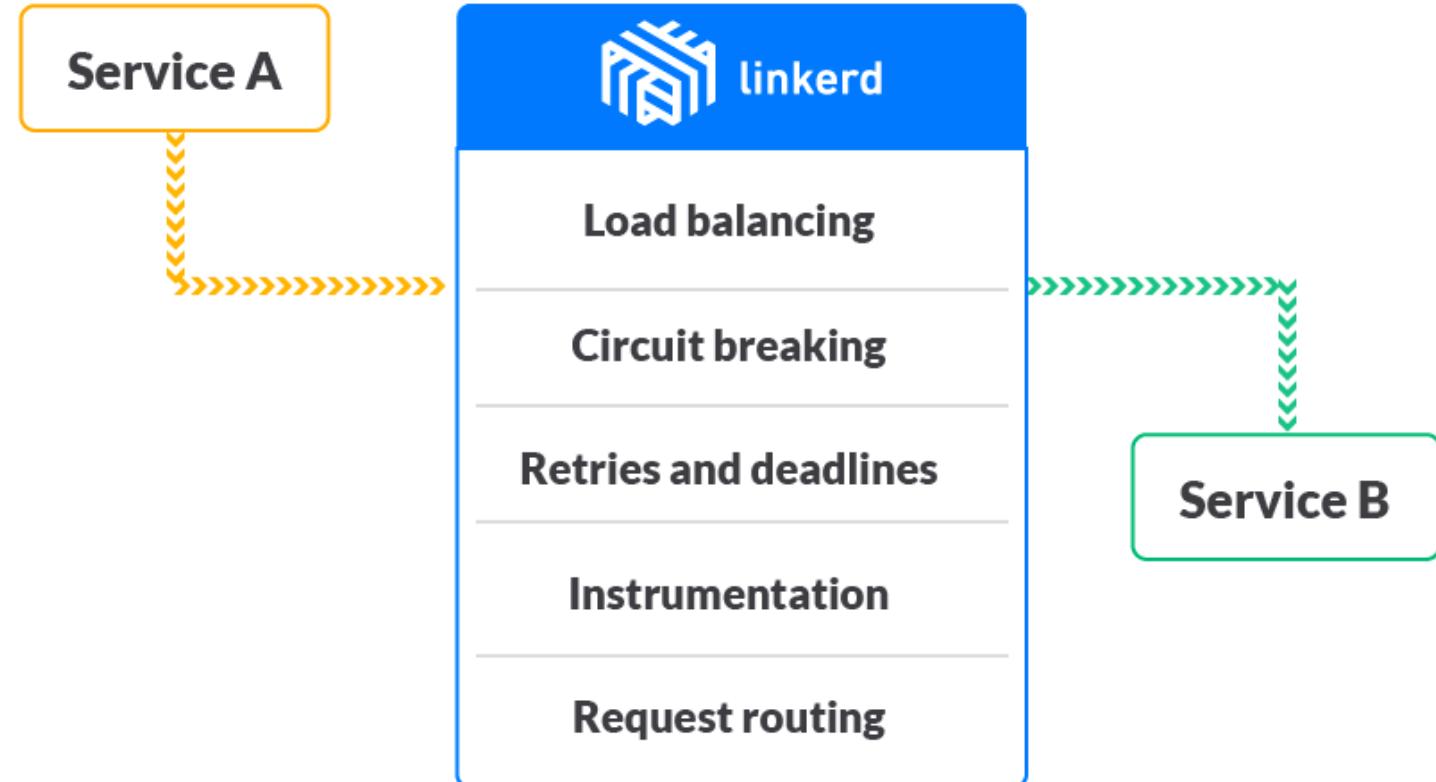
© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Service Mesh implementations

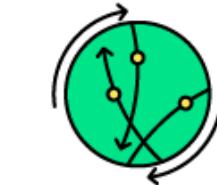
- Envoy
 - And hence Istio
- Linkerd
- Consul



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



Service discovery



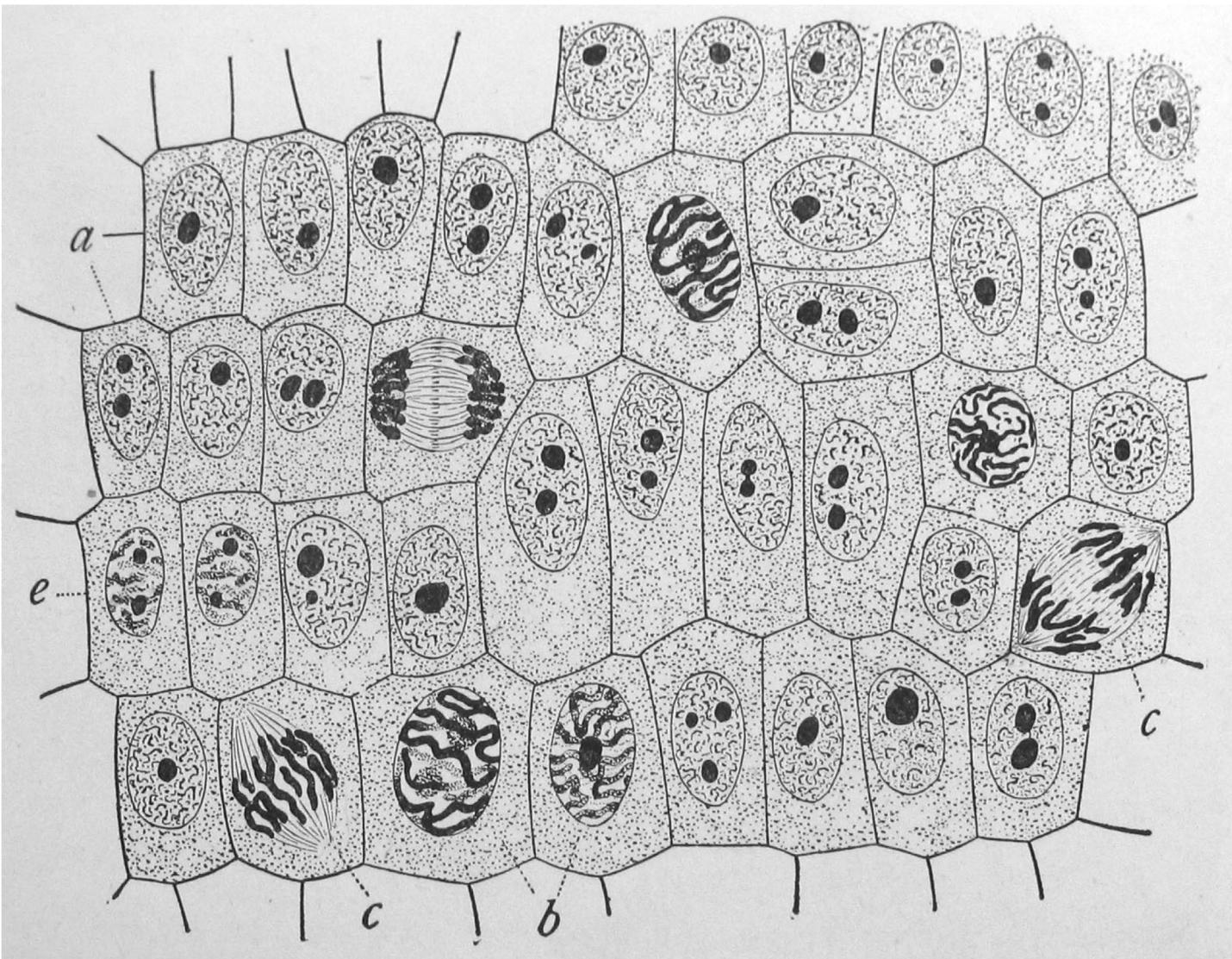
Routing policy

Metrics collection

The cell is the basic structural, functional, and biological unit of all known living organisms



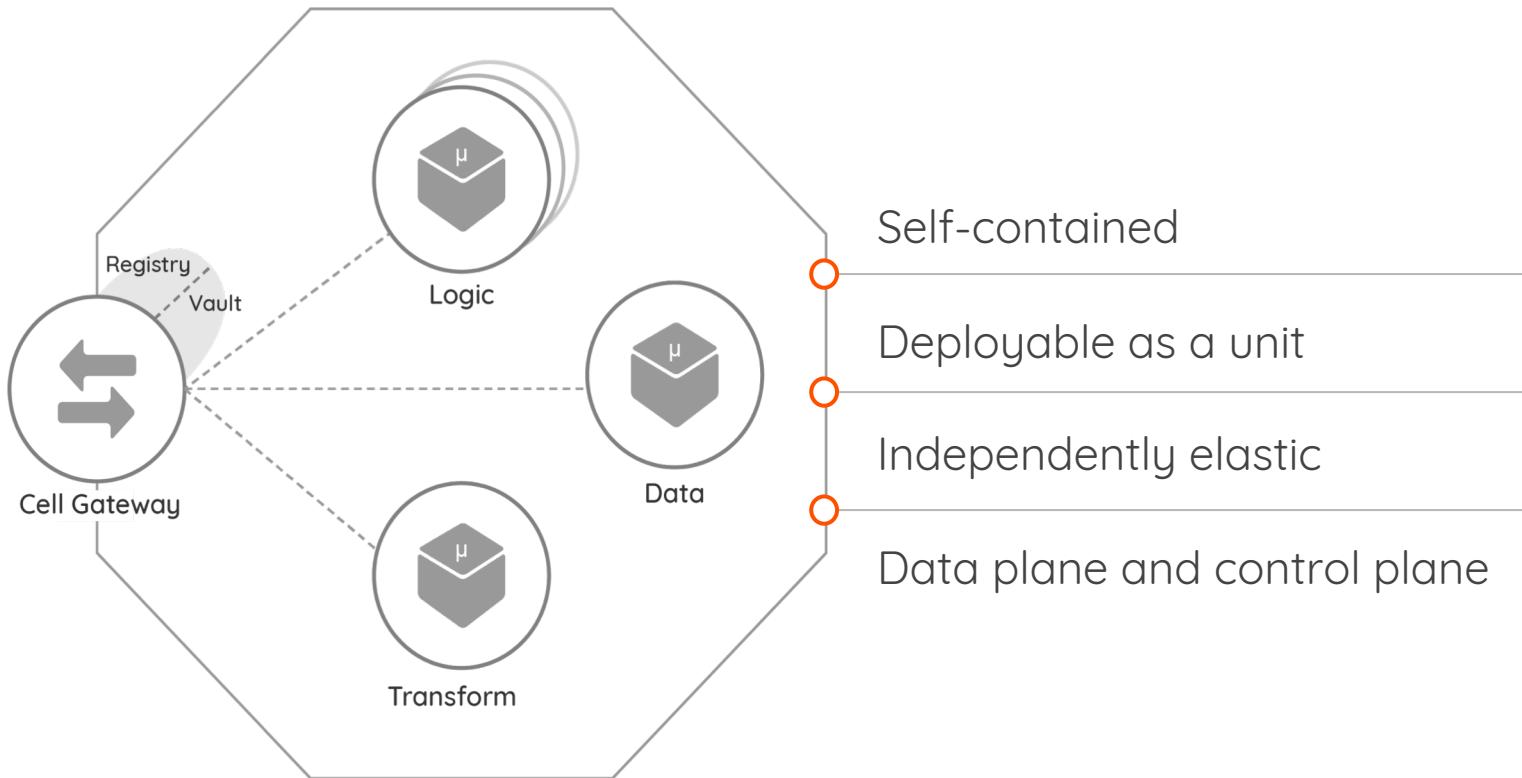
© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Cells Are Building Blocks For The Composable Enterprise

<https://github.com/wso2/reference-architecture/blob/master/reference-architecture-cell-based.md>



BoundedContext

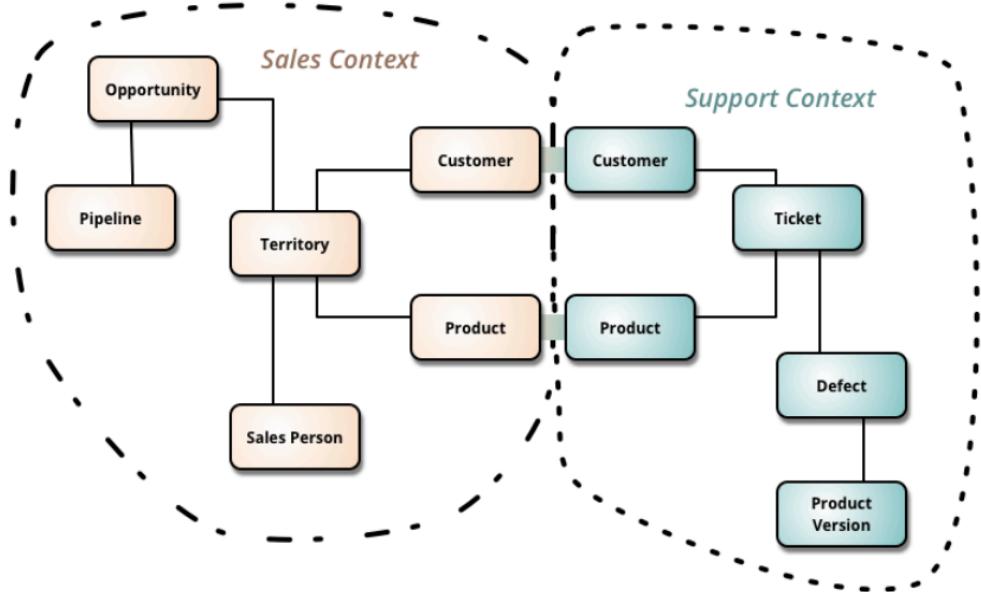
15 January 2014



- ❖ TEAM ORGANIZATION
- ❖ REQUIREMENTS ANALYSIS
- ❖ APPLICATION INTEGRATION
- ❖ DOMAIN DRIVEN DESIGN

Martin Fowler

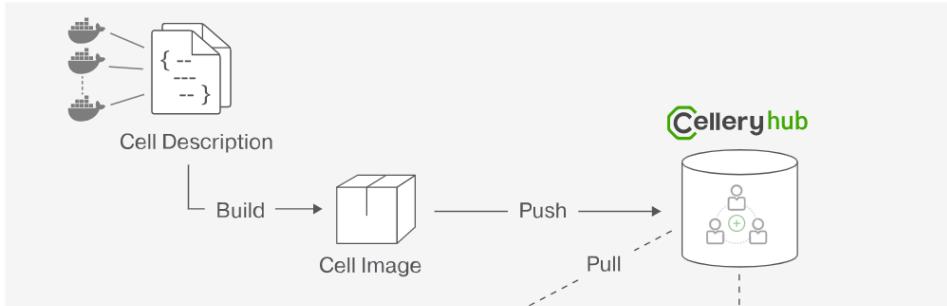
Bounded Context is a central pattern in Domain-Driven Design. It is the focus of DDD's strategic design section which is all about dealing with large models and teams. DDD deals with large models by dividing them into different Bounded Contexts and being explicit about their interrelationships.



Cellery

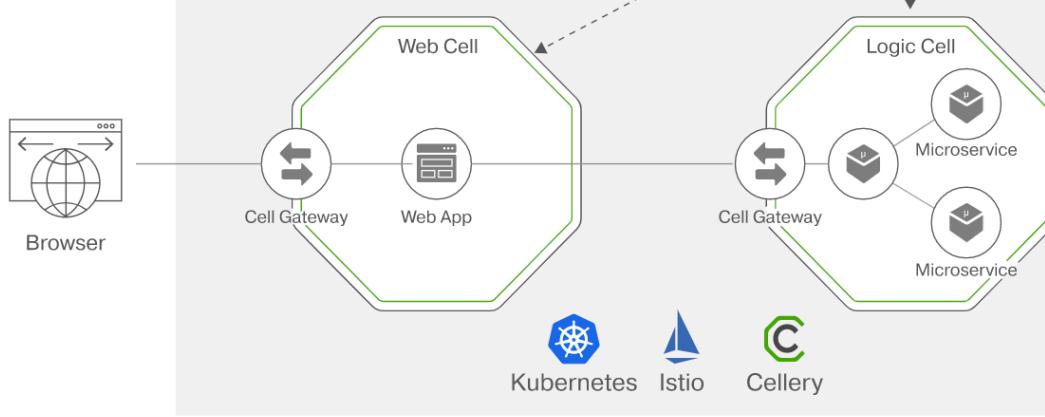
<https://celery.io>

DEVELOP



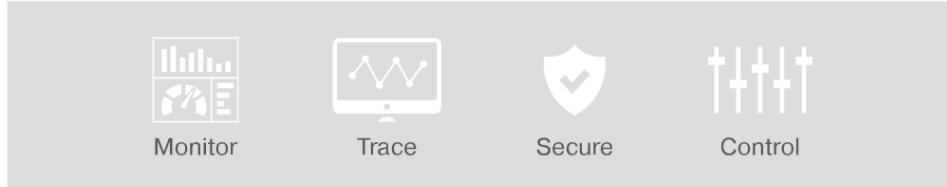
Cellery SDK

RUN



Cellery Mesh

MANAGE



Cellery Observability

Resources

- <http://www.slideshare.net/chris.e.richardson>
- <http://martinfowler.com/articles/microservices.html>
- <http://www.thoughtworks.com/insights/blog/microservices-nutshell>

