ORACLE

Monoliths in a Microservices World

Phil Wilkins
OCI
May 2023

Speaker

Phil Wilkins

Cloud Developer Evangelist

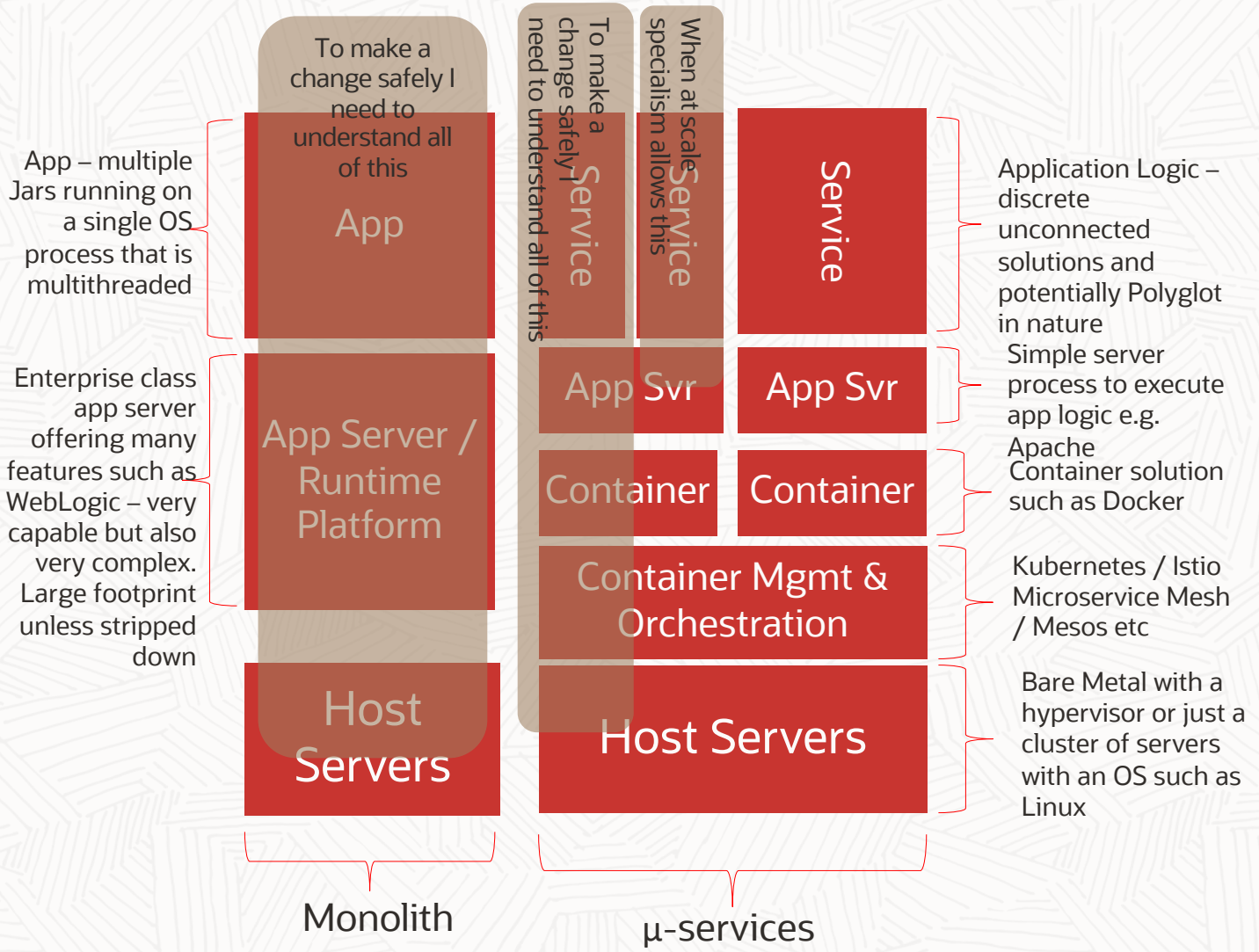
Philip.Wilkins@Oracle.com
https://bit.ly/devrel-slack-emea @Phil Wilkins

mp3monster.org / cloud-native.info / oracle-integration.cloud
linkedin.com/in/philwilkins
github.com/mp3monster
@mp3monster

What is the driver for μ-services?

Development simplicity?

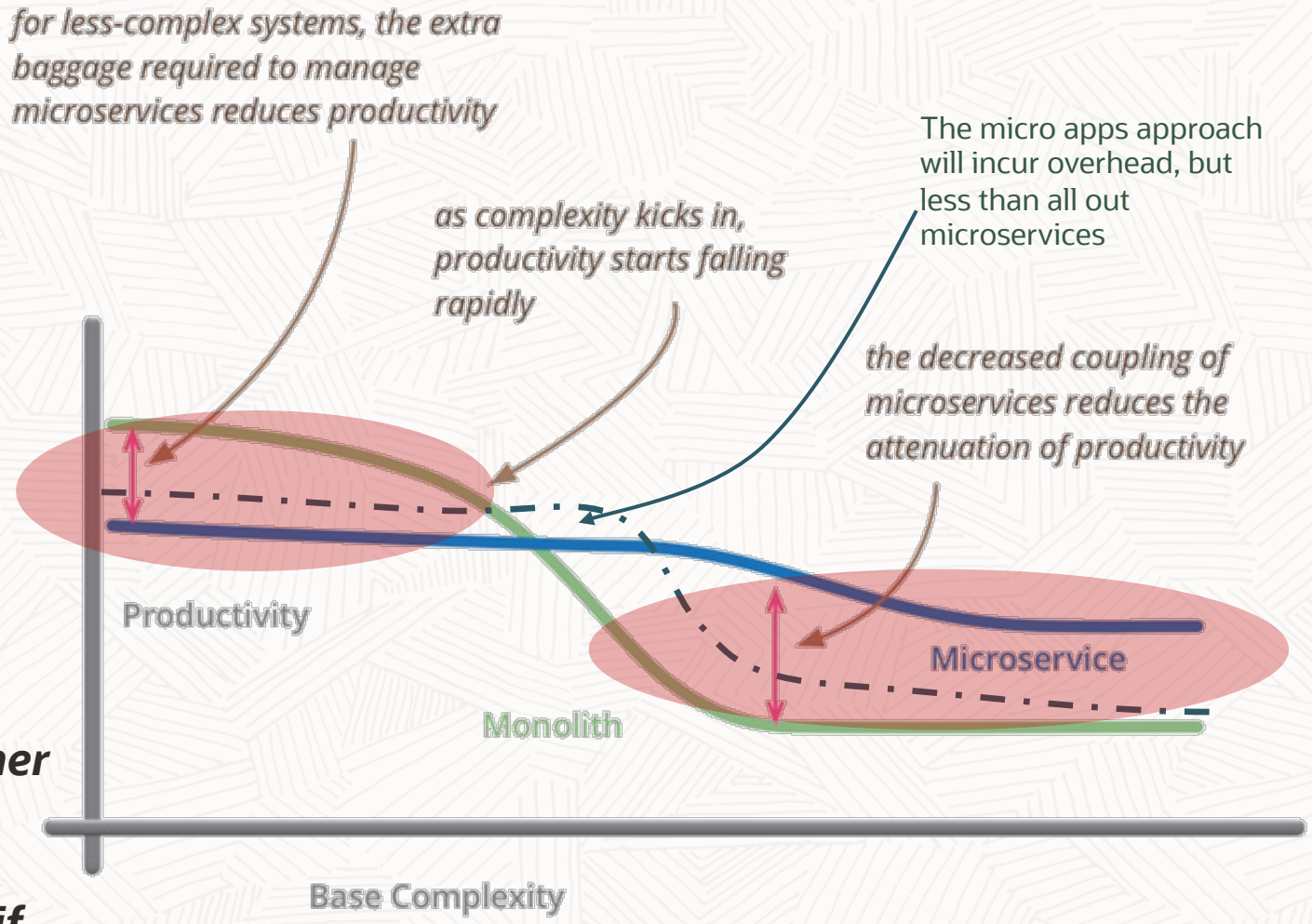
- Both approaches have complexity/ overhead for …
  - Monitoring
  - Repos & deployment
  - Container/host configuration
- µ-service requires additional effort …
  - Storage – persistent volumes, etc
  - Configs for deployment, pod definitions, config maps…
  - Automation to exploit container orchestration benefits
- µ-service simplicity comes with scale when…
  - workload is shared or re-usable
  - Automation is built & maintained



***Question** – does something need simplifying, when should we introduce more abstraction?*

Greater productivity?

- Martin Fowler made the case that the benefit of µ-service relates to the underlying solution complexity
- Gartner has posited the idea of µ-apps as a trade-off
  - Less fine-grained than µ-services
  - Leverage containerization
  - Simplified use of App servers



Martin Fowler
https://martinfowler.com/bliki/MicroservicePremium.html

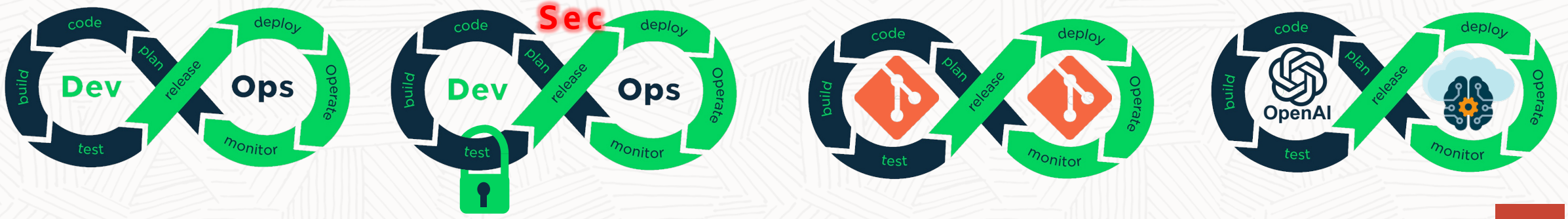
Question – ***How do we decide if whether we can and should transition?***

µ-service needs engineering discipline – if we have that culture – then from the outset – will this give us longer as an operable monolith?

To work with development processes – *Dev |Git|AI|Sec| Ops ?*

- µ-services is an architectural principle – not tools or processes
- *{x}*Ops is an organizational approach, processes supported by tools
- It is easy to link to *{x}*Ops with microservices as they've evolved in the same timeframe
- µ-services complexity have driven *{x}*Ops requirements
- People also confuse CI/CD with *{x}*Ops

Question **– *What prevents us applying these ideas to our existing solutions?***



When looking at μ-services – have we related back to our biz needs?

- **Are we being slowed down?**
  - Is the development process ineffective?
  - Code is maintainable – changing architecture wont fix this
  - Are the governance controls and organizational decisions the source of breaks on deployment speed?
- **Ability to modernize at lightspeed?**
  - Apply change and improvement
  - Meet new business needs
- **Scalability – what scaling demands exist?**
  - Is demand really unpredictable (or is it the business and IT not communicating)?
  - Demand fluctuates unpredictably – automate and dynamically respond – when should we stop the scaling?
- **Resilience what do we want?**
  - Consider monitoring and prevention → as prevention is always better than cure
  - Secure and safe

Question* – *What are we looking for with software development?

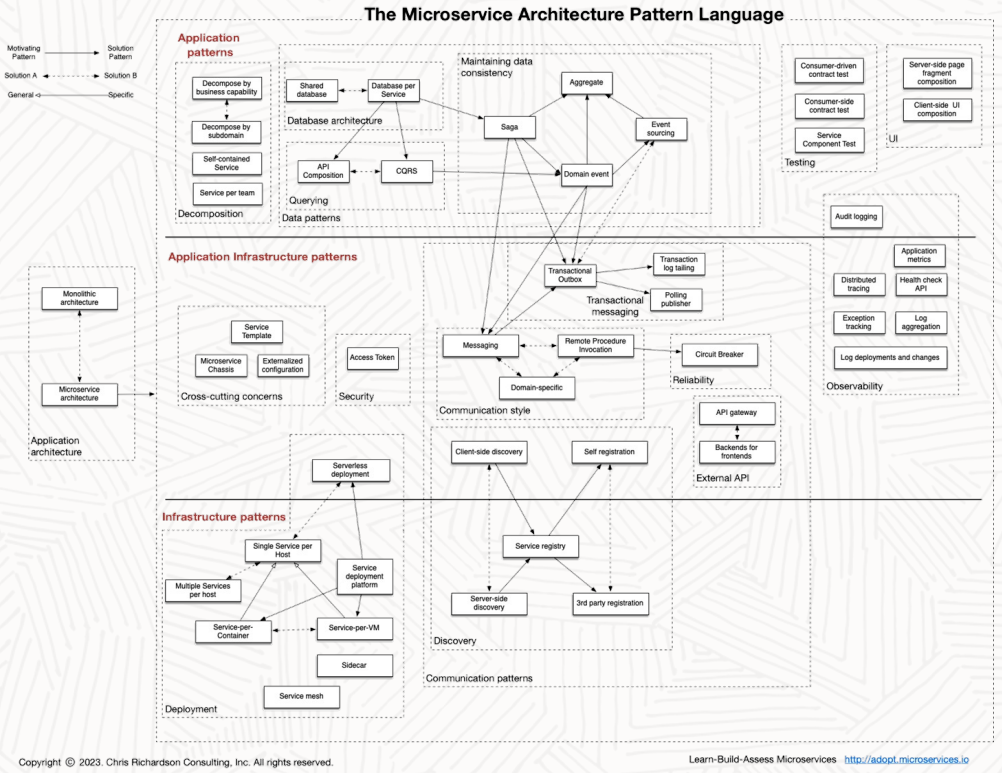
Challenges of adopting μ-services

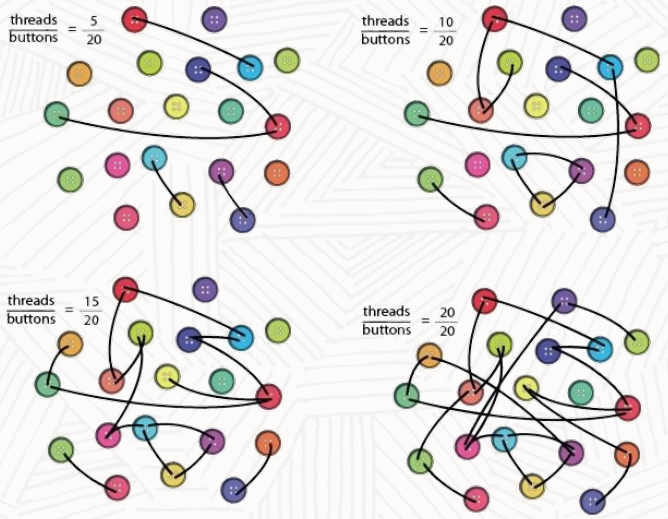
Culture

- Your strategy may be to build and deliver through microservices BUT …
- If your culture isn't oriented to support the ways of working to be effective e.g.
  - No collaboration for cross-cutting concerns
  - Tech silo rather than domain aligned
- Microservices without a culture of automation will be like spinning plates

Design complexity

- Distribution will result in more layers
- Abstraction and separation of layers – increases complexity
  - The smallest component is the simplest
  - The big picture becomes more complex
- Whatever we do we still need to address …
  - Data consistency & integrity
  - End-to-end understanding of what is happening – as more different parts are involved
- Increased re-use means potential for greater impact for change
- Big ball of mud + μ-service = Big ball of mud^2

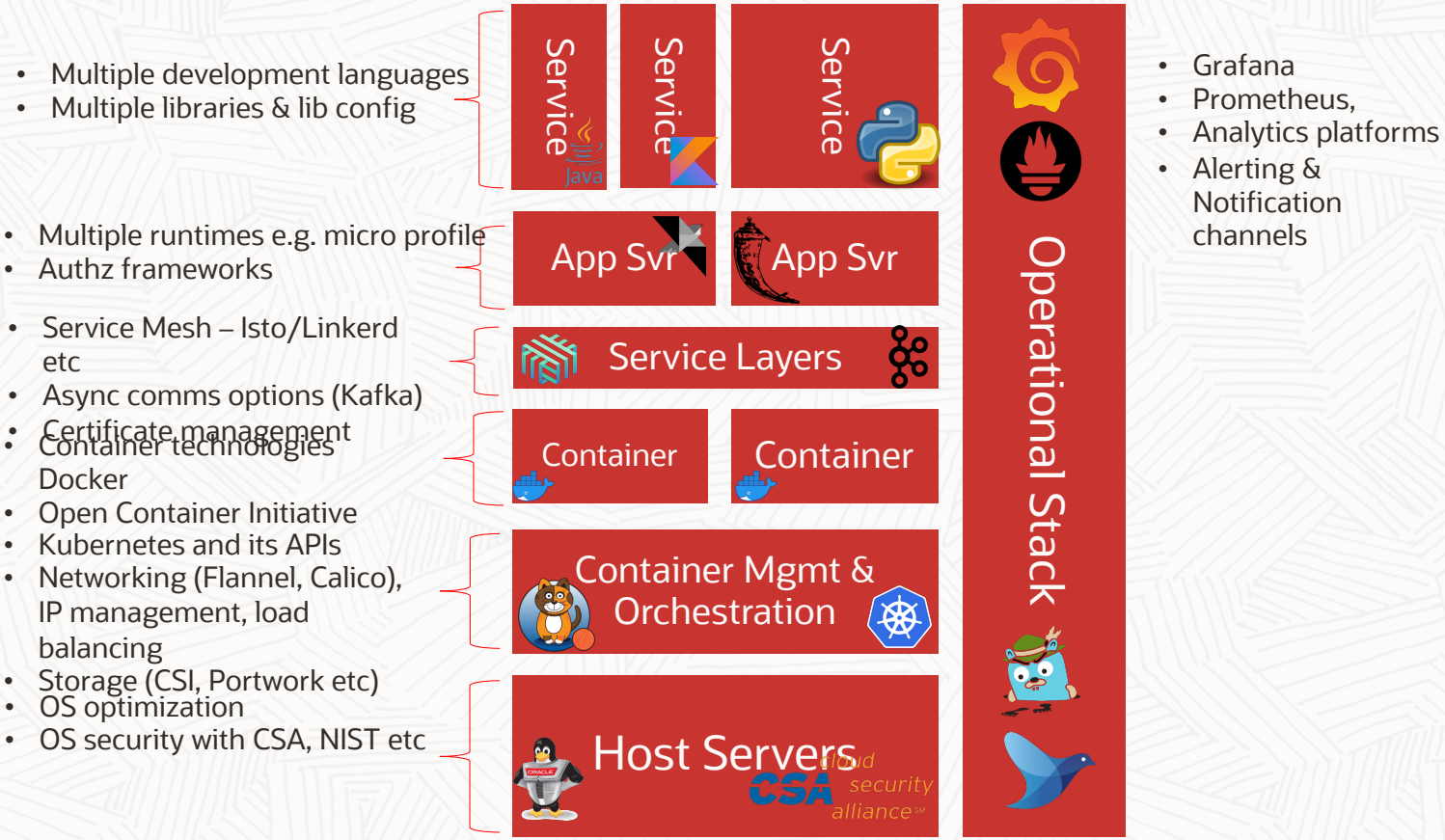




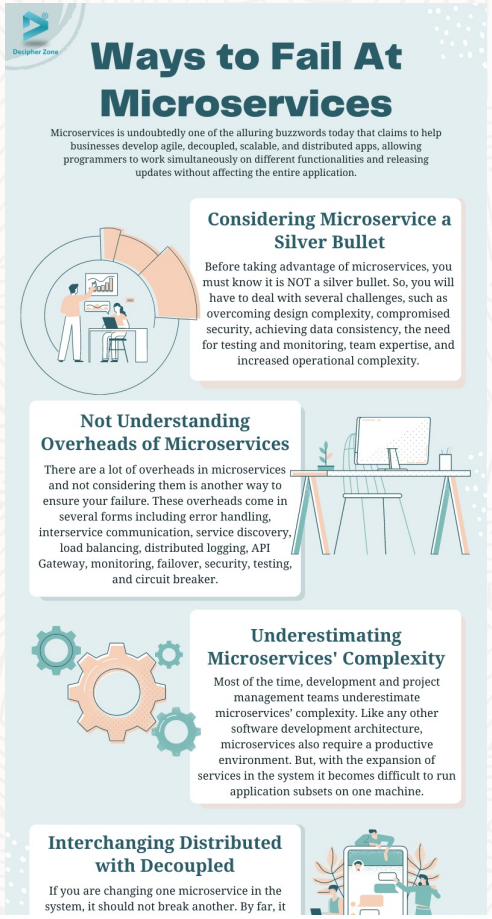
https://androidxhunter.wordpress.com/2017/03/10/the-big-ball-of-mud-antipatten/

Skills

- Operational tooling and processes
  - DevOps
  - Automate as far as ROI allows
- Understanding of many layers
- Development can become polyglot – but how polyglot can you manage
- Polyglot development means differing development processes
- Organization – make sure Conway's law isn't going to bite
- There is no substitute for good engineering practices



µ-service – a challenging & risky journey

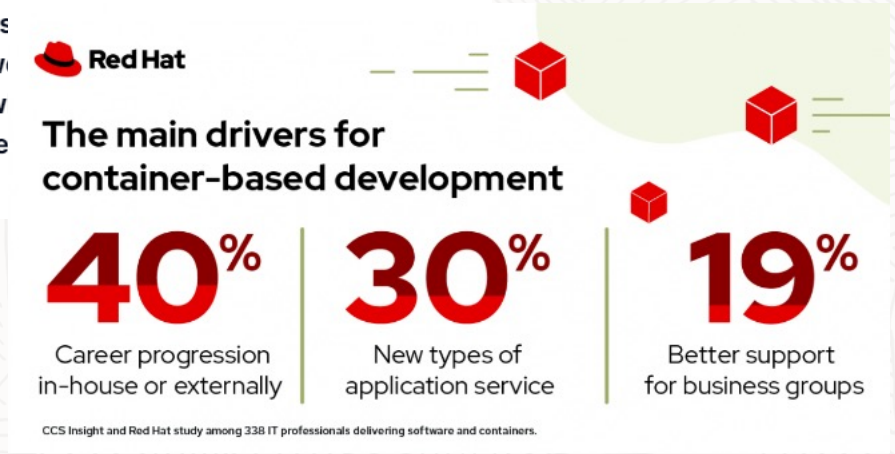


> *"Don't confuse architectural elegance with business value. In other words, microservices are discussed loudly and often amongst architects and developers, with little regard for the business."*
> *—Ross Garrett*

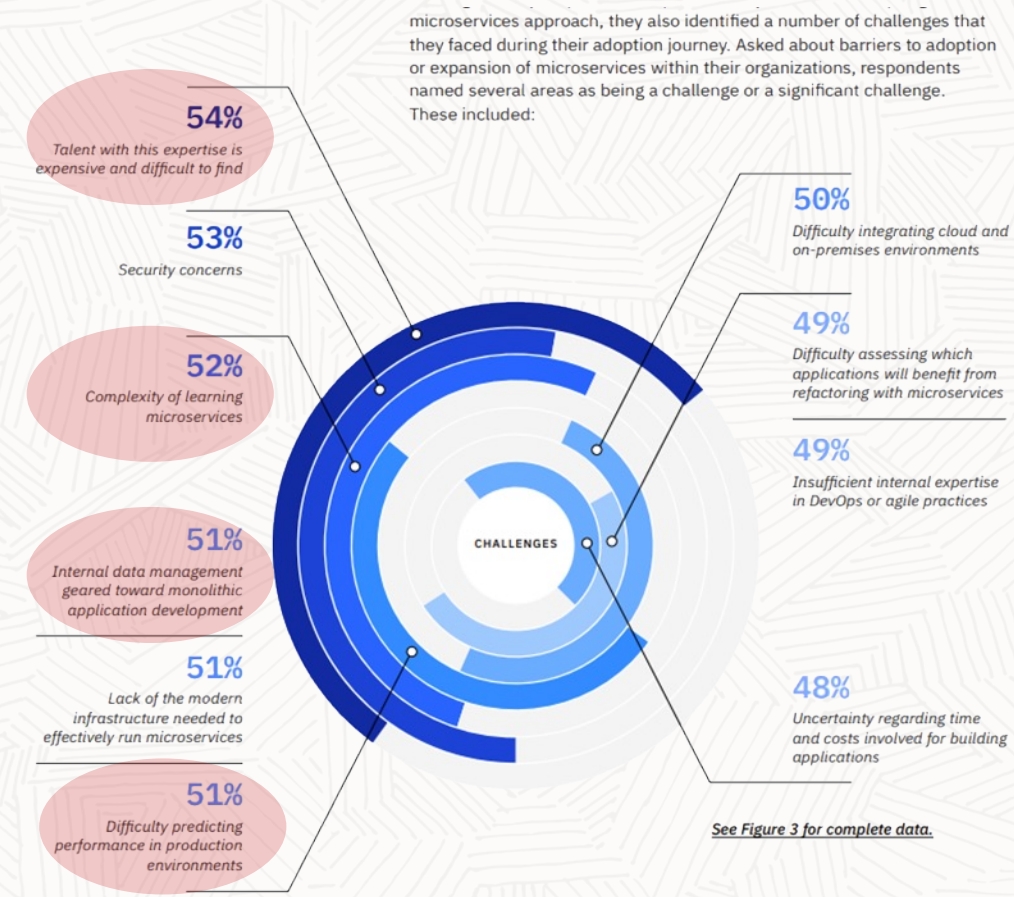
Istio is moving back to the monolith, but it doesn't mean you have to do the same

A couple of days ago, software company Istio confirmed they are moving back from a microservices architecture to somet monolith to ease their product development and requirements with less effort than before. The su widely adopted product by DevOps teams to man architectures indeed. This fact is not ancillary to in the following.

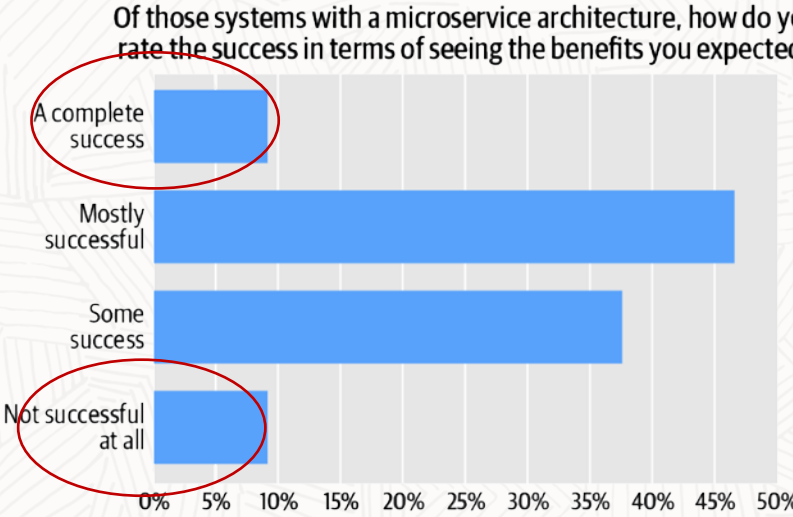
I remain convinced that it is much easier to partition an existing "brownfield" system than to do so upfront with a new, greenfield system. You have more to work with. You have code you can examine, you can speak to people who use and maintain the sys looks like – you have a w easier for you to know w wrong or been too aggre process.

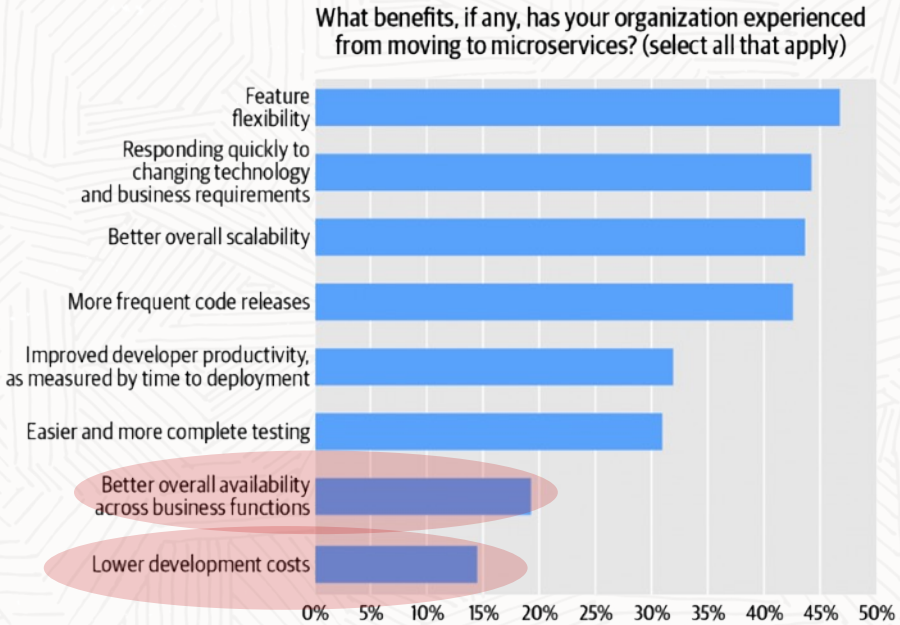


Experience from using μ-services



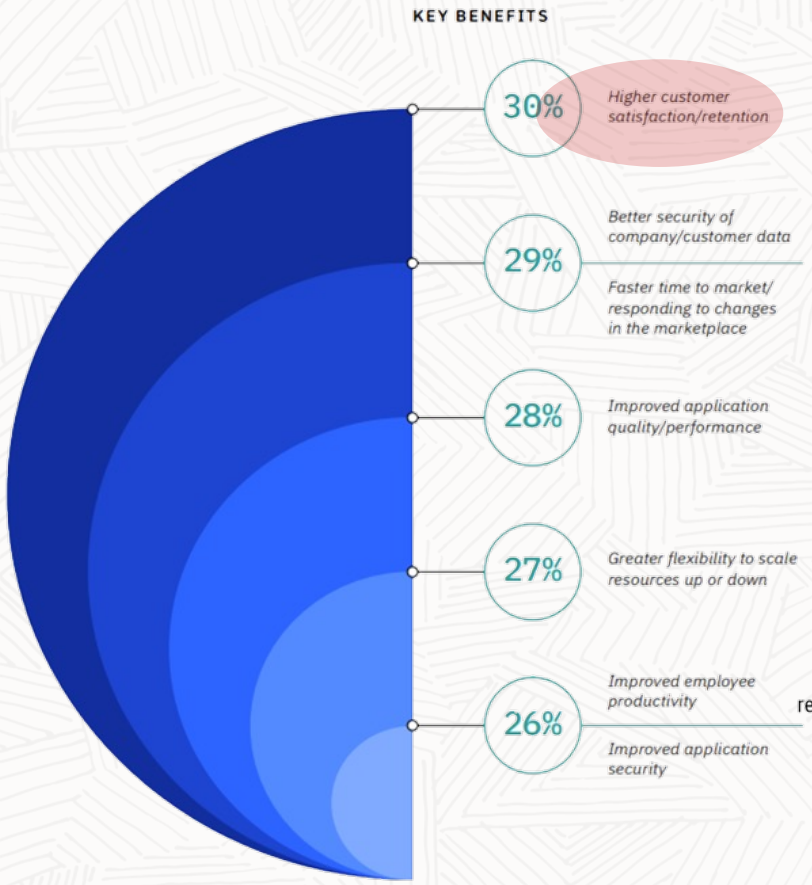
https://www.ibm.com/downloads/cas/OQG4AJAM



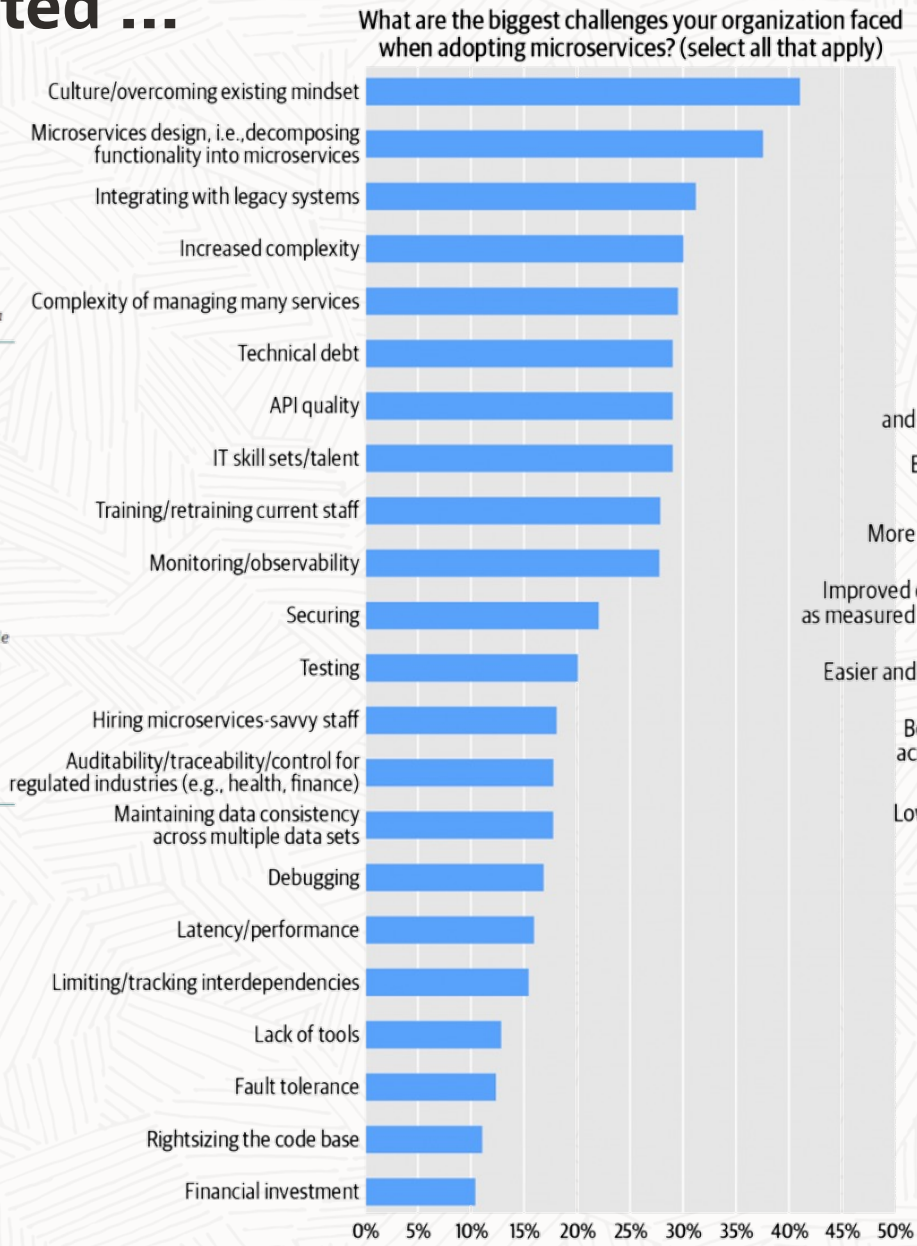


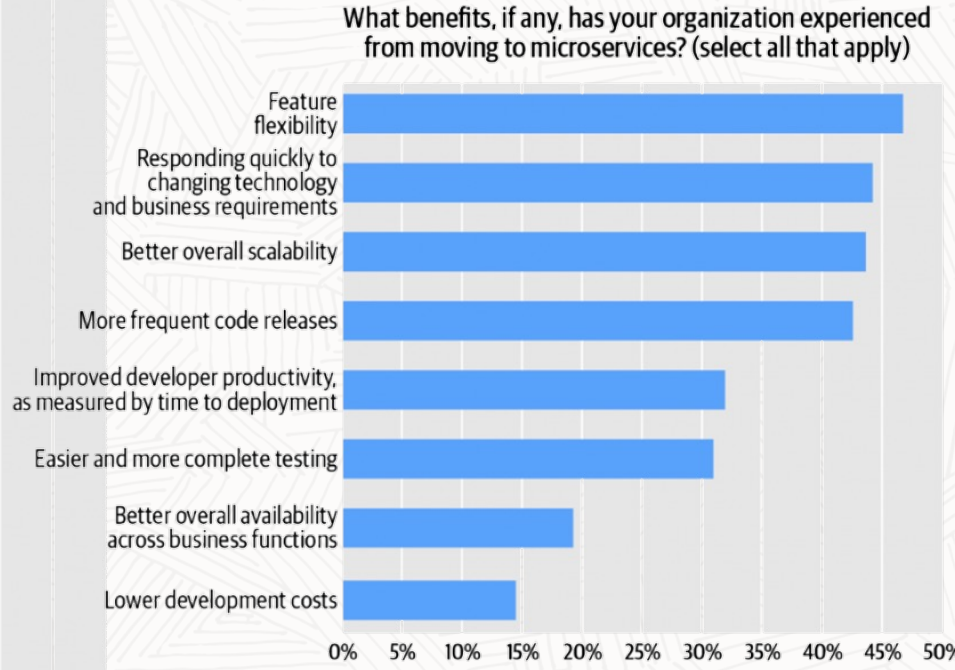
https://www.oreilly.com/radar/microservices-adoption-in-2020/

The experience reported …



Microservices in the enterprise, 2021: Real benefits, worth the challenges
https://www.ibm.com/downloads/cas/OQG4AJAM





https://www.oreilly.com/radar/microservices-adoption-in-2020/

μ-services *and back*

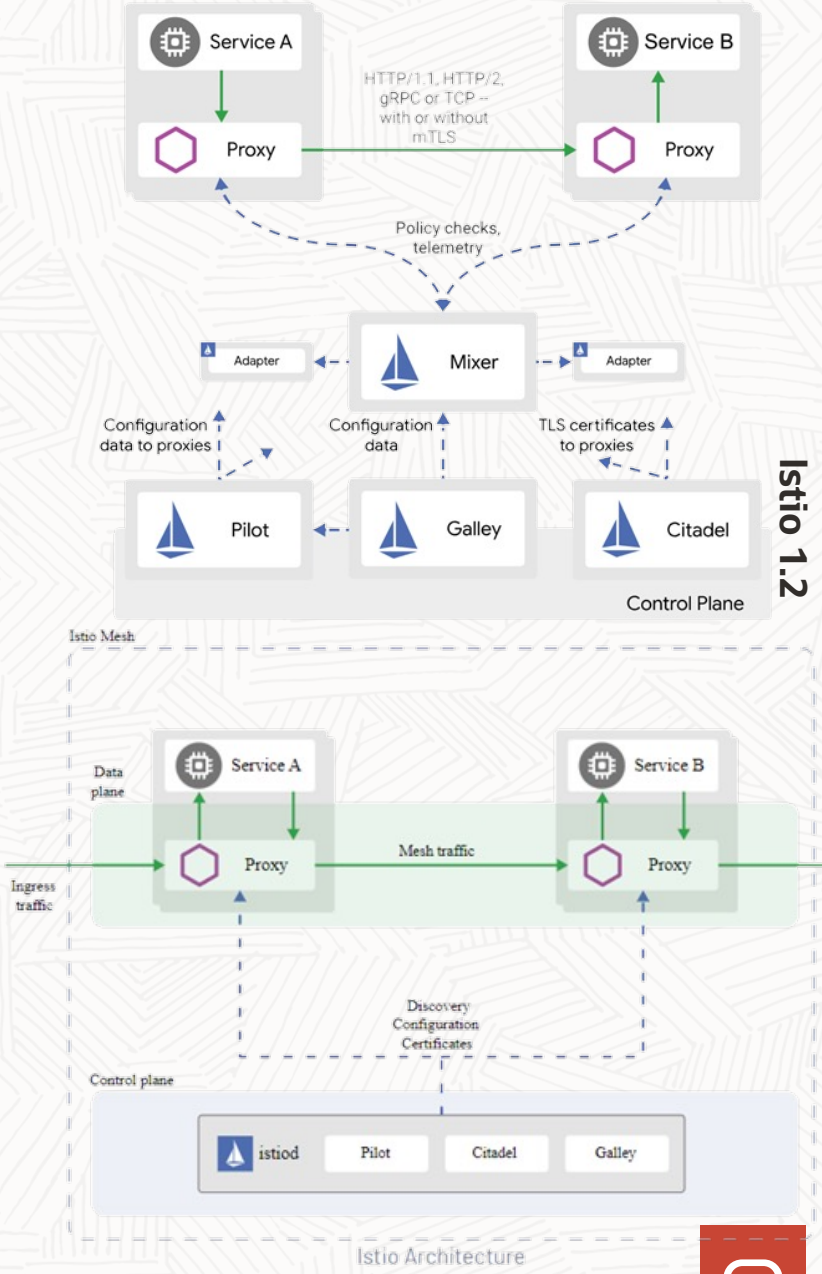
Istio becomes monolithic Istiod

Separate services of …

- Pilot (Traffic Management)
- Citadel (Certificates & Secrets management)
- Galley (Configuration management)
- Mixer (Policy & Telemetry – *deprecated as part of v1.5*)

Consolidated into Istiod, providing all the capabilities of Pilot, Citadel & Galley.

- Proxy's talk to daemon on the control plane.
- Modular design maintained



Real-world example - Segment

Segment – a data analytics platform, now part of Twilio

- Adopted microservices to help with fault isolation and aid observability
- Saw initial benefits in delivery, but …

Problems came from

- Operational overhead of the microservices
- Many services had a functional commonality – addressed by building libraries, BUT that slowed development – testing took longer
- Forking code base undermined maintainability

Consolidated this area of code – based found the still have productivity and high performance

https://www.infoq.com/news/2020/04/microservices-back-again/
Thanks to Alexandra Noonan for sharing the story

"If microservices are implemented incorrectly or used as a band-aid without addressing some of the root flaws in your system, you'll be unable to do new product development because you're drowning in the complexity.“

Real-world examples

ShopKeep (part of Lightspeed)

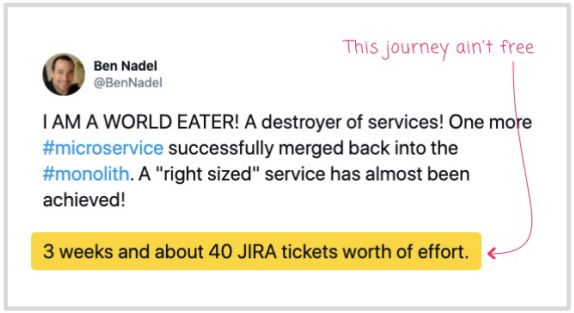
- Classic microservices story – building monolith – worked fine until hit issues of scaling
- 1st attempt failed –
  - Did't get properly grasp challenges that monolith had
  - If monolith architecture has lost cohesion and quality – what chance will you microservices have?
- 2nd time around –
  - Tried to bite too much off at once
  - Partitioned the wrong way – resulting in tightly coupled services

SHOPKEEP

Thanks to Paddy Carey for sharing the story
https://www.youtube.com/watch?v=0jODVkkwiMc

InVision

- Issue of people/process than technical – with release pipeline not quick enough
- Conway's law bit – team to support 'legacy' steadily accumulated microservices – small team – lots of small discrete pieces of functionality for a small team – increasingly impractical

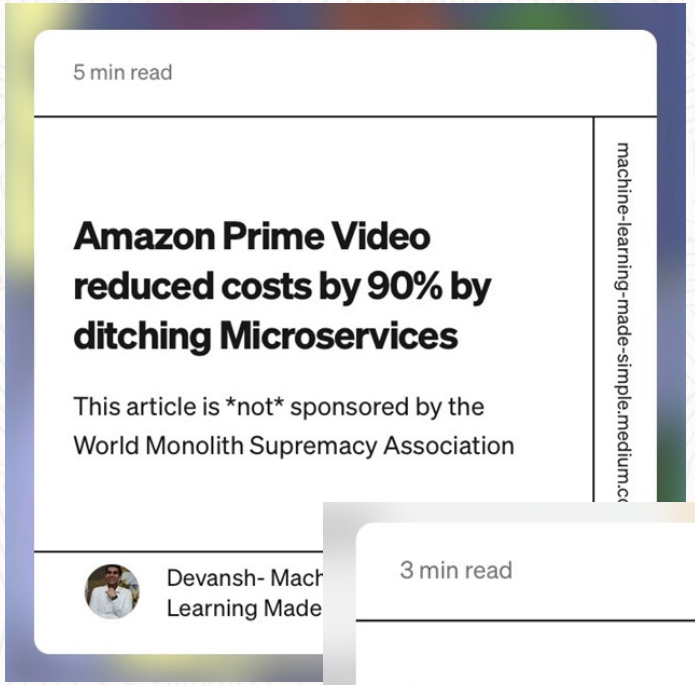


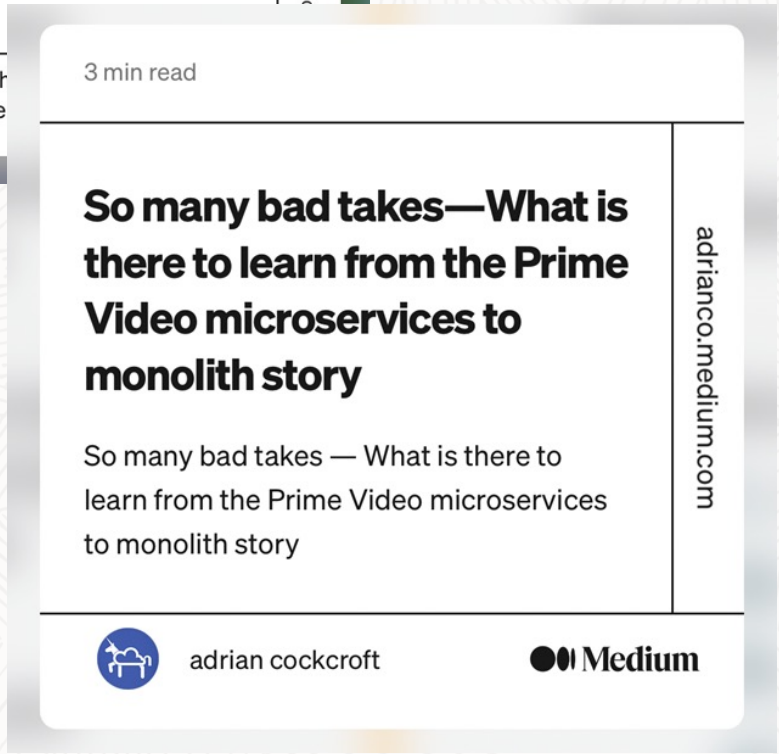
InVision

Thanks to Ben Nadel for sharing the story
https://www.bennadel.com/blog/3944-why-ive-been-merging-microservices-back-into-the-monolith-at-invision.htm

Amazon!!

- Some of these headlines are a bit clickbait BUT
- Background …
  - Needed service to examine video for encoding & audio sync errors
  - Solution too slow & costly
  - Elected to use Lambda and Step Functions
  - Organization with a culture that gets microservices
- In my opinion …
  - Technology before design – wrong way around
  - Cost issues could have been spotted with a bit of strategic thinking



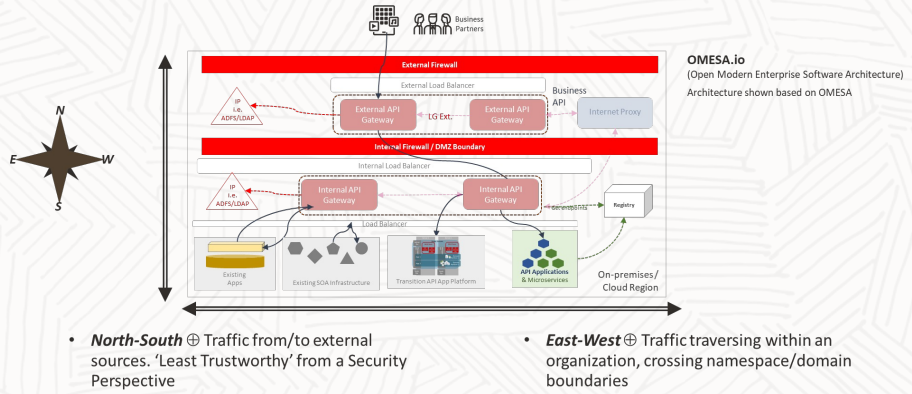


Strategies to minimize monolith challenges, Cohabit & transition

Internal API Gateways and Message brokers – Enabling Anti-corruption layer to help protect modularity

We can consider using internal API gateway(s) to …

- Enforce rate limiting to avoid possible runway processes swamping you with API calls
- Capture usage data for billing, cross-charging, and demand forecasting
- Creating points of abstraction
  - Gateways can mask changes in deployment for consumers
  - Enforce loose coupling – some APIs are intended operational purposes NOT for others to build new applications against
- Gateway as a focus of security management
- Provide easier points for measuring utilization (investment value)
- Use of Gateway can be used to support …
  - Enforcement of design & development approaches
  - API First (developing contract visibility etc)

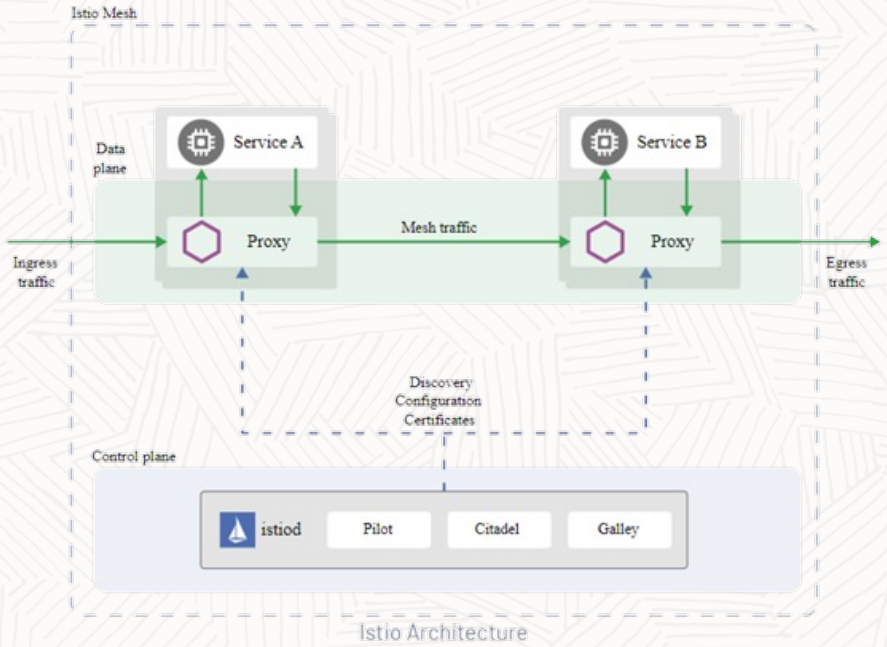


- ***North-South*** ⊕ Traffic from/to external sources. 'Least Trustworthy' from a Security Perspective
- ***East-West*** ⊕ Traffic traversing within an organization, crossing namespace/domain boundaries

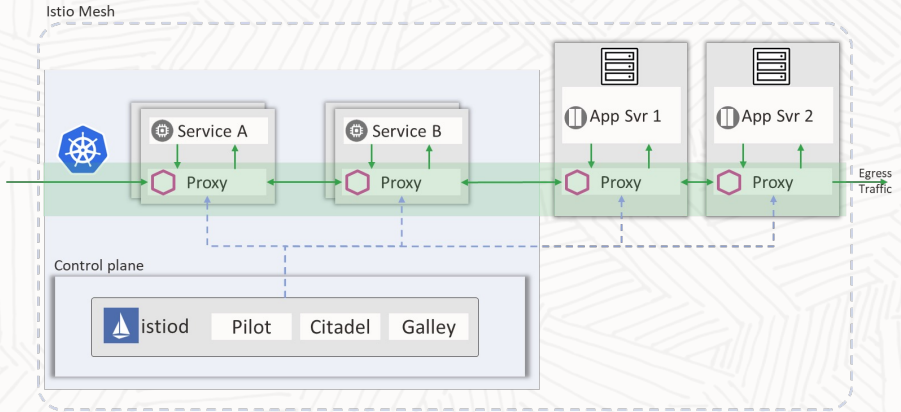
Service Mesh proxy for monoliths

- Service mesh helps development & operations by …
  - Abstracting and routing traffic between end-points
  - Securing traffic between endpoints (Authz, encryption …)
  - Observability & monitoring
  - Extensibility - Extend proxies with WASM
- Nothing here unique to a µ-services, so…
  - Deploy Envoy proxy in front of Monolith
  - Proxy needs to talk to ***Istiod***
  - Monolith will appear to all services in the same manner
- This strategy is well-developed and proven
  - BookInfo demo app includes Istio with a VM deployment
  - Standard Istio documentation includes VM deployment guide
  - Reference case with Bluecore

Typical Istio architecture …

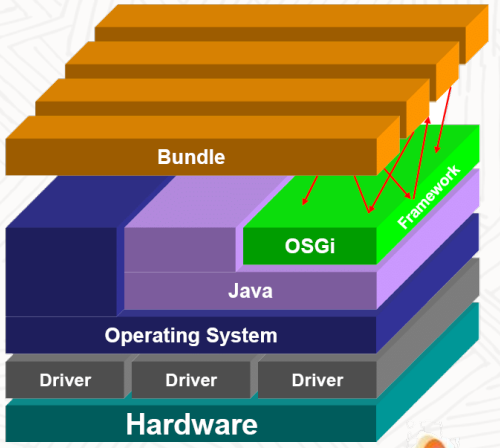


Istio architecture with Monoliths …



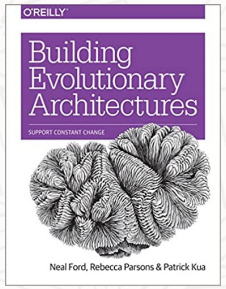
Good design aka modular monolith …

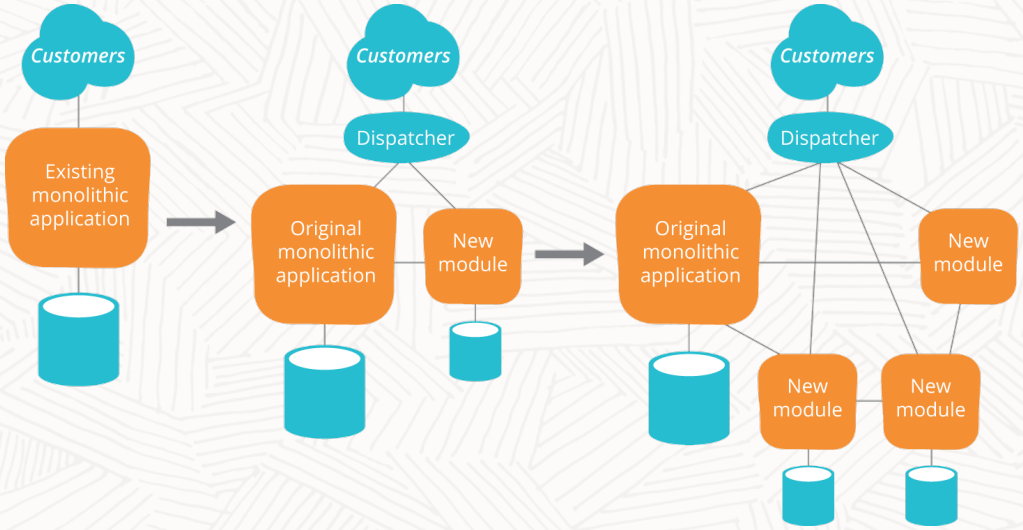
- We can still apply good design principles …
  - Domain Driven Design
  - Design Patterns such as Inversion of Control
  - API First
- Exploit language modularization features and decoupling
  - OSGi (Apache Felix, Celix …)
  - Python packages
  - Go Modules ….
- Automate, automate, automate …
  - Just because we're not deploying as small microservices doesn't mean we can't reap the rewards of automation
  - Load balancing + active monitoring instead of Kubernetes health management



Evolutionary Architecture

- Develop and measure fitness functions – use metrics to guide incremental change
  - Metrics more than dev/unit testing
- If the organization is adversely impacting architecture – Inverse Conway's Law Maneuver
  - Cross-functional teams
  - Inter team communication
  - Product over project
- Changes are always small increments keeping system stability.
- Identify redundant functionality/dead code and remove
- Bounded context views when overlayed with implementation will help identify journey
- Cross contexts using APIs/Messaging etc





https://continuousdelivery.com/implementing/architecture/

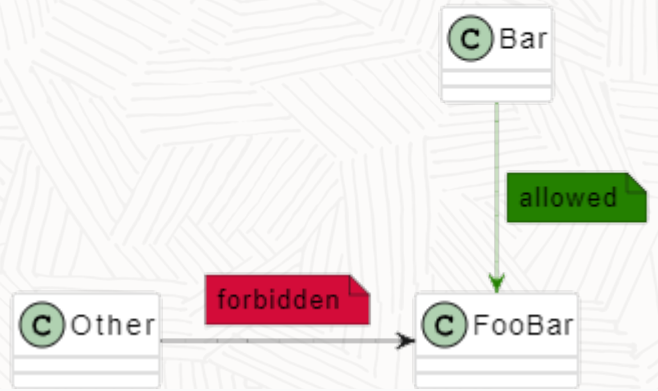
Culture of Discipline – Structured Monolith that can be (de)composed

Class Dependency Checks

Tools like ArchUnit/ArchUnitNet (xUnit extension) can be used to flag abuse of module boundaries..

- Unit tests can ensure code modularity is respected
- Does need discipline to create tests

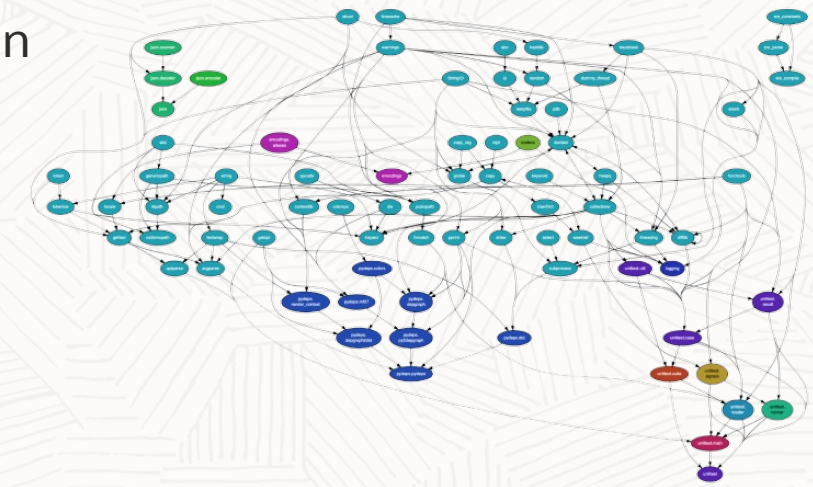
Tools like JDepend will help identify coupling




```
classes().that().haveNameMatching(".*Bar")
    .should().onlyHaveDependentClassesThat().haveSimpleName("Bar")
```

Software Bill of Material (SBOM) tooling to help control dependencies…

- Primary goal is to support Supply Chain issues, which can also help with modularity
- Smaller dependencies can point to control of coupling
- Needs your code to be built as modules
- Need to have SBOM analysis/visualization to help

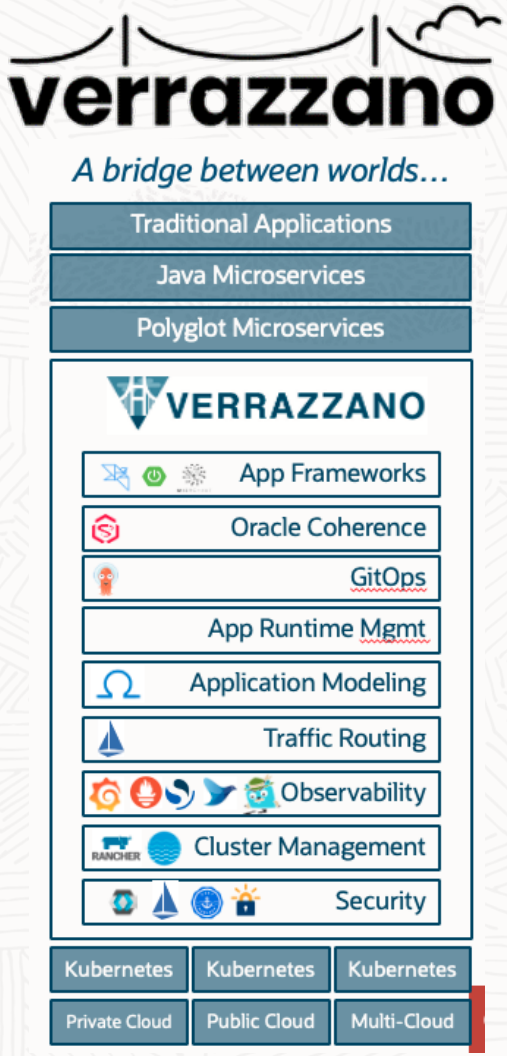


Container Instances

- Container Instances service – available from most hyperscalers
  - Deploy solutions from a container e.g. Docker
  - Could view Container is largely an installation abstraction (i.e. replace MSI, .DEB, RPM mechanisms)
  - Include monitoring, auto restart, etc.
  - None of the complexity of K8s
- Use of containers – create the opportunity for some isolation – can serve as a stepping stone to μ-service

Adopt µ-service tools without tearing apart your monolith

- Verrazano (similar solutions VMWare Tanzu, Google Anthos etc)
  - Containers are your installation tech
  - Tools to help containerize and deploy monoliths
  - Completely Open Source – curated tech stack designed to support Monolith (e.g., WebLogic – J2EE workloads) and services
  - Deploying all the different building blocks is addressed for you e.g., Istio, Keycloak, setting up monitoring ….
- CNCF projects can work with Monoliths e.g.
  - Fluentd, Fluentbit – deploy anywhere (Fluentbit's origins are IoT)
  - Grafana & Prometheus can be deployed anywhere – its just software
  - So we adopt a lot of µ-service tools and techniques in a more incremental manner



Conclusion

- Applying μ-services can be hard, and failing is easy …
  - Even the best have got it wrong
  - We think about technology – but we need to have our organization, culture … in order
- μ-services are an **amplifier**
- μ-services aren't the goal – **business value is the goal** - μ-services is a way to get there
- You don't need to be building μ-services to benefit from cloud-native technologies
- Monoliths can cohabit and participate with microservices
- Adaption of a Martin Fowler quote – ***don't start with μ-services UNLESS there is a clear immediate business need. DO start with MODULARITY. Move to microservices when you're in good health and need it***

Monoliths are *Okay*
μ-services are *Okay*

balls of mud are *BAD*

μ-service *NOT* a fix for poor practice & org issues

ORACLE

Questions & goodies…

Thank you

Phil Wilkins

Cloud Developer Evangelist

Philip.Wilkins@Oracle.com
http://bit.ly/odevrel_slack @Phil Wilkins

mp3monster.org /
cloud-native.info / oracle-integration.cloud
linkedin.com/in/philwilkins
github.com/mp3monster
@mp3monster