

Architecture: Microservices

17-313: Foundations of Software Engineering

<https://cmu-313.github.io>

Michael Hilton and **Chris Timperley**

Fall 2025

Smoking Section

- Last full row



Learning Goals

- Understand the difference between traditional monolithic and modular software architectures
- Explore various approaches to modularity, assess their respective benefits and drawbacks, and reason about **when to use them** and **how to use them well**
 - Monoliths, Plugins, Services, Microservices, Modular Monoliths

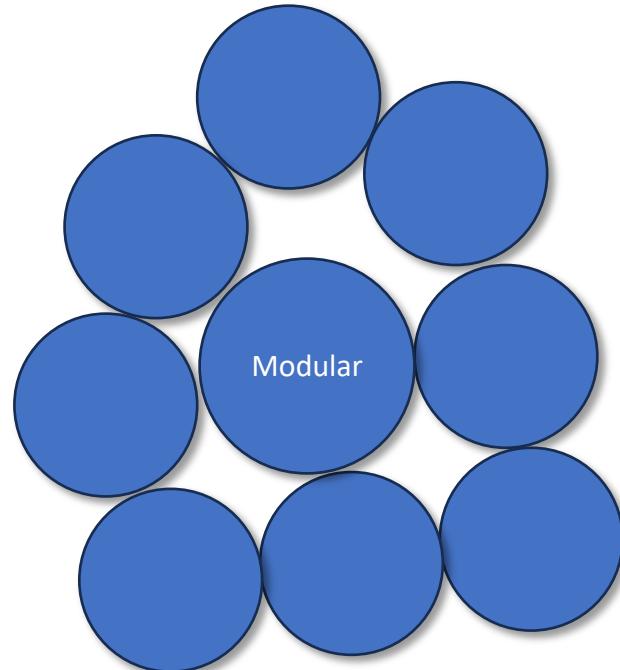
Outline

- Monoliths vs. Modular Architecture
- Service-Based Architecture
 - Case Study: Chrome Web Browser
- Microservices
 - Principles of Microservices
 - Advantages and Challenges of Microservices

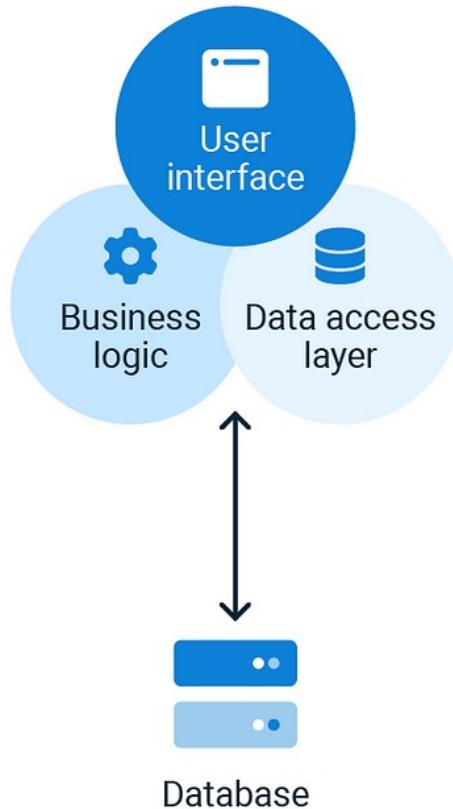
Monolithic vs. Modular Architectures



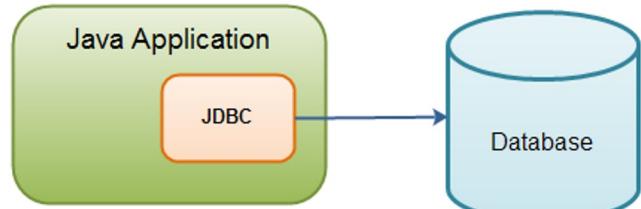
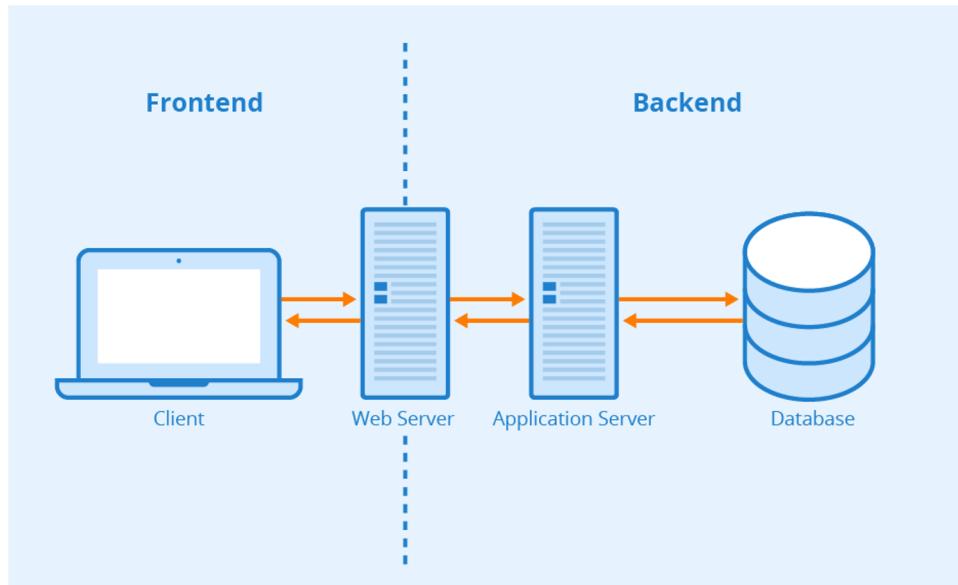
Monolithic



Monolithic Architecture

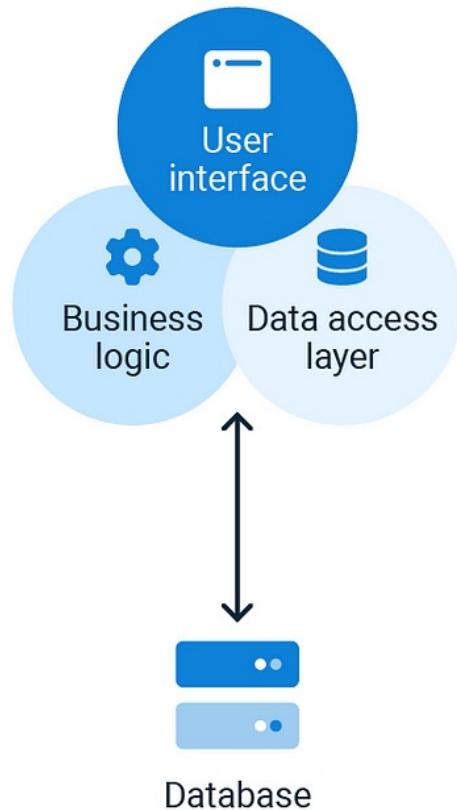


Monolithic Styles

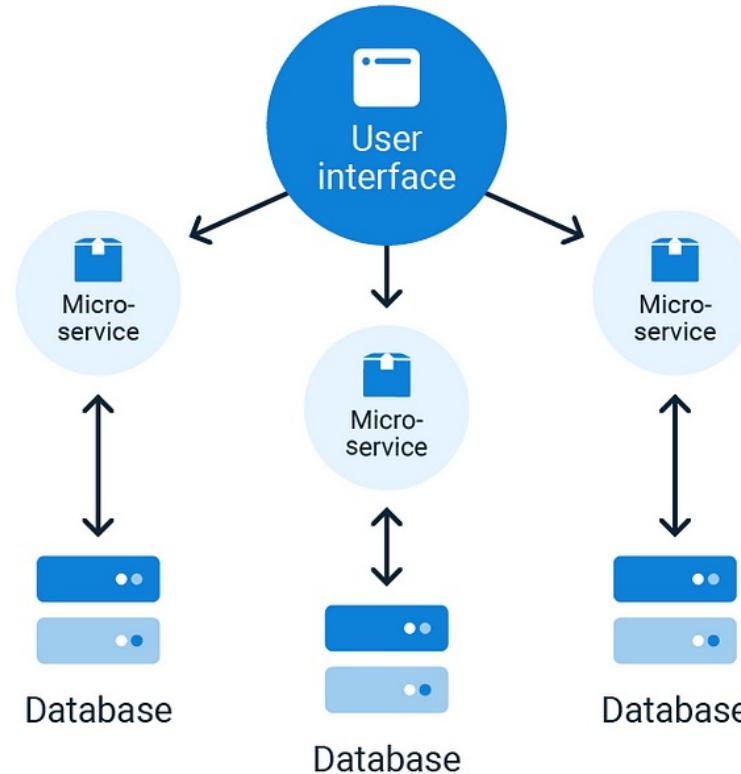


Source: <https://www.seobility.net> (CC BY-SA 4.0)

Monolithic Architecture



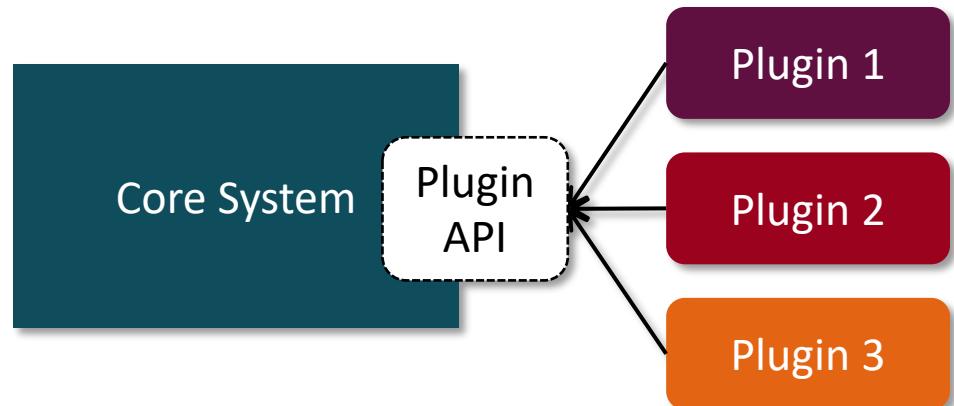
Microservice Architecture



Modularity comes in many ways

- **Plug-in architectures**

- Distinct code repositories, linked-in to a monolithic run-time
- Examples: Linux kernel modules, NodeBB themes, VS Code extensions
- Separates development, but runs as “one”

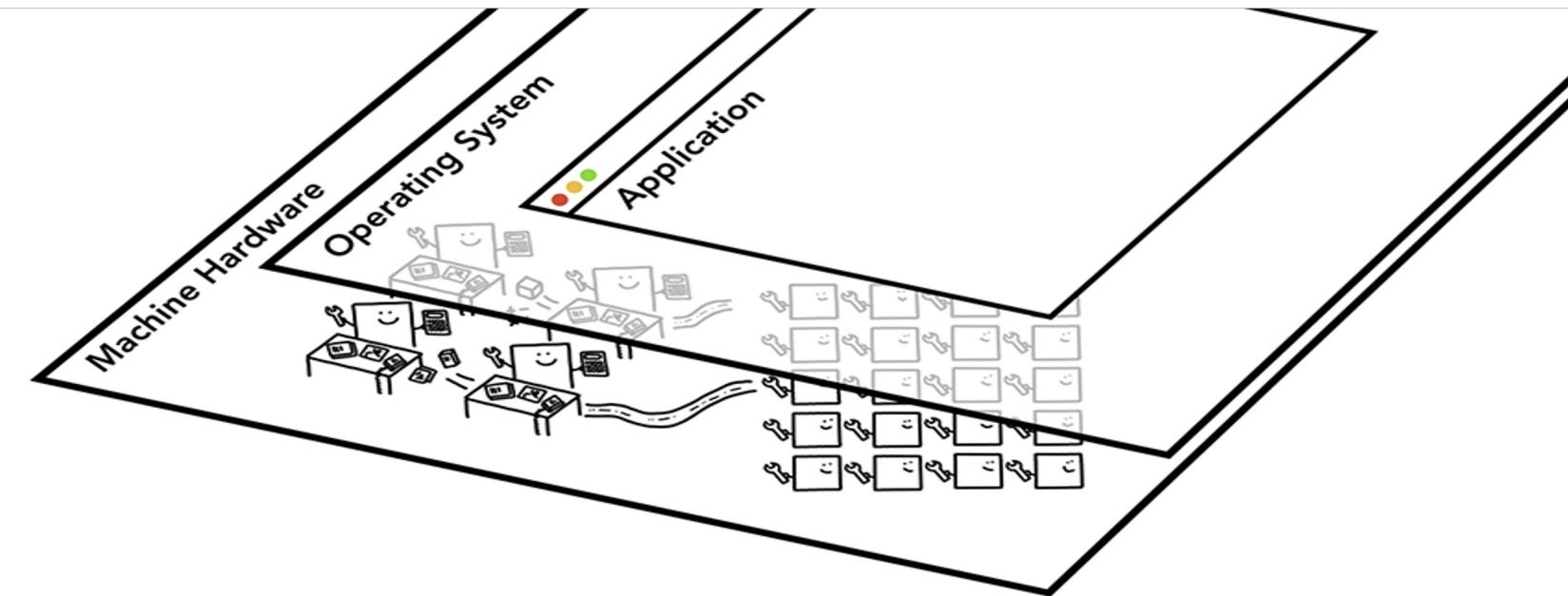


Modularity comes in many ways

- Plug-in architectures
- Service-oriented architectures
 - Distinct processes communicating via messages (e.g., Web browsers)
 - Separates run-time resource management and failure / security issues.
- Distributed micro-services
 - Independent, autonomous services communicating via web APIs
 - Separates almost all concerns

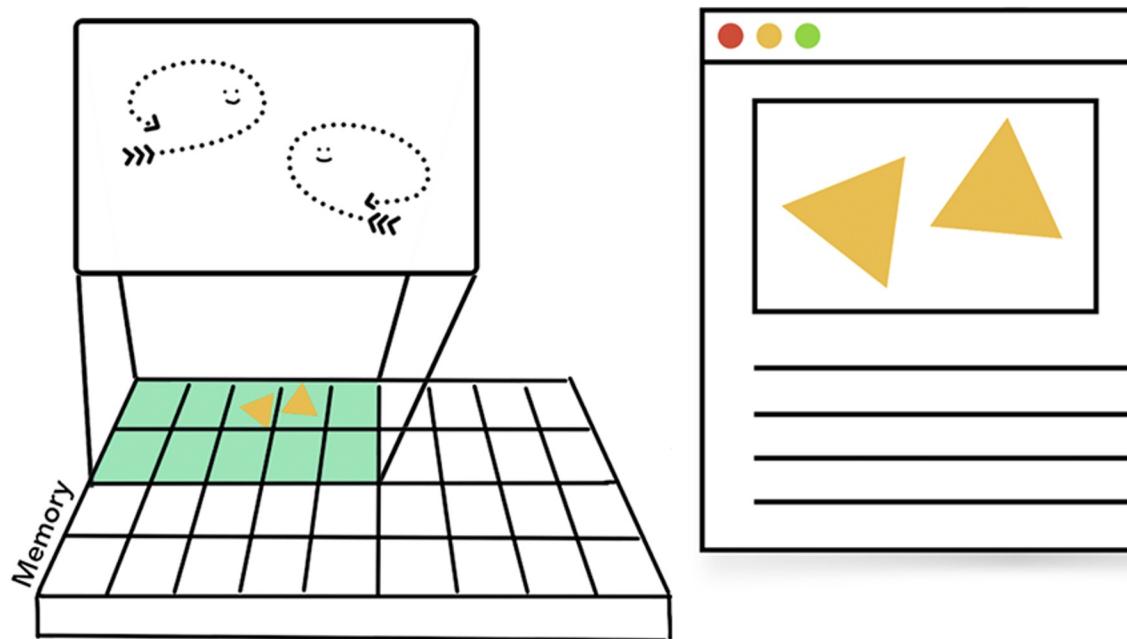
SERVICE-BASED ARCHITECTURES

Case Study: Web Browsers



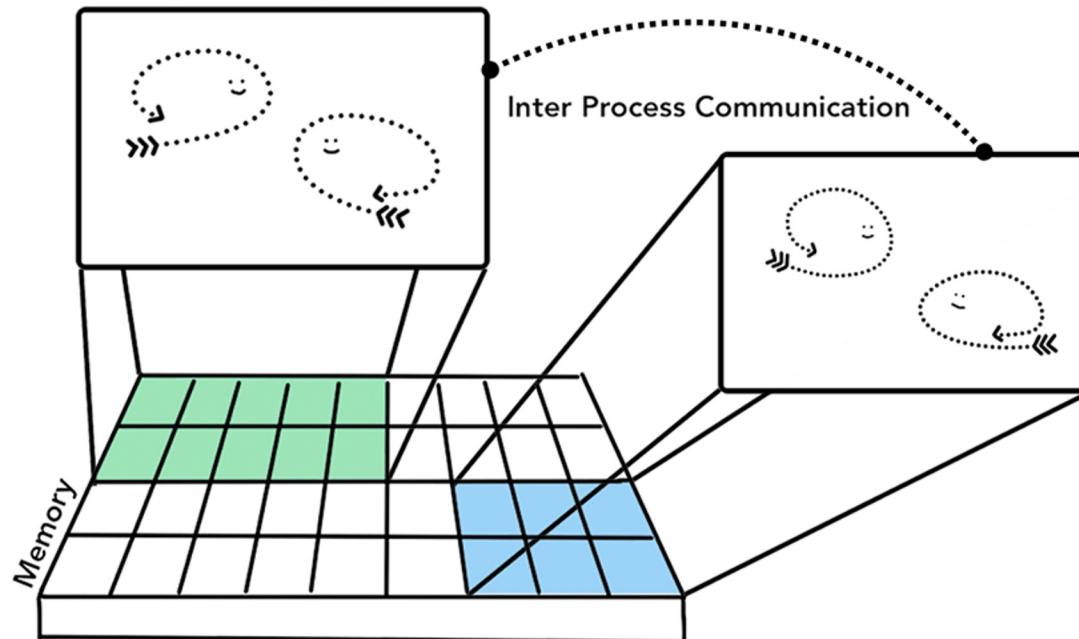
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Multi-threaded browser in single process



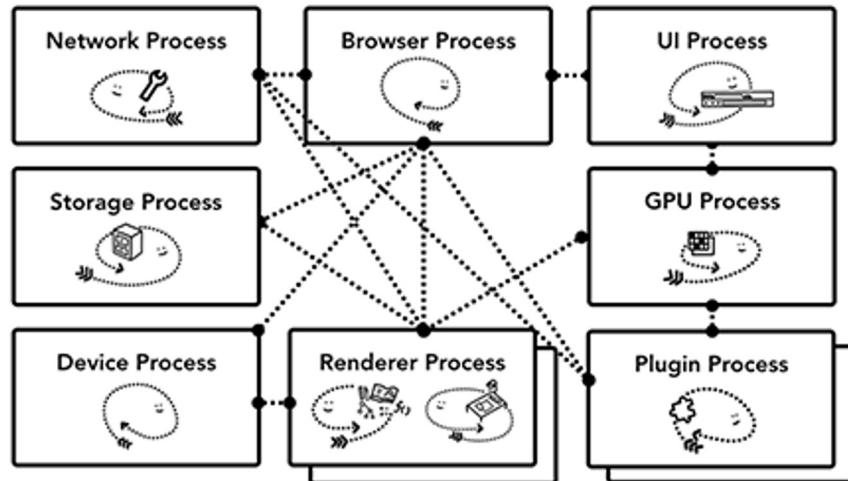
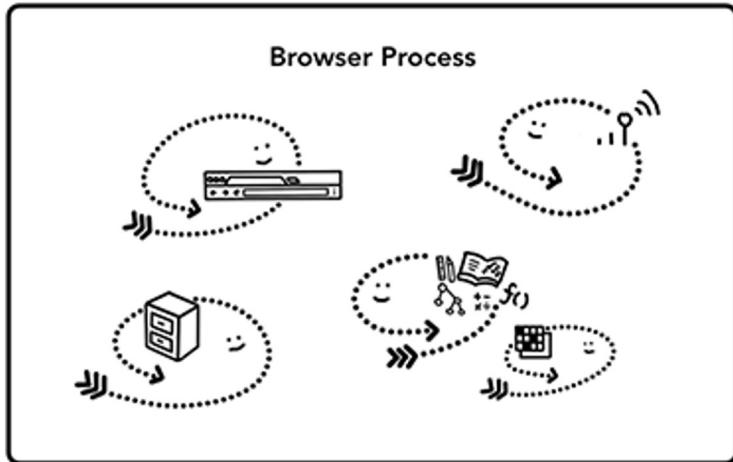
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Multi-process browser with IPC



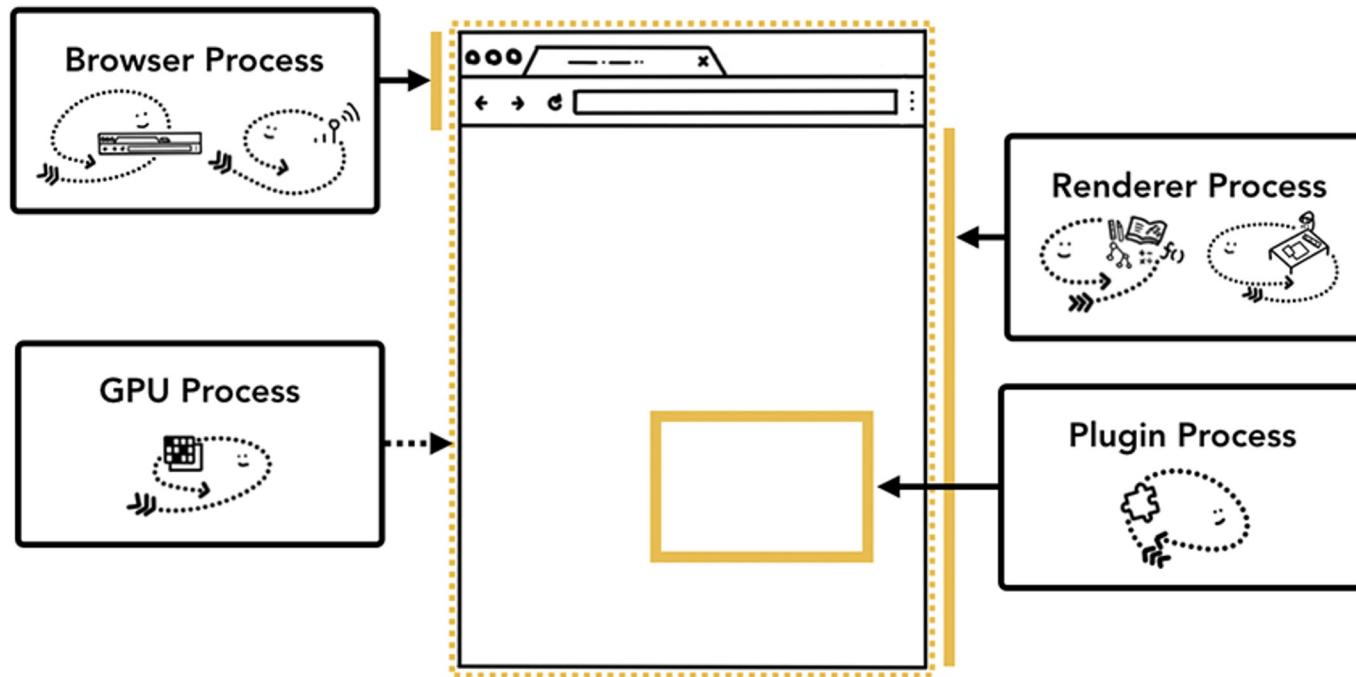
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



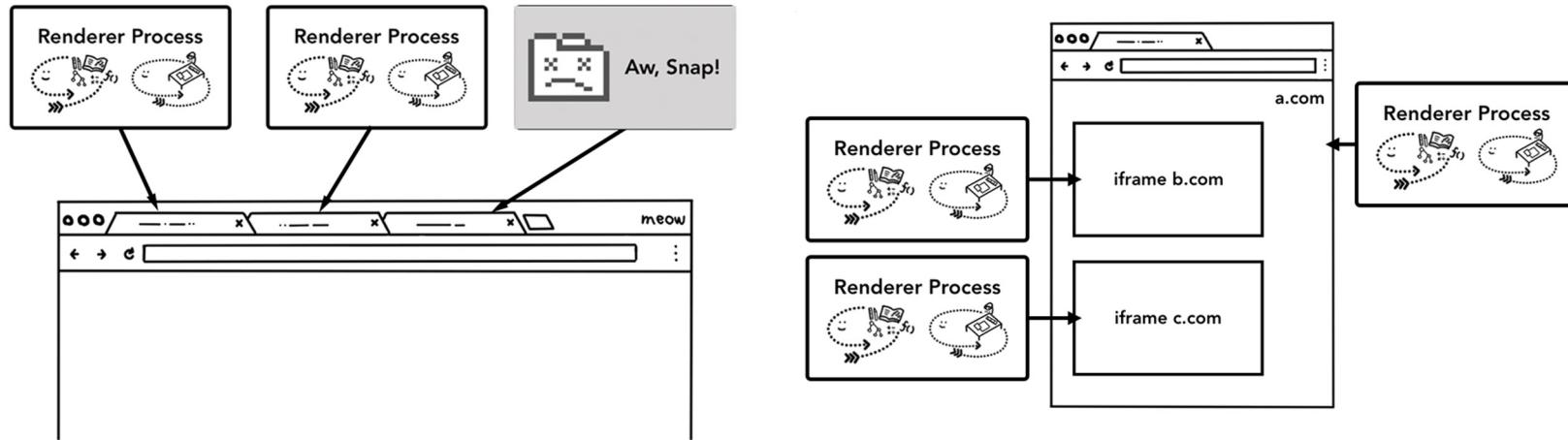
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



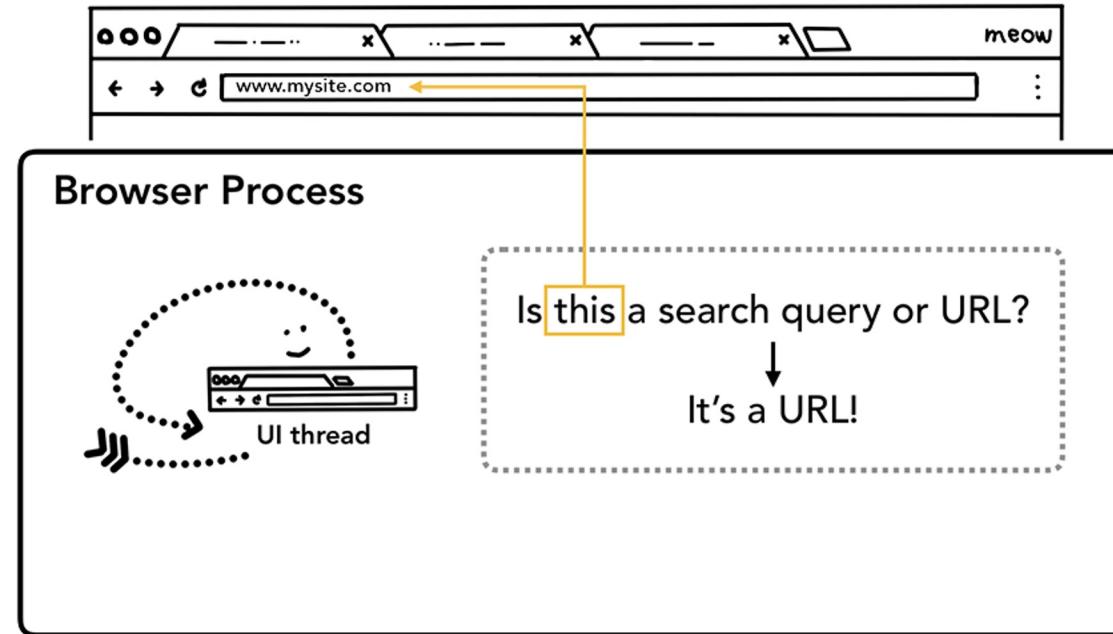
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



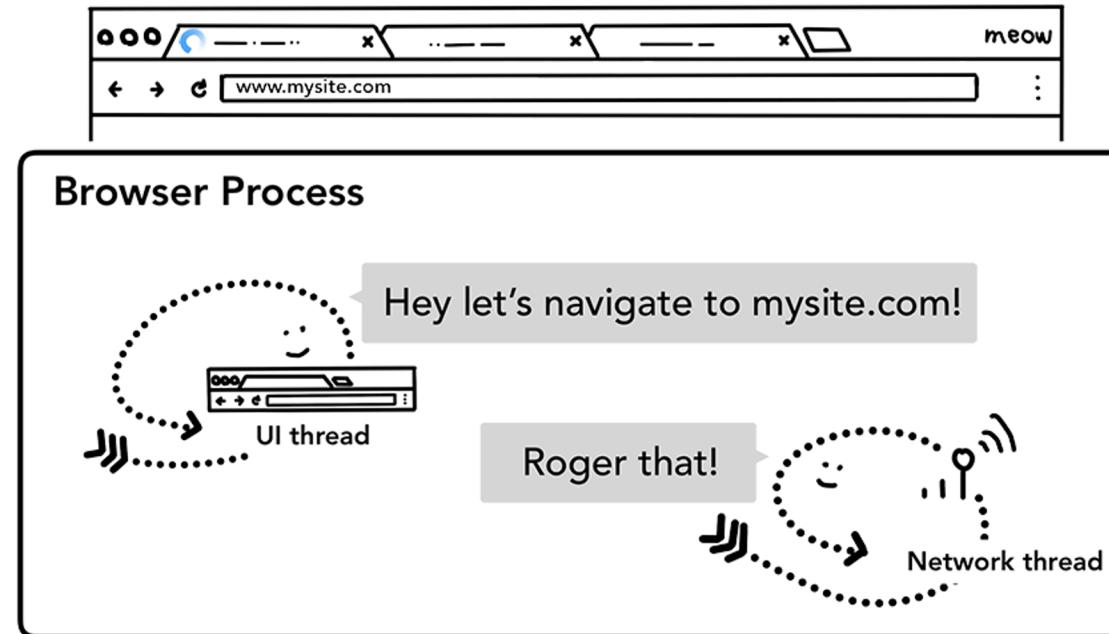
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



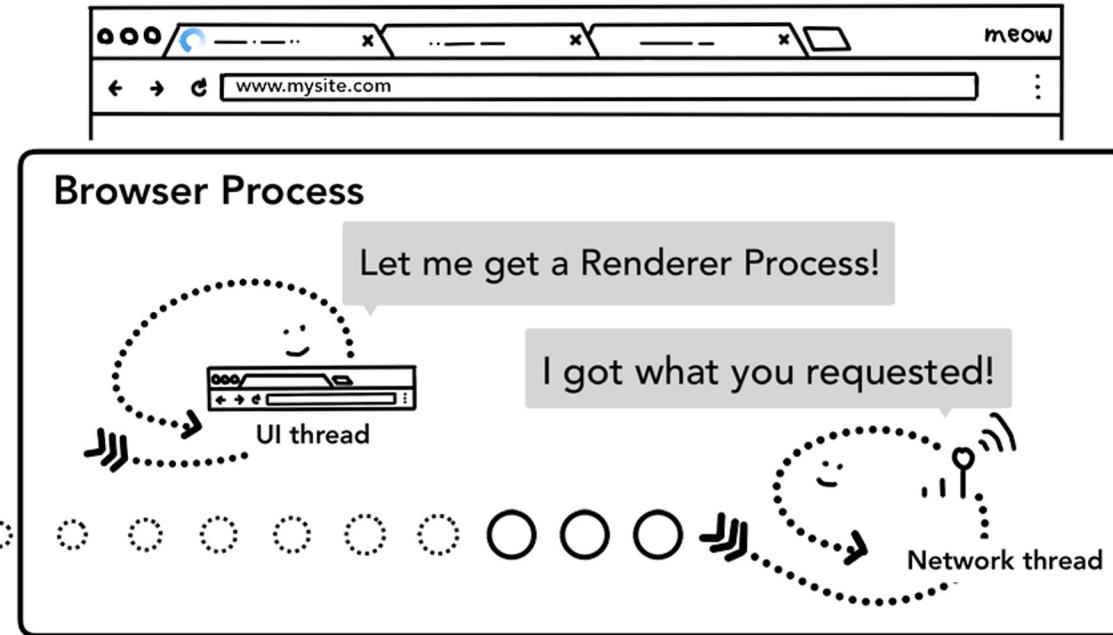
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



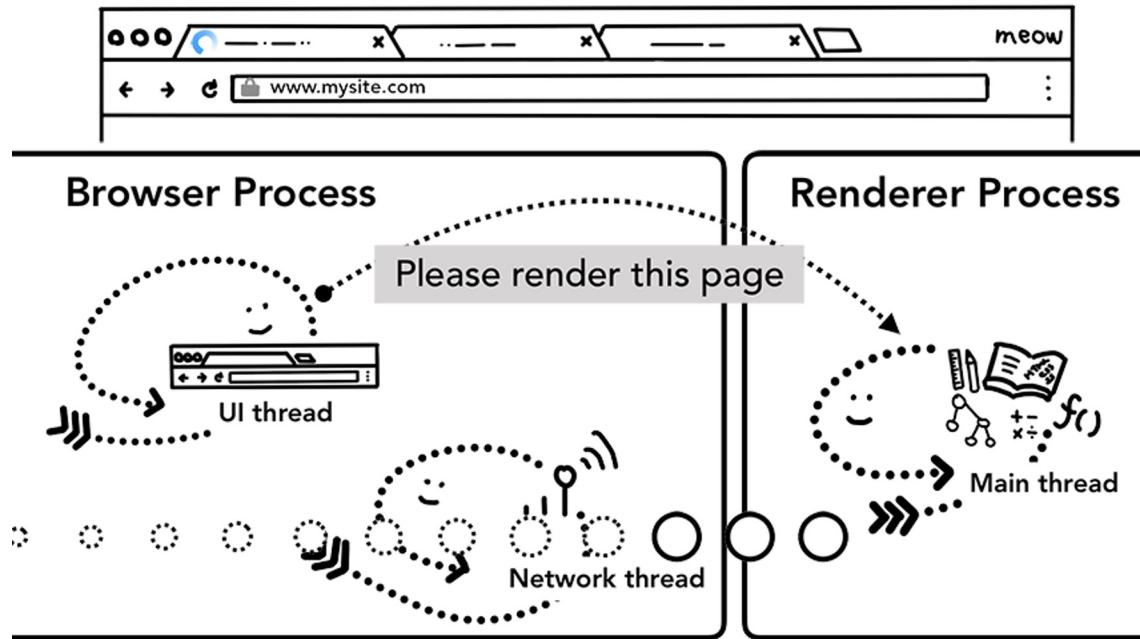
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



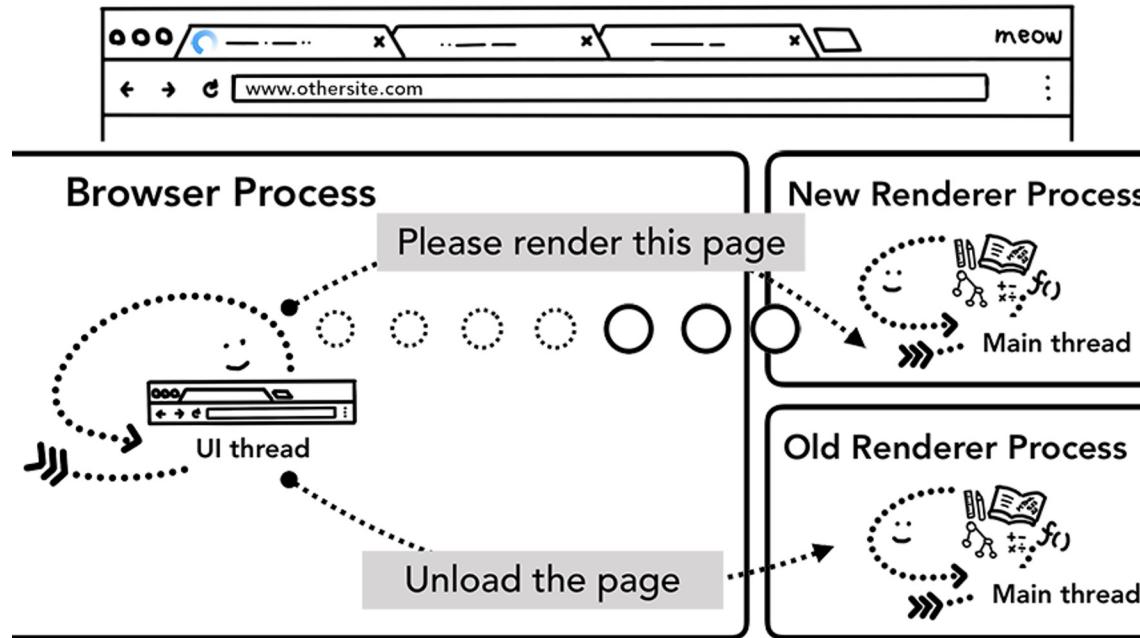
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests

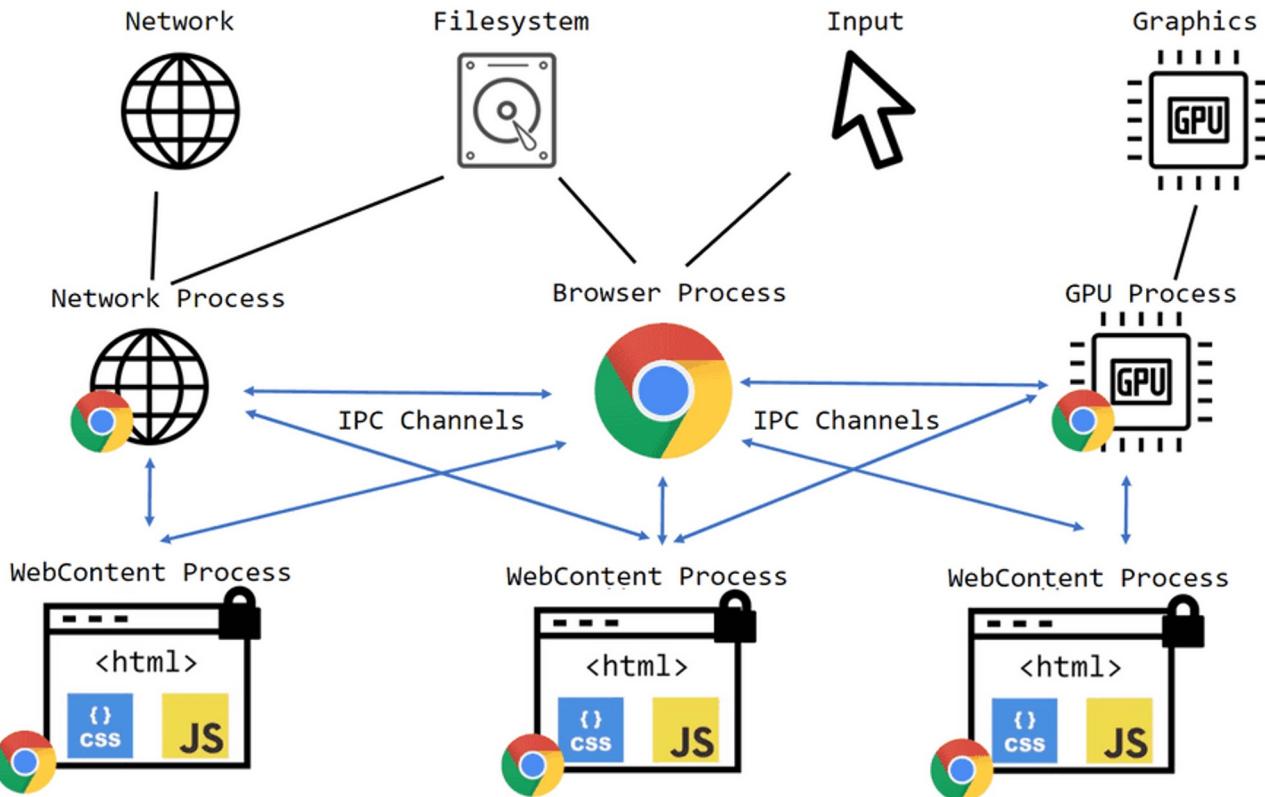


Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



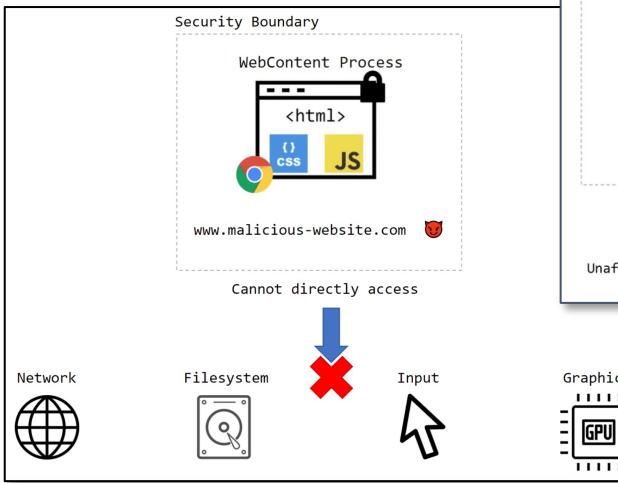
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)



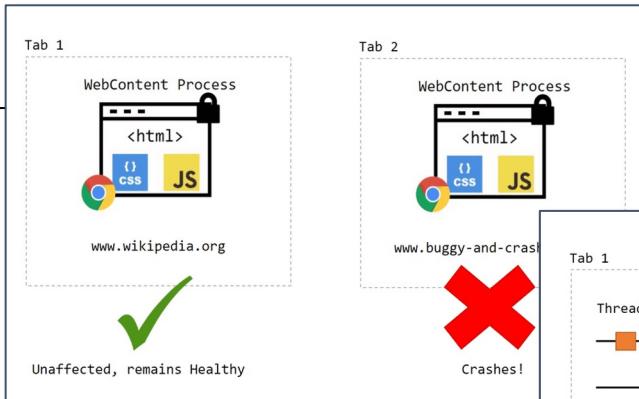
<https://webperf.tips/tip/browser-process-model>

Multi-Process: Benefits

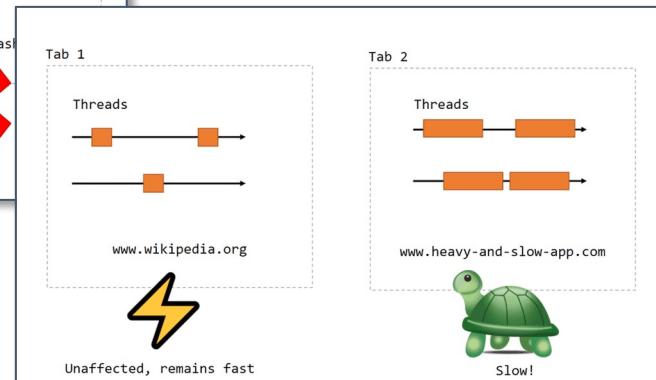
Security



Reliability



Performance



<https://webperf.tips/tip/browser-process-model/>

Multi-Process: Costs and Trade-offs

- **Memory Overhead**
 - spinning up new processes requires additional memory allocation
- **Process Creation Overhead**
 - more expensive to create a new process rather than simply a new thread in an existing process
- **IPC Overhead**
 - communicating across processes is slower than keeping communication completely localized within a single process

Service-Based Architectures: Pros and Cons

-  **Pros**

- Ability to change components independently
- Independent processes (Isolation, Security)
- Focusing on doing one thing well

-  **Cons**

- Increased complexity
- Increased cost and overheads
- Difficult to ensure data consistency across different services

MICROSERVICES



Microservices



UBER

GROUPON®

NBC
COMCAST

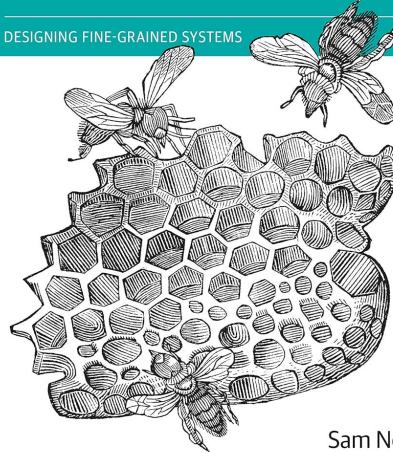
*“Small **autonomous** services
that work well together”*

Sam Newman

O'REILLY®

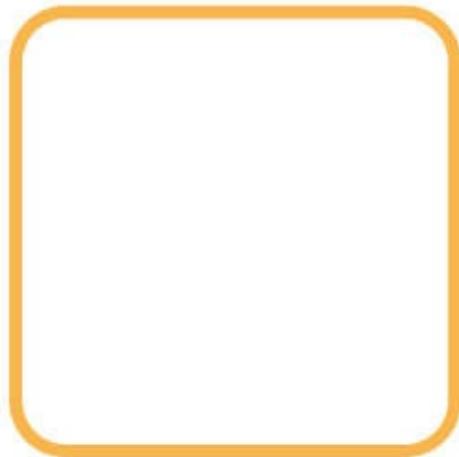
Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



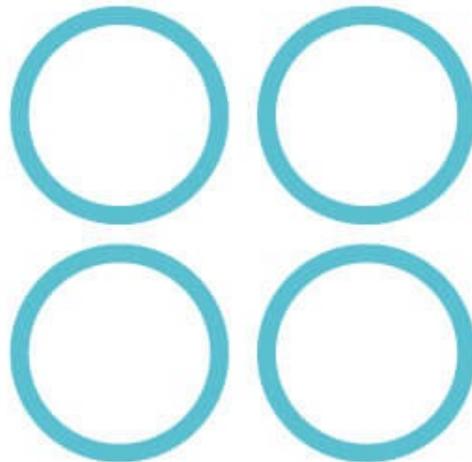
Sam Newman

Monolith vs. Service-Based vs. Microservice



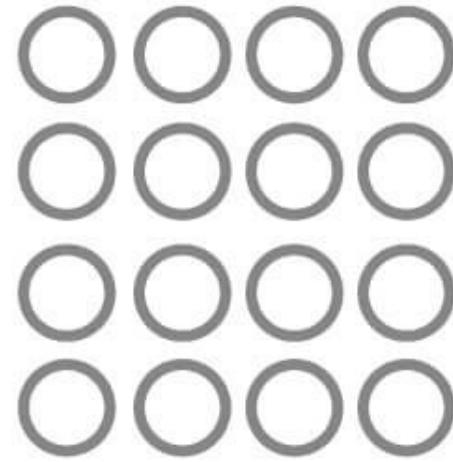
MONOLITHIC

Single Unit



SOA

Coarse- grained



MICROSERVICES

Fine-grained

Netflix Microservices



The image is a screenshot of the Netflix homepage. The main focus is the movie "the MENU" featuring Ralph Fiennes. Below the movie image are buttons for "Play" and "More info". To the right of the movie image are volume and subtitle settings. The page then displays the "Top 10 Movies in the US Today" with numbered thumbnails for "the MENU", "AMERICAN RENEGADES", "BACK ACTION", "FALLEN", "You, Me and Dupree", and "LUCAS'S WORLD". Below this is a "Your Next Watch" section with thumbnails for "THE LAST KICK", "UGLY BETTY", "KINDA PREGNANT", "MANIFEST", "WEDNESDAY", and "STRANGER THINGS". At the bottom, there is a "New on Netflix" section with thumbnails for "KINDA PREGNANT", "ANUJA", "Bogotá: CITY OF THE LOST", "the MENU", "PUSHKIN STORIES", and "LUCAS'S WORLD". A footer message "Because you watched Emily in Paris" is at the bottom, followed by a "Systems Department" link.

Why Can't Netflix Use a Monolithic Architecture?

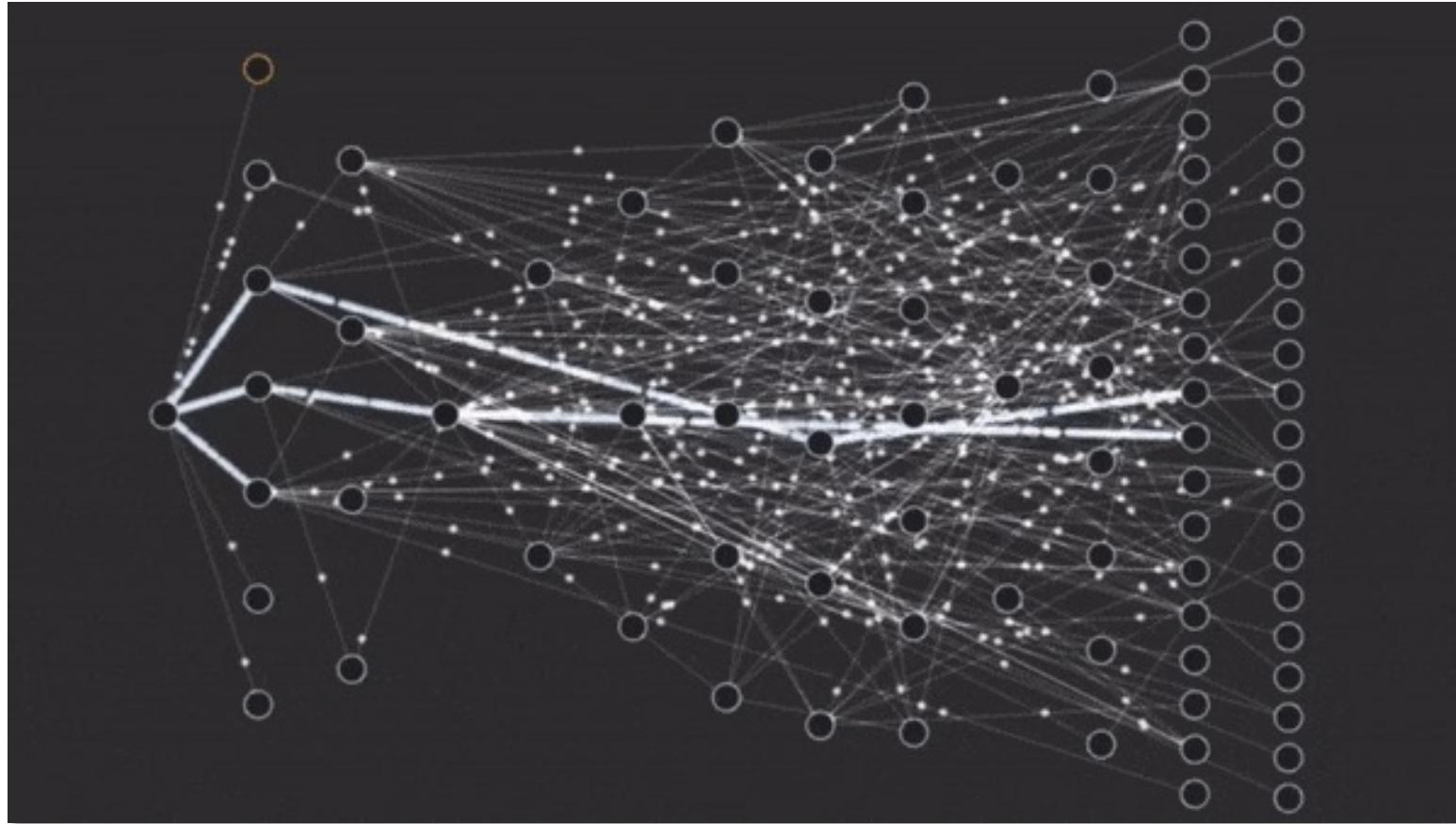


- Requires an architecture that can handle **various computational demands**
- Need scalability must support **millions of users worldwide**
- Need fault tolerance to maintain a **seamless user experience**
- New features and improvements need to be **rolled out rapidly**

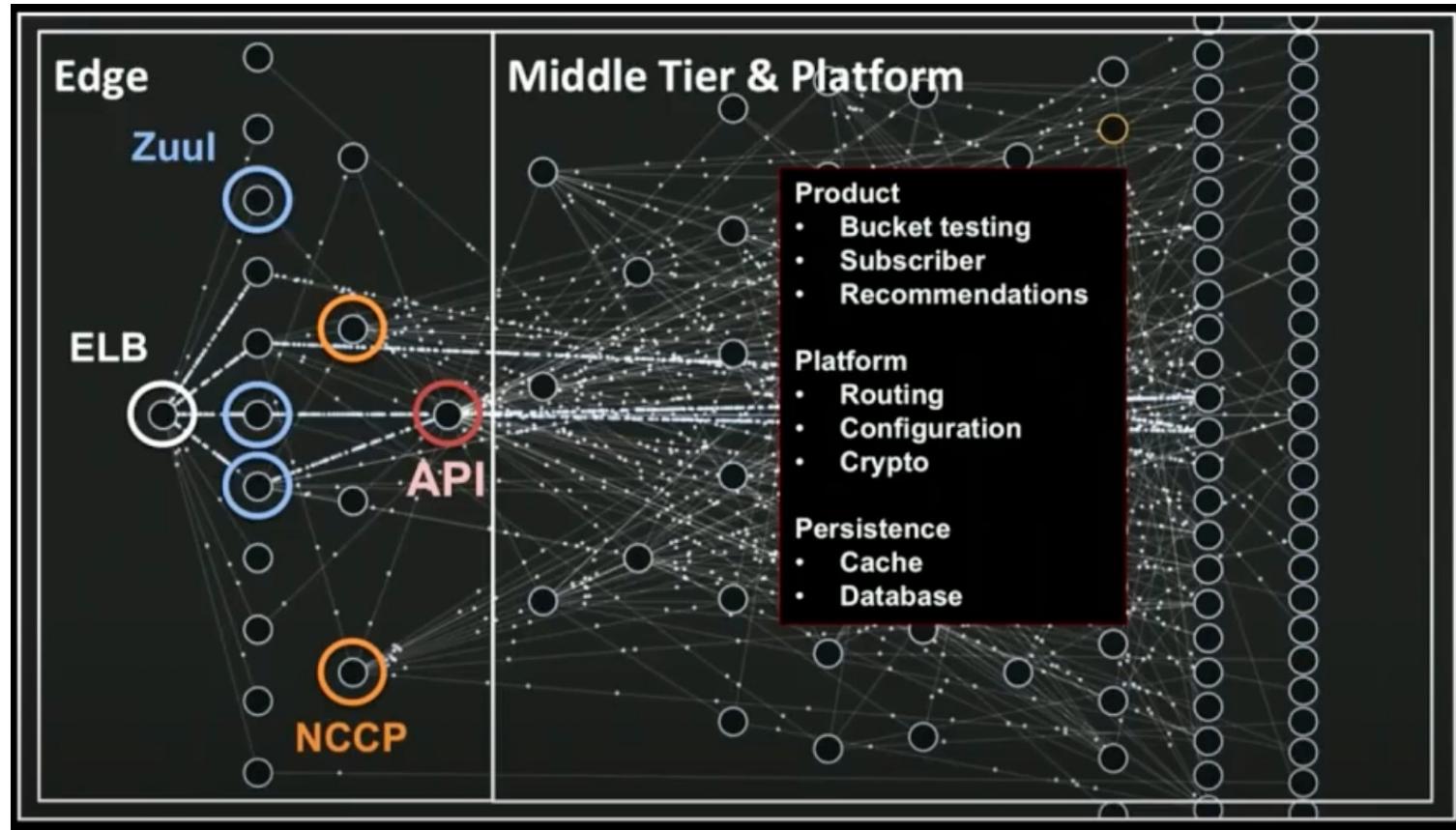
Netflix Microservices



- User Subscriptions
- Banner Ad
- Popular Shows
- Trending Now
- Continue Watching
- My List (saved shows)
- Notifications
- User Management
- Subtitles
- ...

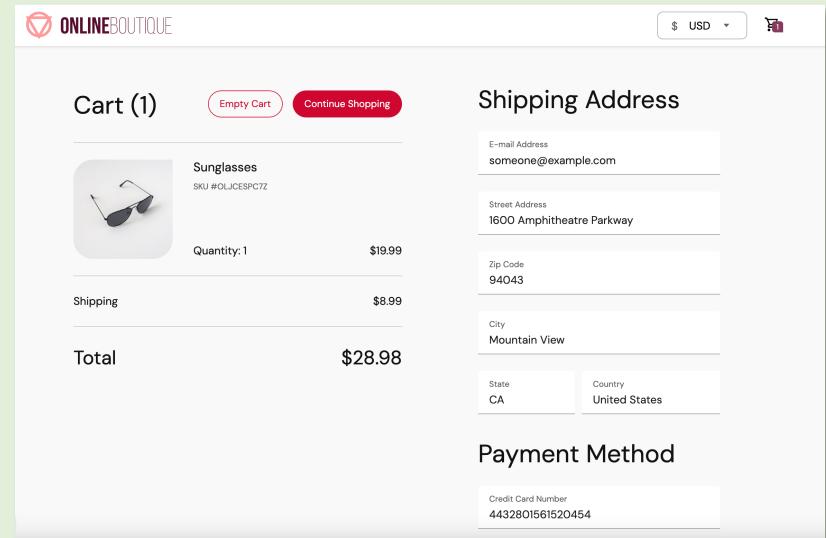
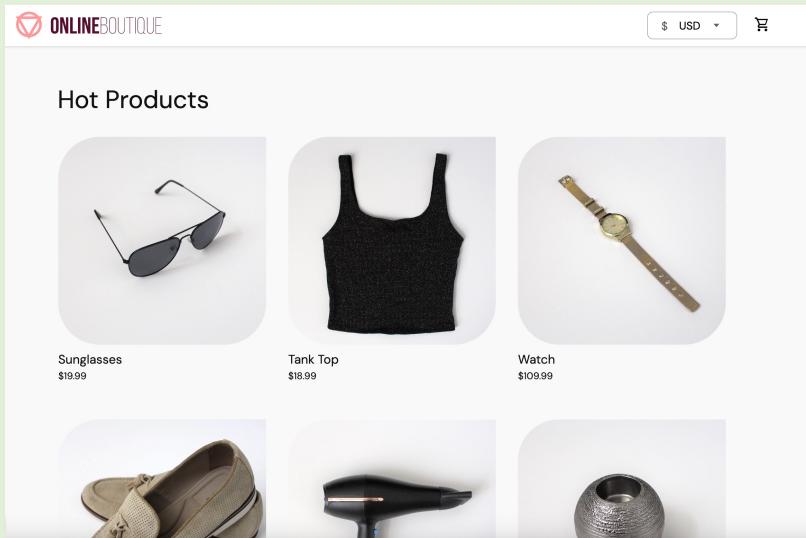


https://www.youtube.com/watch?v=V_oxbj-a1wQ



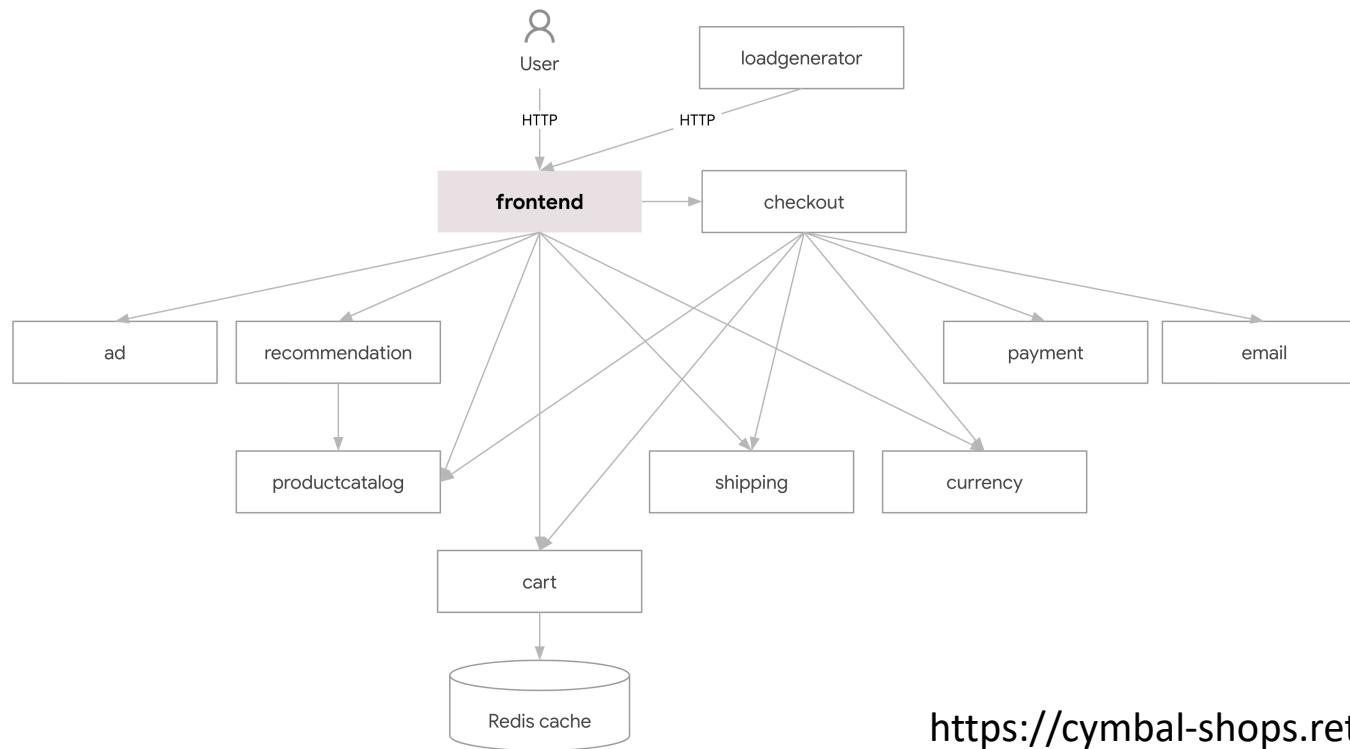
https://www.youtube.com/watch?v=V_oxbj-a1wQ

Online Boutique: Guess some microservices



<https://cymbal-shops.retail.cymbal.dev>

Online Boutique: Microservice Architecture



<https://cymbal-shops.retail.cymbal.dev>

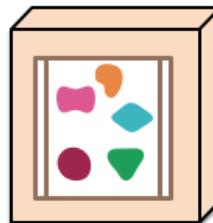
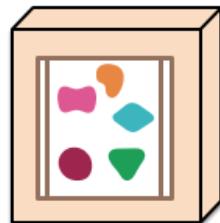
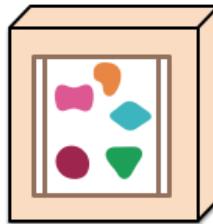
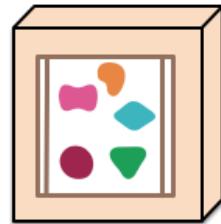
| Service | Language | Description |
|---------------------------------------|---------------|---|
| frontend | Go | Exposes an HTTP server to serve the website. Does not require signup/login and generates session IDs for all users automatically. |
| cartservice | C# | Stores the items in the user's shopping cart in Redis and retrieves it. |
| productcatalogservice | Go | Provides the list of products from a JSON file and ability to search products and get individual products. |
| currencyservice | Node.js | Converts one money amount to another currency. Uses real values fetched from European Central Bank. It's the highest QPS service. |
| paymentservice | Node.js | Charges the given credit card info (mock) with the given amount and returns a transaction ID. |
| shippingservice | Go | Gives shipping cost estimates based on the shopping cart. Ships items to the given address (mock) |
| emailservice | Python | Sends users an order confirmation email (mock). |
| checkoutservice | Go | Retrieves user cart, prepares order and orchestrates the payment, shipping and the email notification. |
| recommendationservice | Python | Recommends other products based on what's given in the cart. |
| adservice | Java | Provides text ads based on given context words. |
| loadgenerator | Python/Locust | Continuously sends requests imitating realistic user shopping flows to the frontend. |

Scalability

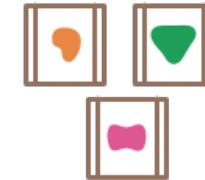
A monolithic application puts all its functionality into a single process...



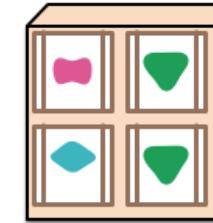
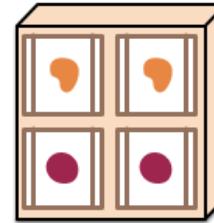
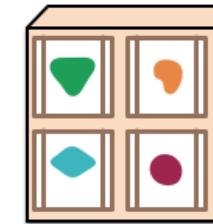
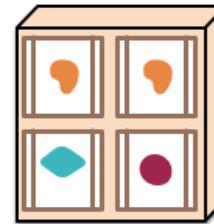
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



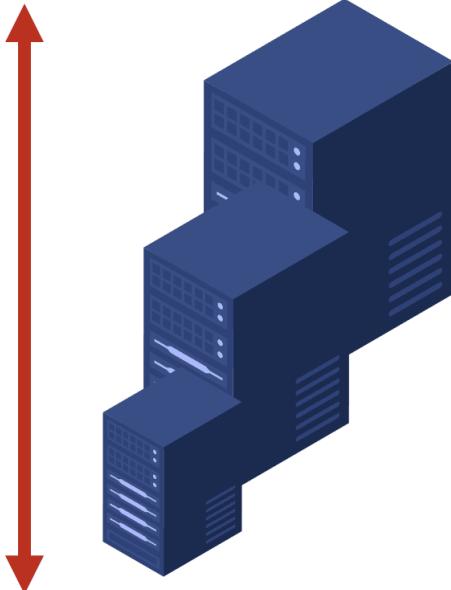
... and scales by distributing these services across servers, replicating as needed.



Types of Scaling: Vertical vs. Horizontal

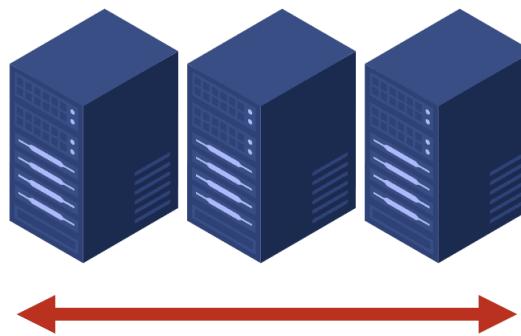
Vertical Scaling

Increase or decrease the capacity of existing services/instances.

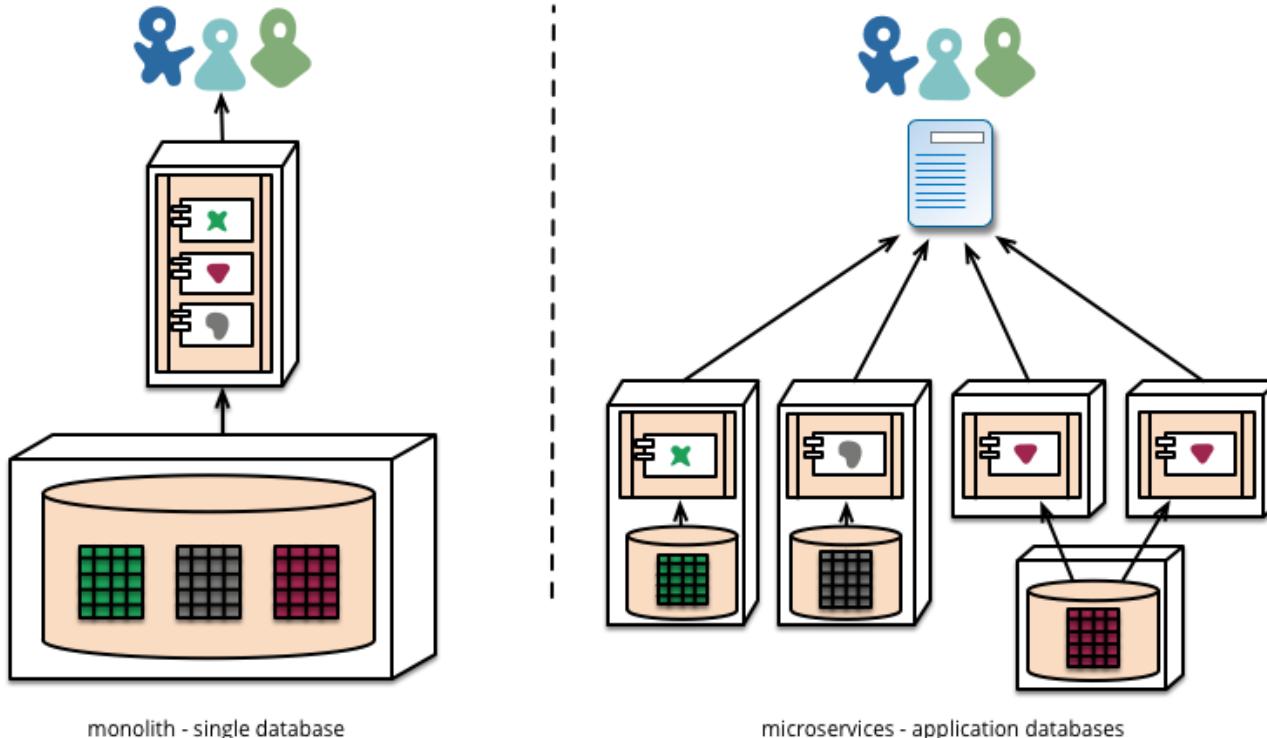


Horizontal Scaling

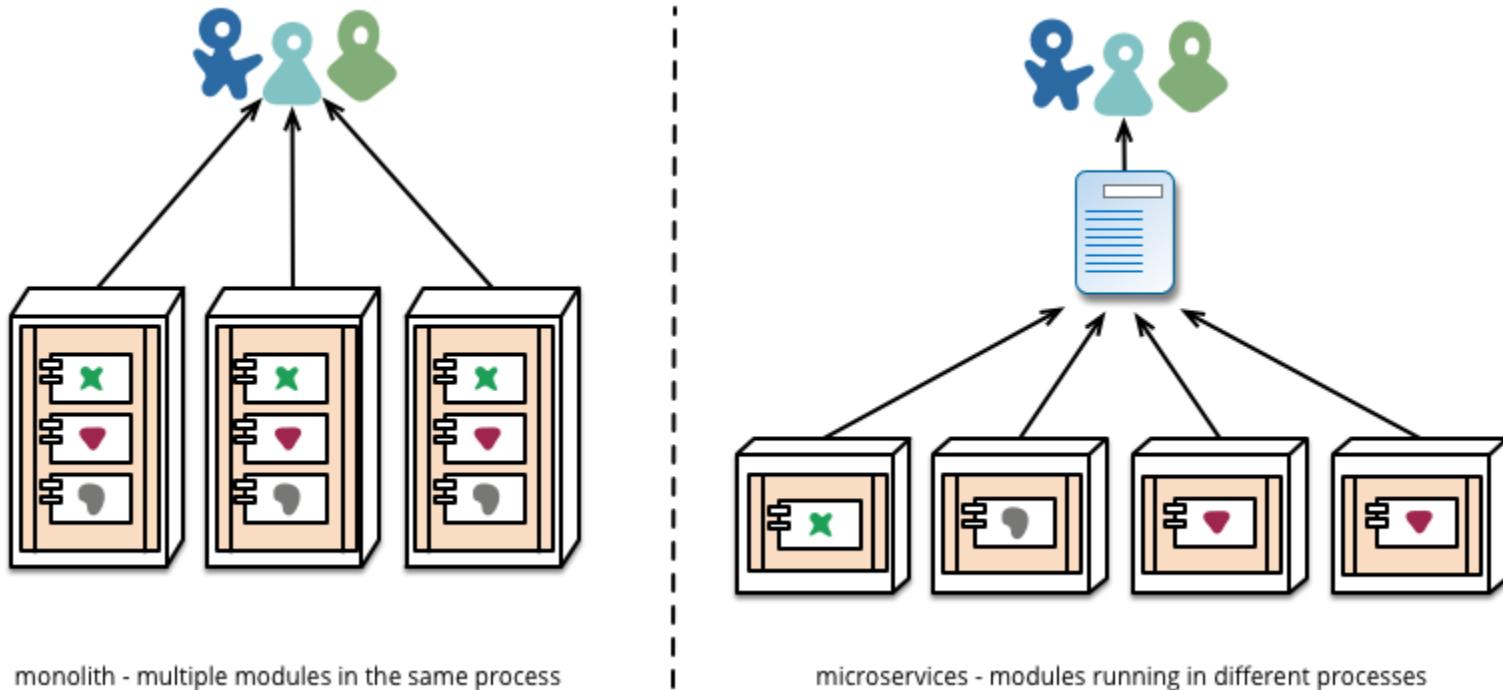
Add more resources like virtual machines to your system to spread out the workload across them.



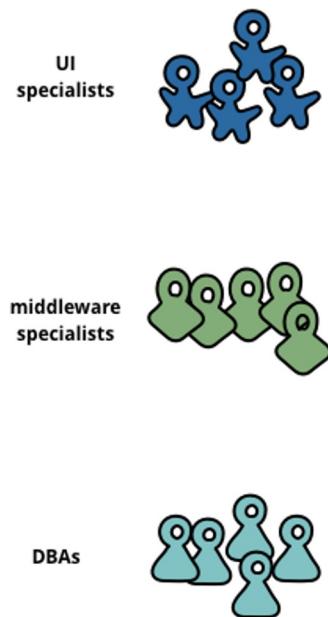
Data Management and Consistency



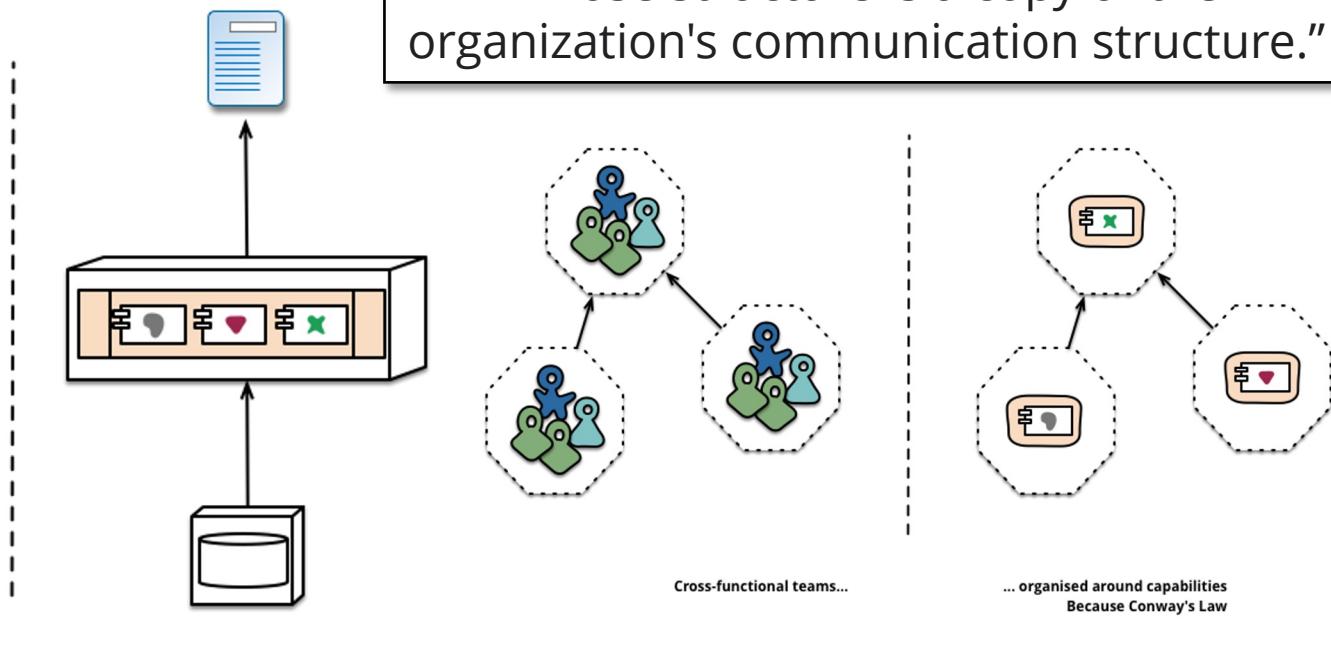
Deployment and Evolution



Conway's Law



Siloed functional teams...



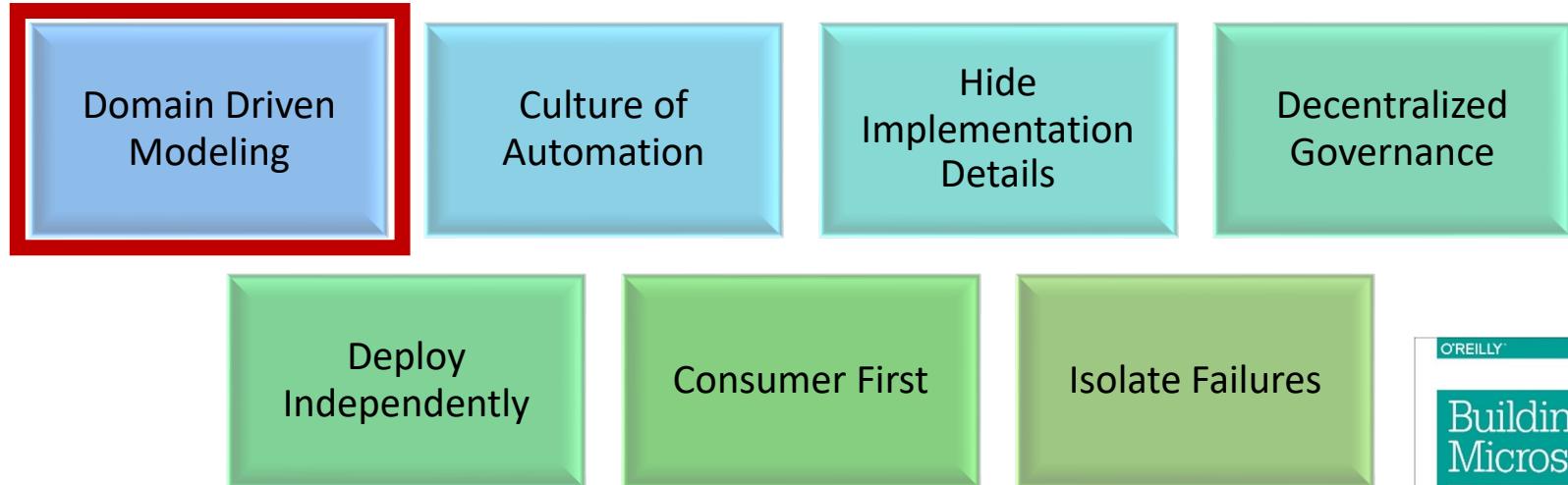
“Products” not “Projects”

YOU BUILD IT YOU RUN ~~AWAY~~ IT

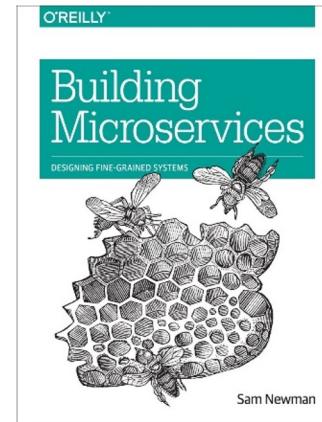
"The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service."

-- Werner Vogels in "A conversation with Werner Vogels" in ACM Queue, May 2006

MICROSERVICES: PRINCIPLES

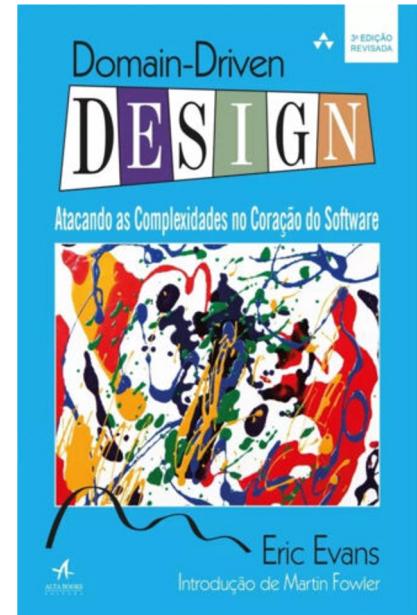


Sam Newman's Principles of Microservices

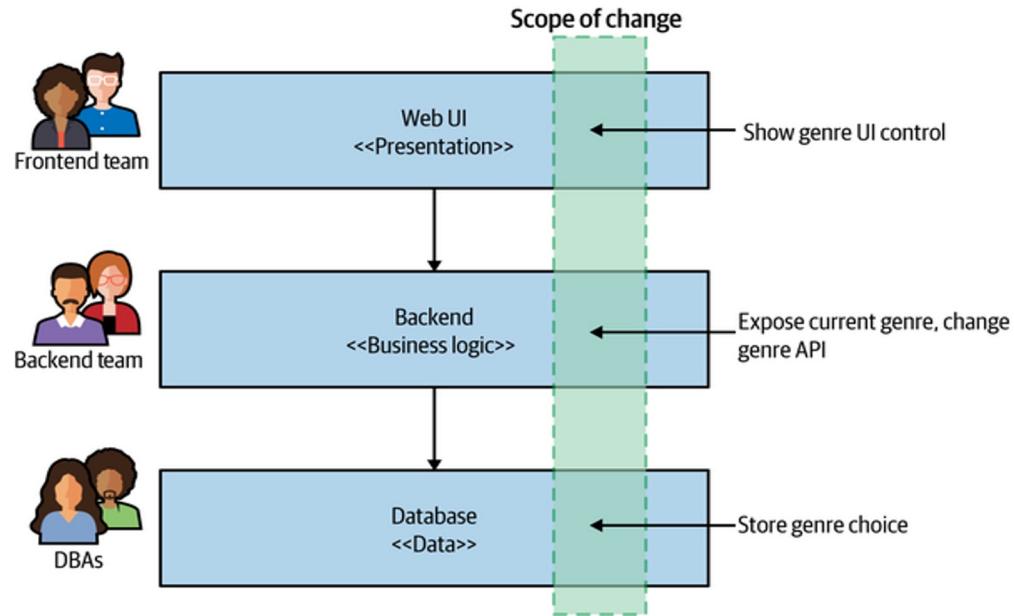
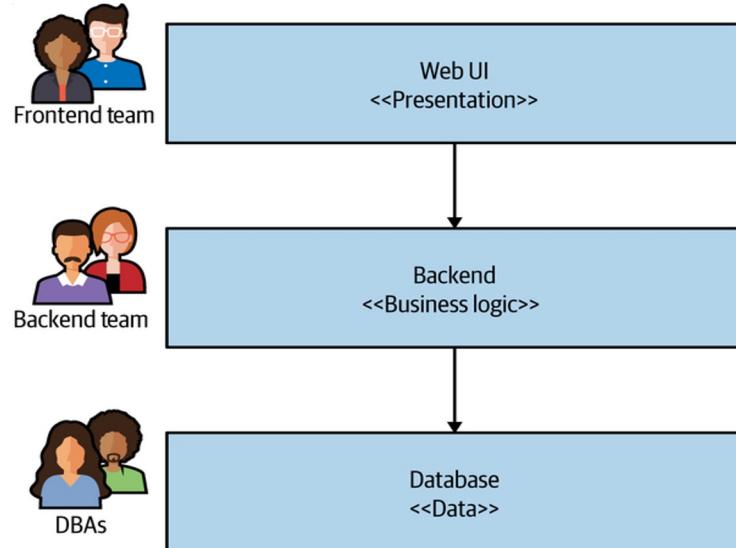


Domain-driven modeling

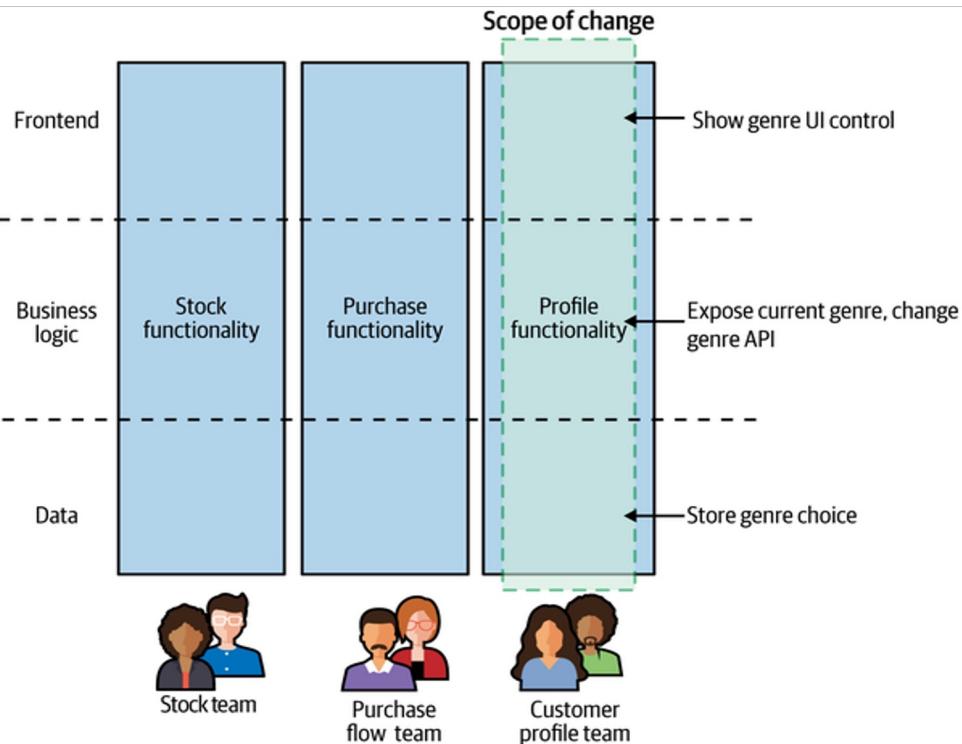
- Model services around business capabilities



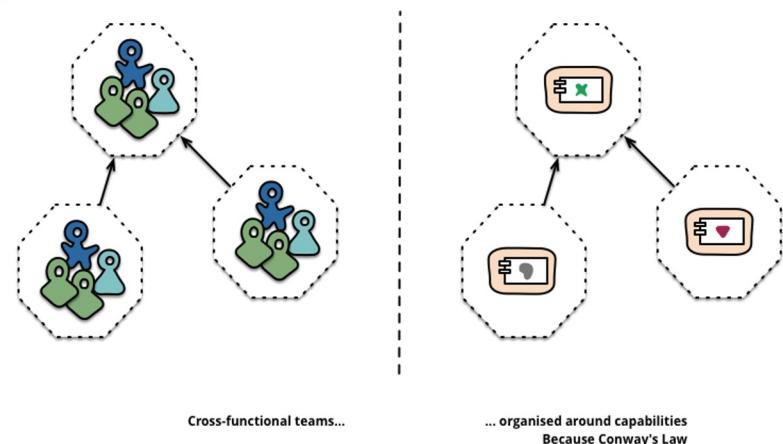
Domain-driven modeling

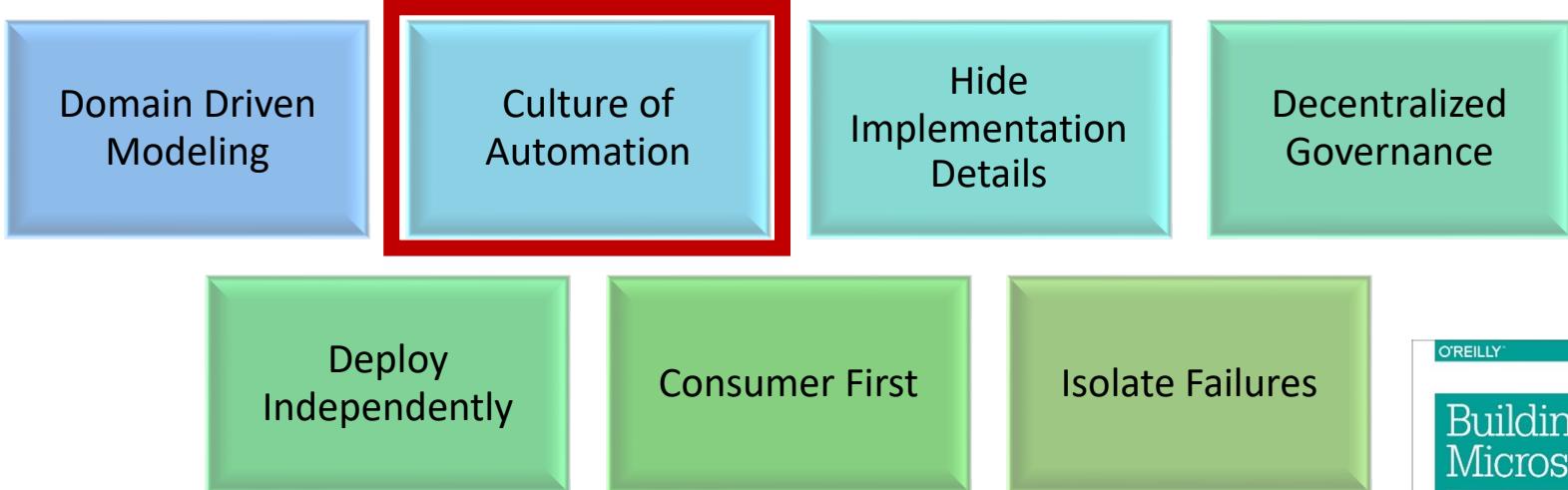


Domain-driven modeling

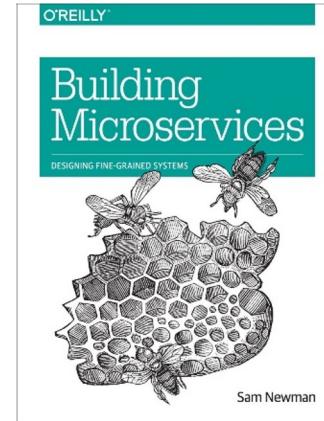


Remember Conway's Law?





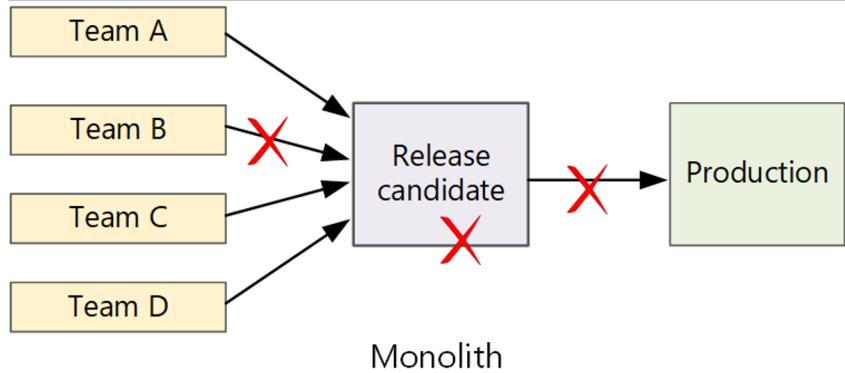
Sam Newman's Principles of Microservices



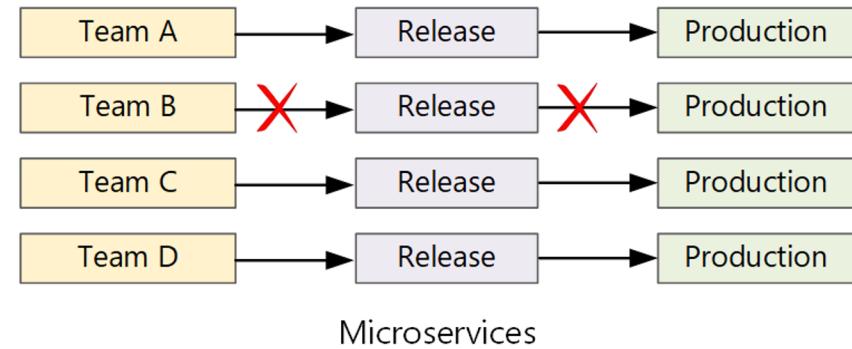
Culture of Automation

- API-Driven Machine Provisioning
- Continuous Delivery
- Automated Testing

Continuous Delivery

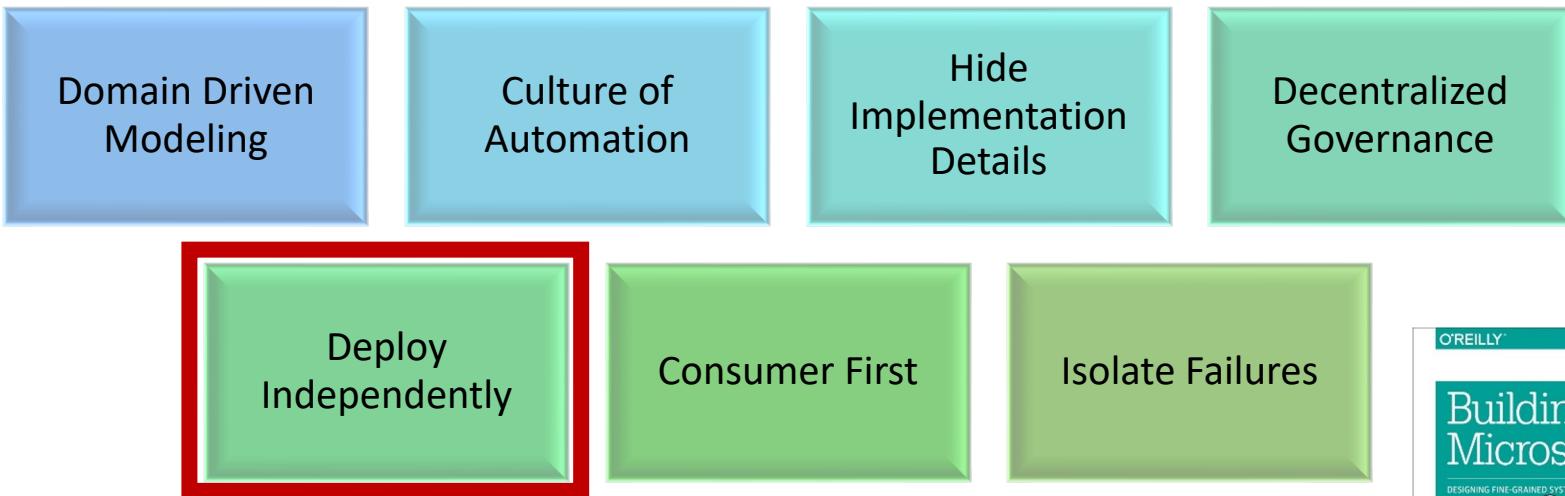


Monolith

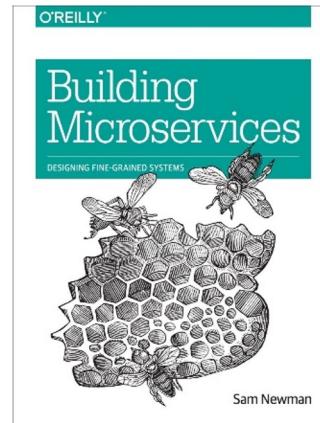


Microservices

Image Source: <https://learn.microsoft.com/en-us/azure/architecture/microservices/ci-cd>



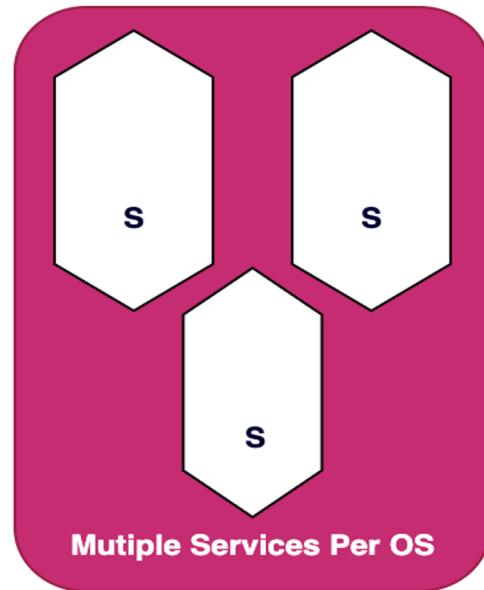
Sam Newman's Principles of Microservices



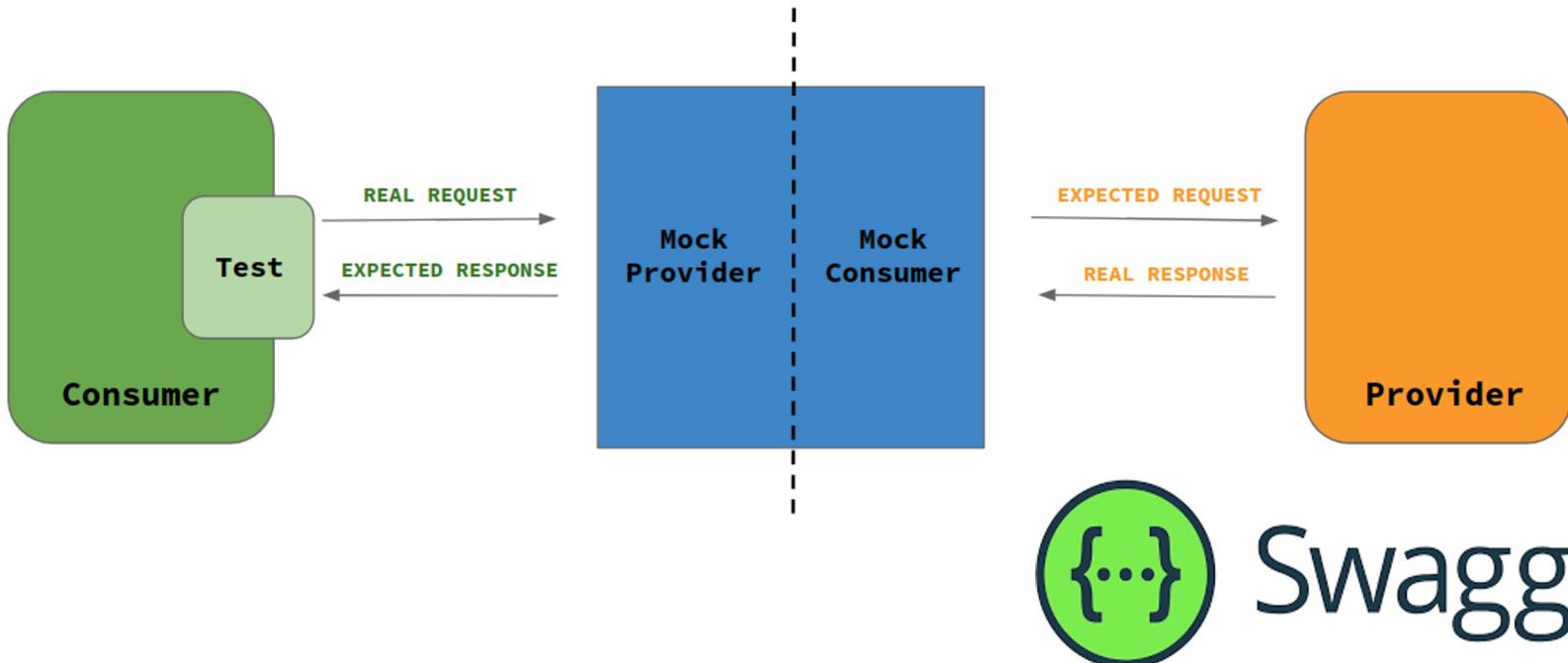
Deploy Independently

- One Service Per OS
- Consumer-Driven Contracts
- Multiple coexisting versions

One Service Per OS

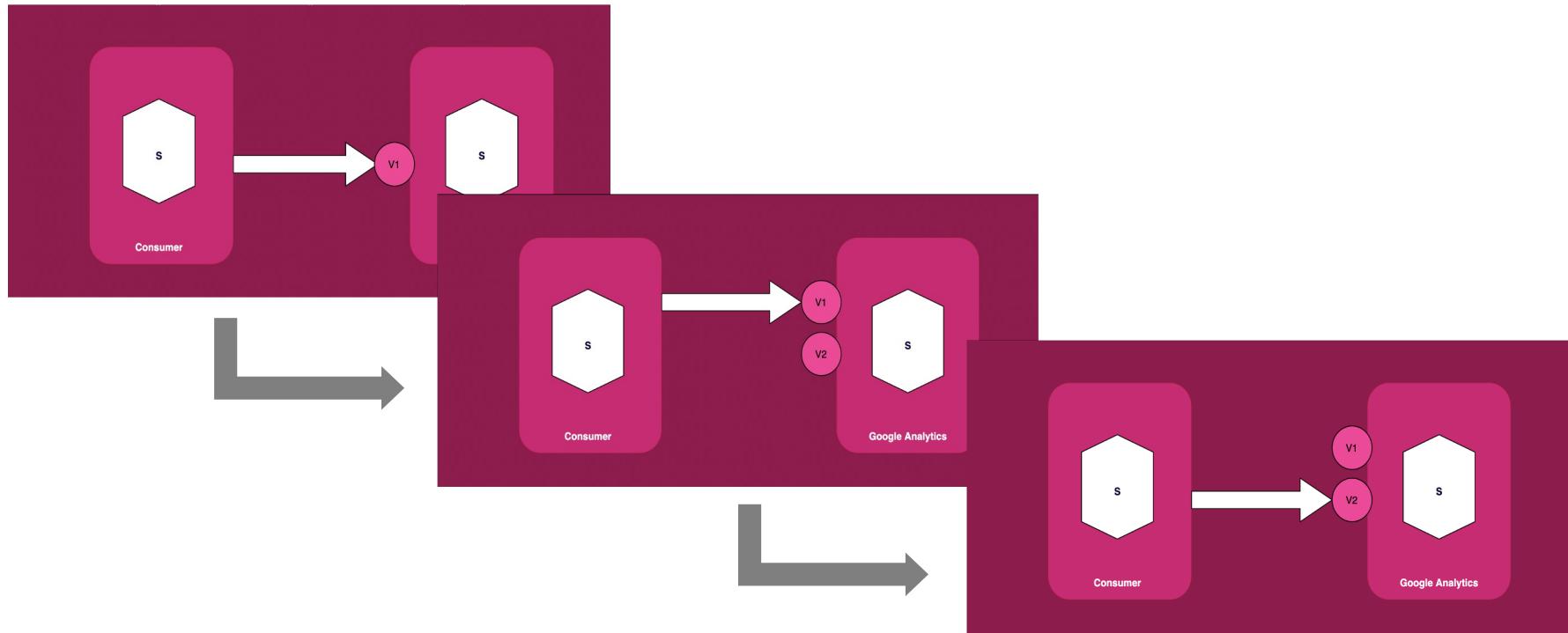


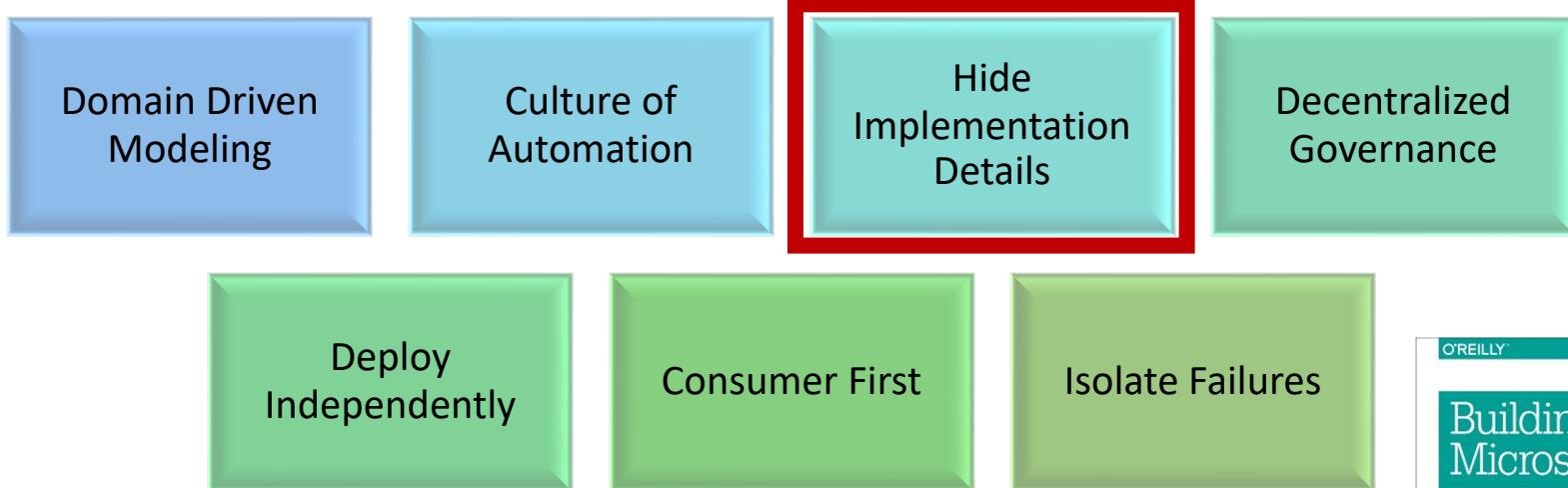
Consumer-Driven Contracts



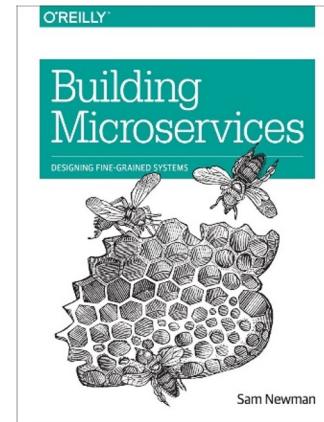
<https://medium.com/@japneetkaur11/contract-testing-with-pact-17909b838de9>

Multiple coexisting versions





Sam Newman's Principles of Microservices

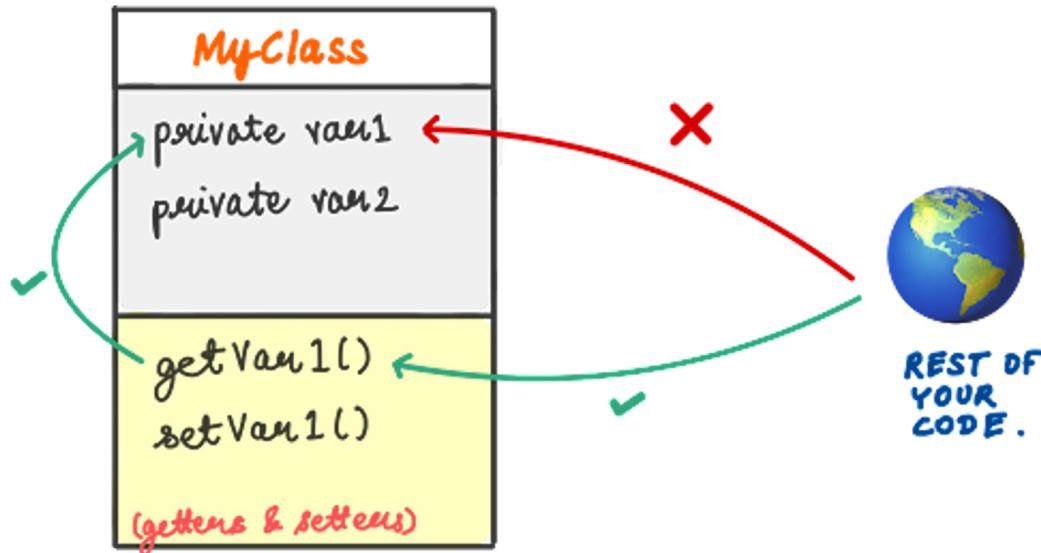


Hide implementation details

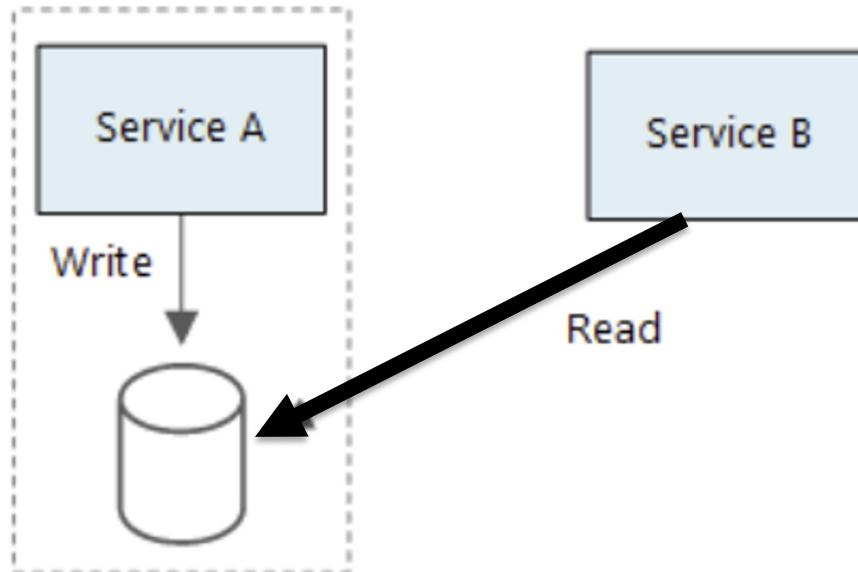
- Design your APIs carefully
- It's easier to expose details later than hide them
- Do not share your database!

Hide implementation details

Recall: Encapsulation in OOP



Sharing database: Anti-pattern



Domain Driven
Modeling

Culture of
Automation

Hide
Implementation
Details

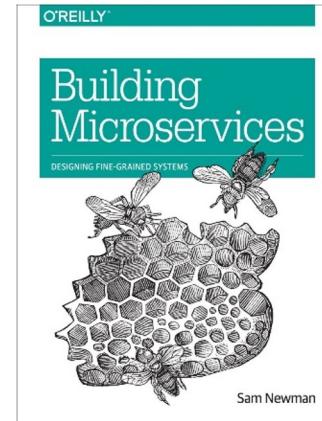
Decentralized
Governance

Deploy
Independently

Consumer First

Isolate Failures

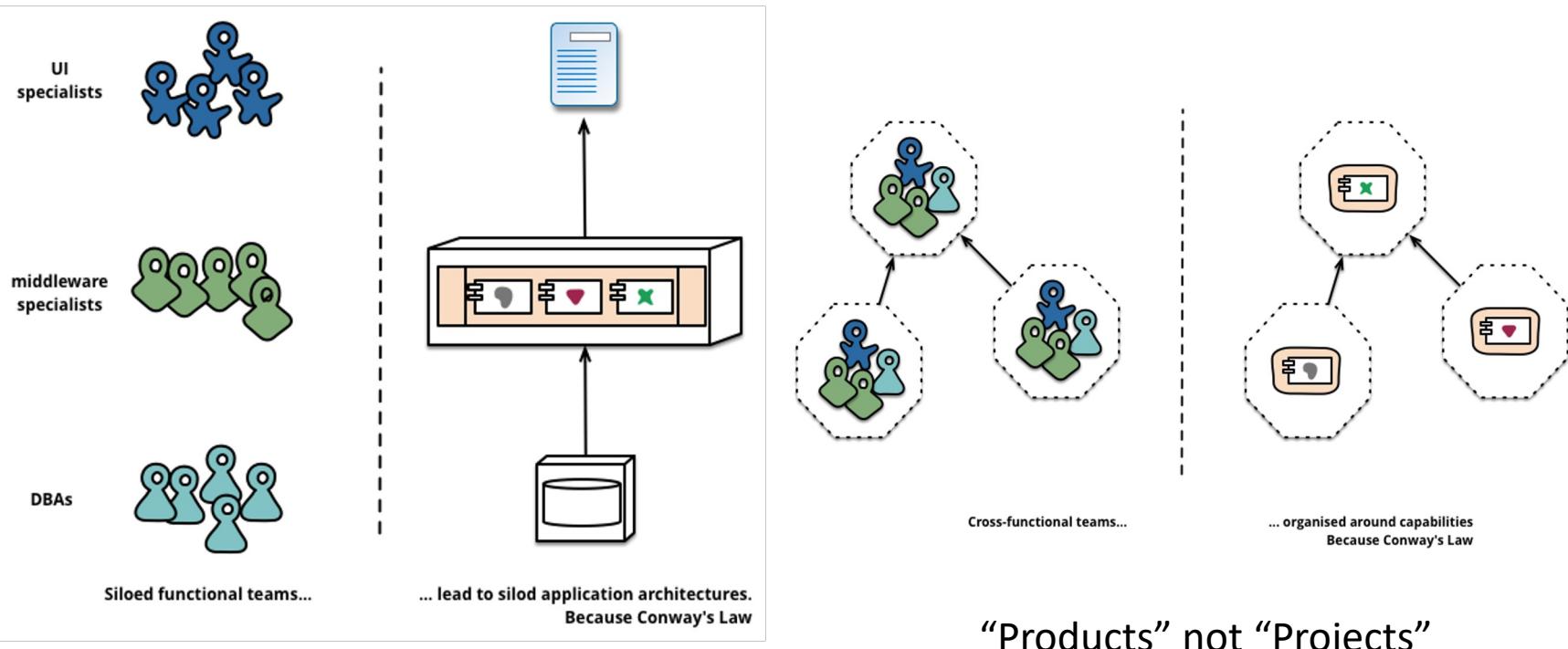
Sam Newman's Principles of Microservices



Decentralized Governance

- Mind Conway's Law
- You Build It, You Run It
- Embrace team autonomy
- Internal Open Source Model

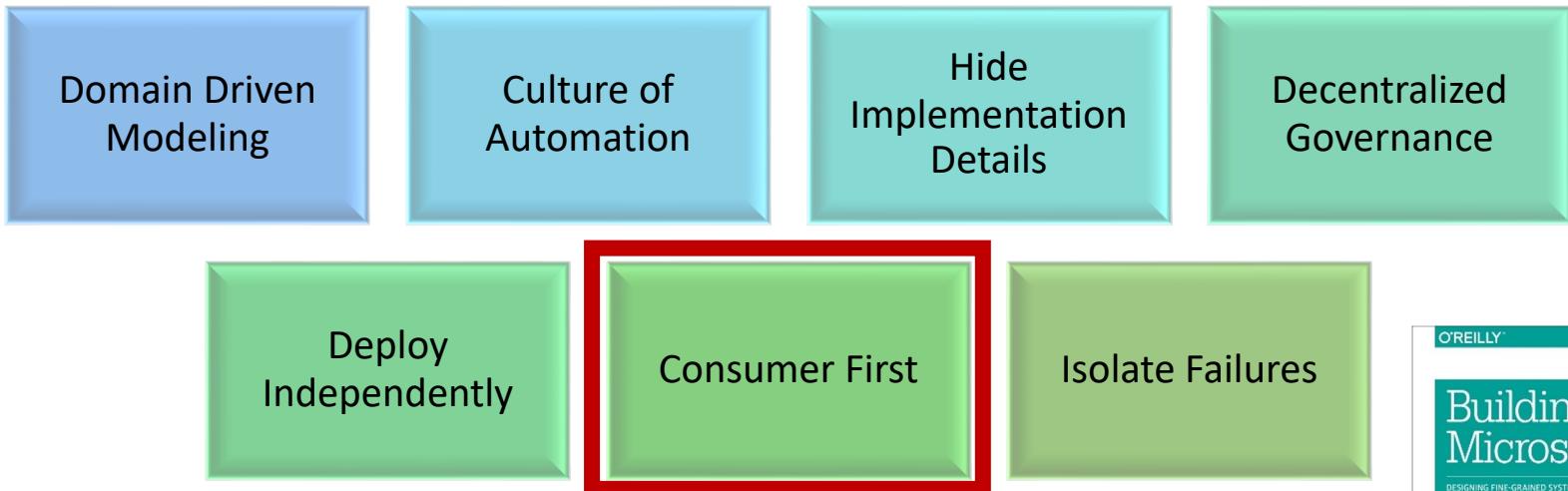
Mind Conway's Law



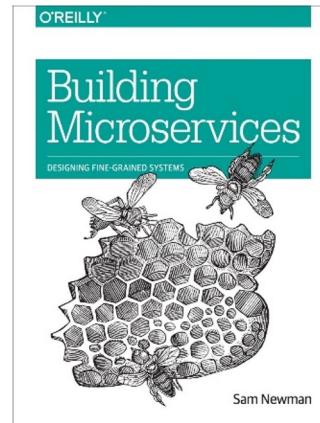
YOU BUILD IT YOU RUN ~~AWAY~~ IT

“The traditional model is that you take your software to the wall that separates development and operations, and throw it over and then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact with the day-to-day operation of their software. It also brings them into day-to-day contact with the customer. This customer feedback loop is essential for improving the quality of the service.”

-- Werner Vogels in “A conversation with Werner Vogels” in ACM Queue, May 2006



Sam Newman's Principles of Microservices



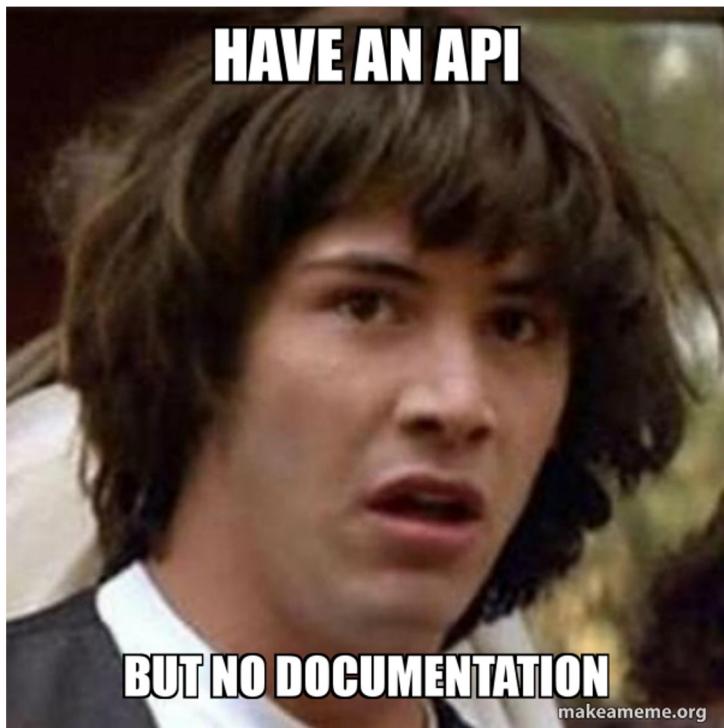
Consumer First

- Encourage conversations
- API Documentation
- Service Discovery

Encourage conversations



Document your APIs!



Swagger

TM

Domain Driven
Modeling

Culture of
Automation

Hide
Implementation
Details

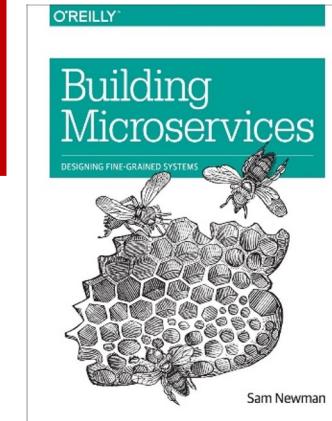
Decentralized
Governance

Deploy
Independently

Consumer First

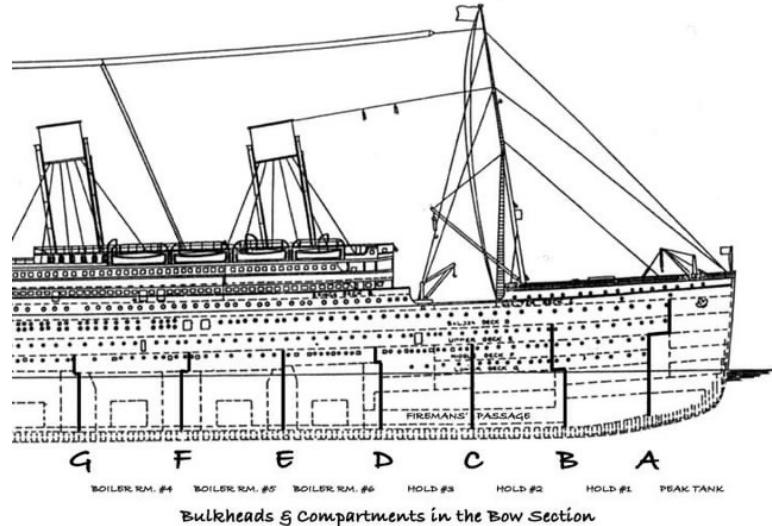
Isolate Failures

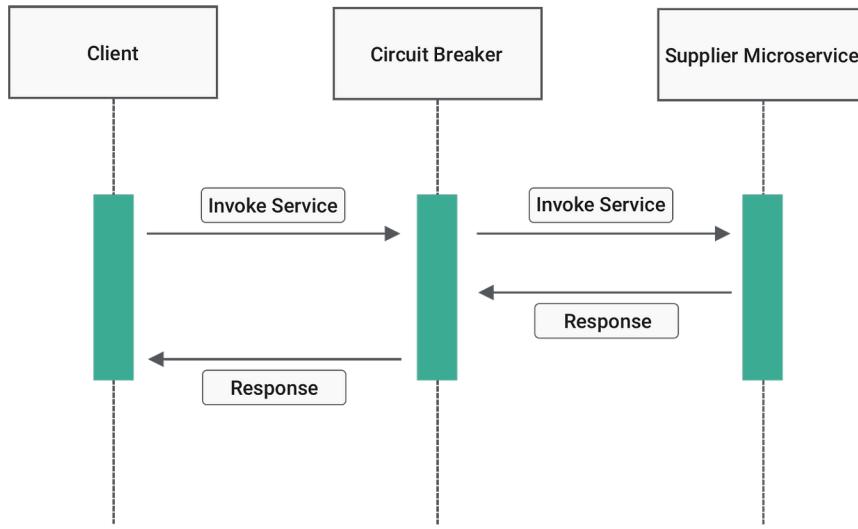
Sam Newman's Principles of Microservices



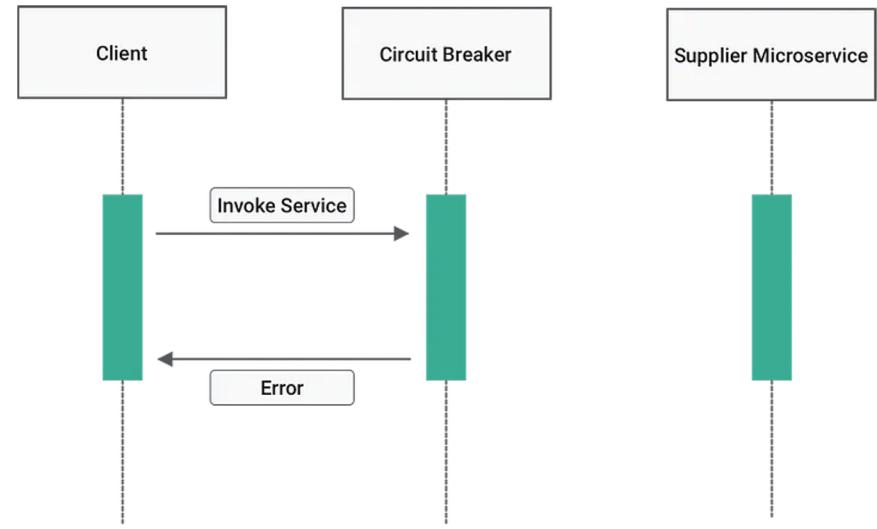
Isolate Failure

- Avoid **cascading failures**
- Timeouts between components
- **Fail fast** aka *Design for Failure*
 - Bulkheading / Circuit breakers





Closed Circuit



Open Circuit

blogs.halodoc.io

Are microservices always the right choice?

Advantages of Microservices

- Ship features faster and safer
- Amenable to horizontal scaling
- Target security concerns via isolation
- Allow the interplay of different systems and languages; no commitment to a single technology stack
- Easily deployable and replicable
- Embrace uncertainty, automation, and faults
- Better alignment with organization structure

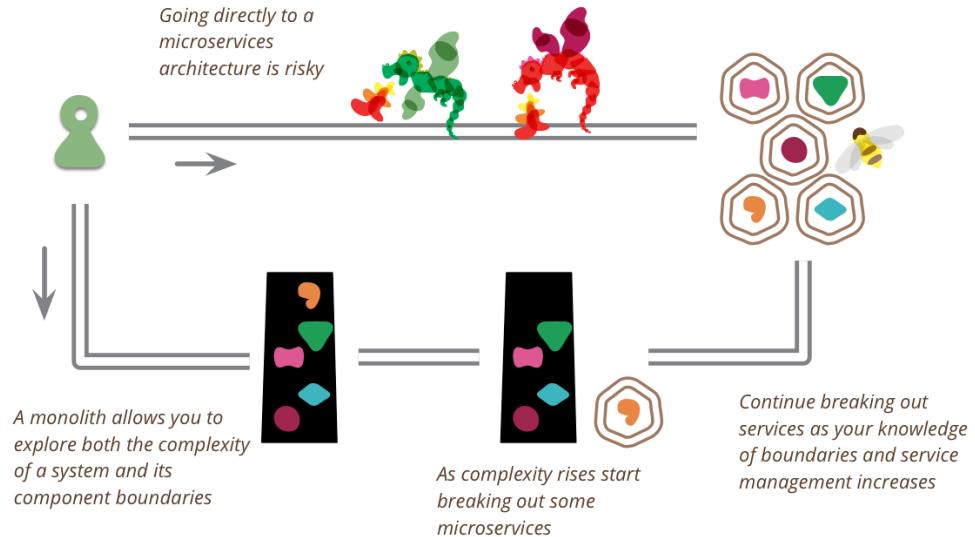
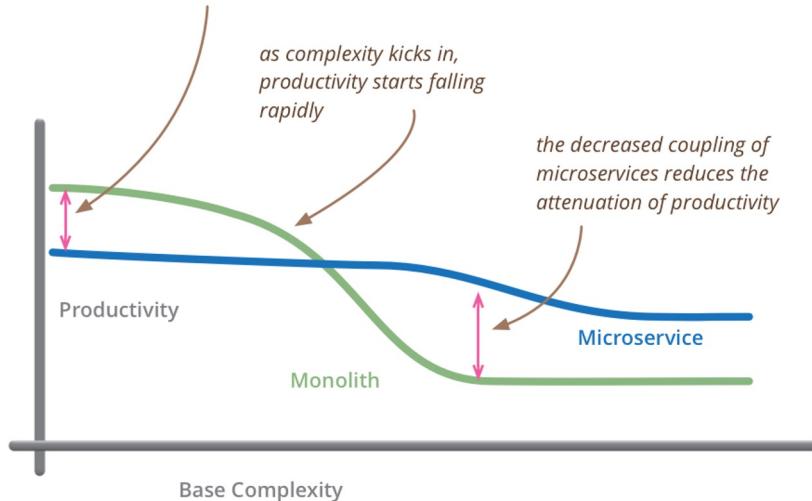
Microservice Challenges

- Too many choices
- Lag between investment and payback
- Complexities of distributed systems
 - network latency, faults, inconsistencies
 - testing challenges
- Monitoring is more complex
- More system states
- More points of failure
- Operational complexity
- Frequently adopted by breaking down a monolithic application

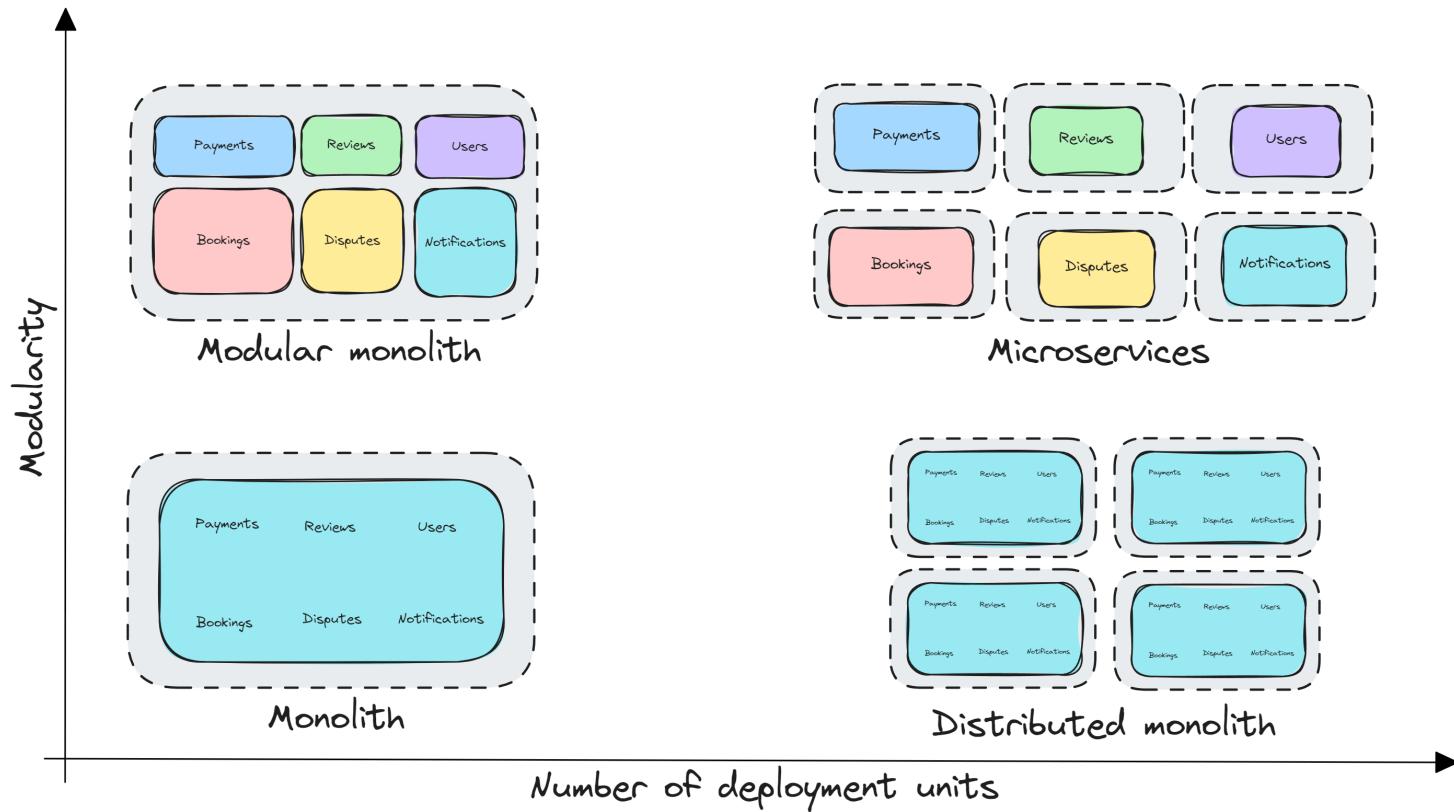


Microservices Overhead

for less-complex systems, the extra baggage required to manage microservices reduces productivity



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



<https://www.milanjovanovic.tech/blog/what-is-a-modular-monolith>