

Microservice Architectures

17-313 Fall 2023

Foundations of Software Engineering

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

Andrew Begel and Rohan Padhye

Inspirations:

Martin Fowler (<http://martinfowler.com/articles/microservices.html>)

Josh Evans @ Netflix (<https://www.youtube.com/watch?v=CZ3wluvmHeM>)

Matt Ranney @ Uber (<https://www.youtube.com/watch?v=kb-m2fasdDY>)

Christopher Meiklejohn & Filibuster (<http://filibuster.cloud>)

Administrivia

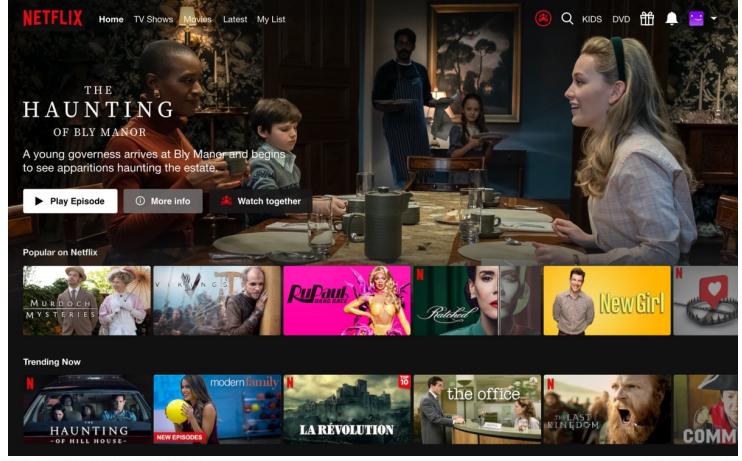
- Mid-term exam next week (Oct 10) in class
- Recitation this week: midterm review (**come prepared!**)
 - Work through problems on the previous midterms – many students found this helpful.
 - Any questions on the previous midterm questions – bring them to recitation to discuss as a class.
- Final Presentations (P5):
Tuesday December 12th, 5:30 pm - 8:30pm, Room TBD

Learning Goals

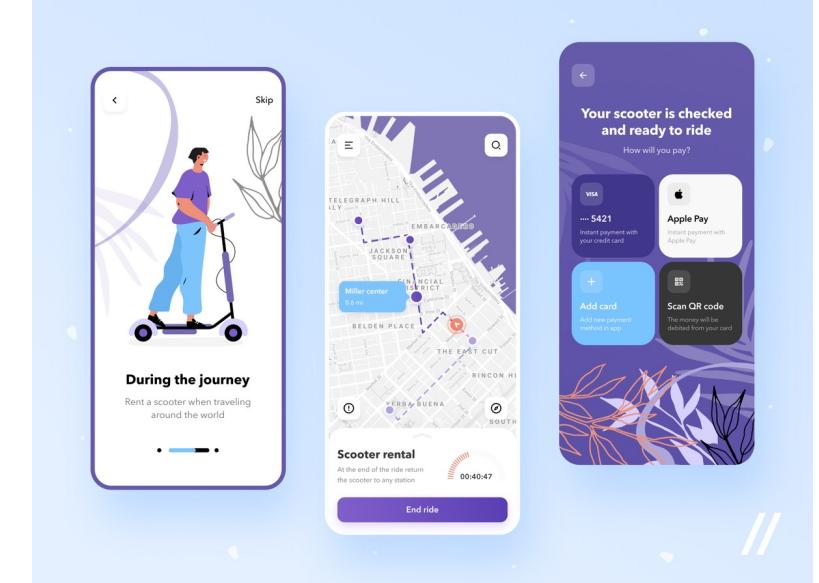
- Contrast the monolithic application design with a modular design based on microservices.
- Reason about how architectural choices affect software quality and process attributes.
- Reason about tradeoffs of microservices architectures.

Before we get to microservices...

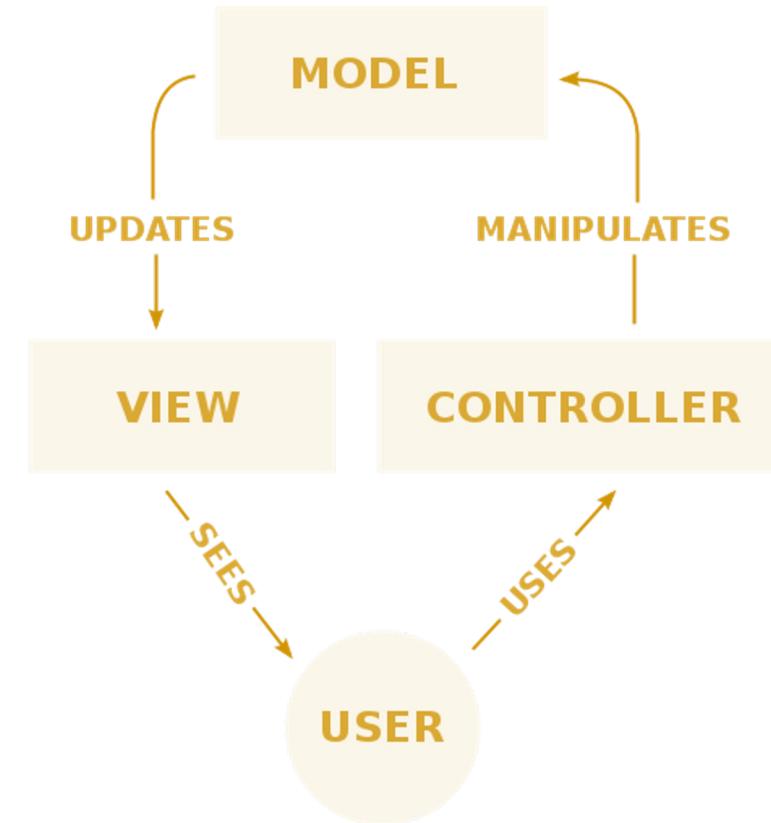
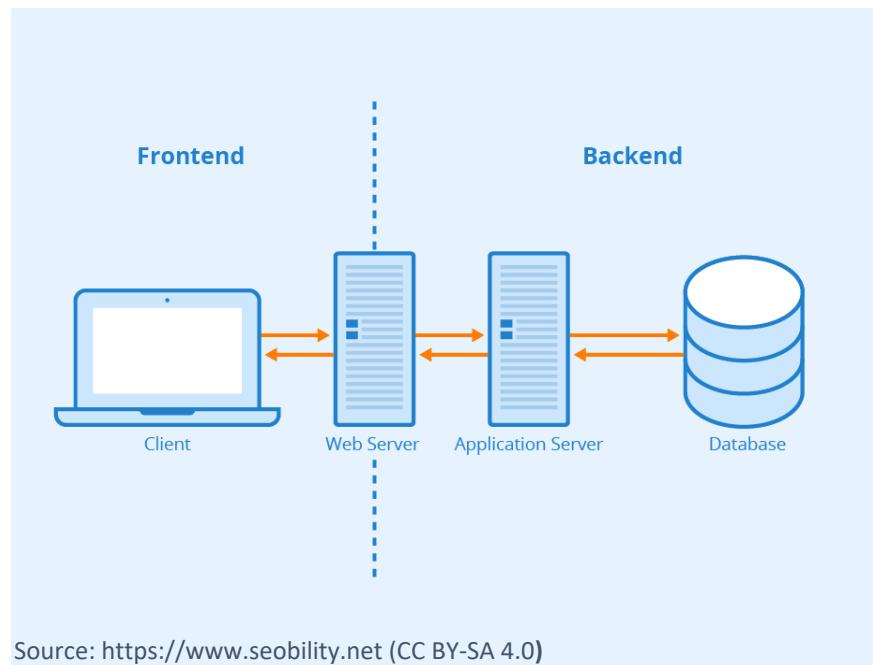
How might these apps be architected?



A screenshot of a document management system. The top navigation bar includes 'Simsics Docs', 'Documents', 'Tags', and 'Users & Groups'. A search bar is present. The main content area shows a search result for a document titled 'Tourism' created on 11/18/17 by 'admin'. The interface includes a sidebar with filters like 'Auto-configuration', 'Auto-segregation', 'Discrimination based on skin color', 'Collections', 'Indigo Era (economist)', 'Dhat syndrome', 'Korean ethnic nationalism', 'Individualism', 'Anarchism', and 'Music'. Below the search results are thumbnail previews of documents and images related to tourism.



Monolithic styles: Client-server or MVC

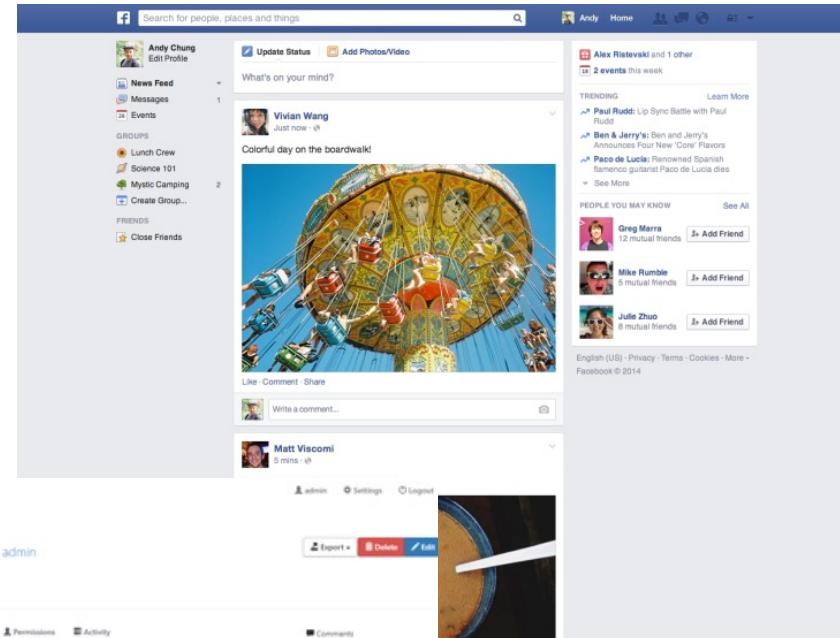


Monoliths make trade-offs on software quality

Several consequences of this architecture on:

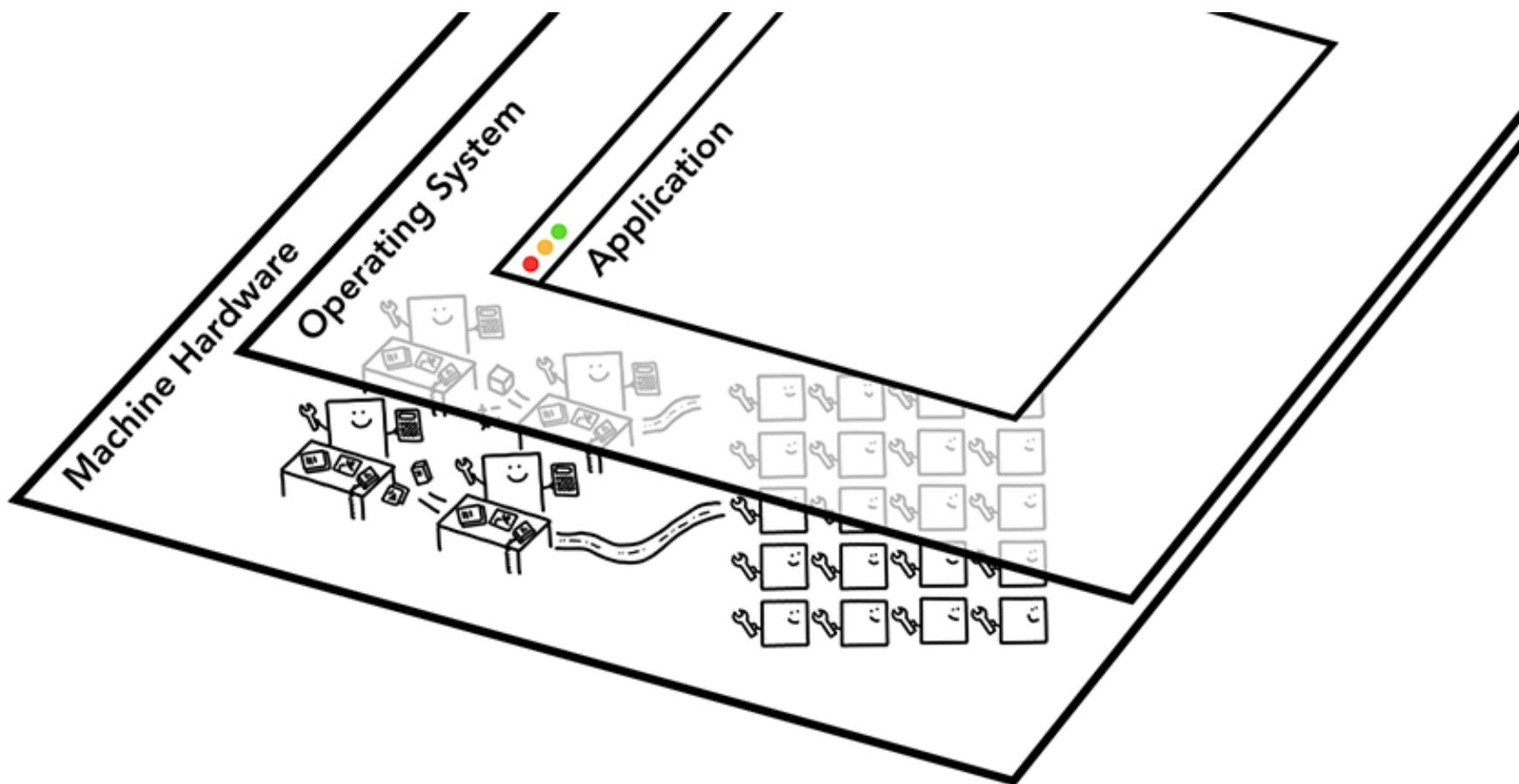
- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership

The screenshot shows two views of the Simeics Docs application. On the left, a search results page displays a list of documents categorized by tags, such as 'Authoritarianism', 'Auto-socialization', 'Discrimination based on skin color', 'Collections', 'Indigo Era (colonial)', 'Dhat syndrome', 'Korean ethnic nationalism', 'Individualism', 'Akanthism', and 'Music'. On the right, a detailed view of a document titled 'Tourism' is shown, featuring a preview image of a city skyline, a list of contributors, and a section for adding comments.



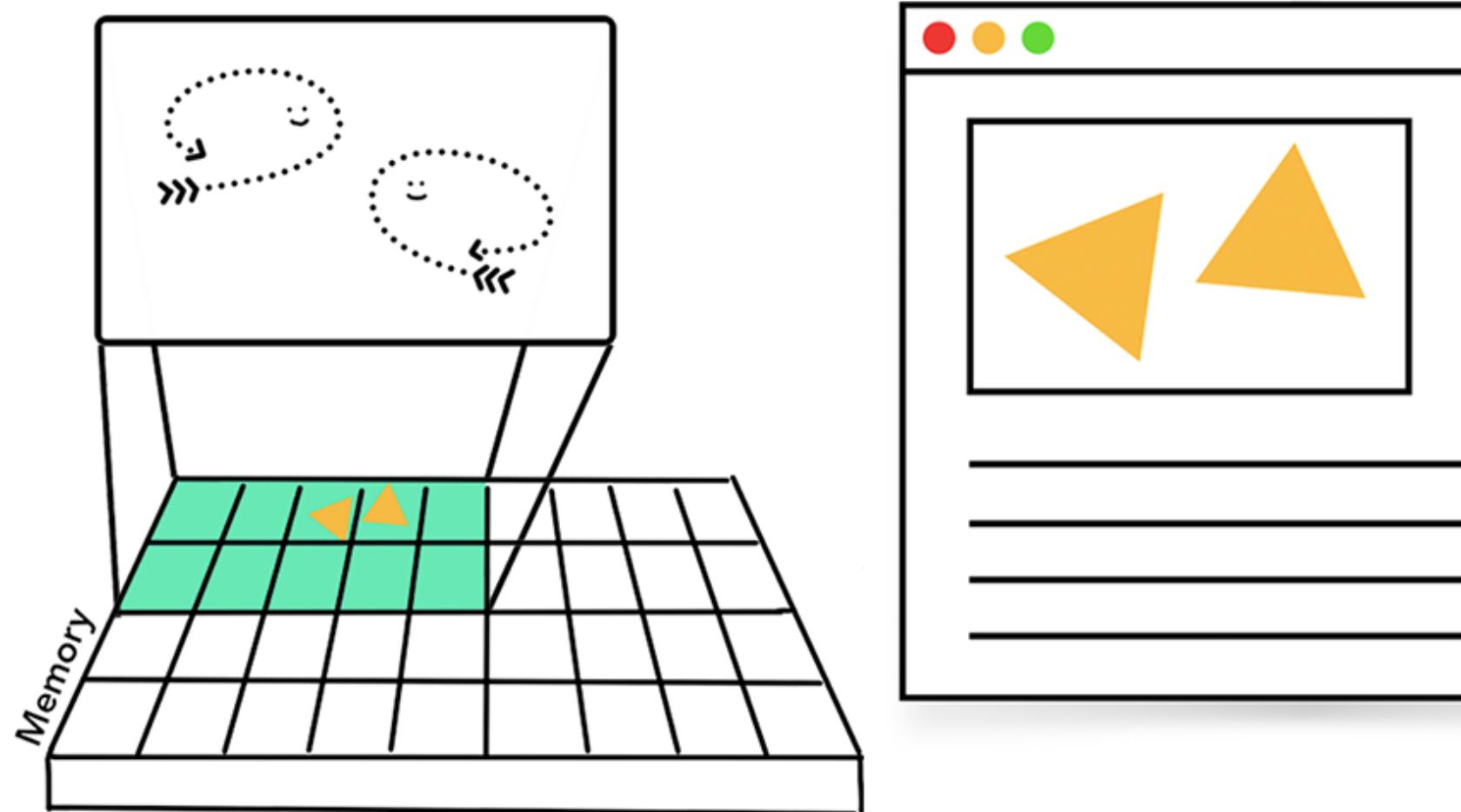
Service-based architecture – Chrome

Web Browsers



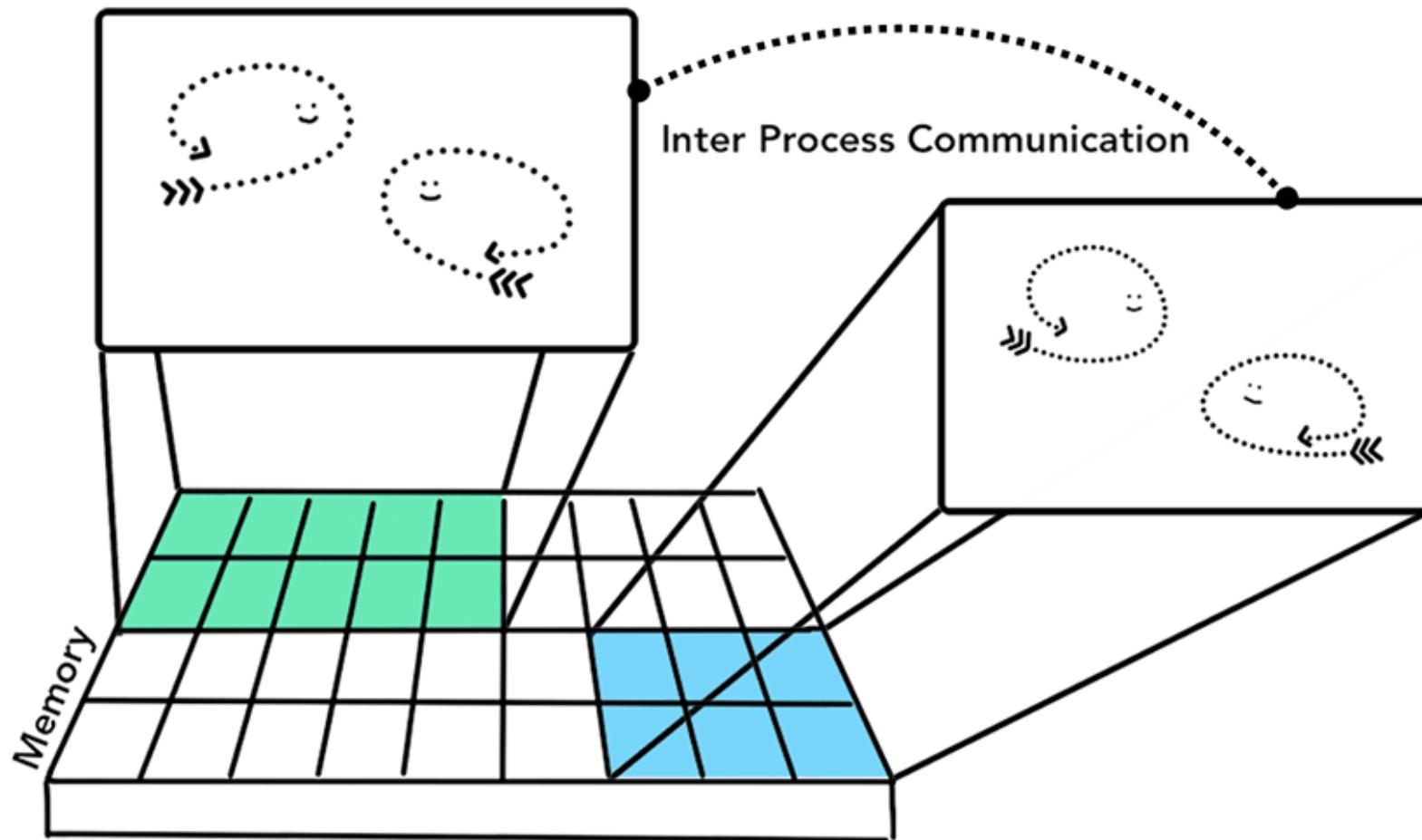
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Browser: A multi-threaded process



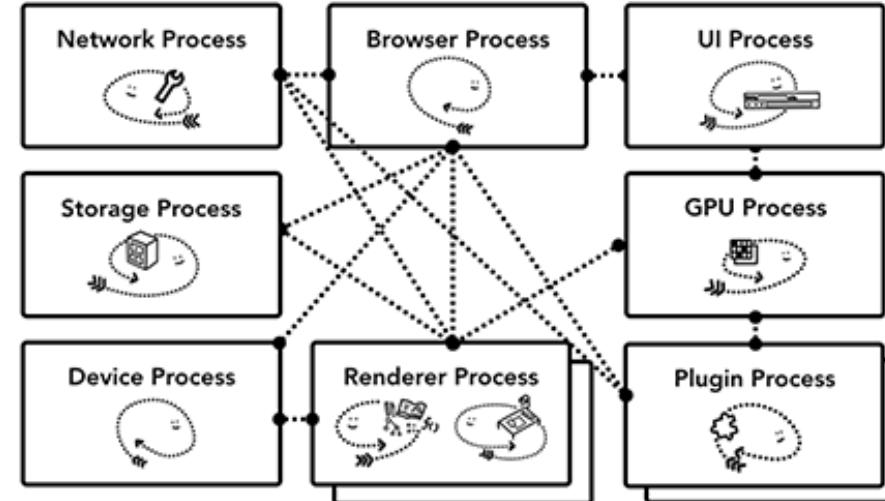
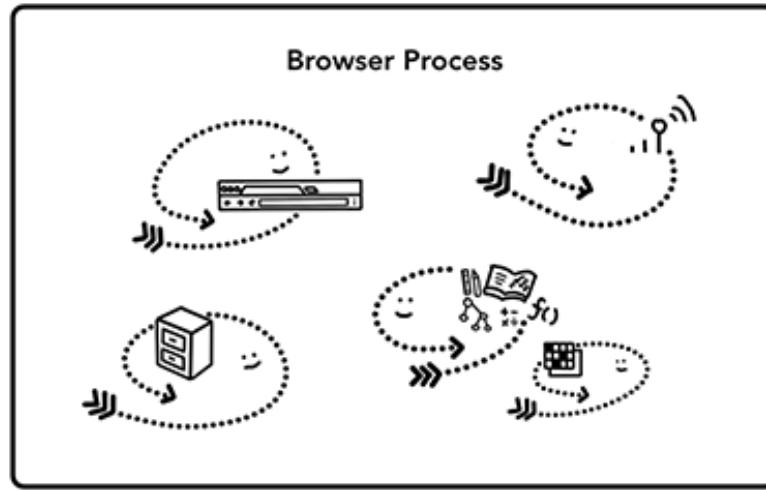
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Multi-process browser with IPC



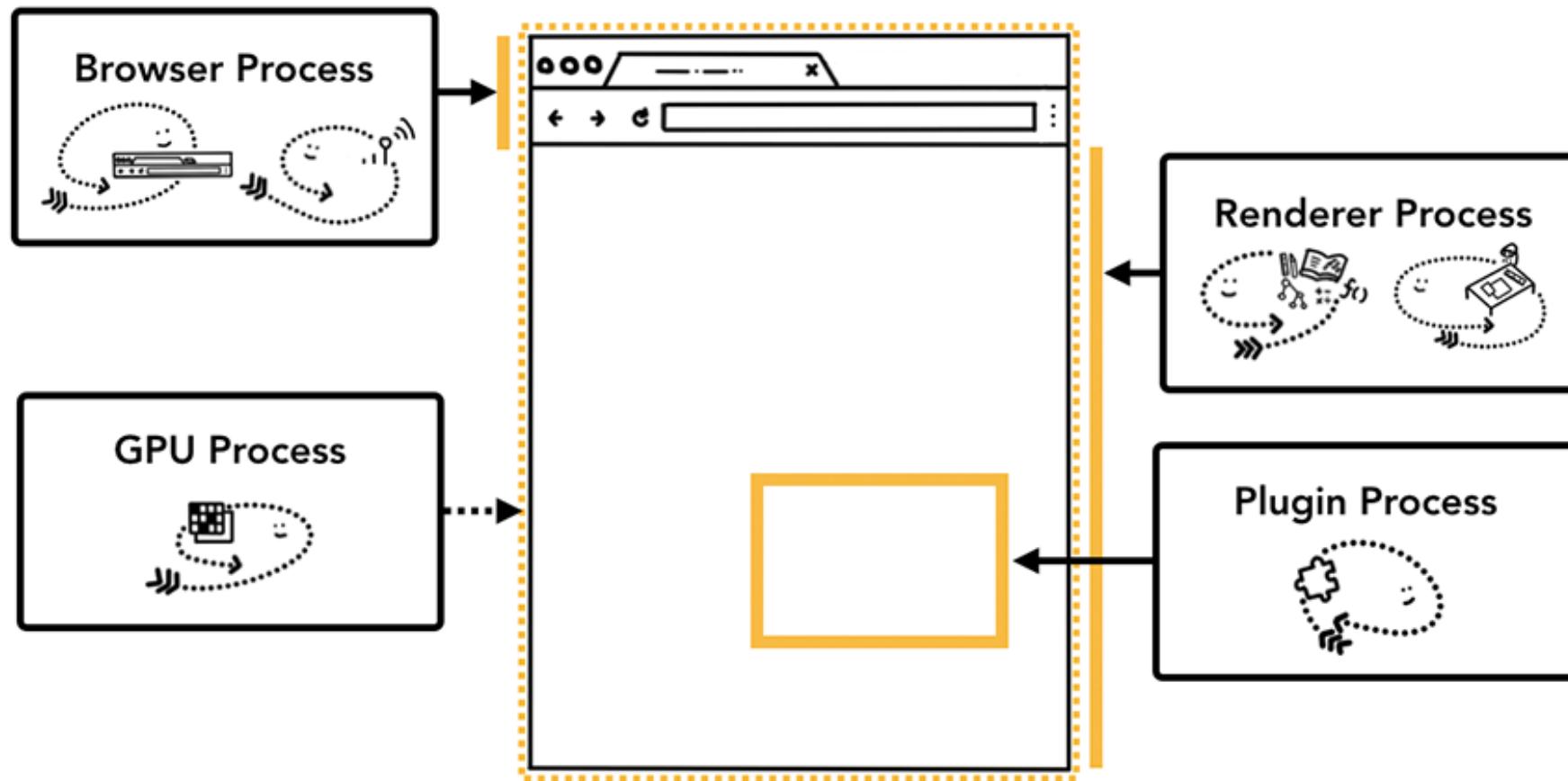
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Browser Architectures



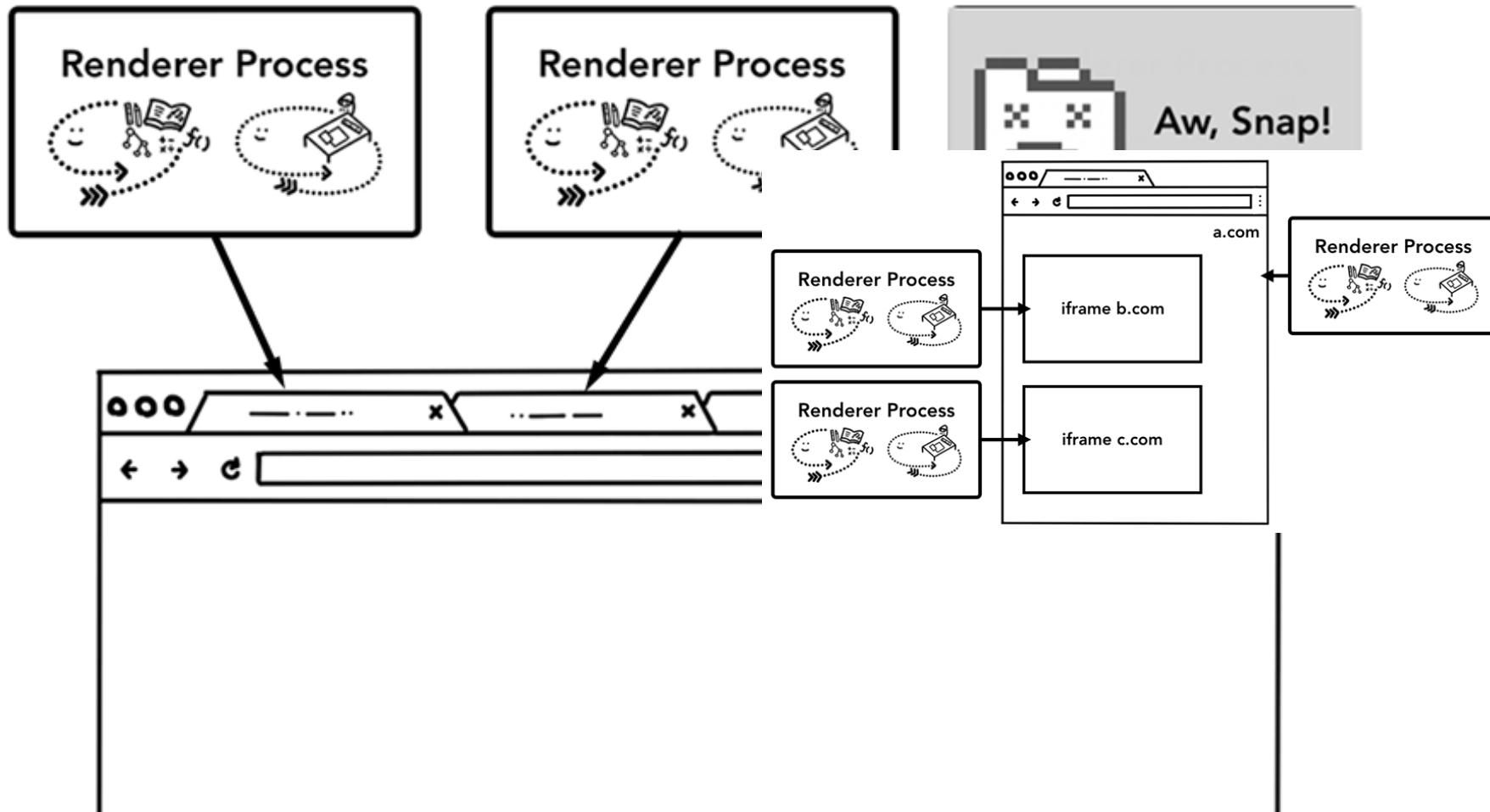
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



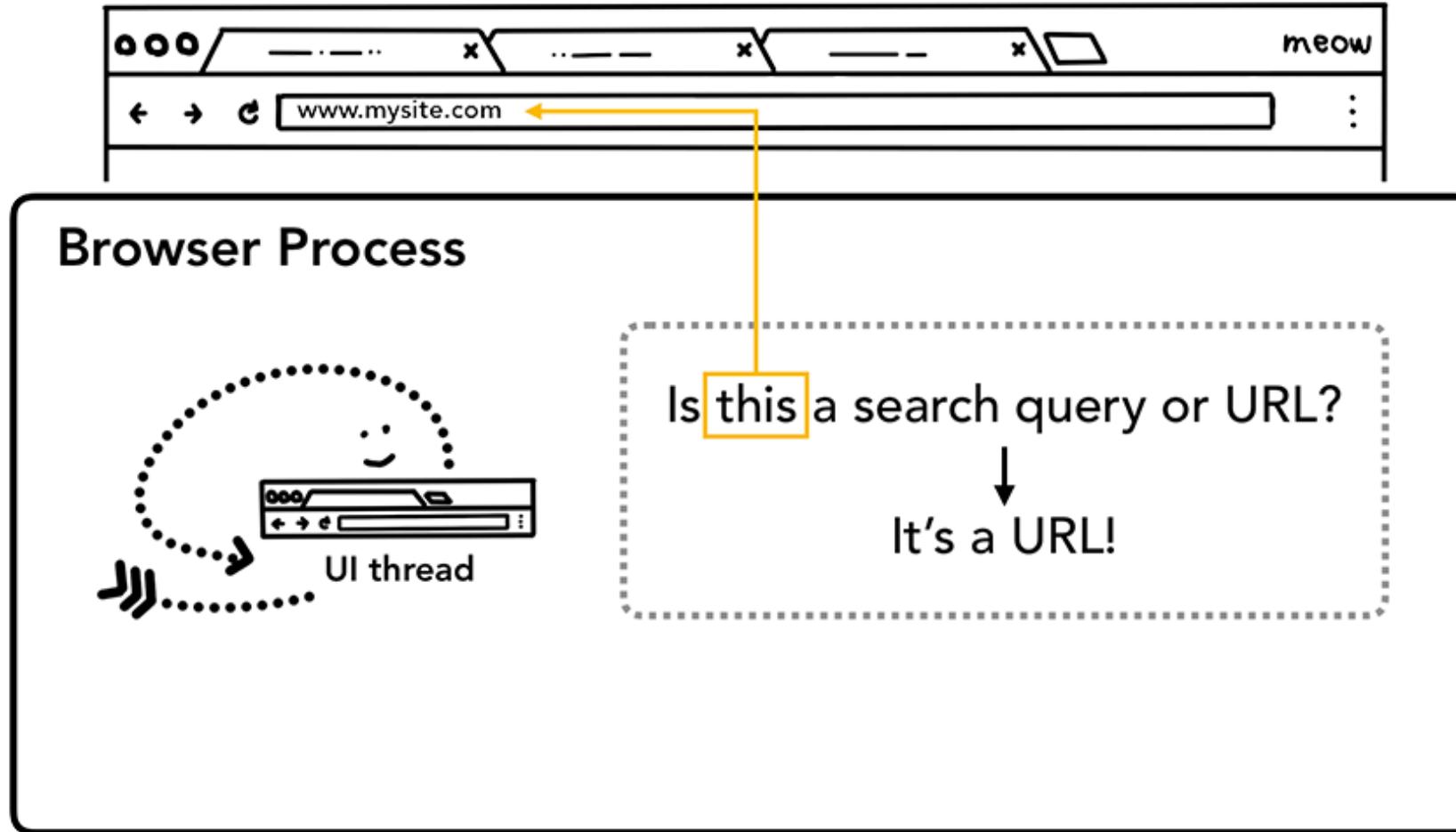
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



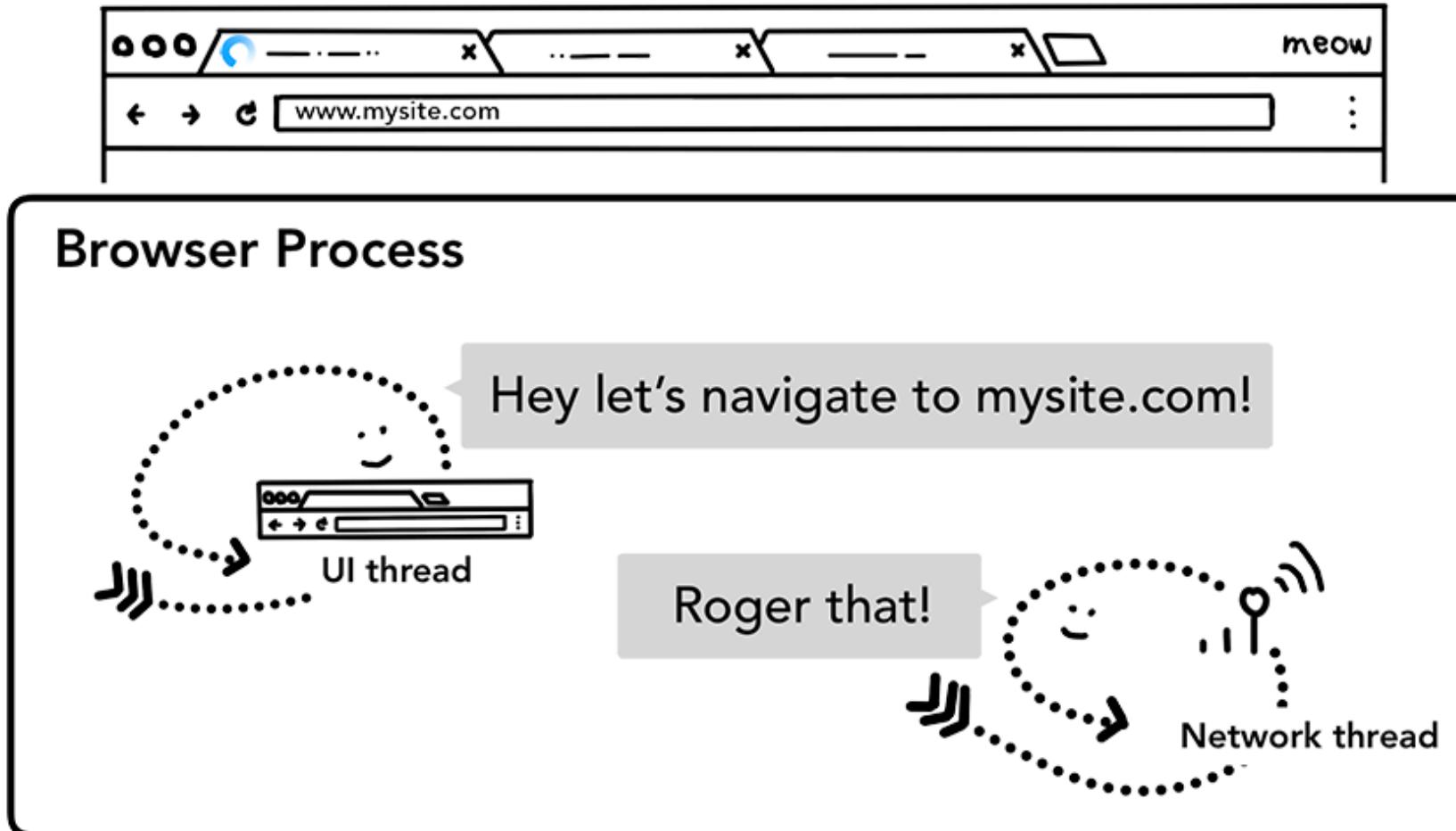
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



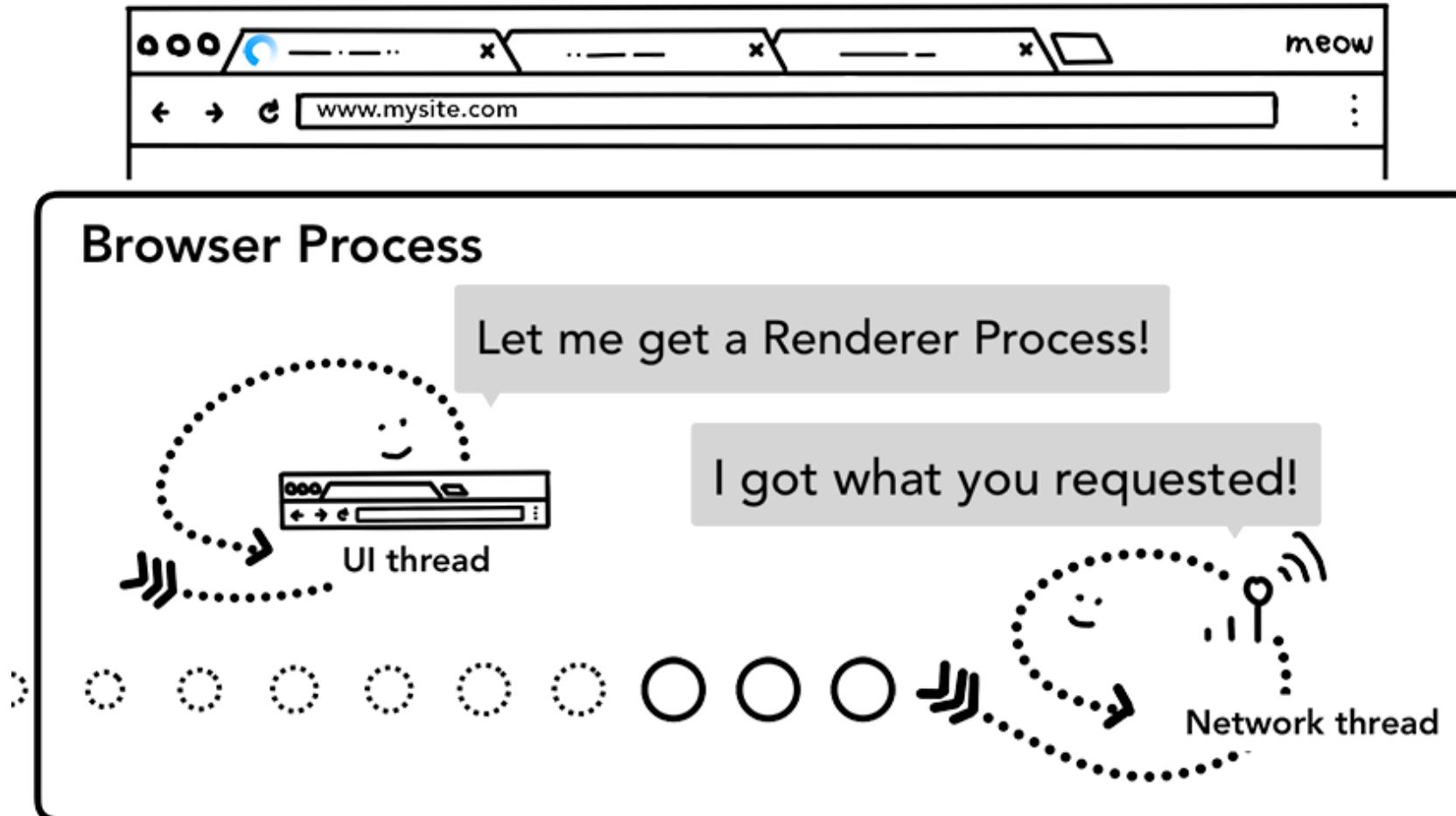
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



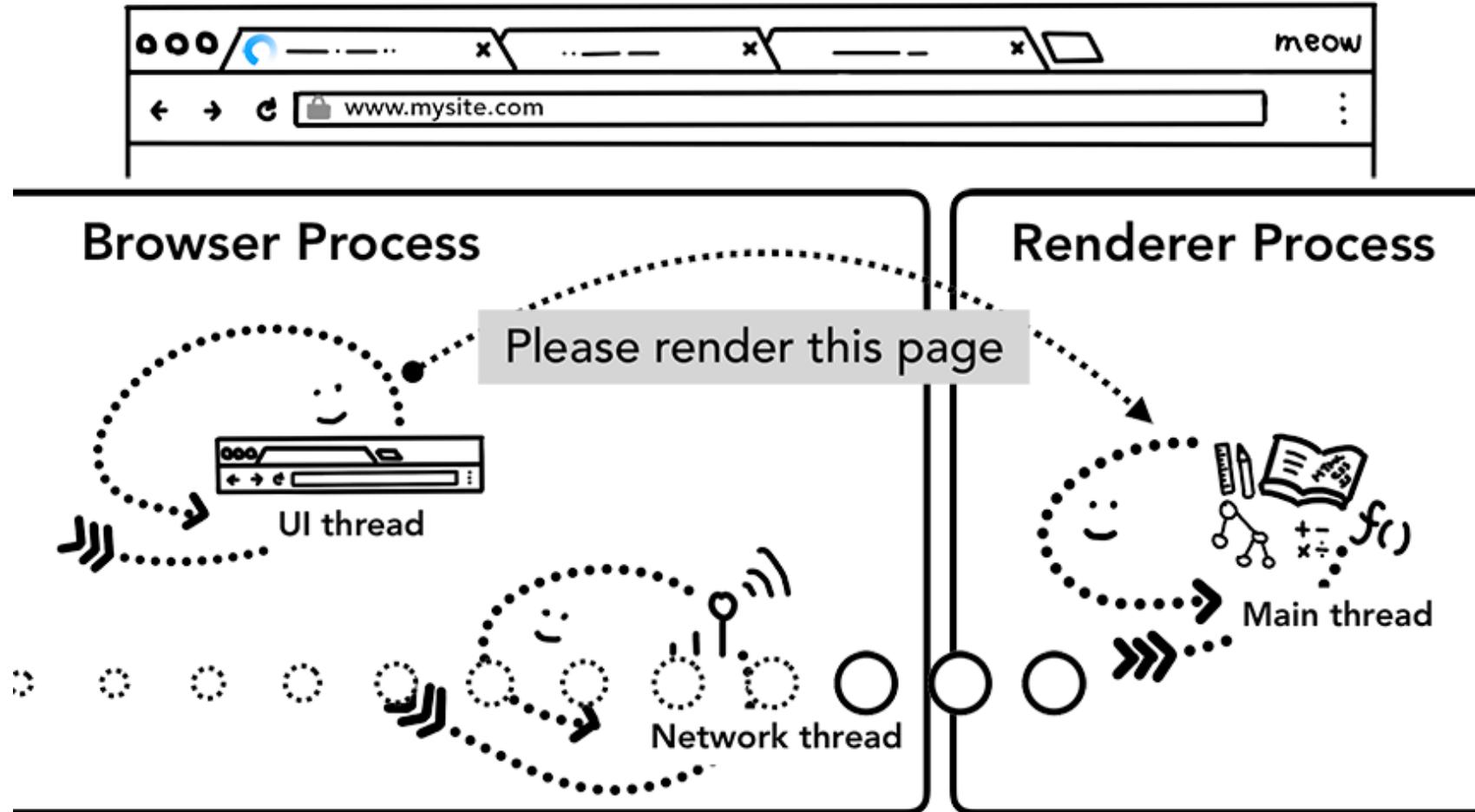
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



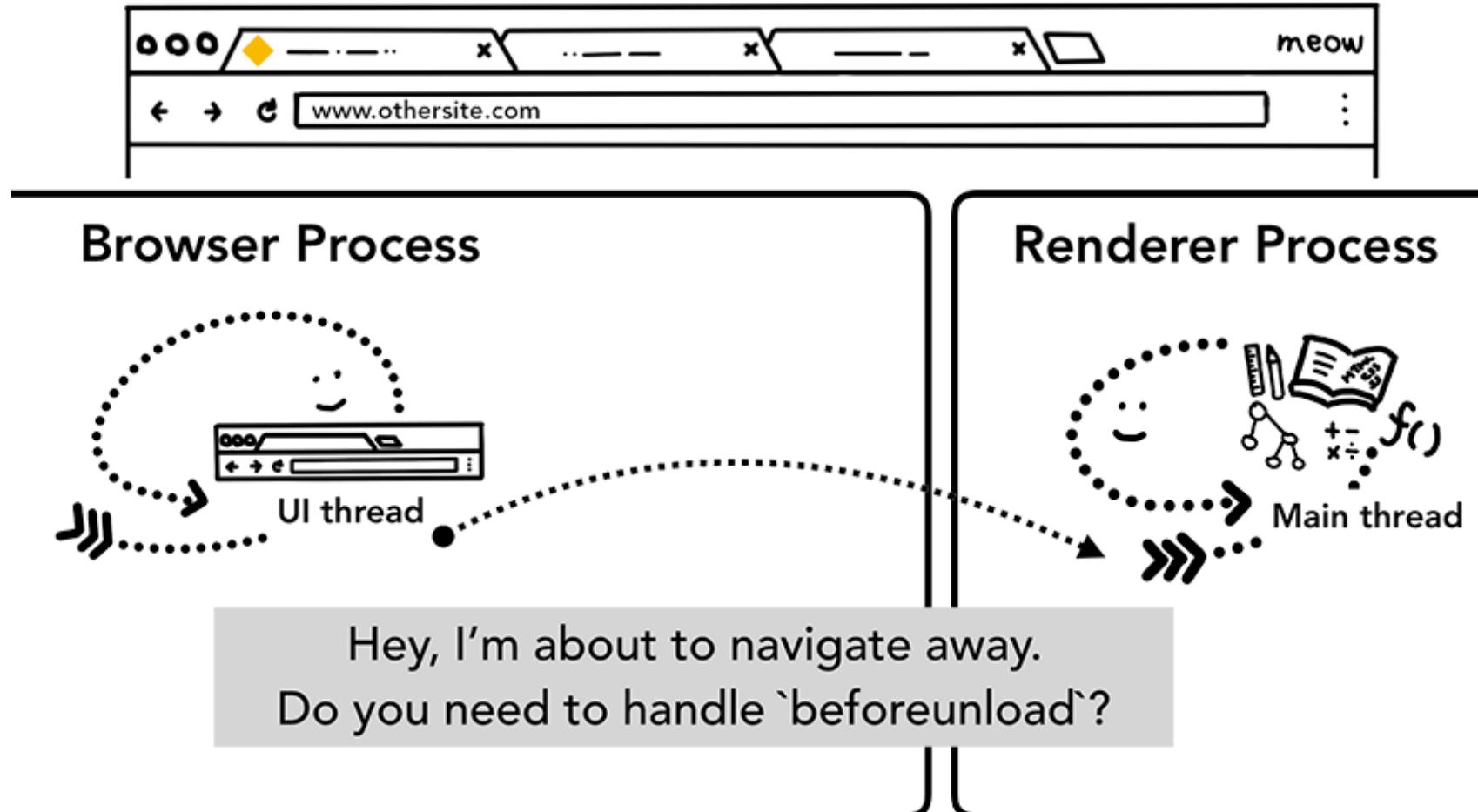
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



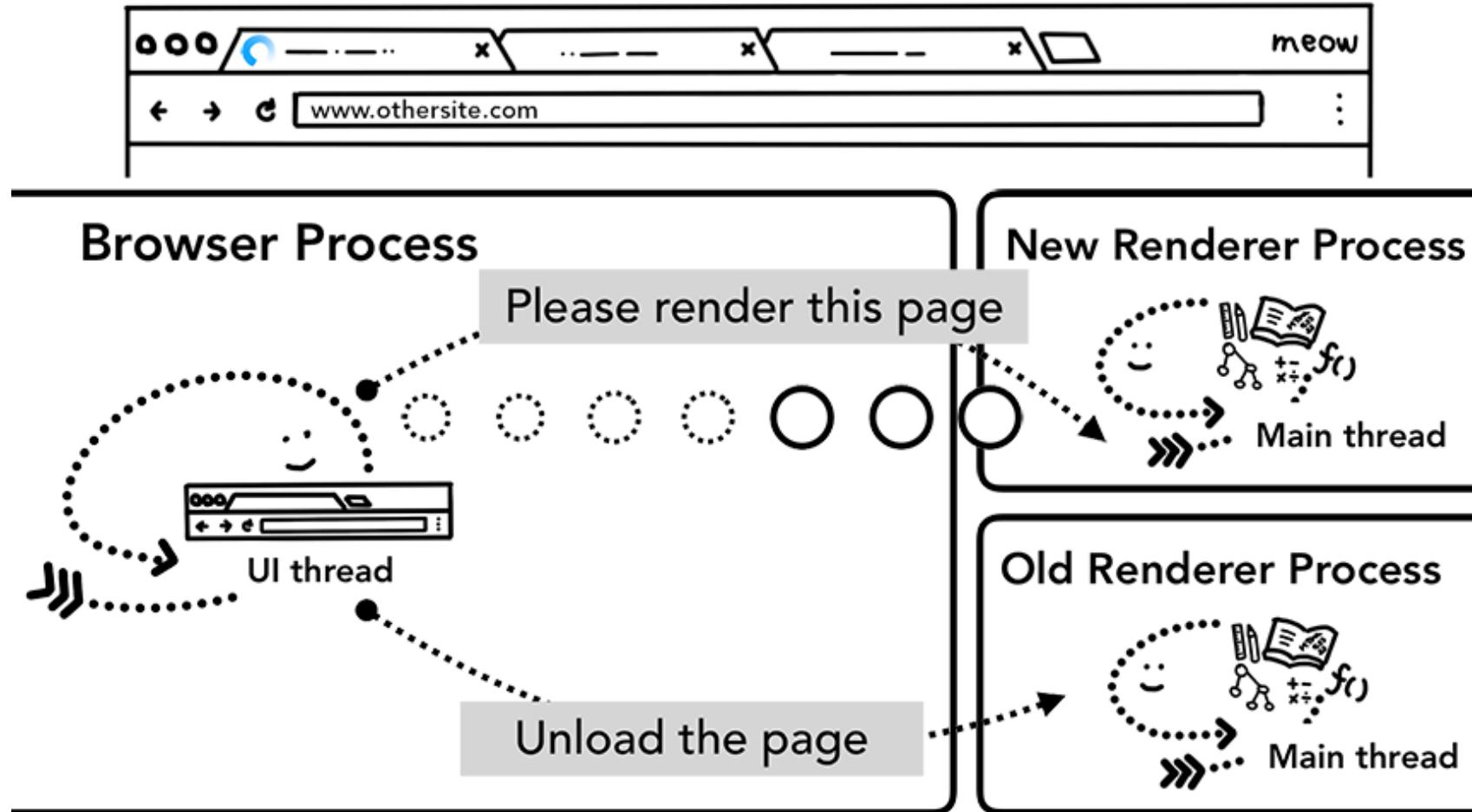
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests



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

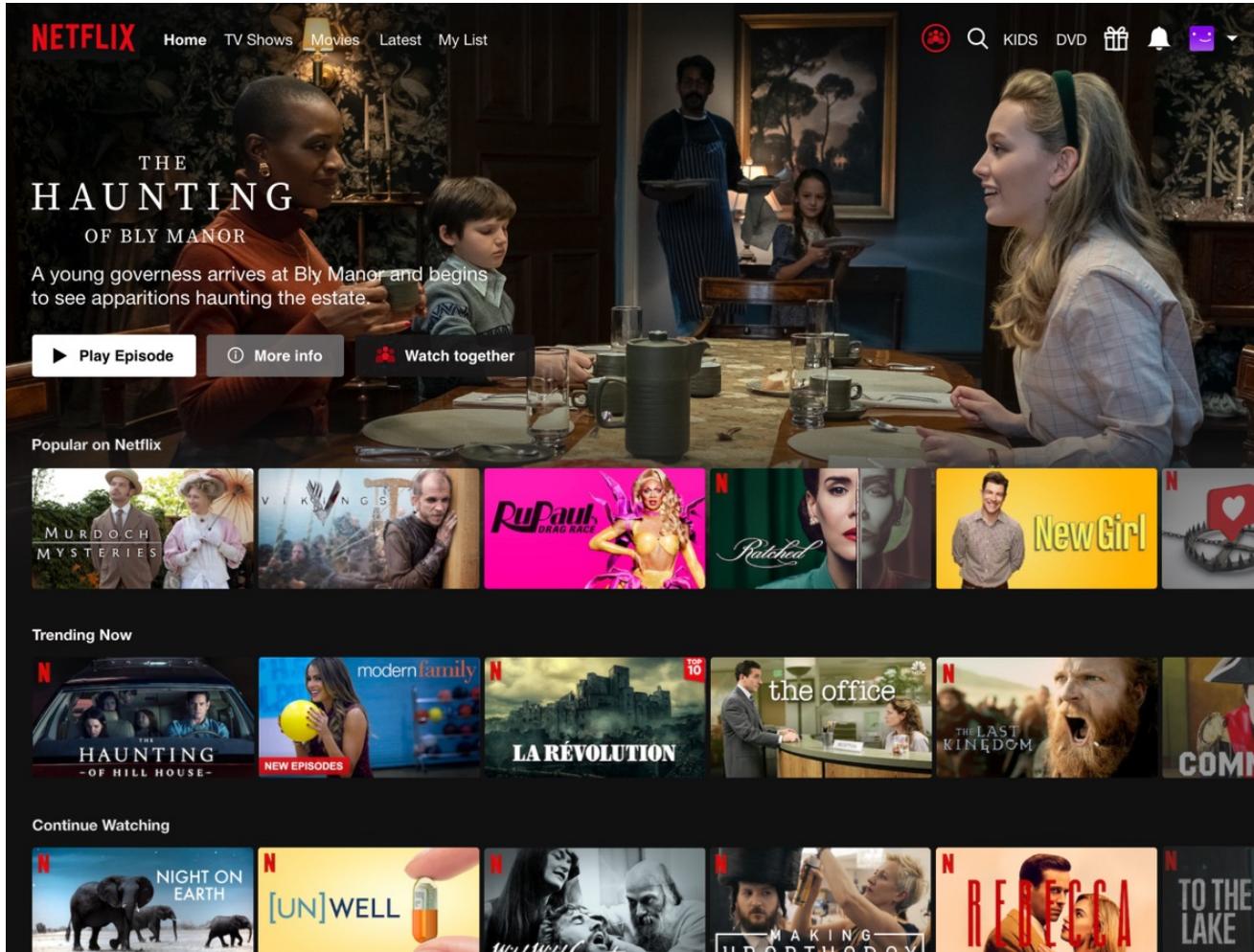
Navigating to a web site uses service requests



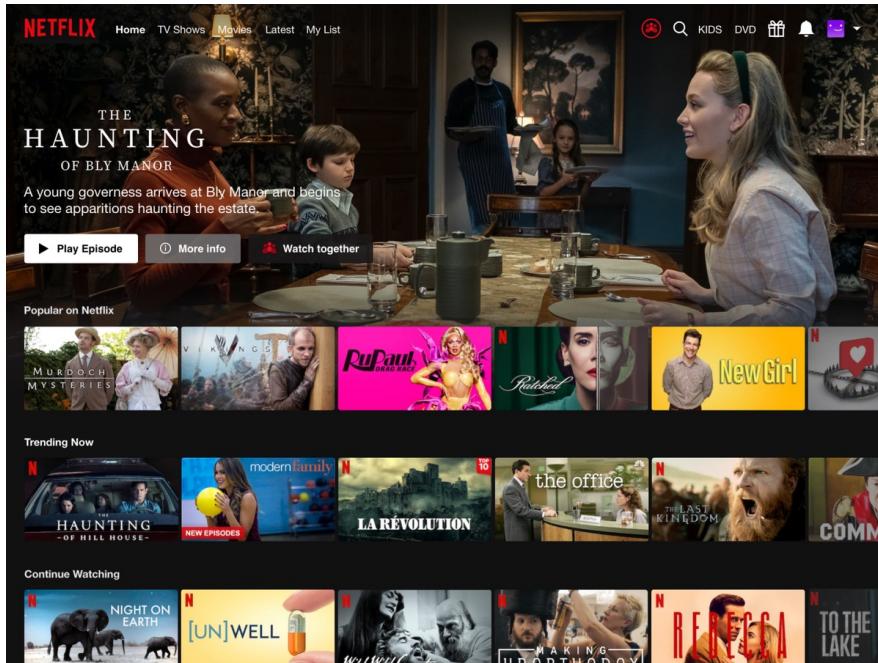
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Microservice architecture – Netflix

Netflix

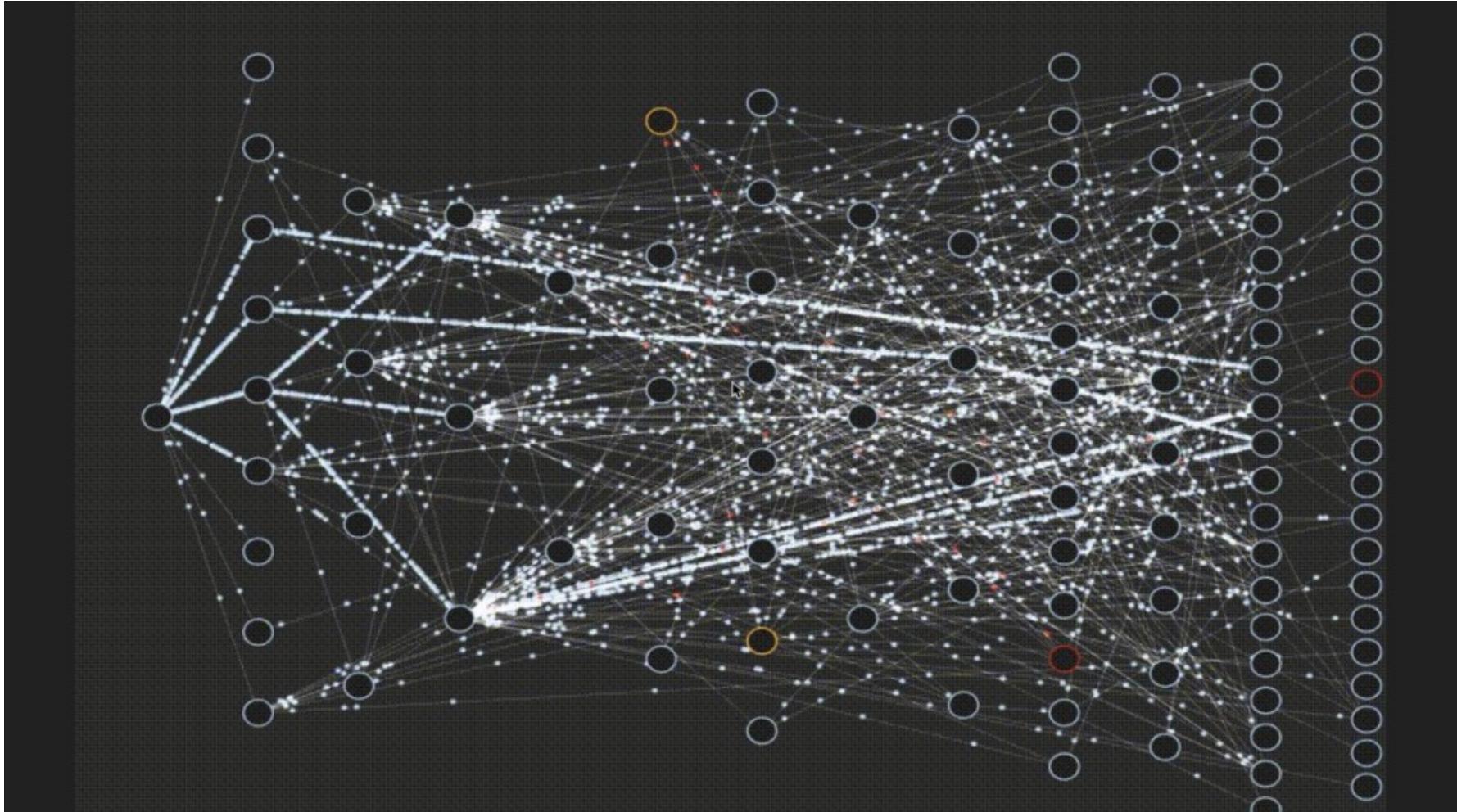


Netflix Microservices – App Boot



- Recommendations
- Trending Now
- Continue Watching
- My List
- Metrics

Netflix Microservices – One Request

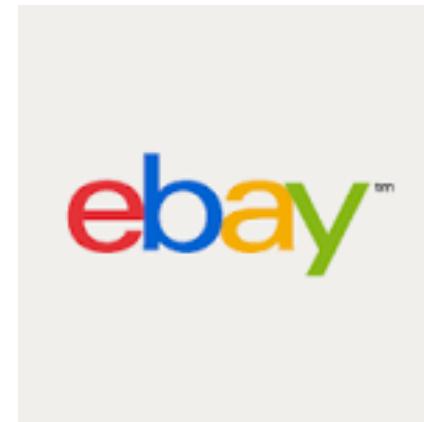
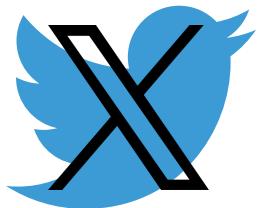


<https://www.youtube.com/watch?v=CZ3wluvmHeM>

Who uses Microservices?



COMCAST



UBER

GROUPON®

Microservices – The Hipster Shop Example

Hipster Shop: Guess some microservices

The image displays two screenshots of the Online Boutique website, illustrating a microservices architecture.

Left Screenshot (Homepage):

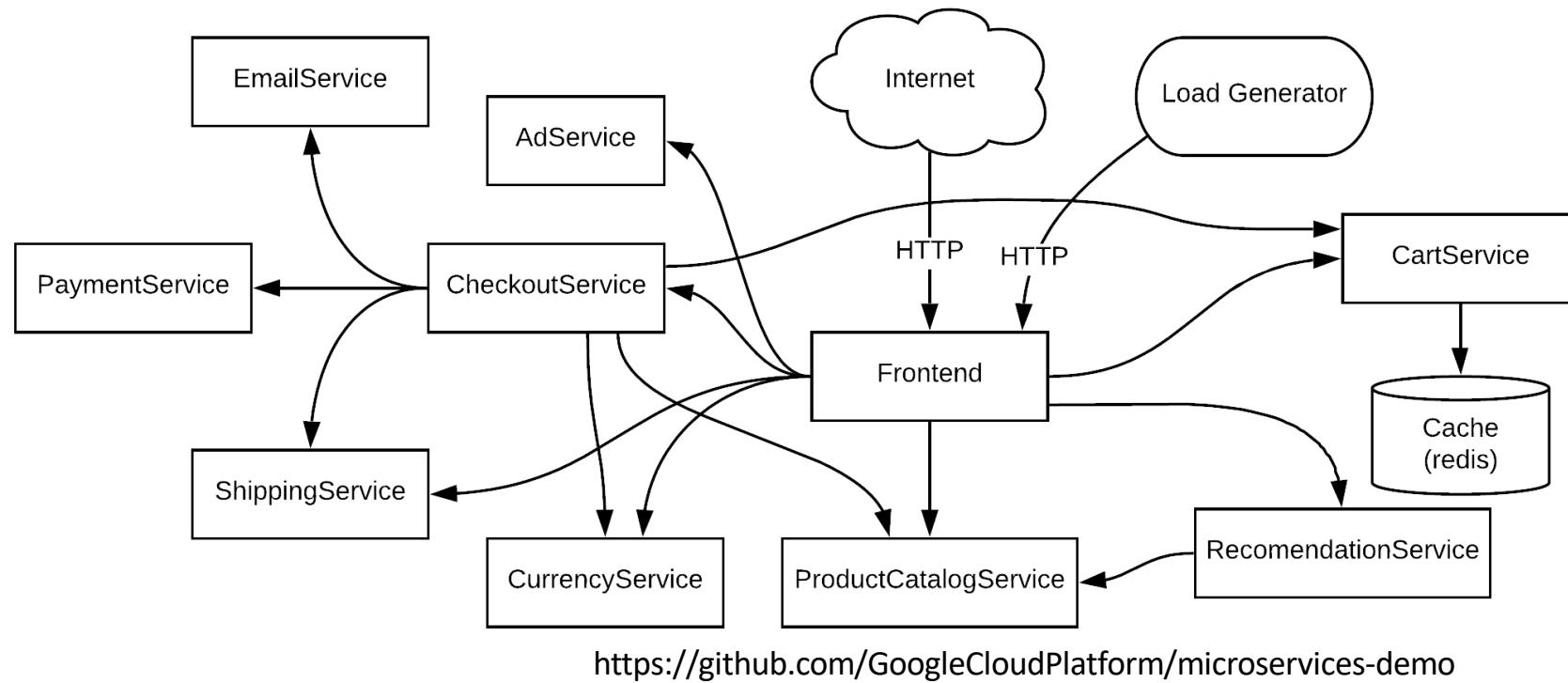
- Header: "ONLINE BOUTIQUE" logo, "Google Cloud" badge, "TRY" button, and "Cart" icon.
- Text: "Free shipping with \$75 purchase!"
- Section: "Hot PRODUCTS" featuring five items: "VINTAGE TYPEWRITER" (TRY 368.34), "VINTAGE CAMERA LENS" (TRY 67.66), "HOME BARISTA KIT" (TRY 671.79), "Terrarium" (TRY 12.99), and "Vintage Camera" (TRY 338.33).

Right Screenshot (Cart Page):

- Header: "ONLINE BOUTIQUE" logo, "Google Cloud" badge, "TRY" button, and "Cart" icon.
- Text: "Free shipping with \$75 purchase!"
- Section: "2 items in your cart" with "EMPTY CART" and "KEEP BROWSING" buttons.
- Items:
 - Home Barista Kit**
SKU: #1YMWNN1N40
Quantity: 1
TRY 671.79
 - Vintage Camera Lens**
SKU: #6EVCHSJNUP
Quantity: 5
TRY 338.33
- Text: "Shipping Cost: TRY 93.23" and "Total Cost: TRY 1103.36"
- Section: "Checkout" with fields for "E-mail Address" (someone@example.com), "Street Address" (1600 Amphitheatre Parkway), "Zip Code" (94043), "City" (Mountain View), "State" (CA), "Country" (United States), "Credit Card Number" (4432 8015 6152 0454), "Month" (January), "Year" (2021), and "CVV" (***). A "PLACE ORDER" button is at the bottom.

<https://onlineboutique.dev>

Hipster Shop Microservice Architecture



Microservices

What are the consequences of this architecture? On:

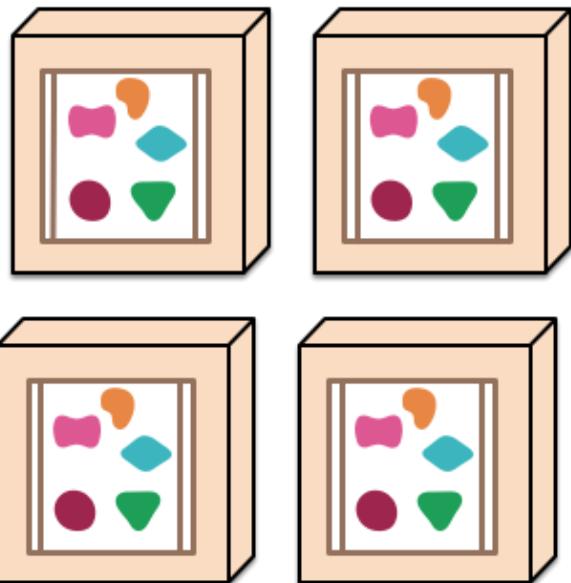
- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership
- Data Consistency

Scalability

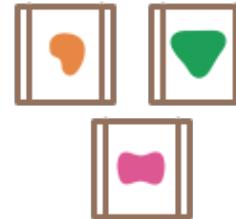
A monolithic application puts all its functionality into a single process...



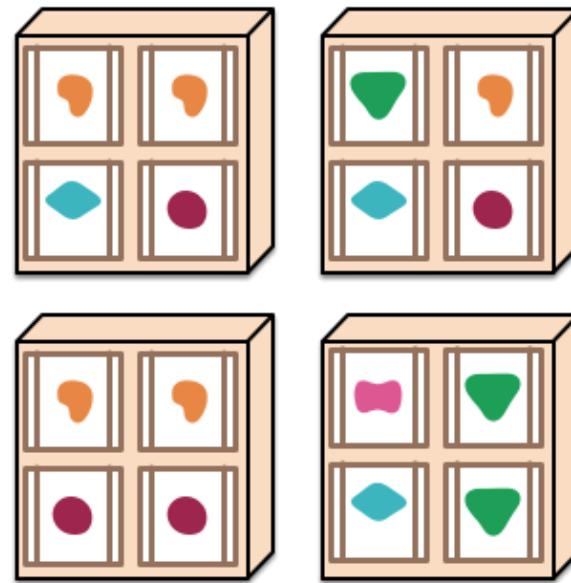
... and scales by replicating the monolith on multiple servers



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

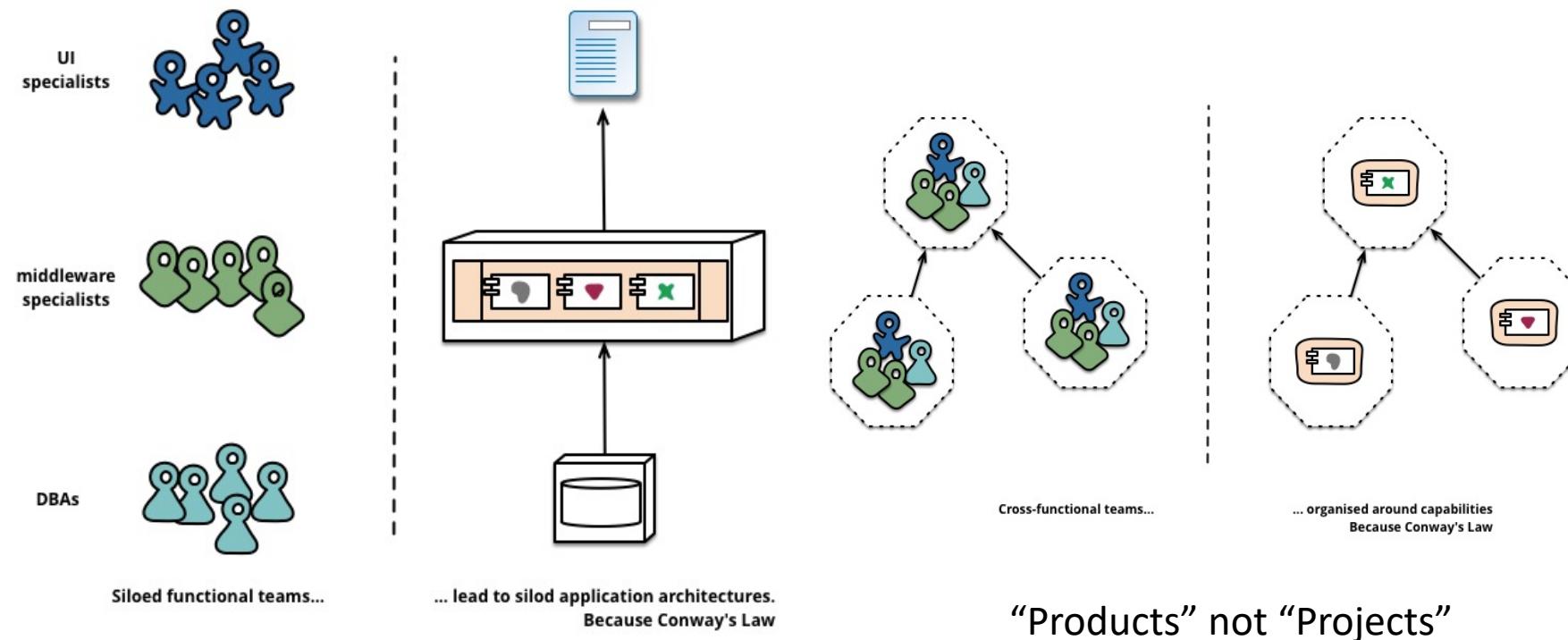


... and scales by distributing these services across servers, replicating as needed.



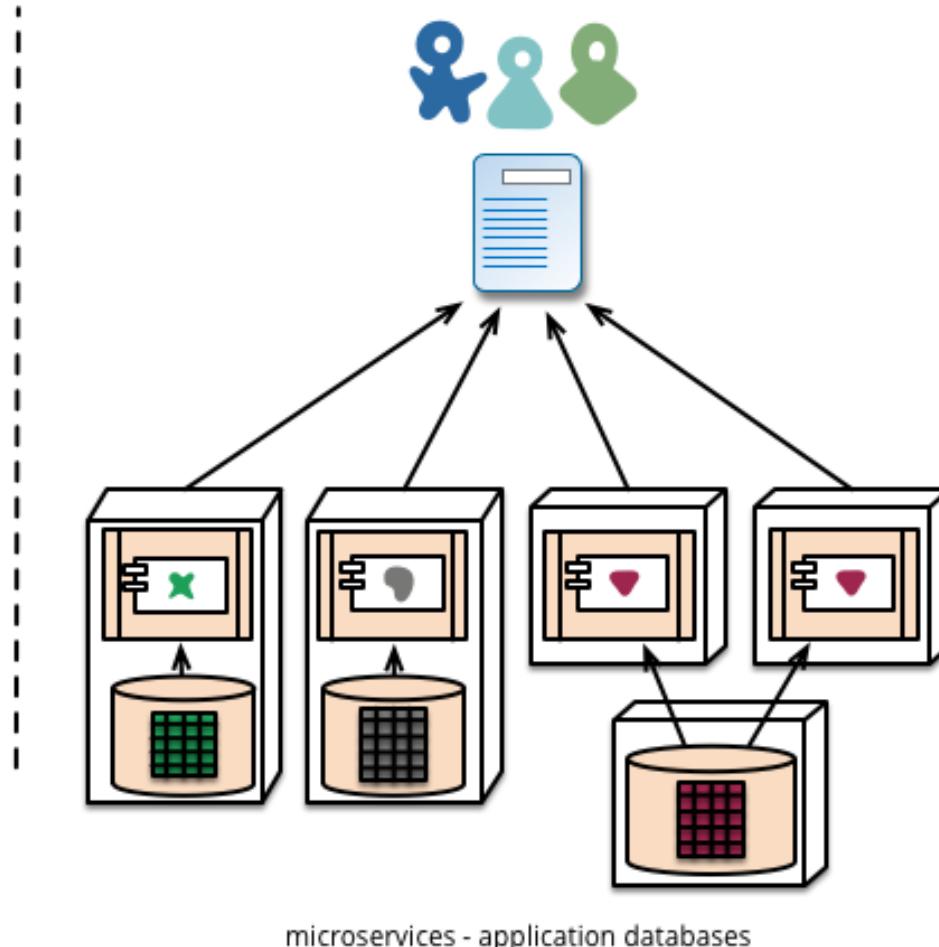
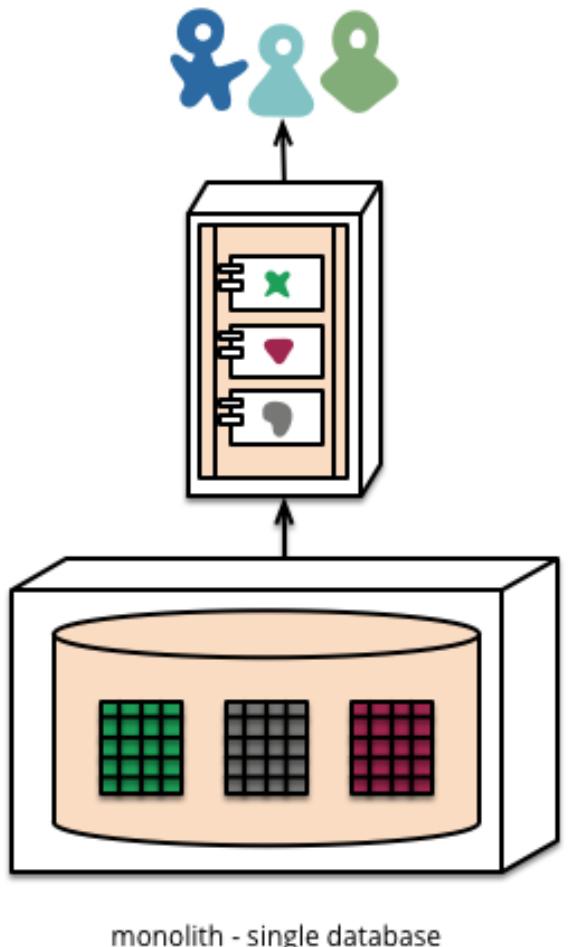
Source: <http://martinfowler.com/articles/microservices.html>

Team Organization (Conway's Law)



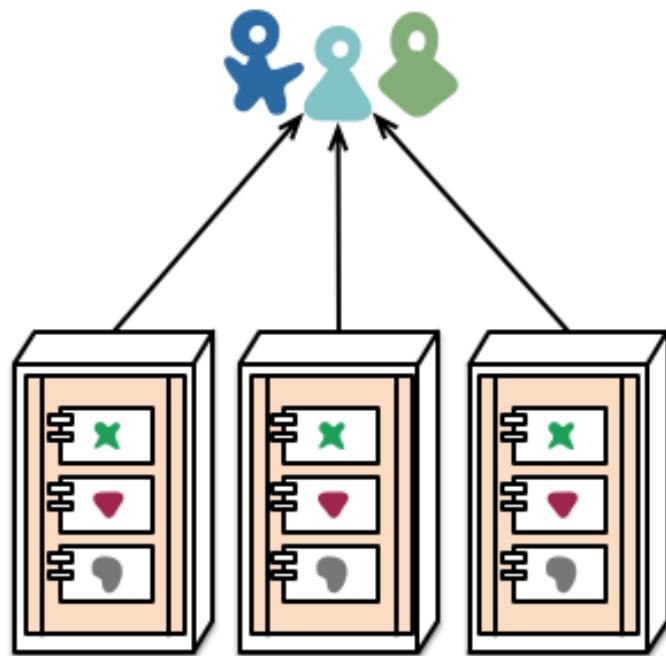
Source: <http://martinfowler.com/articles/microservices.html>

Data Management and Consistency

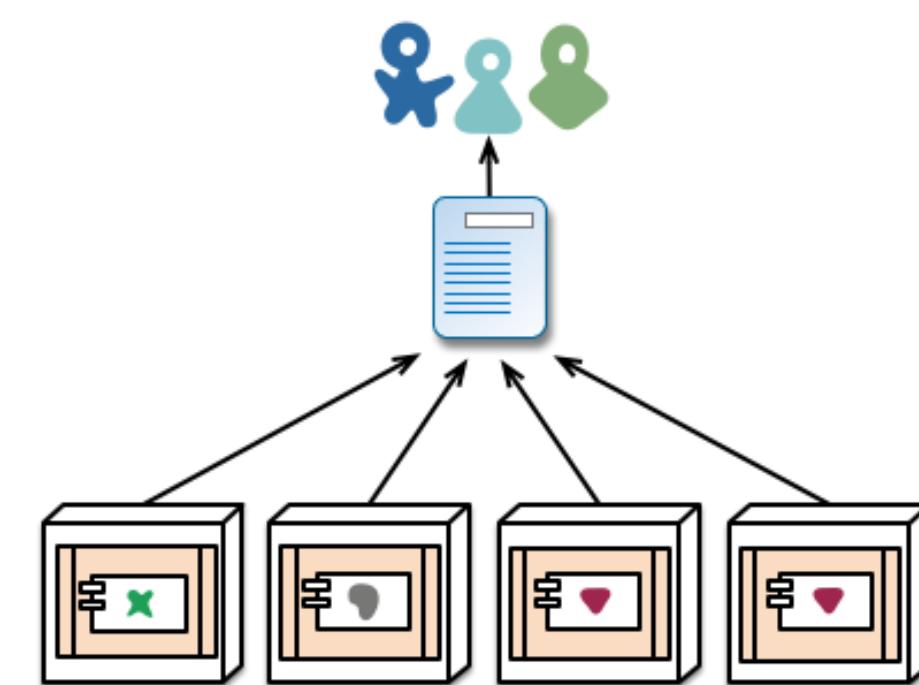


Source: <http://martinfowler.com/articles/microservices.html>

Deployment and Evolution



monolith - multiple modules in the same process



microservices - modules running in different processes

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

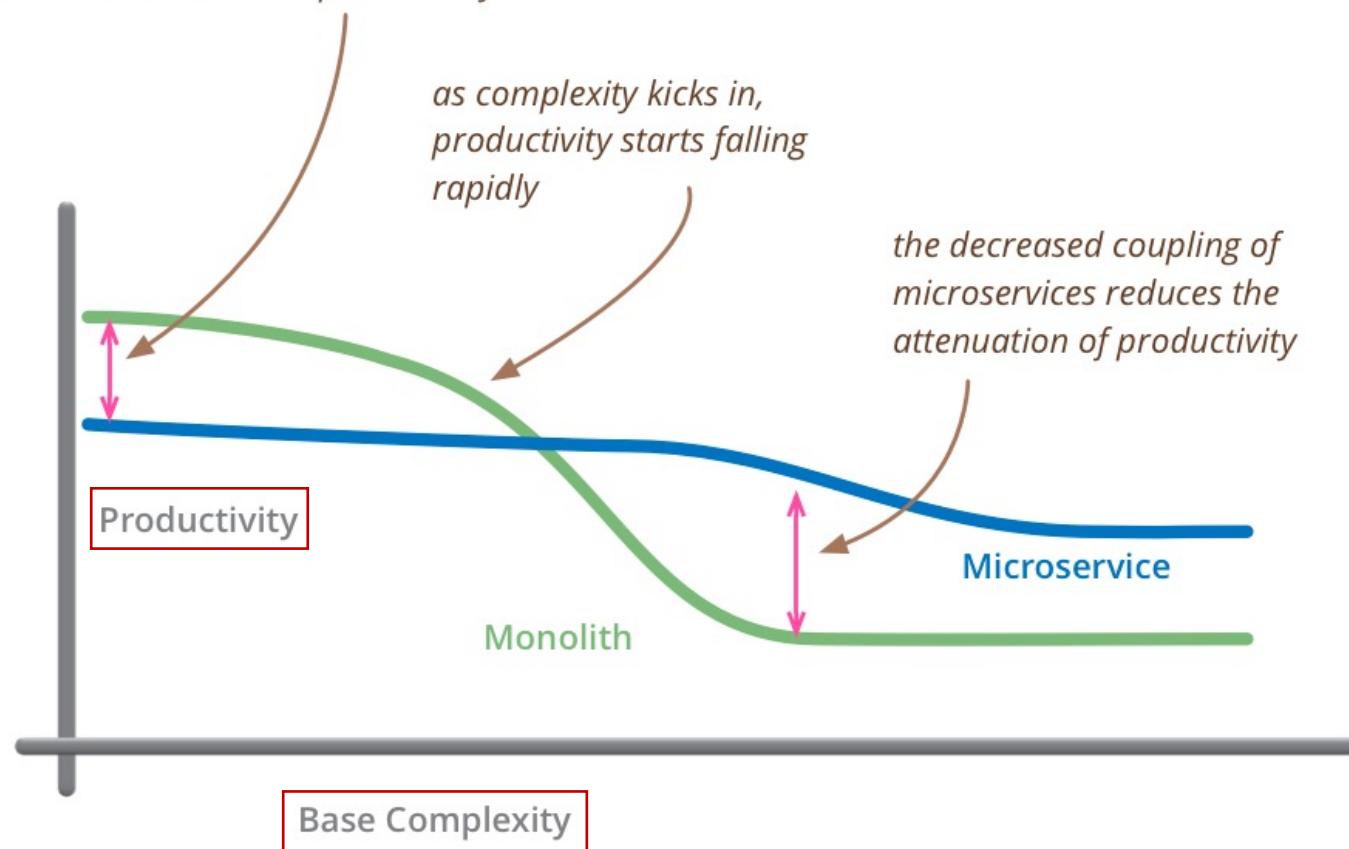
Microservices

- Building applications as suite of small and easy to replace services
 - fine grained, one functionality per service (sometimes 3-5 classes)
 - composable
 - easy to develop, test, and understand
 - fast (re)start, fault isolation
 - modelled around business domain
- Interplay of different systems and languages
- Easily deployable and replicable
- Embrace automation, embrace faults
- Highly observable

Are microservices always the right choice?

Microservices overhead

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



Microservice challenges

- Complexities of distributed systems
 - network latency, faults, inconsistencies
 - testing challenges
- Resource overhead, RPCs
 - Requires more thoughtful design (avoid "chatty" APIs, be more coarse-grained)
- Shifting complexities to the network
- Operational complexity
- Frequently adopted by breaking down monolithic application
- HTTP/REST/JSON communication
 - Schemas? Document API using Swagger, etc.



Taken to the extreme... Serverless (Functions-as-a-Service)

- Instead of writing minimal services, write just functions
- No state, rely completely on cloud storage or other cloud services
- Pay-per-invocation billing with elastic scalability
- Drawback: more ways things can fail, state is expensive
- Examples:
AWS lambda, CloudFlare workers, Azure Functions
- What might this be good for?

More in DevOps & Scaling

