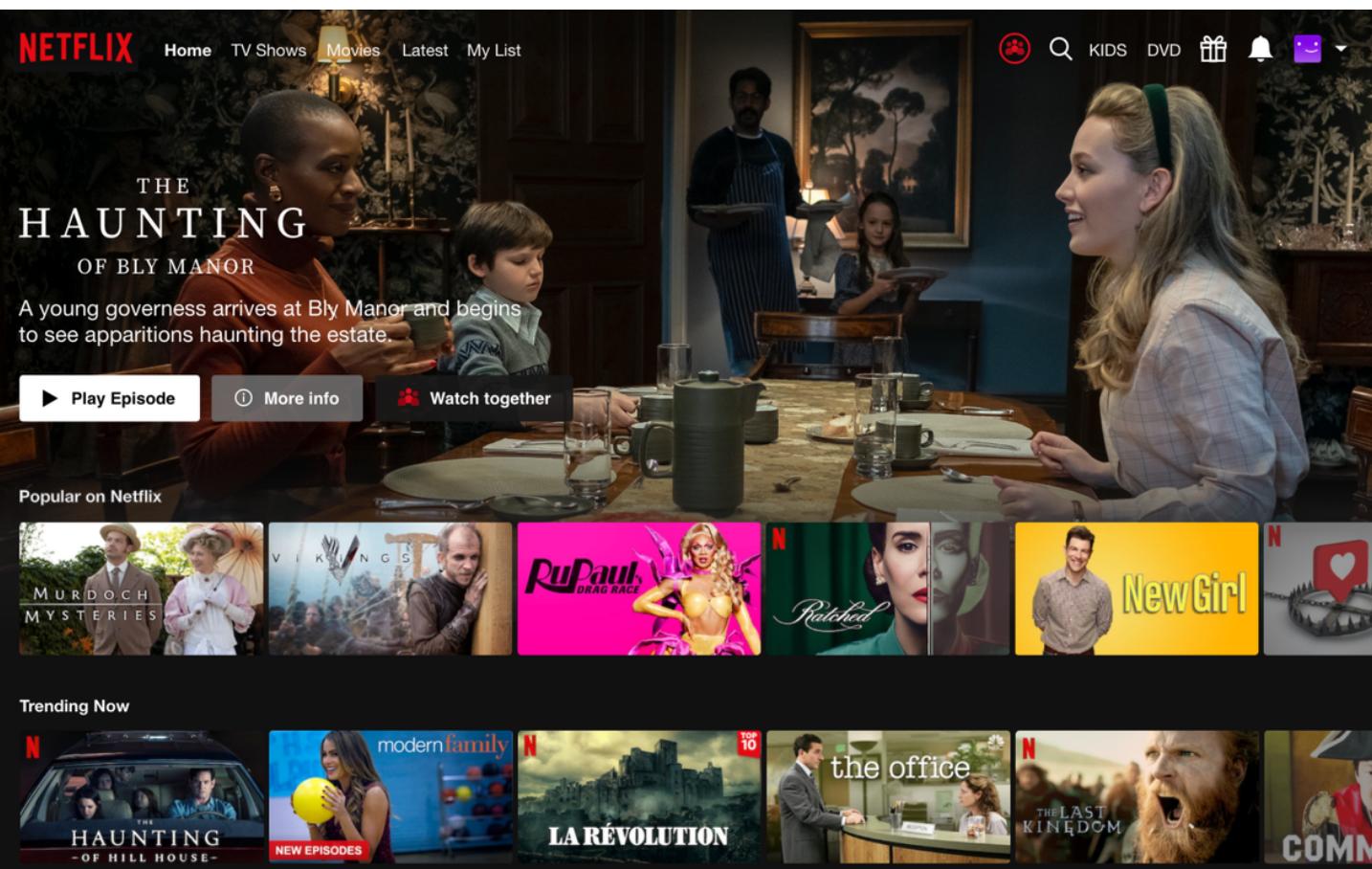
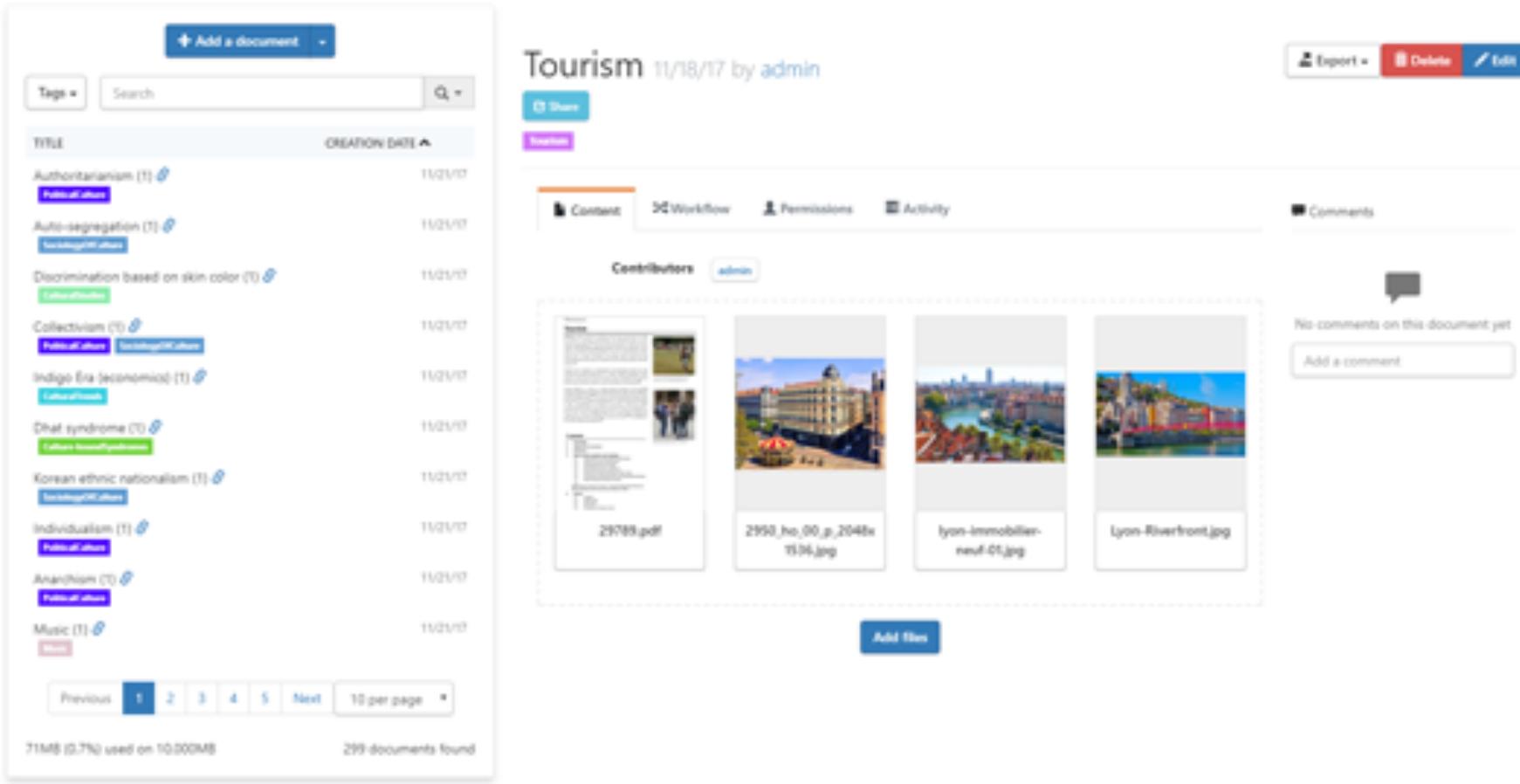


Monolithic Design vs. Microservices

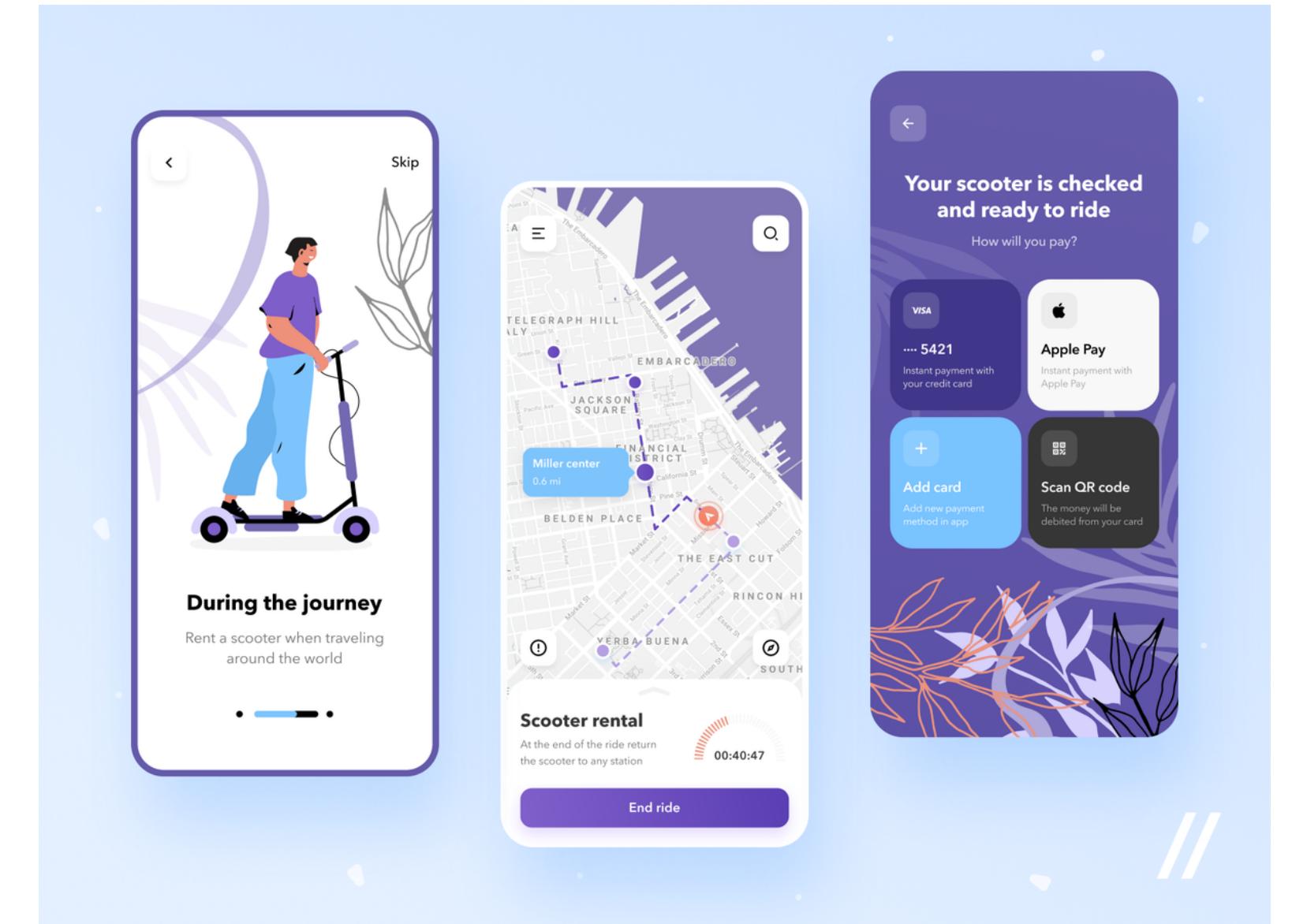
How might these apps be architected?



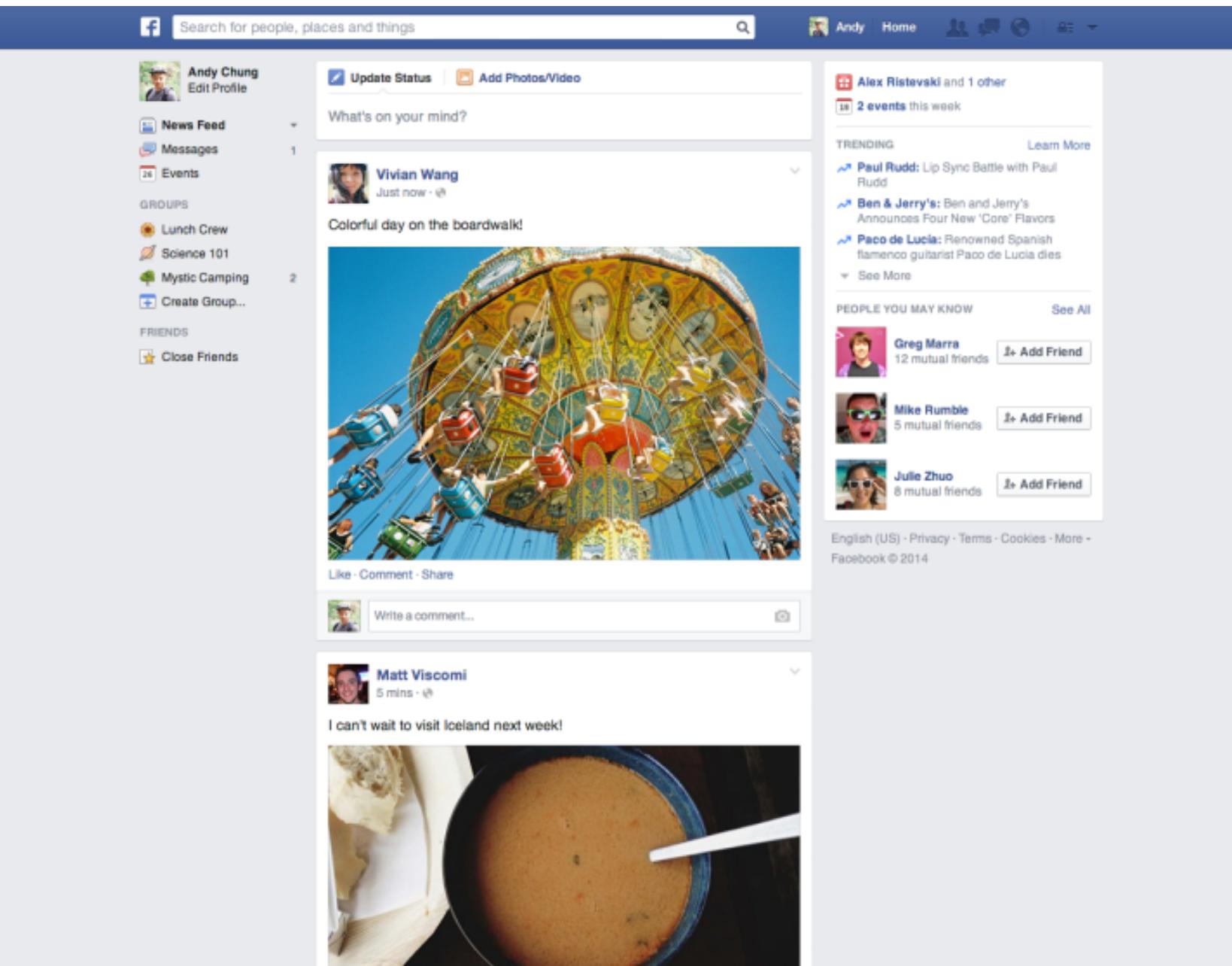
The Netflix homepage for the series 'The Haunting of Bly Manor'. It features a main banner with a woman in a governess costume. Below the banner are sections for 'Popular on Netflix' and 'Trending Now'.



A screenshot of a document management system interface titled 'Tourism'. It shows a list of documents with titles like 'Authoritarianism', 'Auto-segregation', 'Discrimination based on skin color', 'Collectivism', 'Indigo Era (economical)', 'Dhat syndrome', 'Korean ethnic nationalism', 'Individualism', 'Anarchism', and 'Music'. Below the list are preview thumbnails for files named '29789.pdf', '2998_no_90_p_2048w_7534.jpg', 'lyon-immobilier-neuf-ct.jpg', and 'Lyon-Riverfront.jpg'. At the bottom, there are navigation links for 'Previous', 'Next', and '10 per page'.

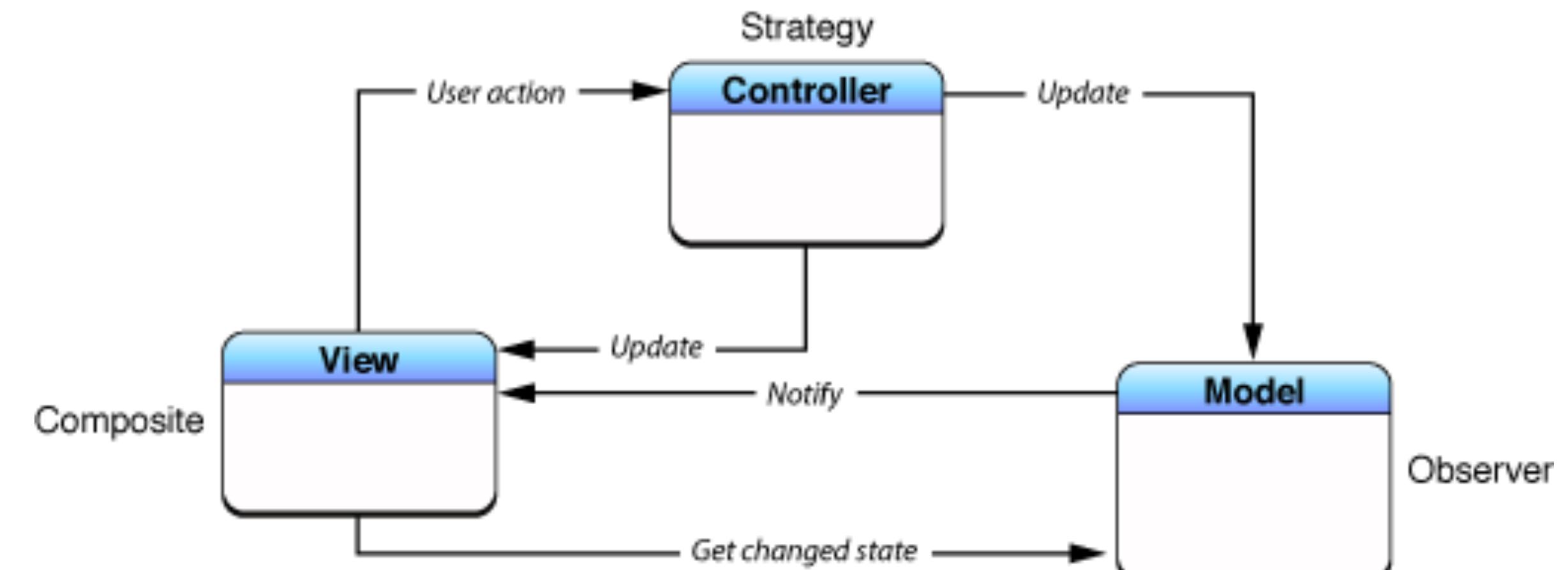
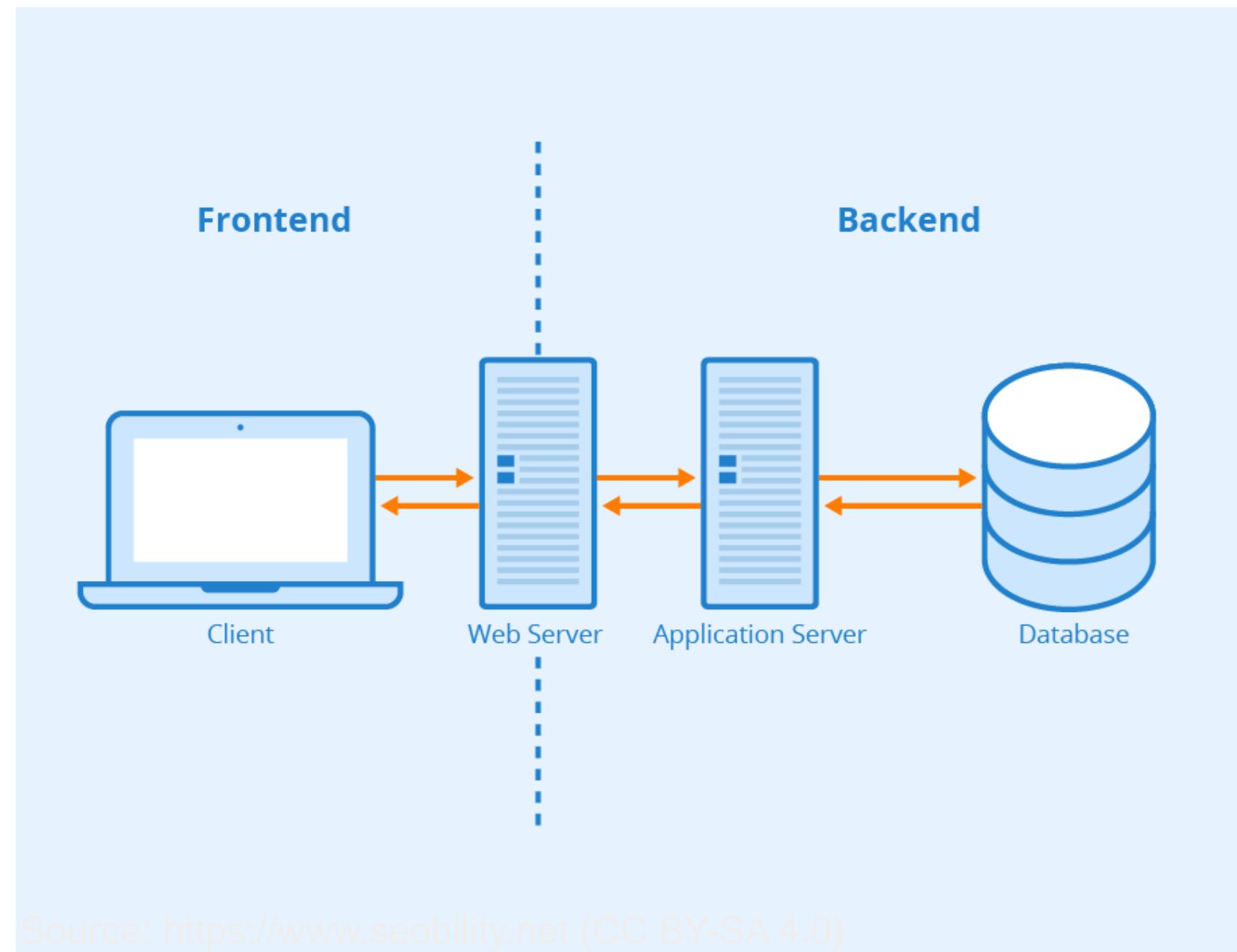


Three screenshots of a mobile application for scooter rental. The first screen shows a person riding a scooter with the text 'During the journey'. The second screen is a map showing a route through San Francisco neighborhoods like Telegraph Hill, Financial District, and Rincon Hill. The third screen shows payment options: Visa, Apple Pay, Add card, and Scan QR code.



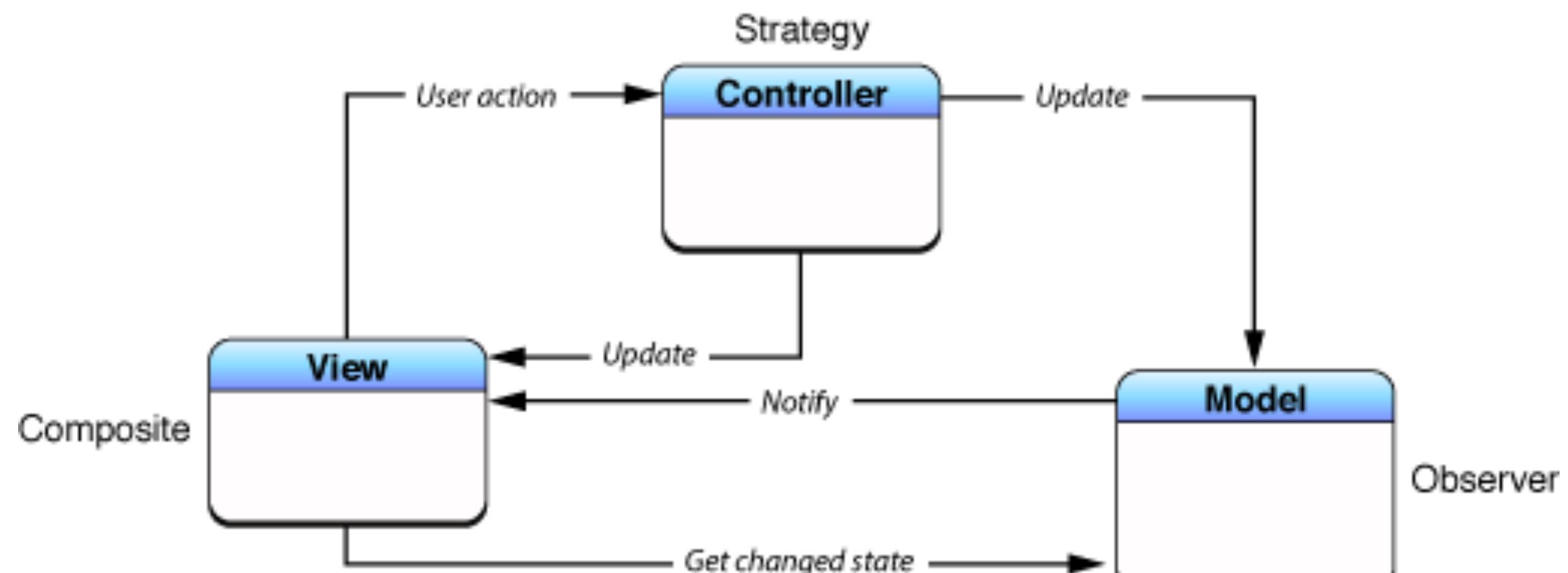
A screenshot of a Facebook news feed. It shows posts from users Vivian Wang ('Colorful day on the boardwalk!'), Matt Visconti ('I can't wait to visit Iceland next week!'), and others. The feed also includes news items about Paul Rudd, Ben & Jerry's, and Paco de Lucia.

Monolithic styles: Client-server or MVC



Brief digression: MVC (Model-View-Controller)

- Views:
 - Reusable views promote consistency
 - Modularity promotes reusability
- Model: separate to allow representation independence
- Controller: "business logic"; very custom



Monoliths make trade-offs on software quality

Several consequences of this architecture on:

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

The image contains three screenshots of monolithic software interfaces:

- Screenshot 1 (Top): Facebook News Feed.** Shows a user's news feed with posts from friends like Andy Chung, Vivian Wang, and Matt Visconti, along with trending news items about Paul Rudd, Ben & Jerry's, and Paco de Lucia.
- Screenshot 2 (Bottom Left): Sismics Docs search results.** Shows a search interface for documents, listing results for terms like "Authoritarianism", "Auto-segregation", "Discrimination based on skin color", "Collectivism", "Indigo Era (economical)", "Dhat syndrome", "Korean ethnic nationalism", "Individualism", "Anarchism", and "Music".
- Screenshot 3 (Bottom Right): Sismics Docs document view.** Shows a detailed view of a document titled "Tourism" created by "admin" on 11/18/17. The view includes tabs for Content, Workflow, Permissions, and Activity, and displays four images related to tourism.

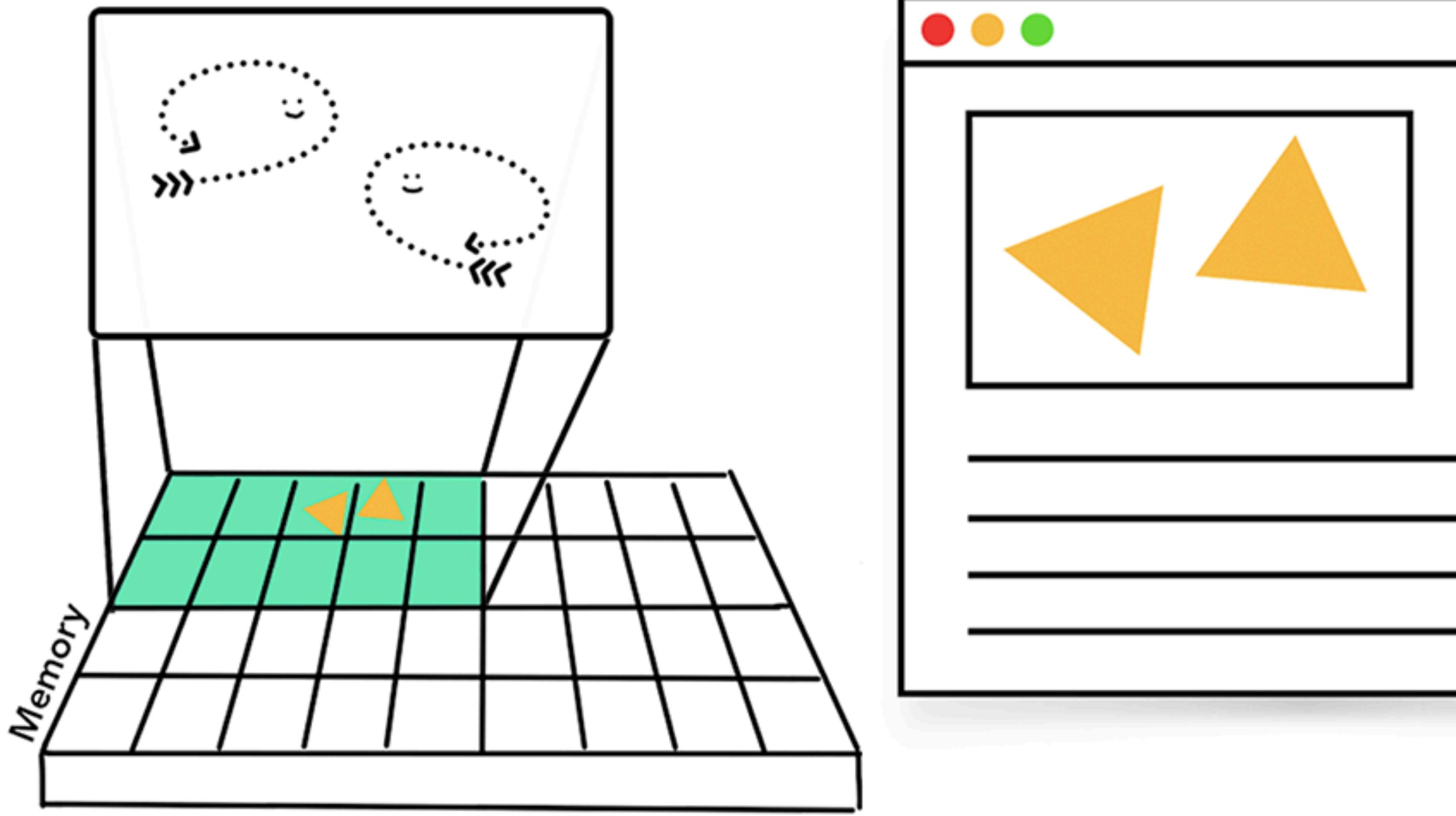
Service-based architecture – Chrome

Web Browsers



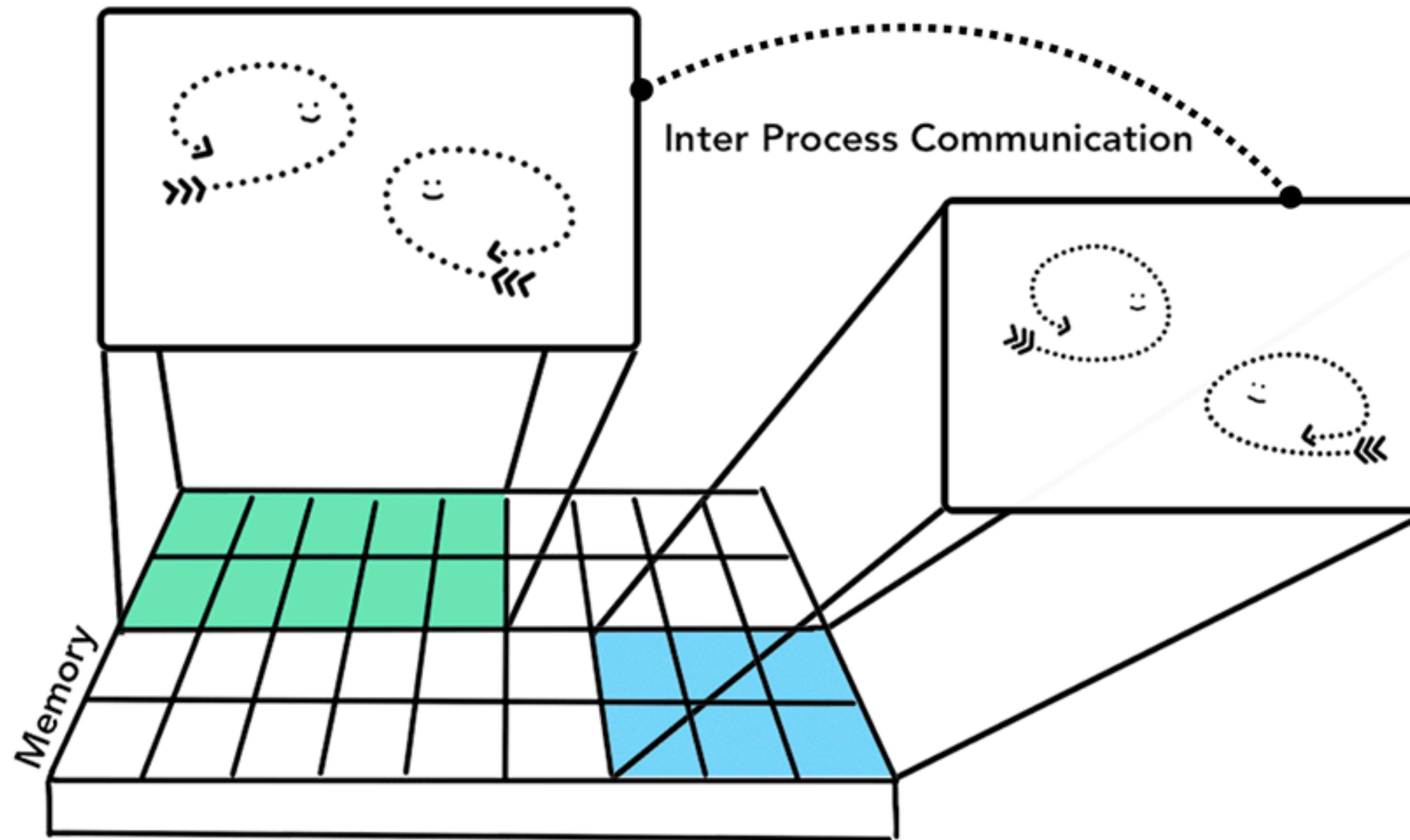
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Browser: A multi-threaded process



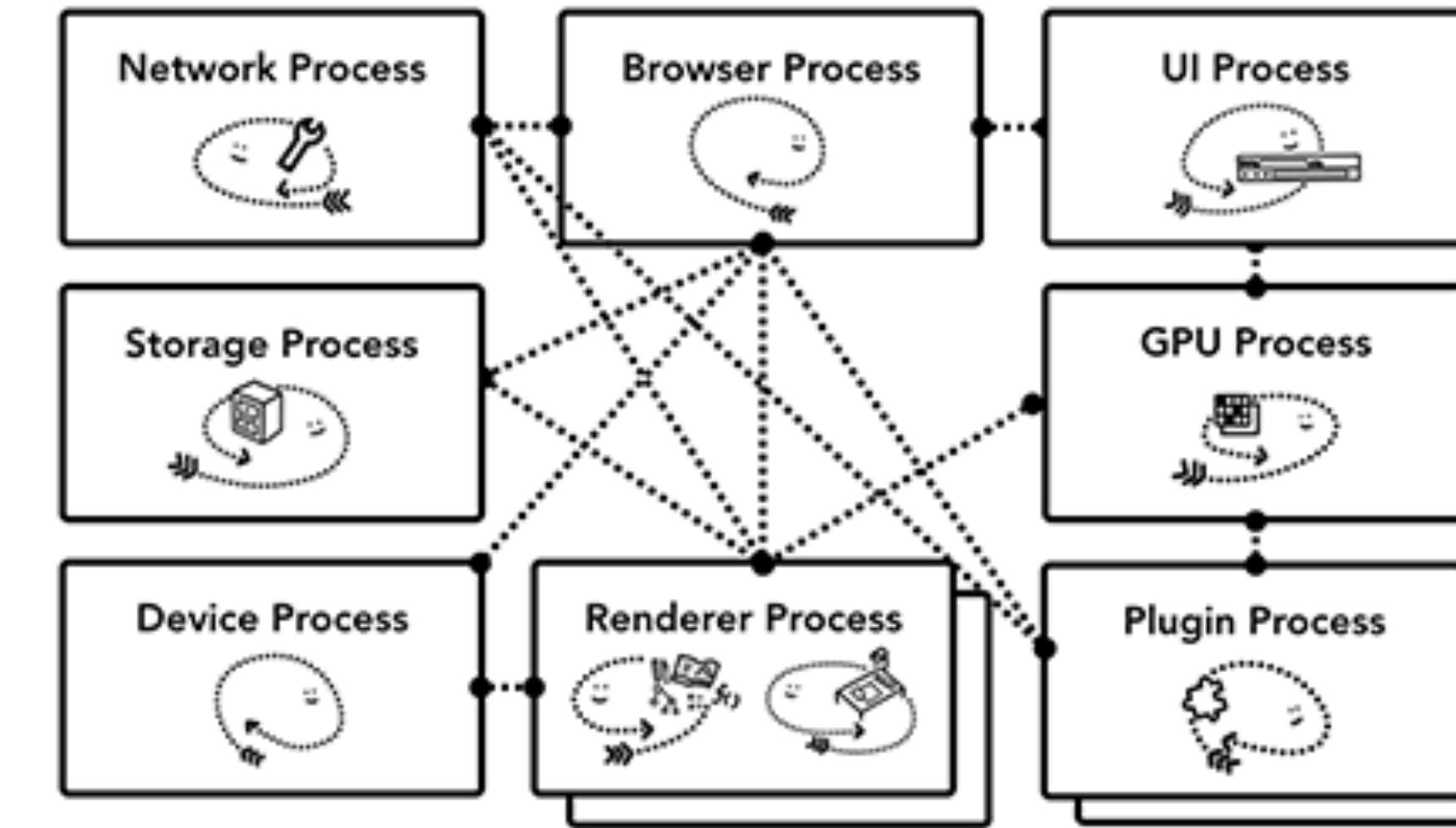
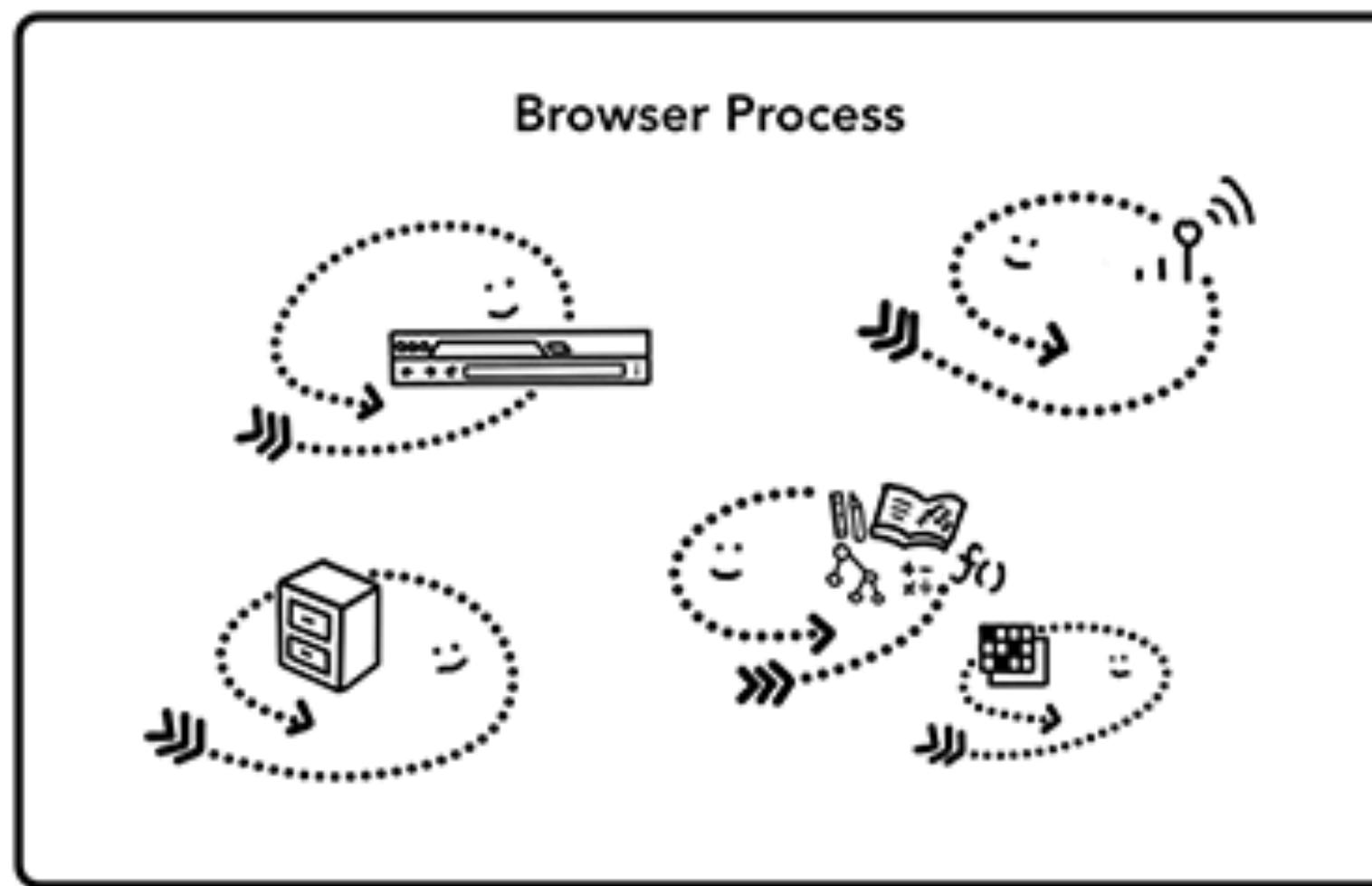
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Multi-process browser with IPC



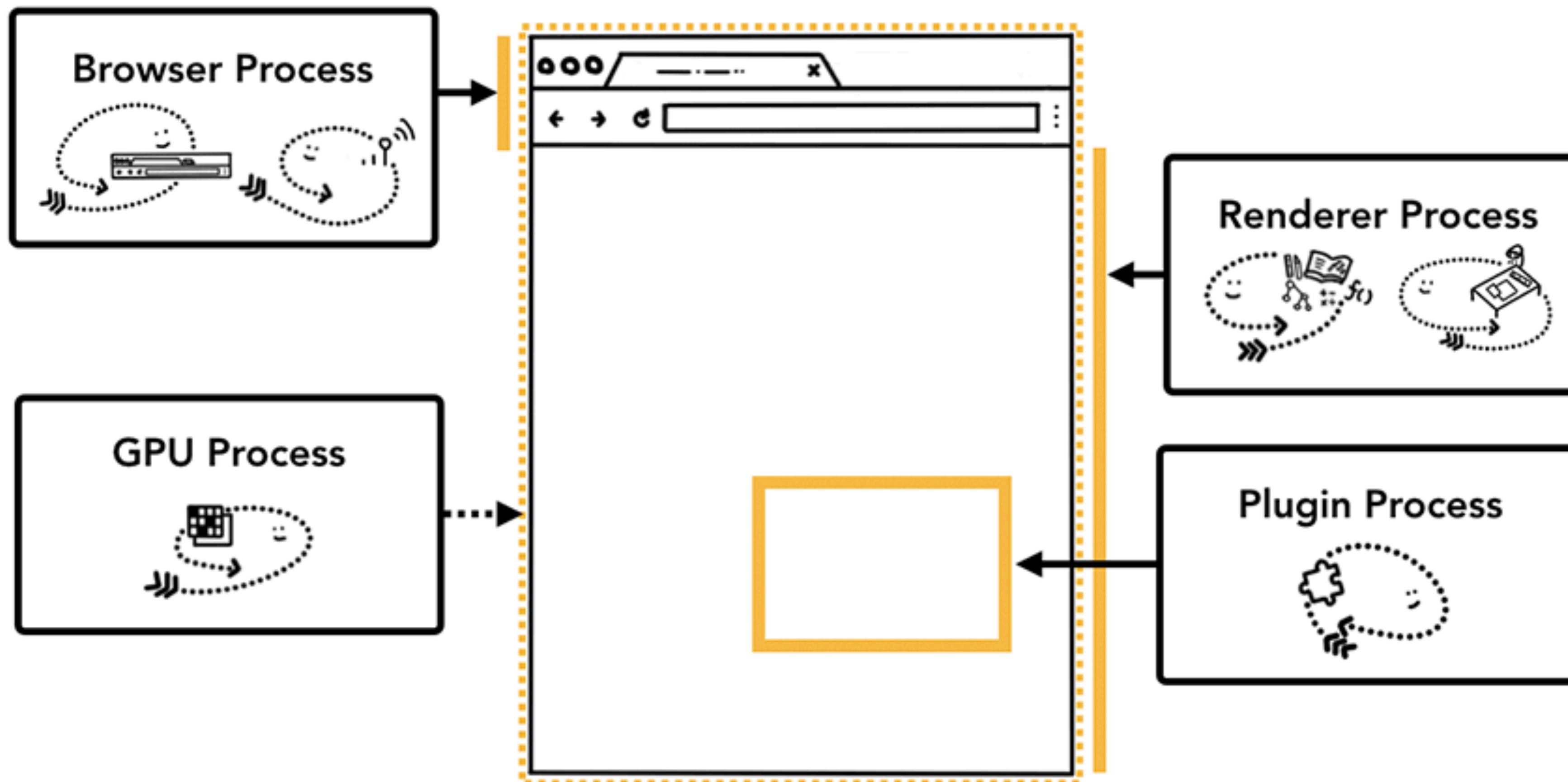
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Browser Architectures



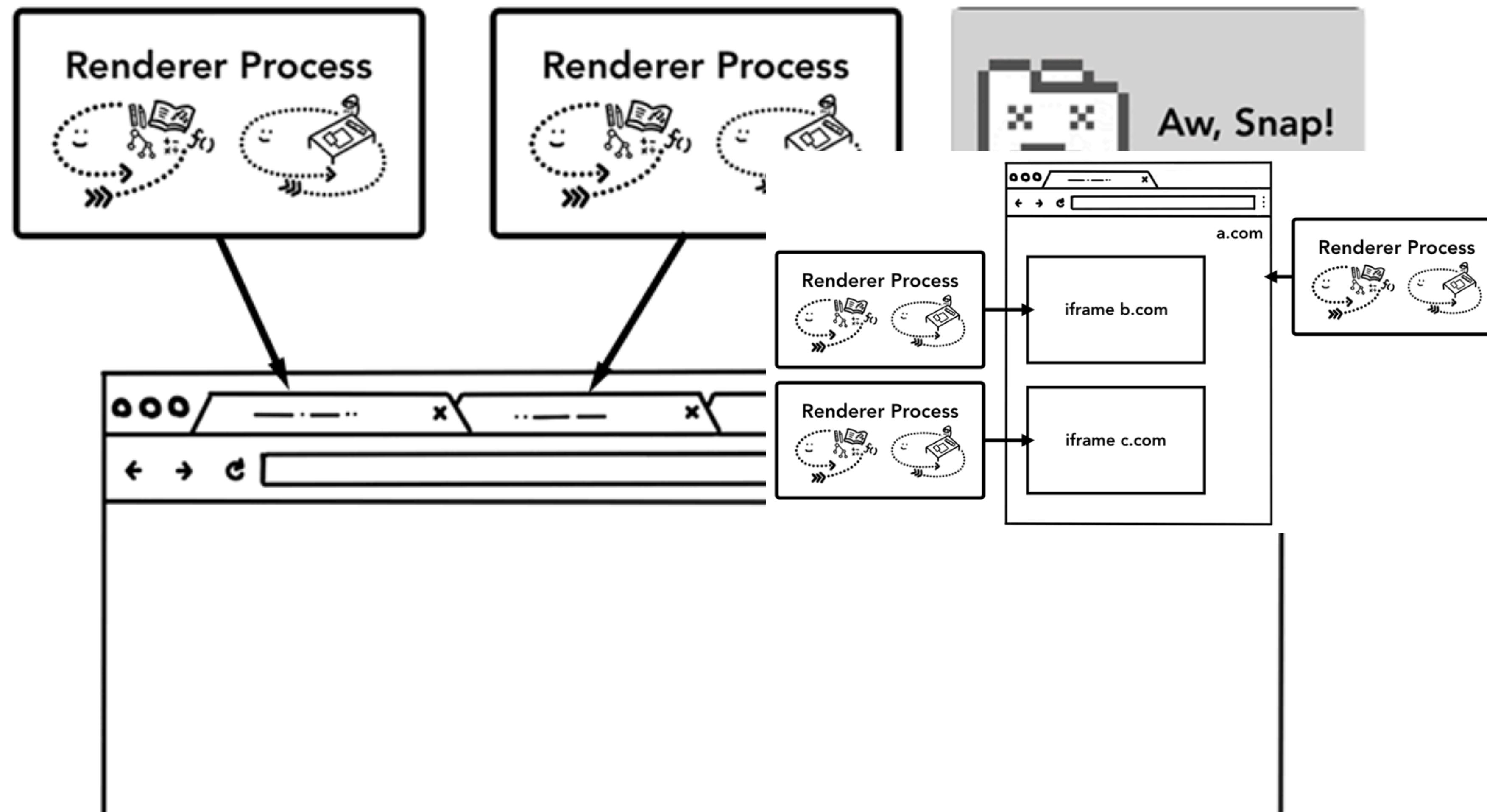
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture



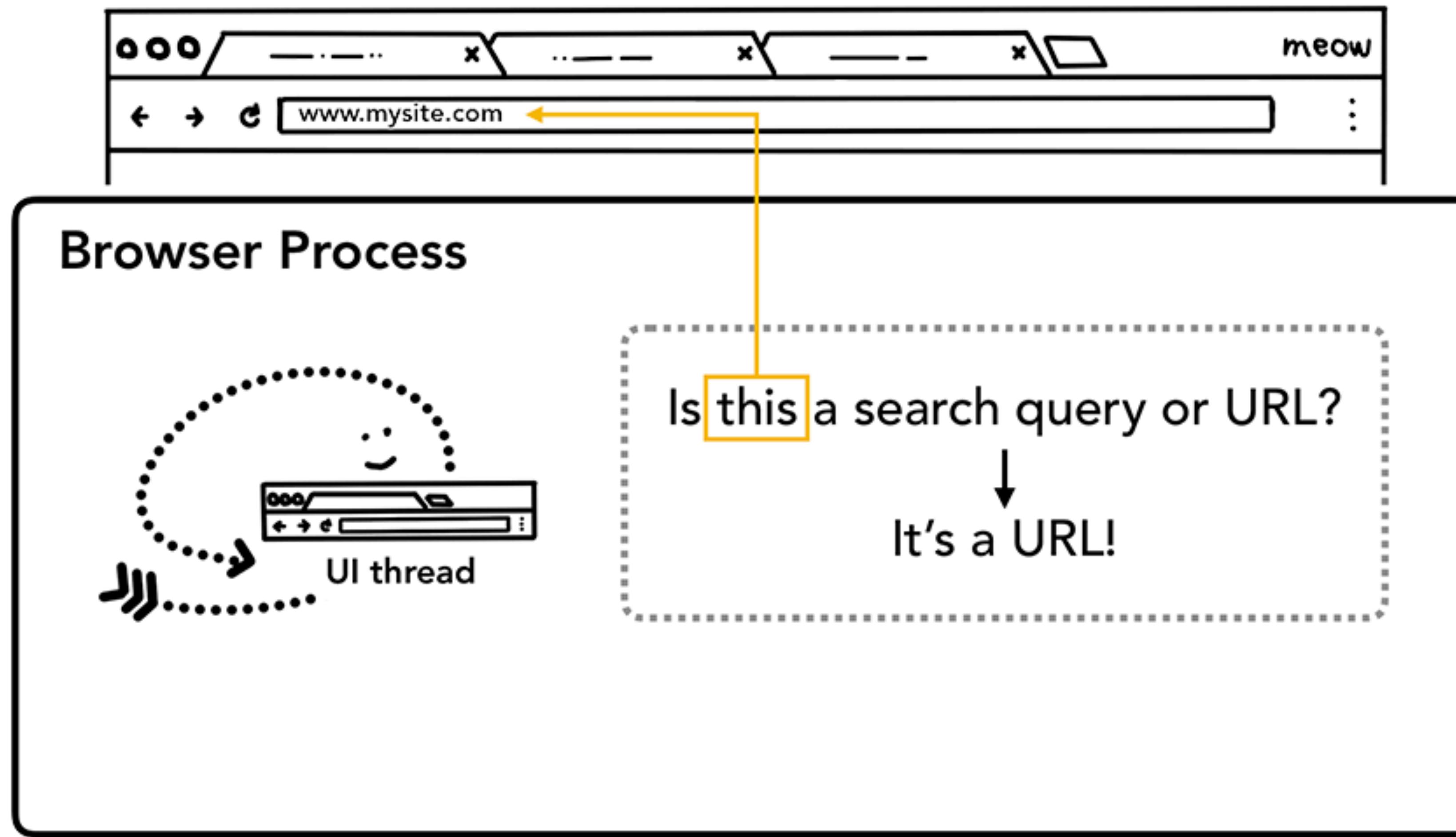
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Service-based browser architecture

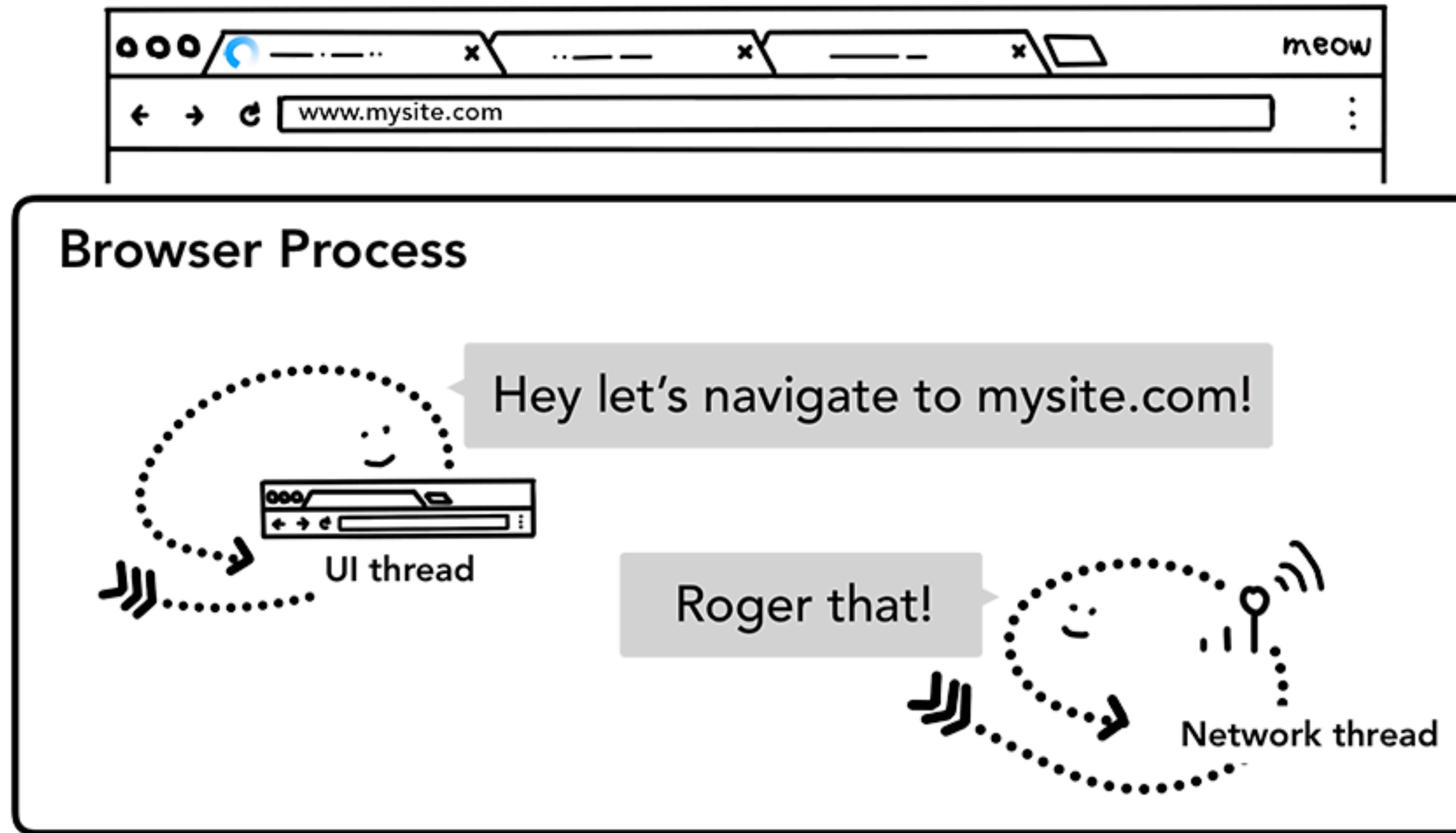


Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

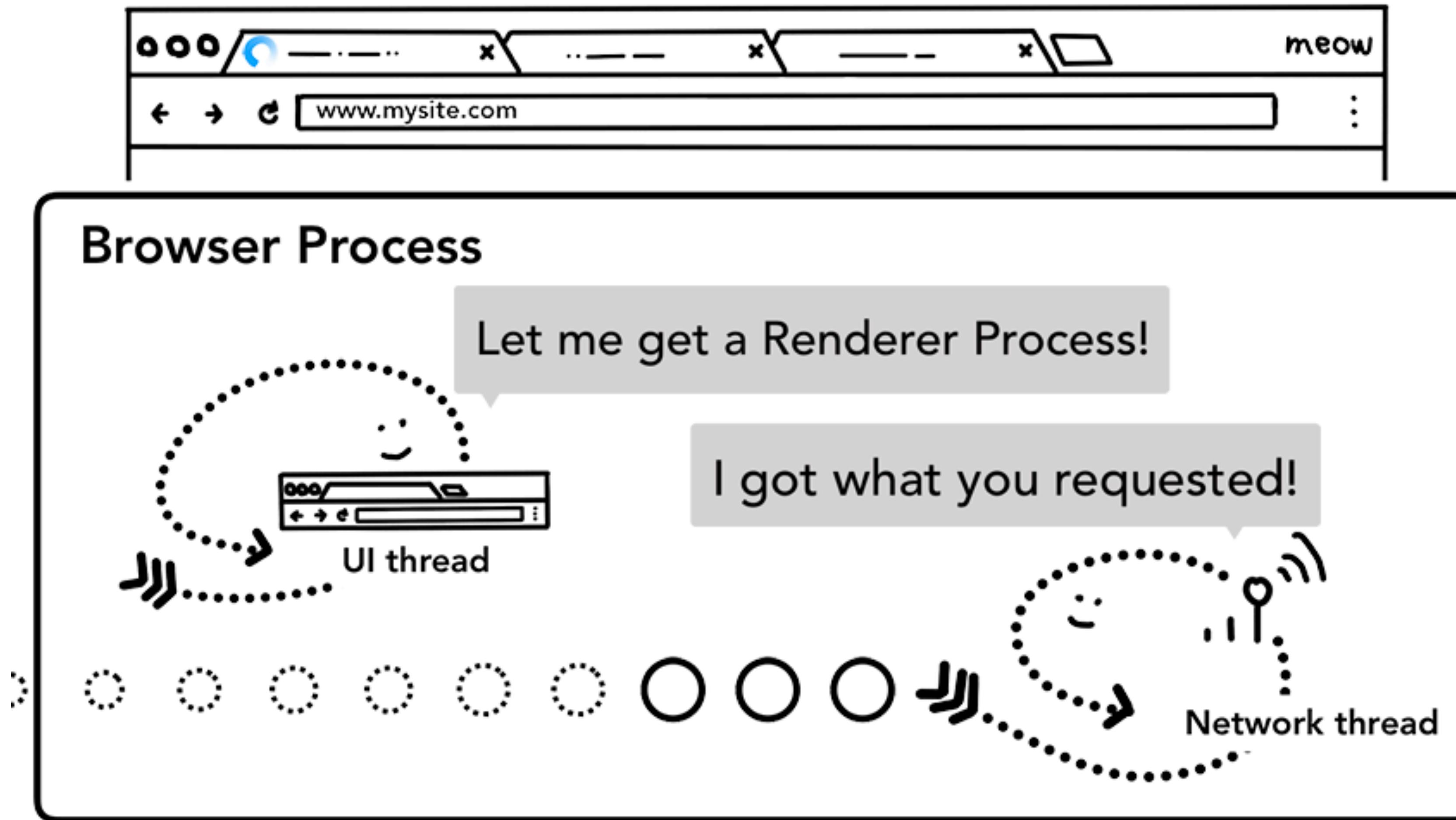
Navigating to a web site uses service requests



Navigating to a web site uses service requests

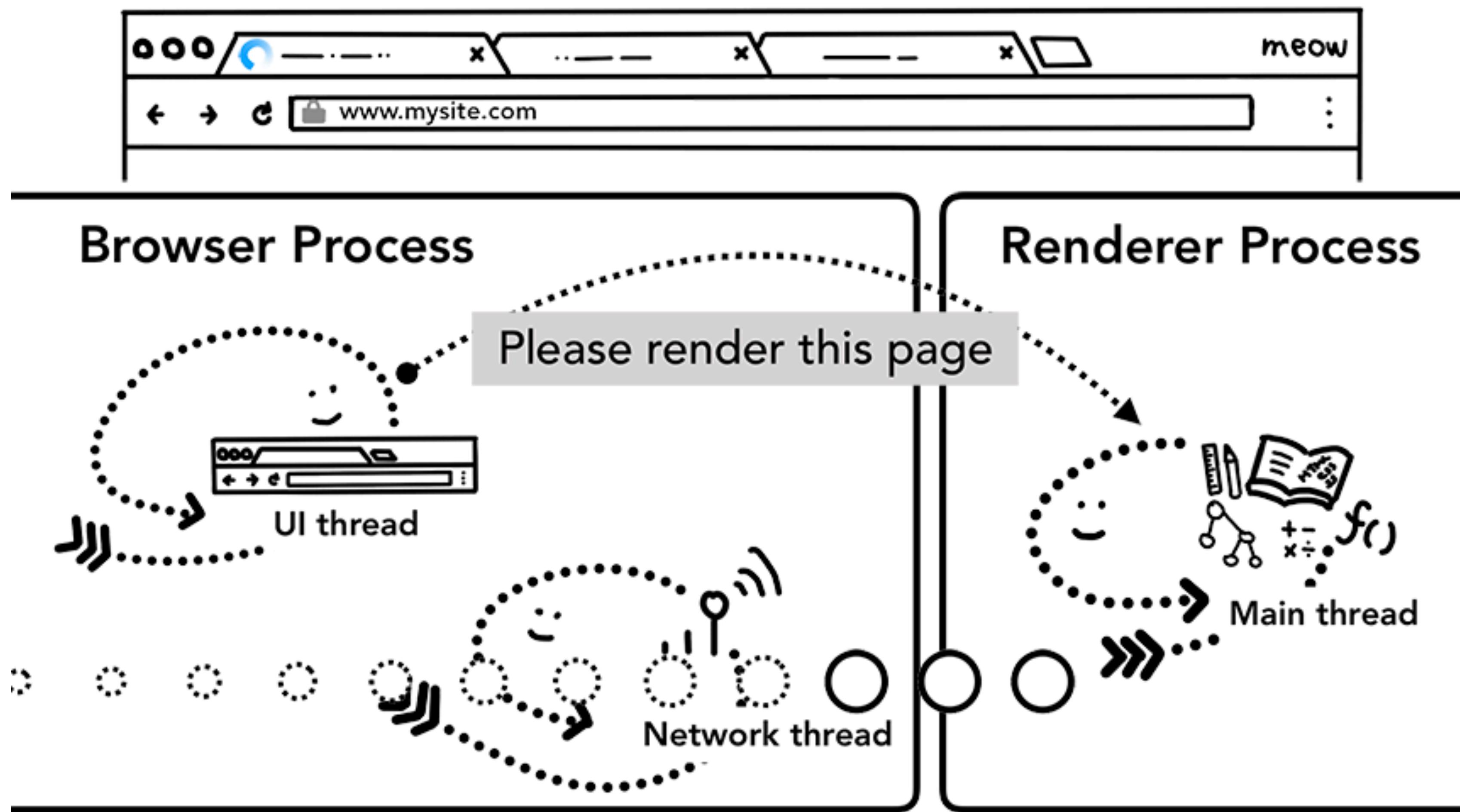


Navigating to a web site uses service requests



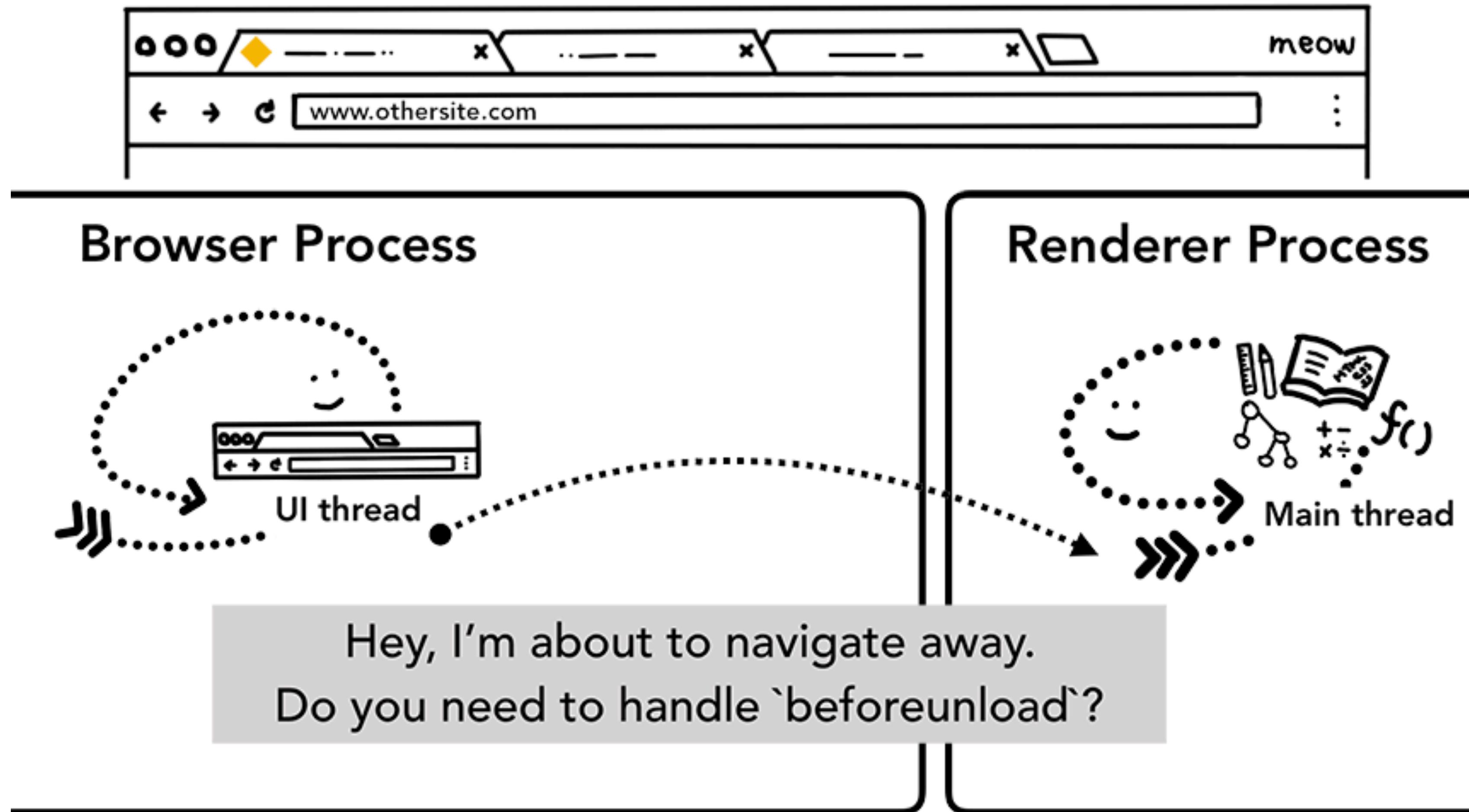
Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests

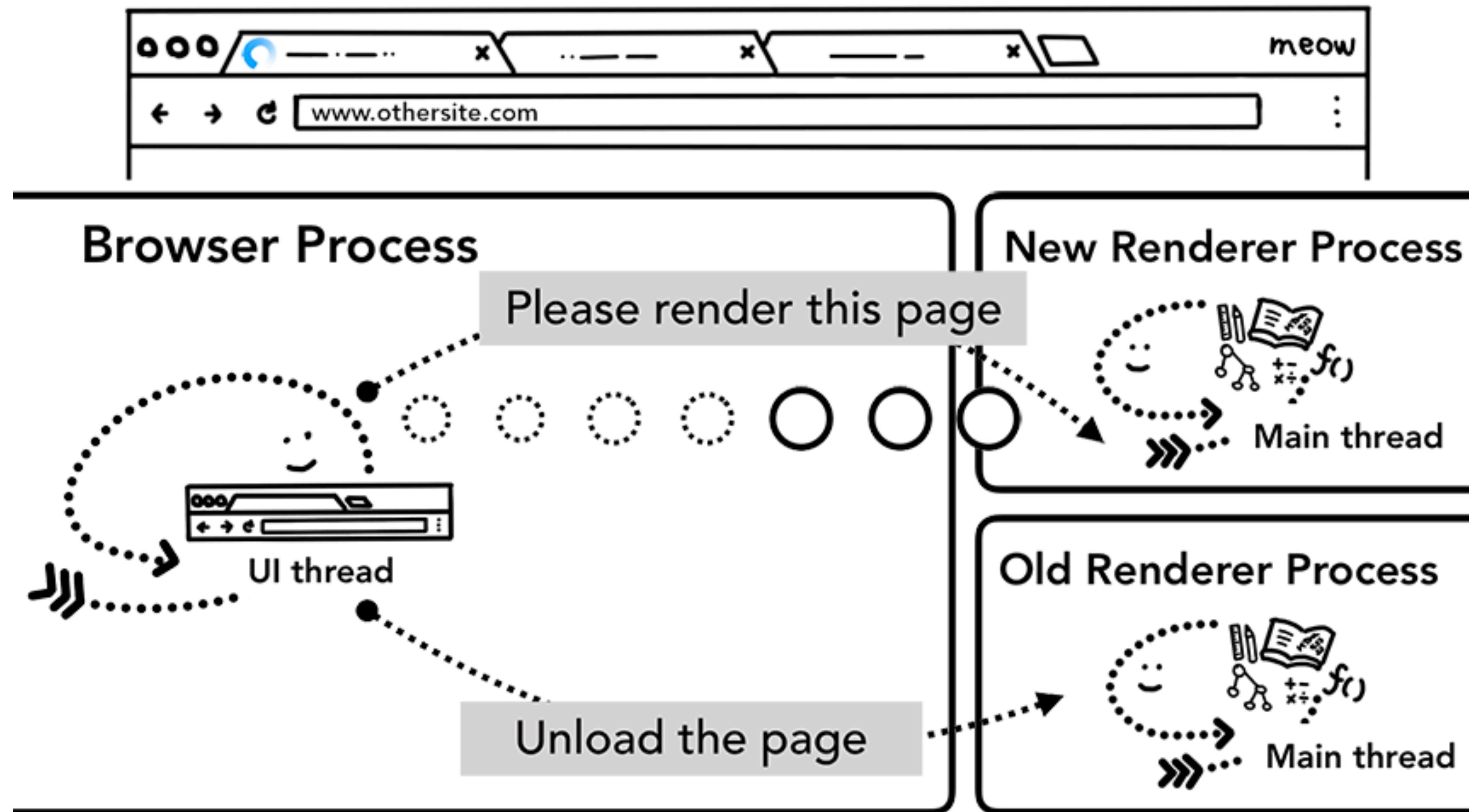


Source: <https://developers.google.com/web/updates/2018/09/inside-browser-part1> (CC BY 4.0)

Navigating to a web site uses service requests

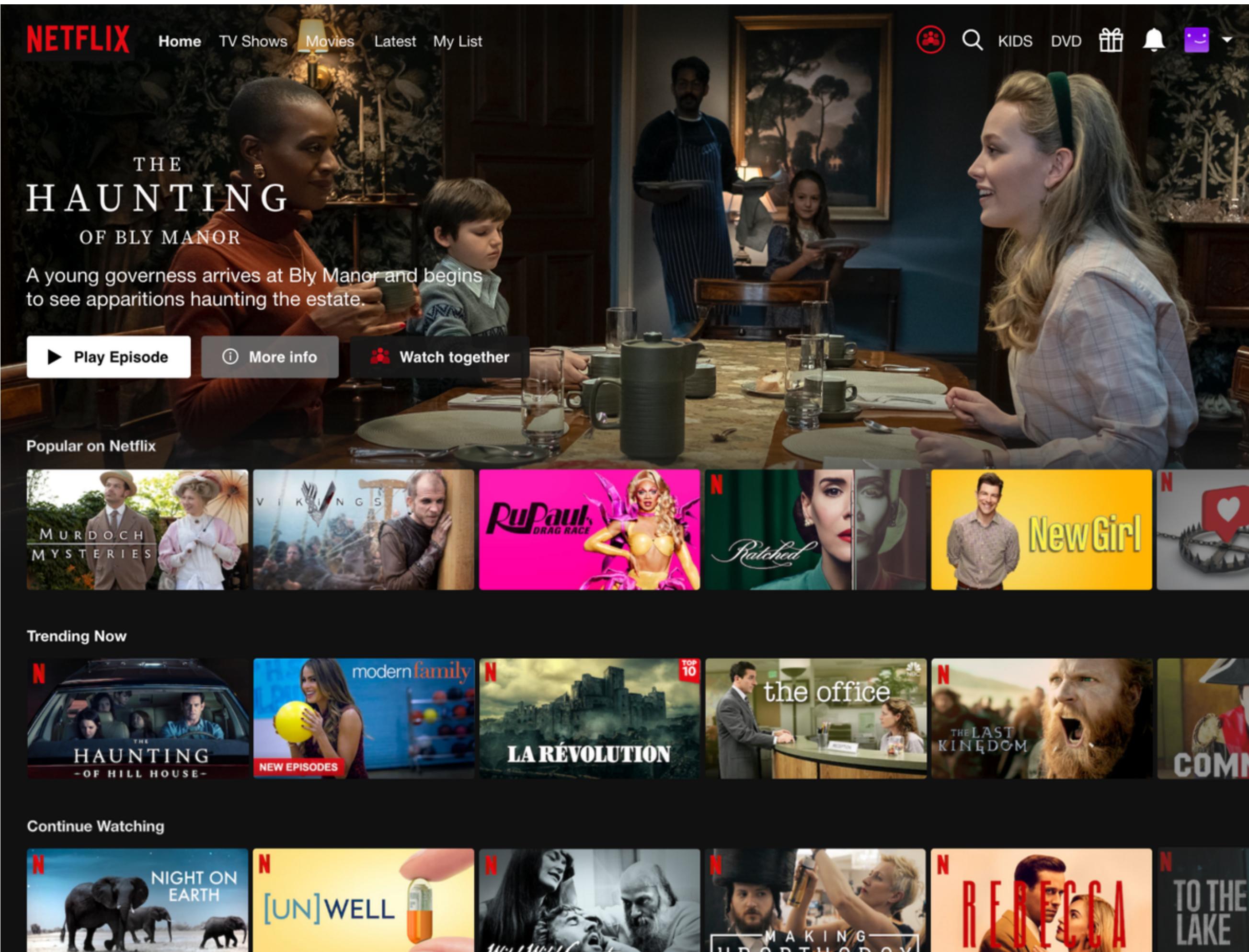


Navigating to a web site uses service requests

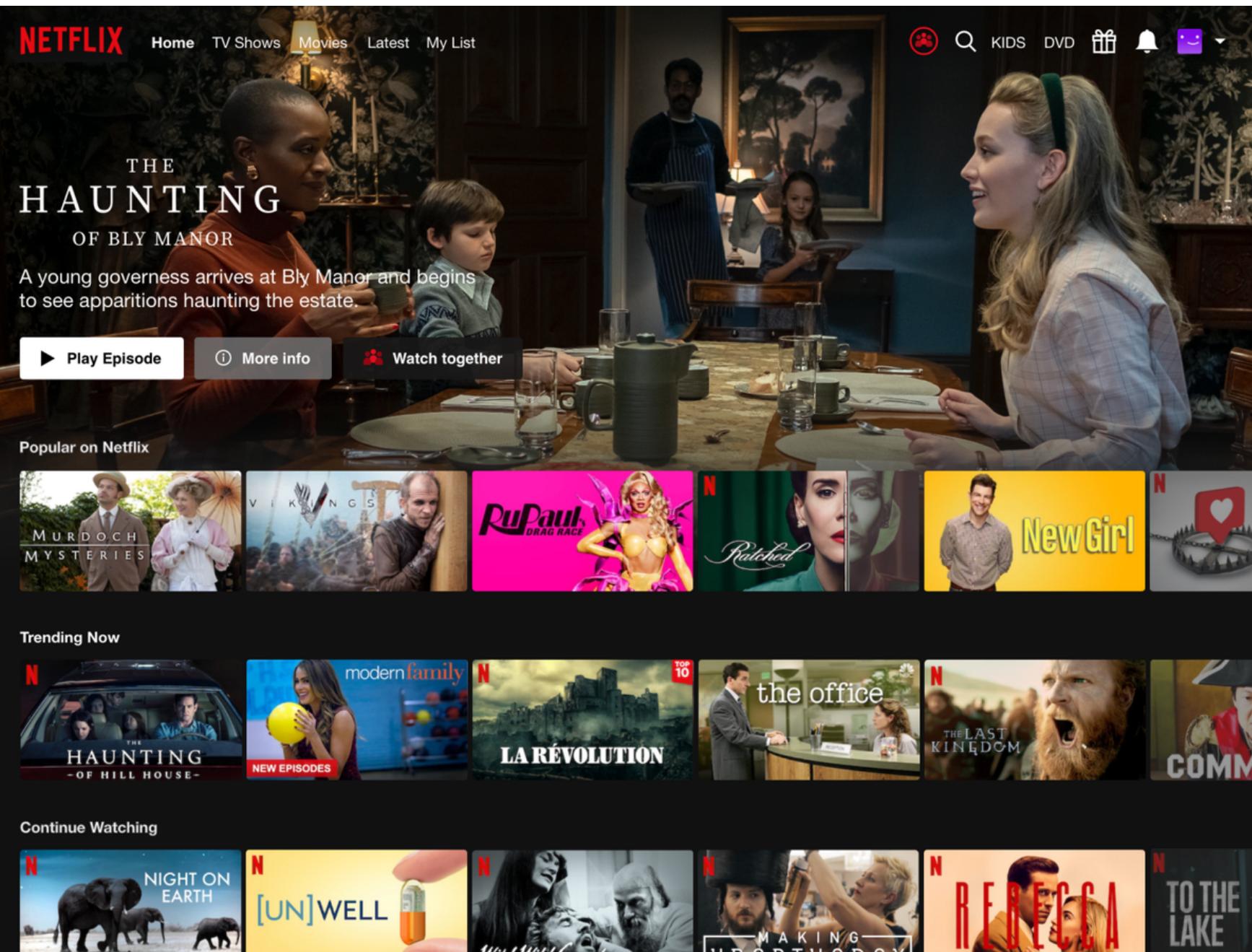


Microservice architecture - Netflix

Netflix



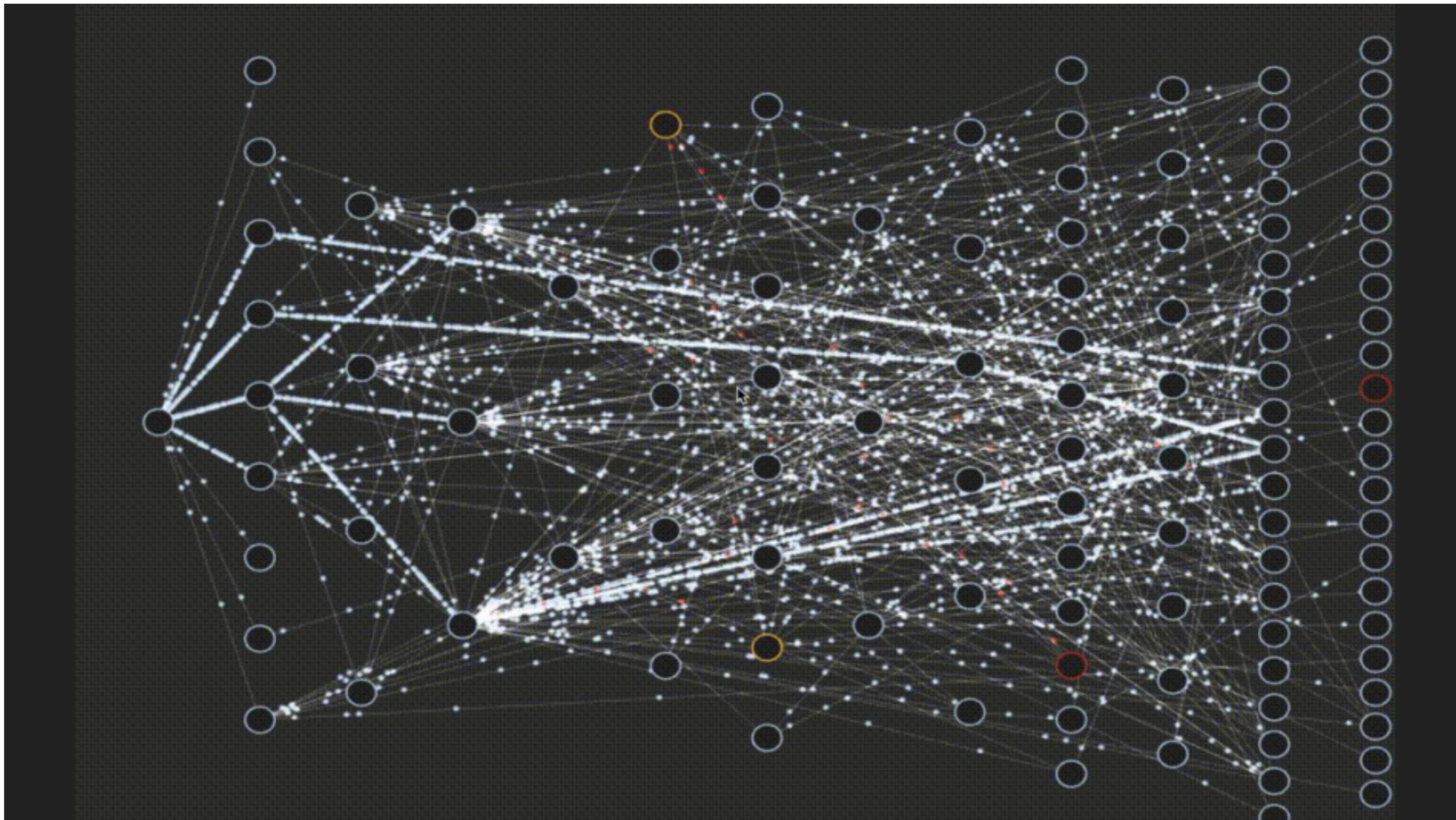
Netflix Microservices – App Boot



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

(as of 2016)

Netflix Microservices – One Request



(as of 2016)

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

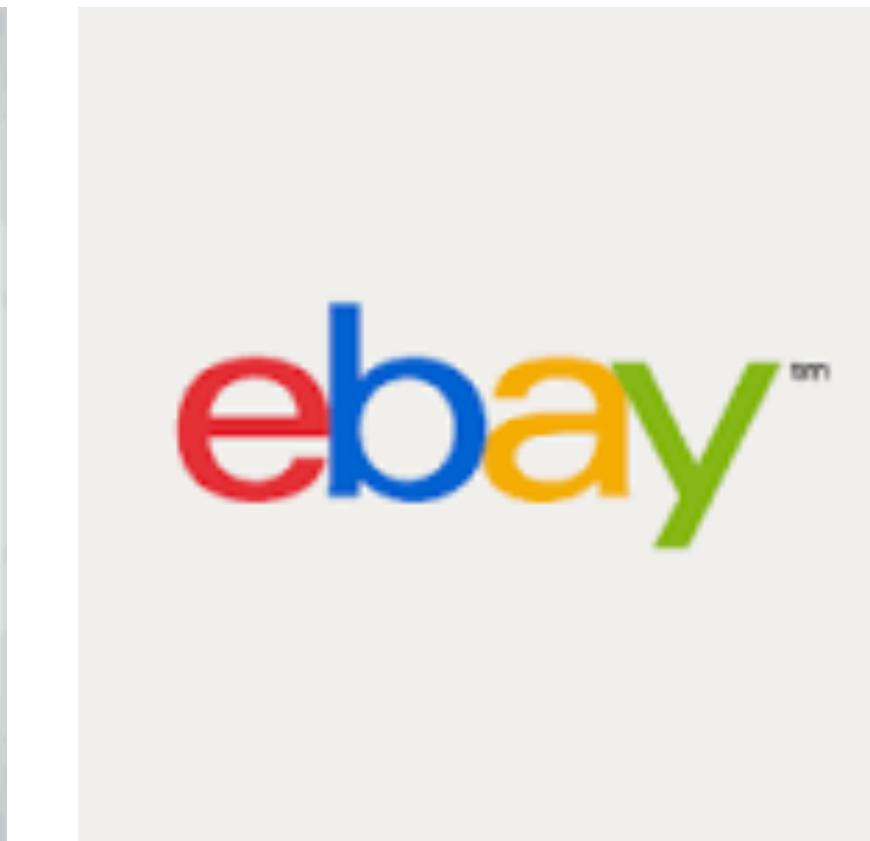
Who uses Microservices?



UBER



COMCAST



GROUPON®



Microservices – The Hipster Shop Example

Online Boutique: Guess some microservices

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

Left Screenshot (Homepage):

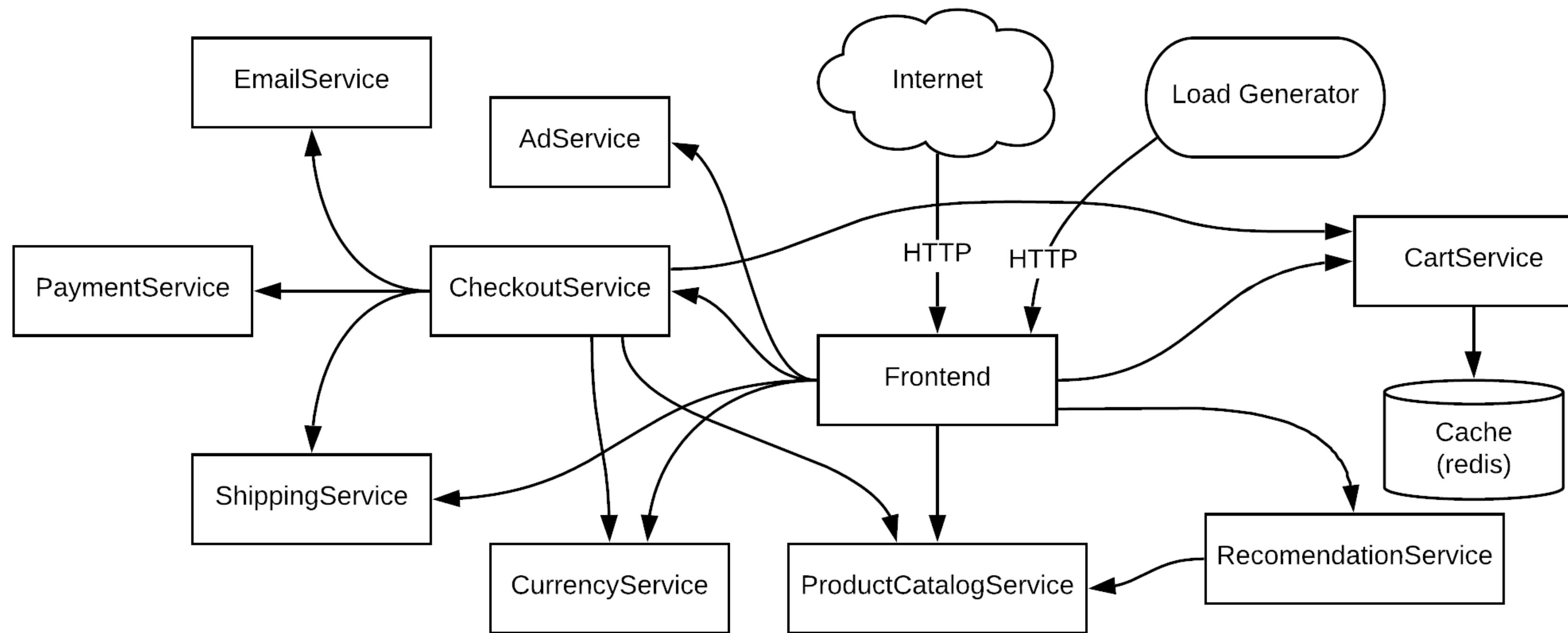
- Header: "Free shipping with \$75 purchase!"
- Logo: "ONLINEBOUTIQUE" with a circular icon.
- Cart: "Cart" button with a count of 0.
- Google Cloud badge.
- Banner: "Online BOUTIQUE EST. 2013" featuring a vintage typewriter.
- Section: "Hot PRODUCTS" with three items:
 - VINTAGE TYPEWRITER (TRY 368.34)
 - VINTAGE CAMERA LENS (TRY 67.66)
 - HOME BARISTA KIT (TRY 671.79)
- Bottom row: Three more products shown as thumbnails.

Right Screenshot (Shopping Cart Page):

- Header: "Free shipping with \$75 purchase!"
- Logo: "ONLINEBOUTIQUE" with a circular icon.
- Cart: "Cart" button with a count of 2.
- Google Cloud badge.
- Text: "2 items in your cart".
- Items:
 - Home Barista Kit (SKU: #1YMWNN1N40) - Quantity: 1, TRY 671.79
 - Vintage Camera Lens (SKU: #66VCHSJNUP) - Quantity: 5, TRY 338.33
- Text: "Shipping Cost: TRY 93.23" and "Total Cost: TRY 1103.36".
- Section: "Checkout".
- Form fields:
 - 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
 - CVV: ...
- Button: "PLACE ORDER" (green).

<https://onlineboutique.dev>

Online Boutique Microservice Architecture



<https://github.com/GoogleCloudPlatform/microservices-demo>

Microservices

What are the consequences of this architecture? On:

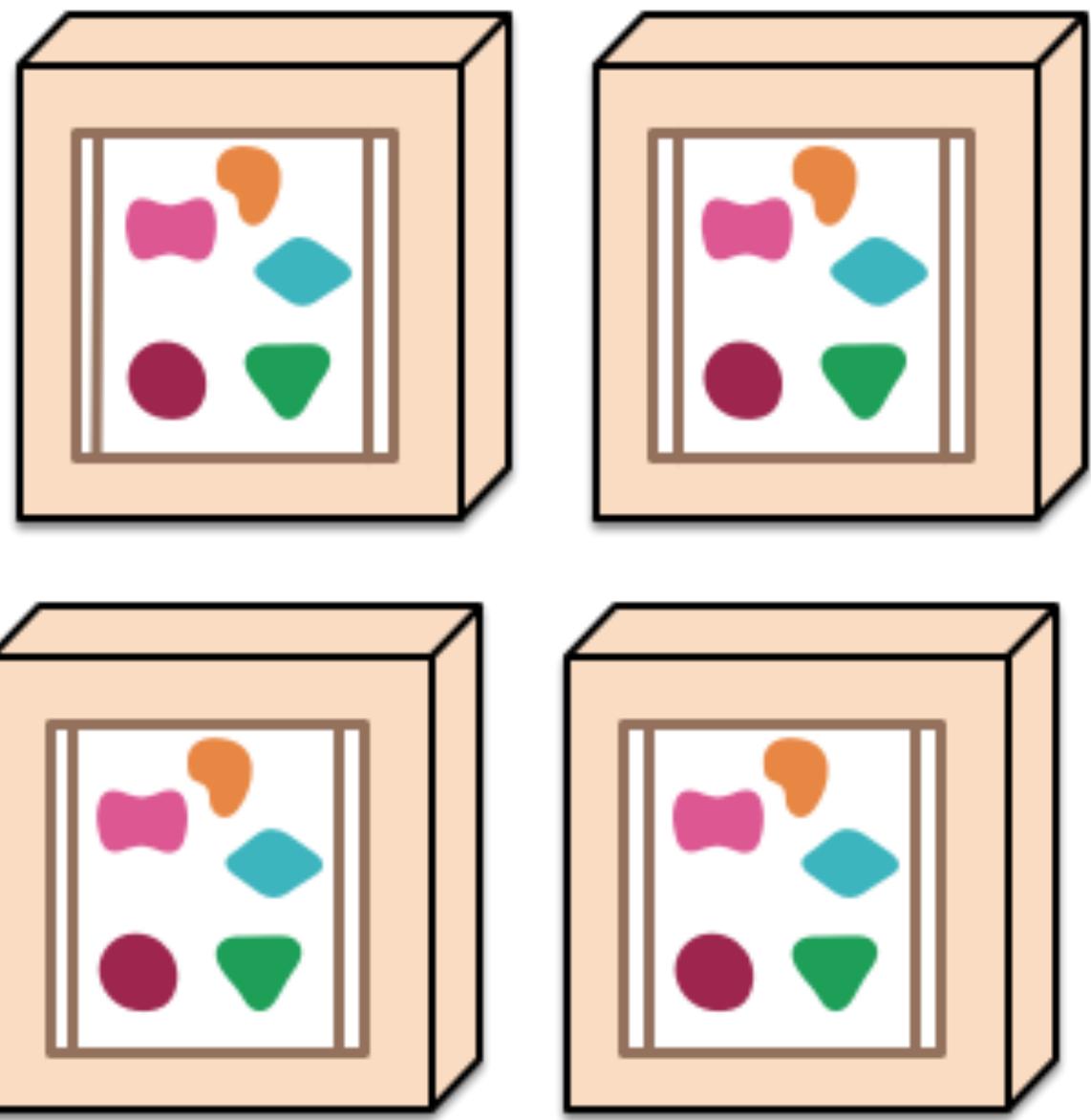
- Scalability
- Reliability
- Performance
- Development
- Maintainability
- Evolution
- Testability
- Ownership
- Data Consistency

Scalability

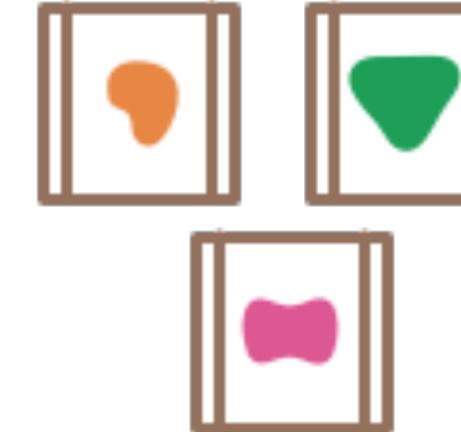
A monolithic application puts all its functionality into a single process...



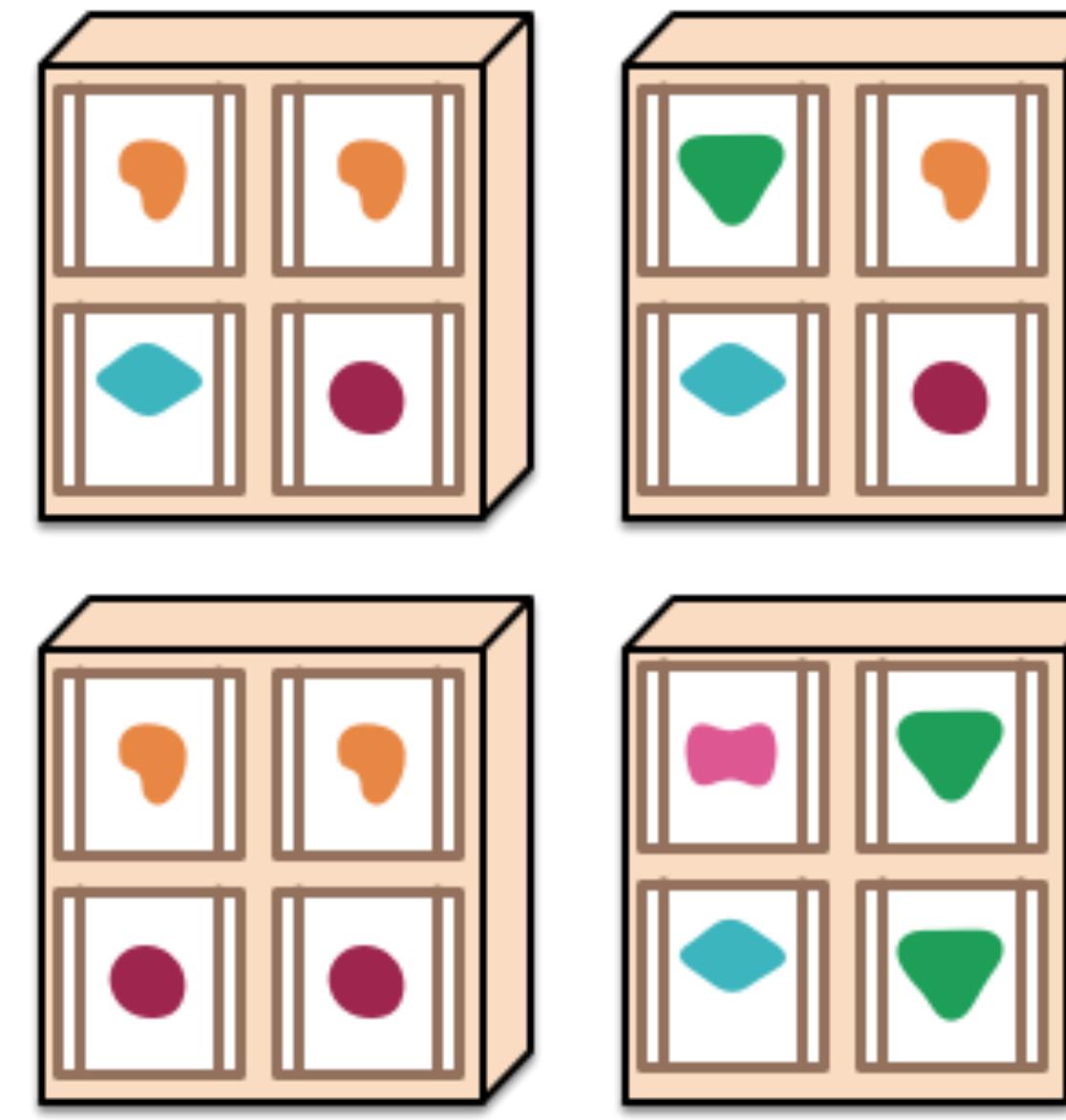
... and scales by replicating the monolith on multiple servers



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

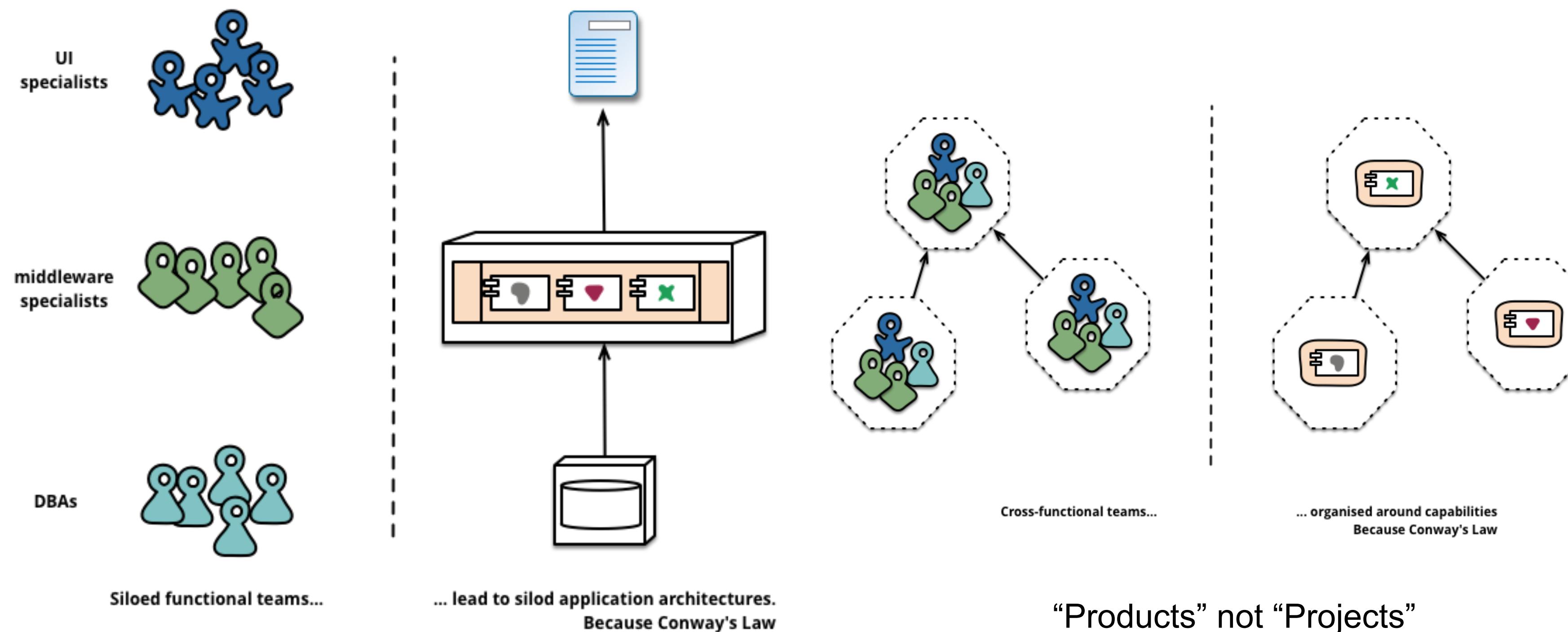


... and scales by distributing these services across servers, replicating as needed.



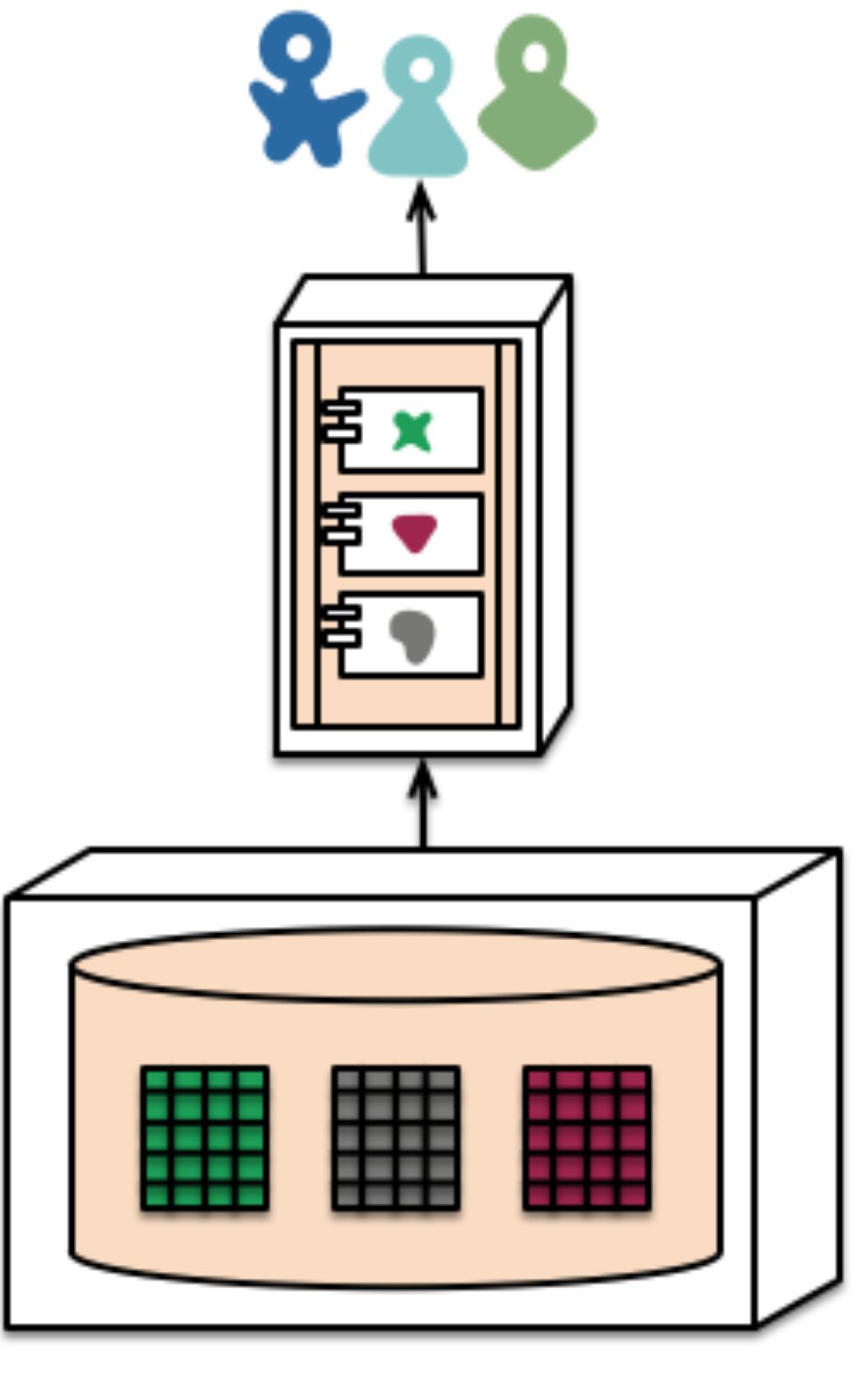
Source: <http://martinfowler.com/articles/microservices.html>

Team Organization (Conway's Law)

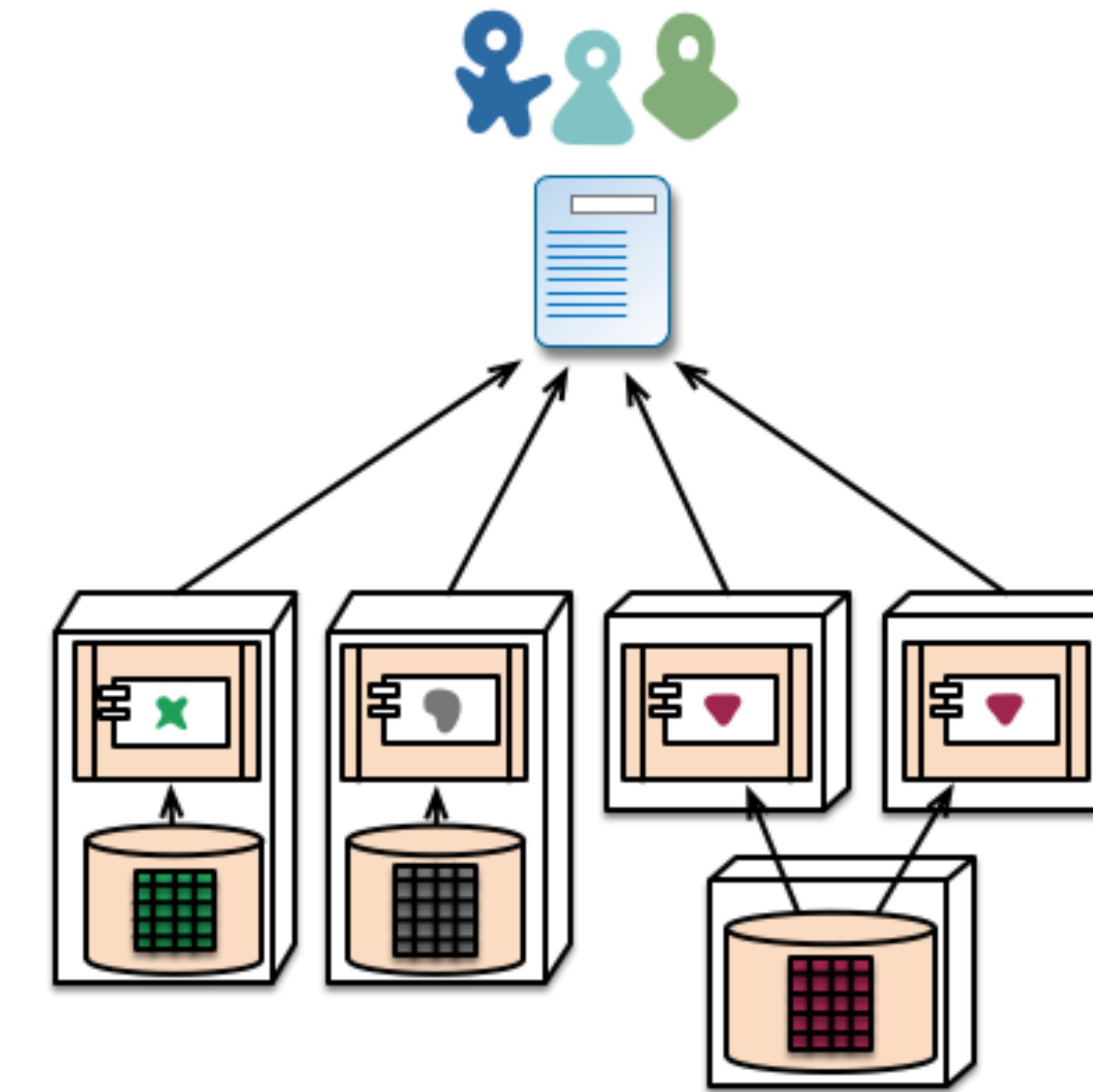


Source: <http://martinfowler.com/articles/microservices.html>

Data Management and Consistency



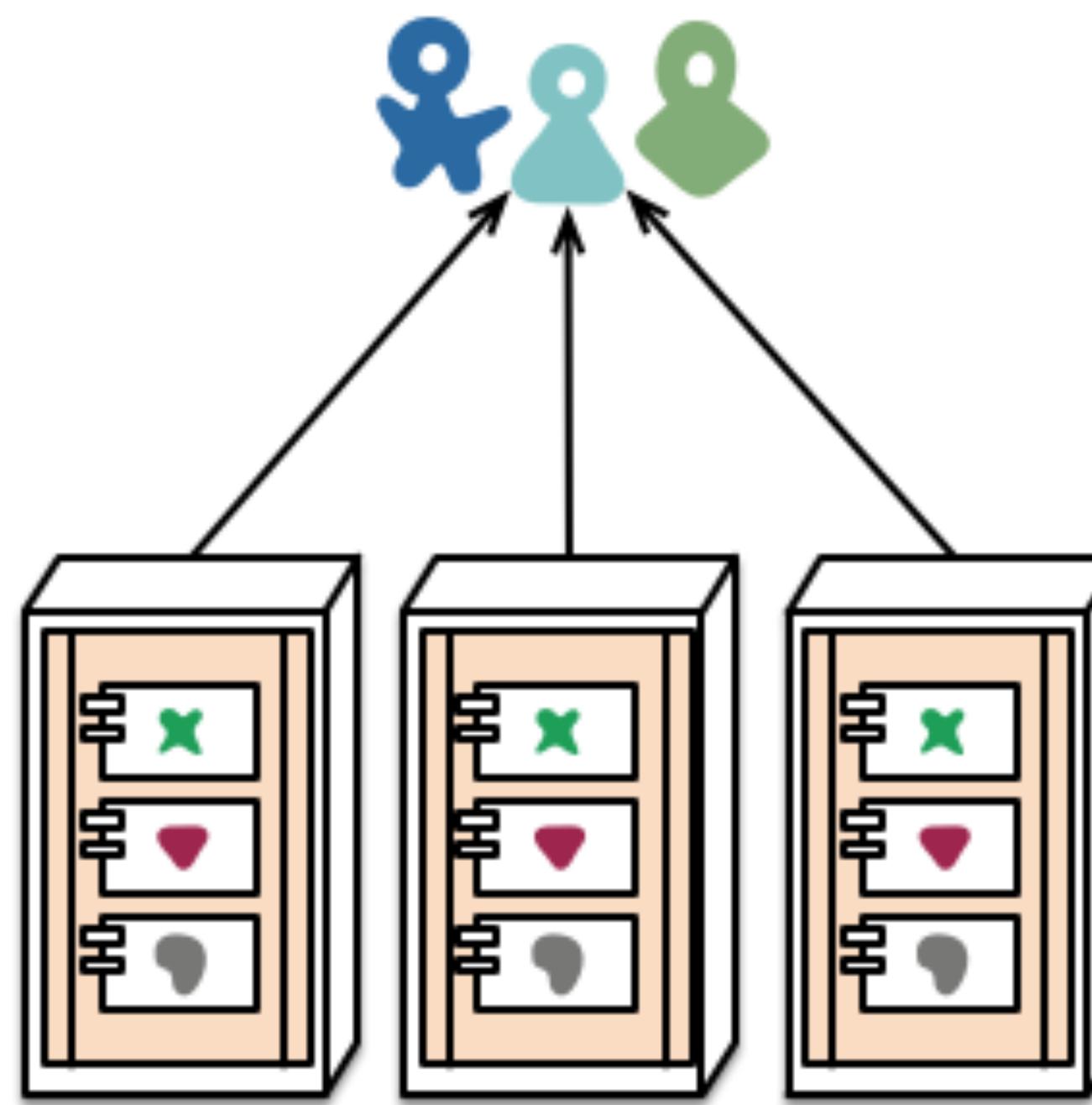
monolith - single database



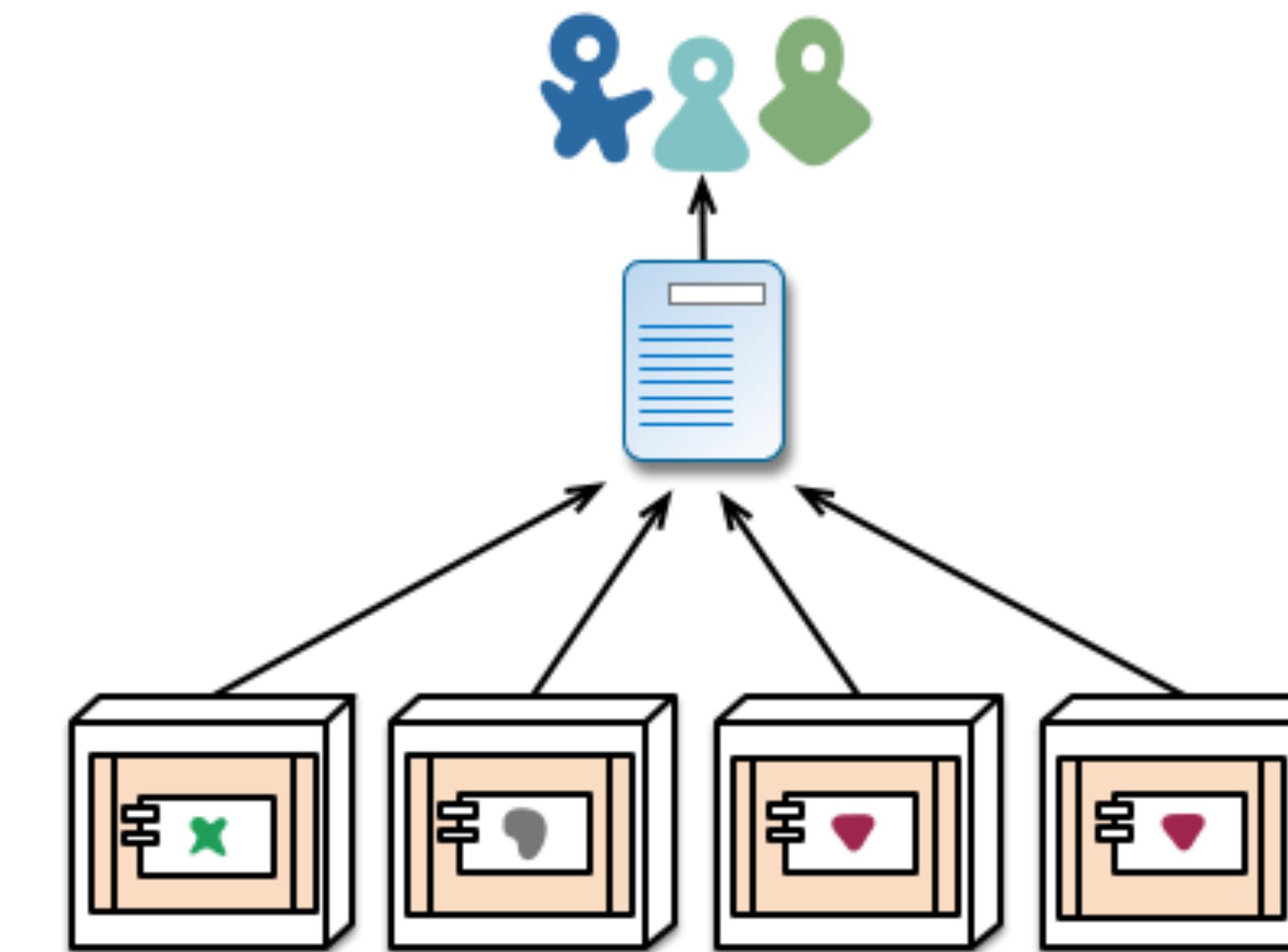
microservices - application databases

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>

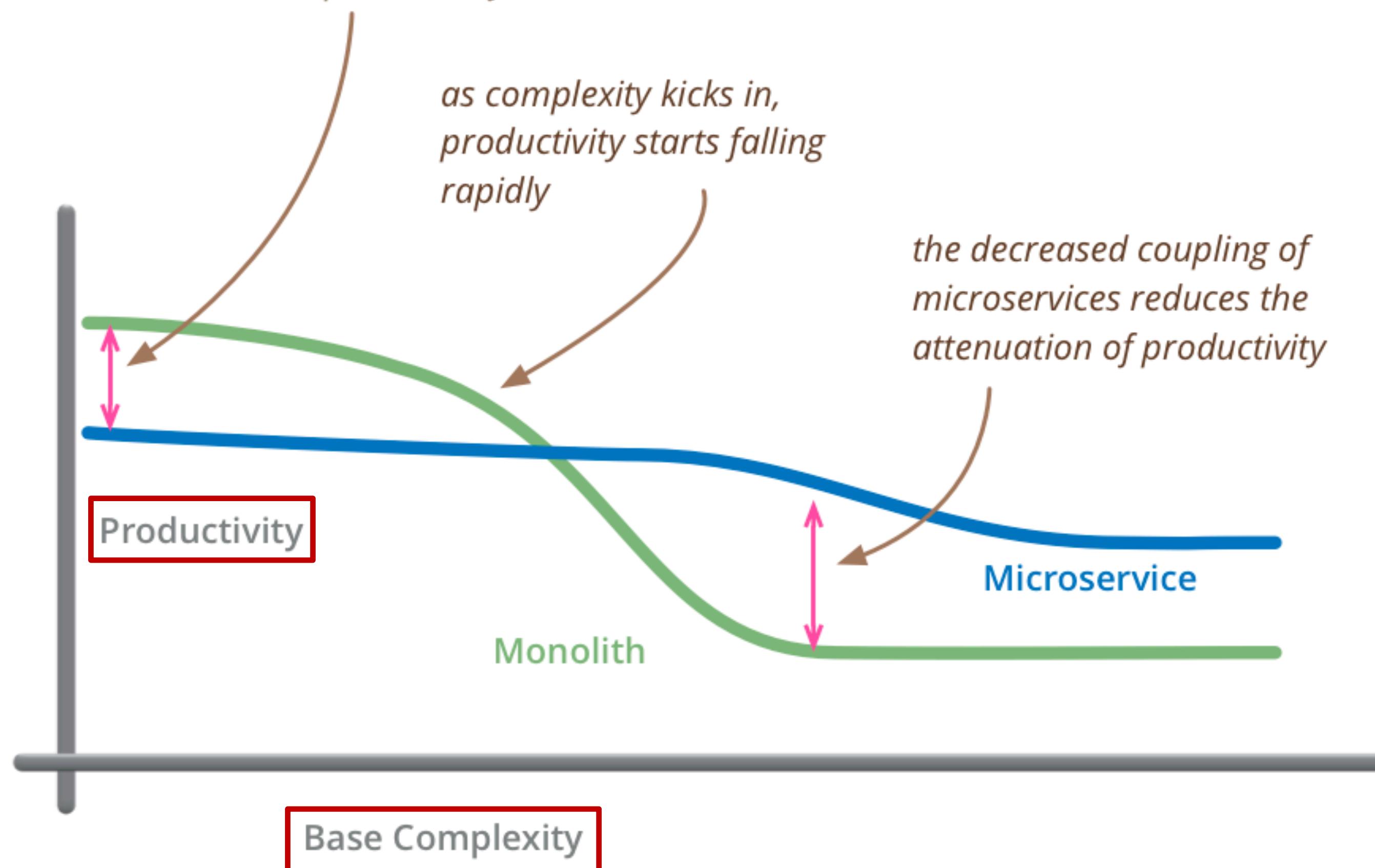
Microservices

- 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?

Serverless

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: API testing and DevOps

