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=CZ3wIuvmHeM)
Matt Ranney @ Uber (https://www.youtube.com/watch?v=kb-m2fasdDY)
Christopher Meiklejohn & Filibuster (http://filibuster.cloud)

Administrativia

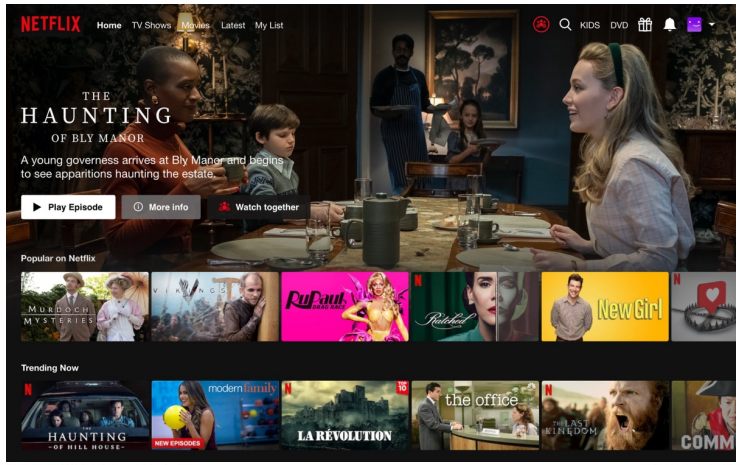
- Mid-term exam next week (Oct 10) in class
- Recitation this week: midterm review (**come prepared**!)
  - Work through problems on the previous midterms – many students found this helpful.
  - Any questions on the previous midterm questions – bring them to recitation to discuss as a class.
- Final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

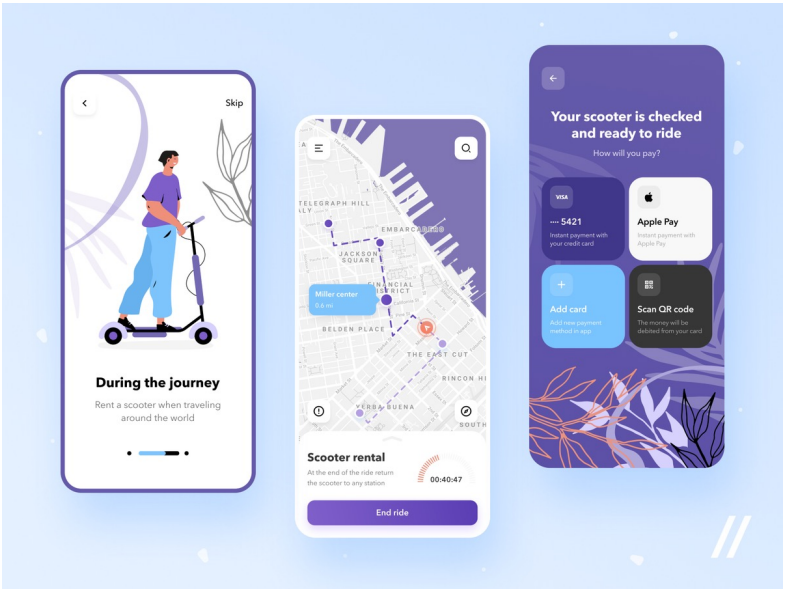
Learning Goals

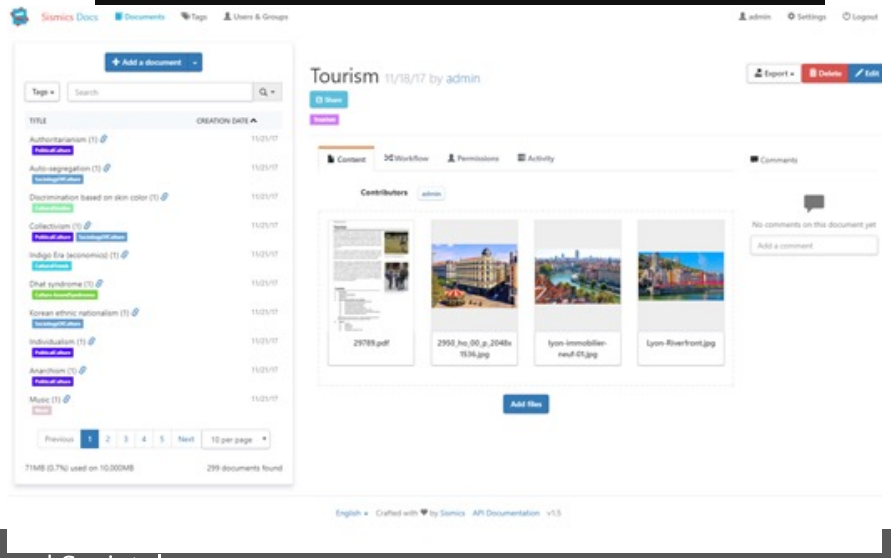
- Contrast the monolithic application design with a modular design based on microservices.
- Reason about how architectural choices affect software quality and process attributes.
- Reason about tradeoffs of microservices architectures.

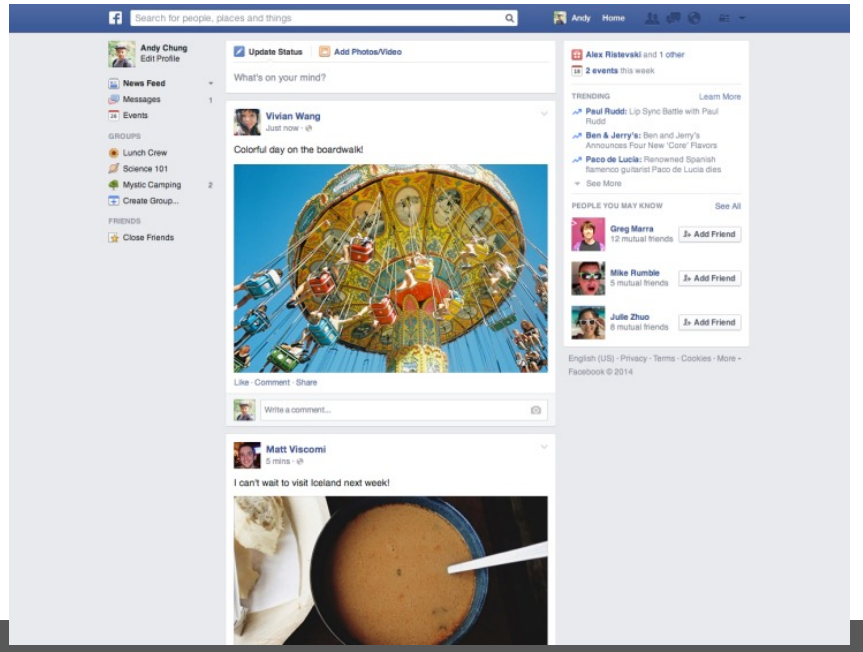
Before we get to microservices…

How might these apps be architected?

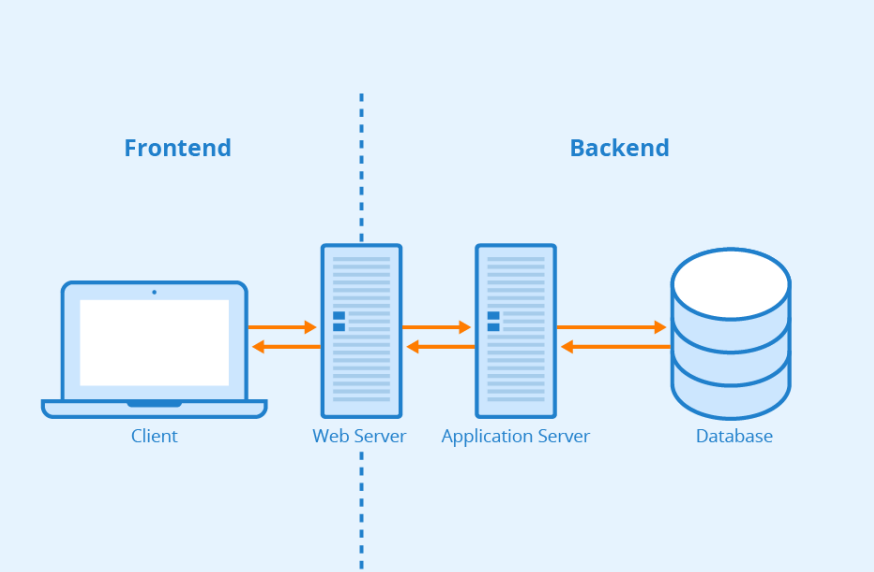




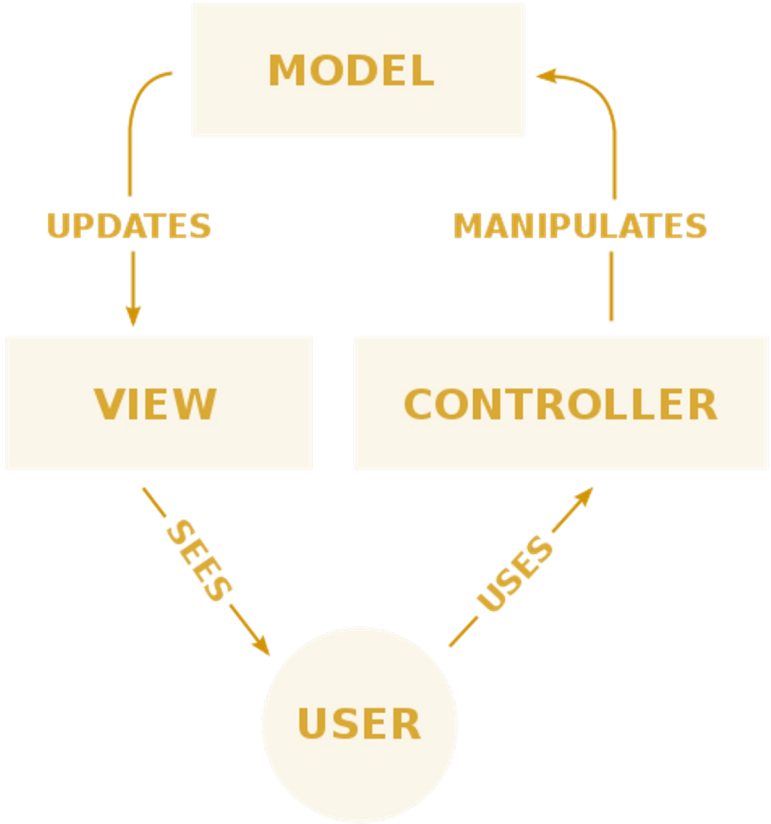




Monolithic styles: Client-server or MVC



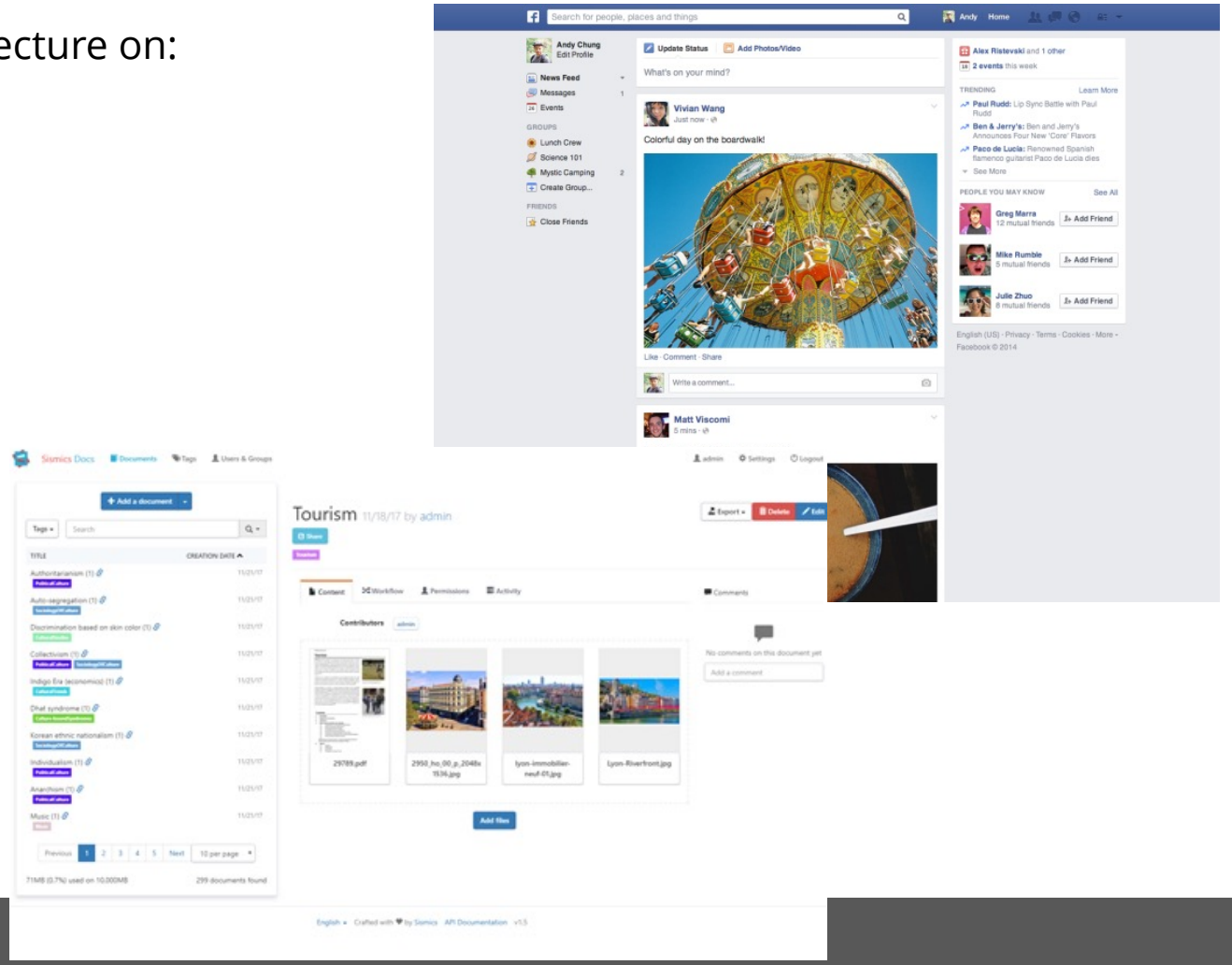
Source: https://www.seobility.net (CC BY-SA 4.0)



Monoliths make trade-offs on software quality

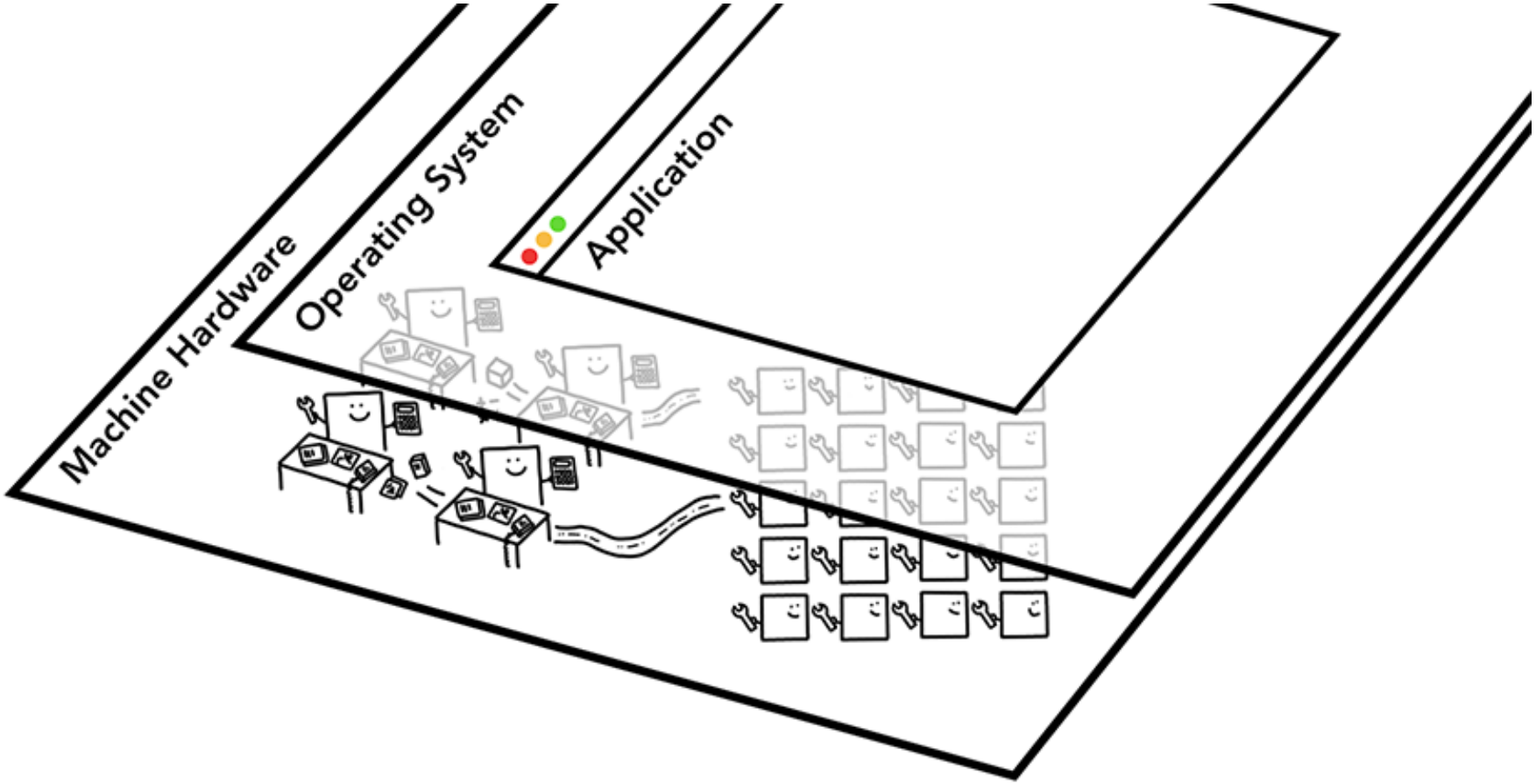
Several consequences of this architecture on:

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



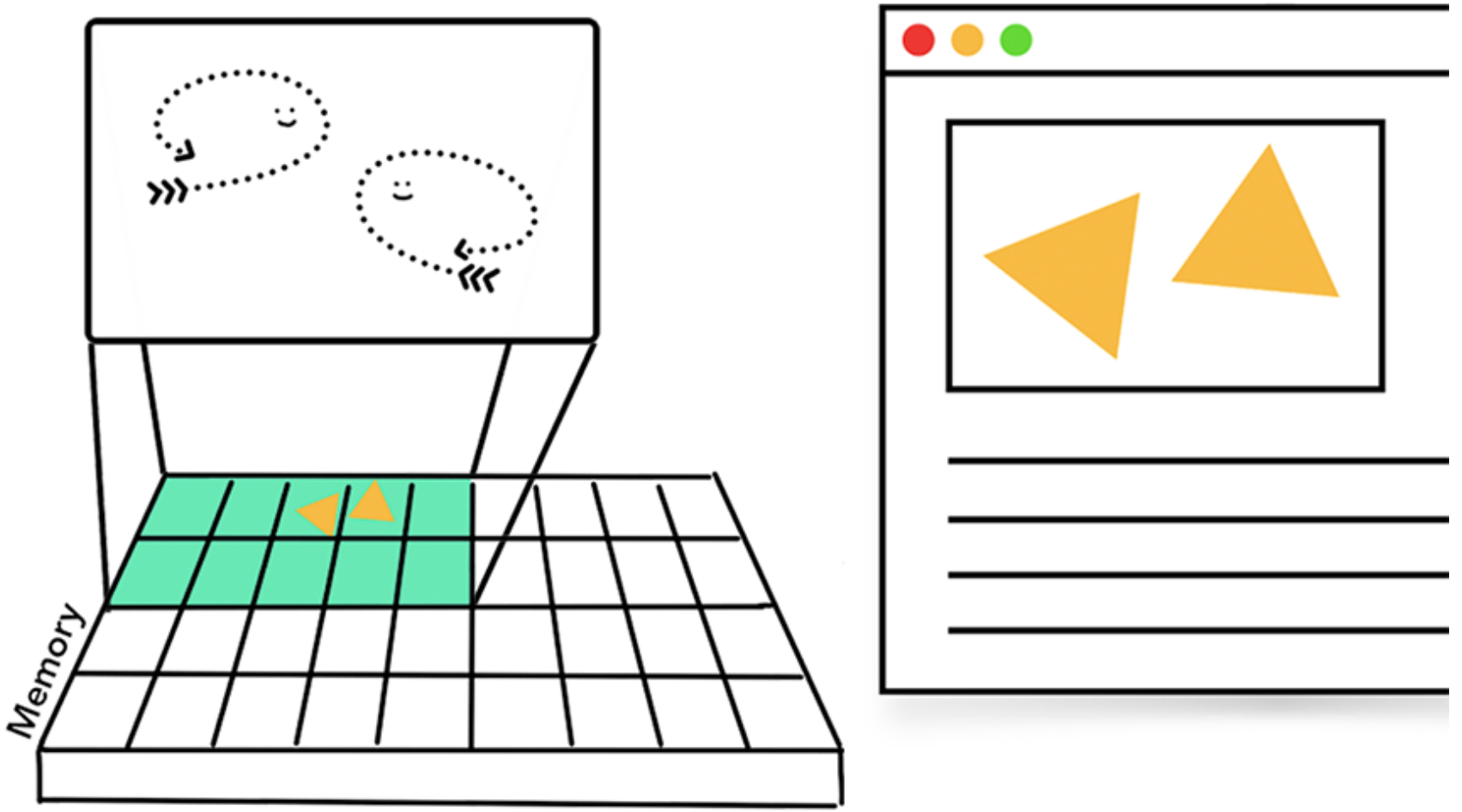
Service-based architecture – Chrome

Web Browsers



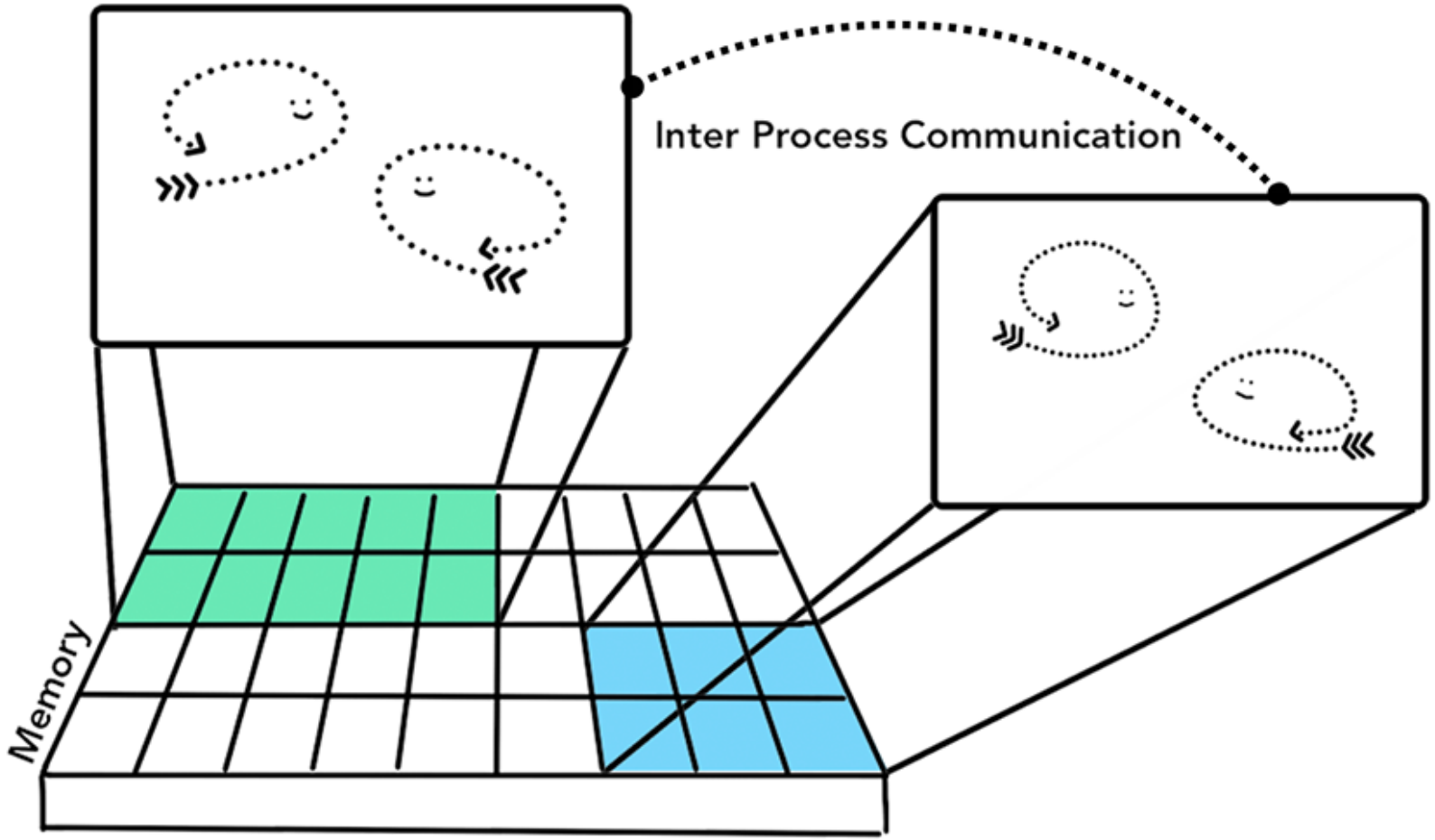
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Browser: A multi-threaded process



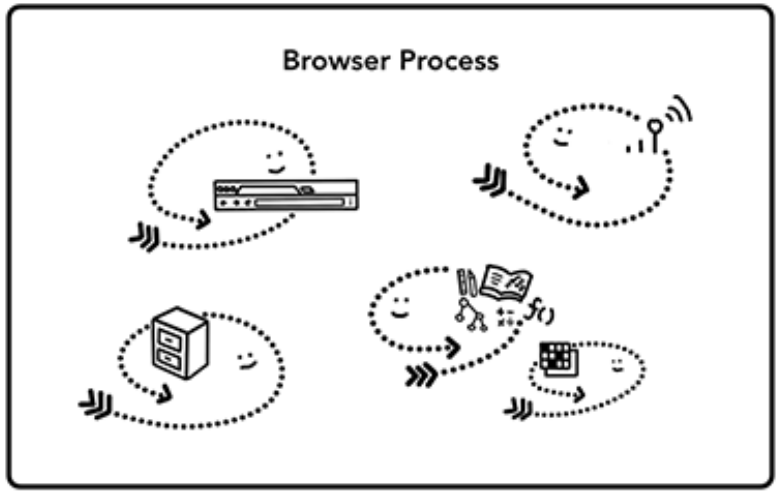
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

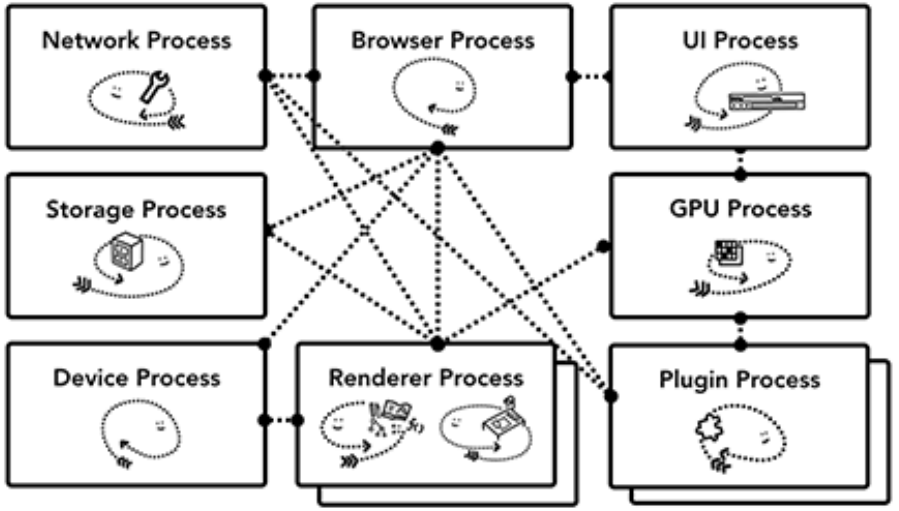
Multi-process browser with IPC



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

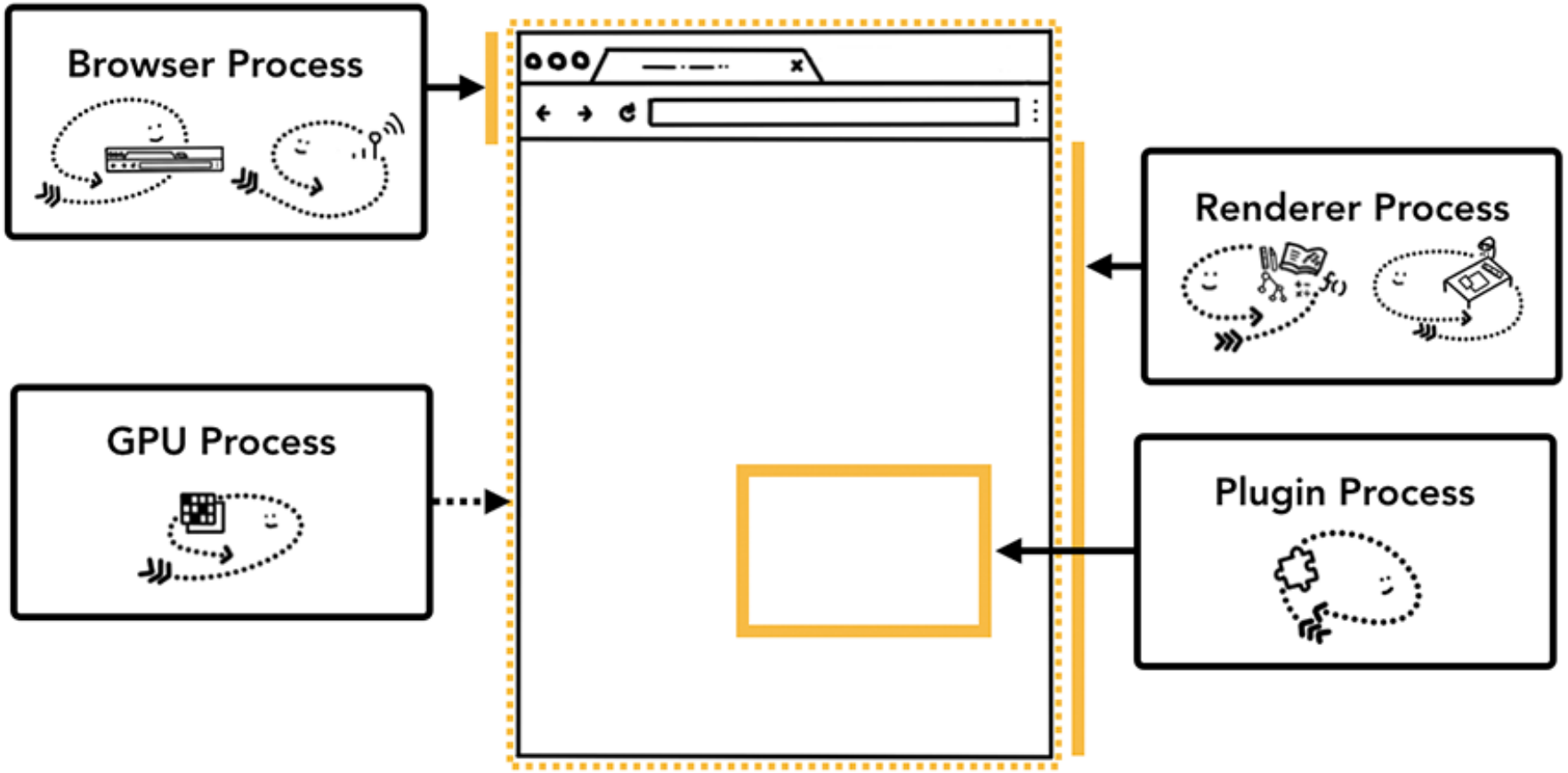
Browser Architectures





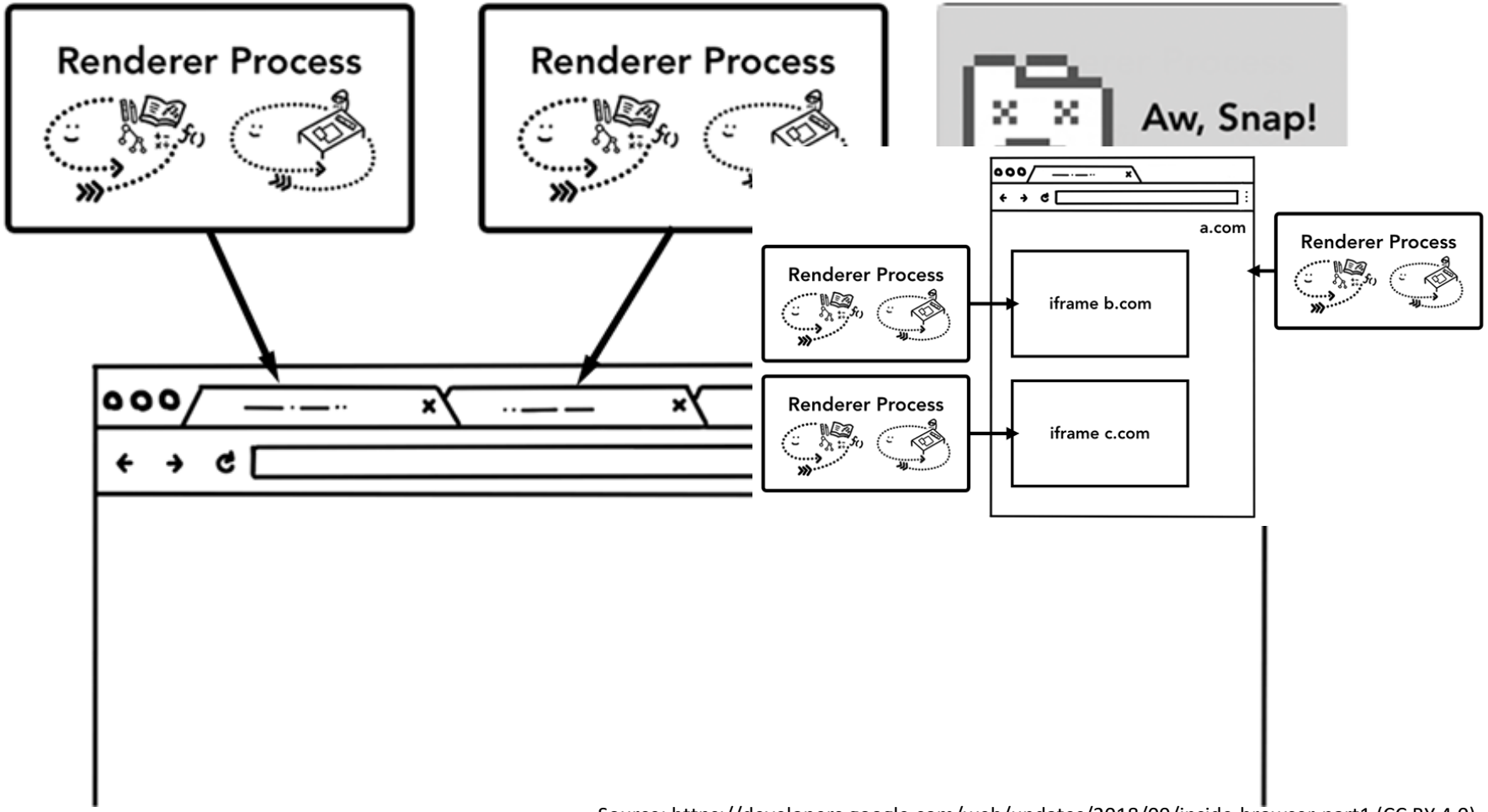
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Service-based browser architecture



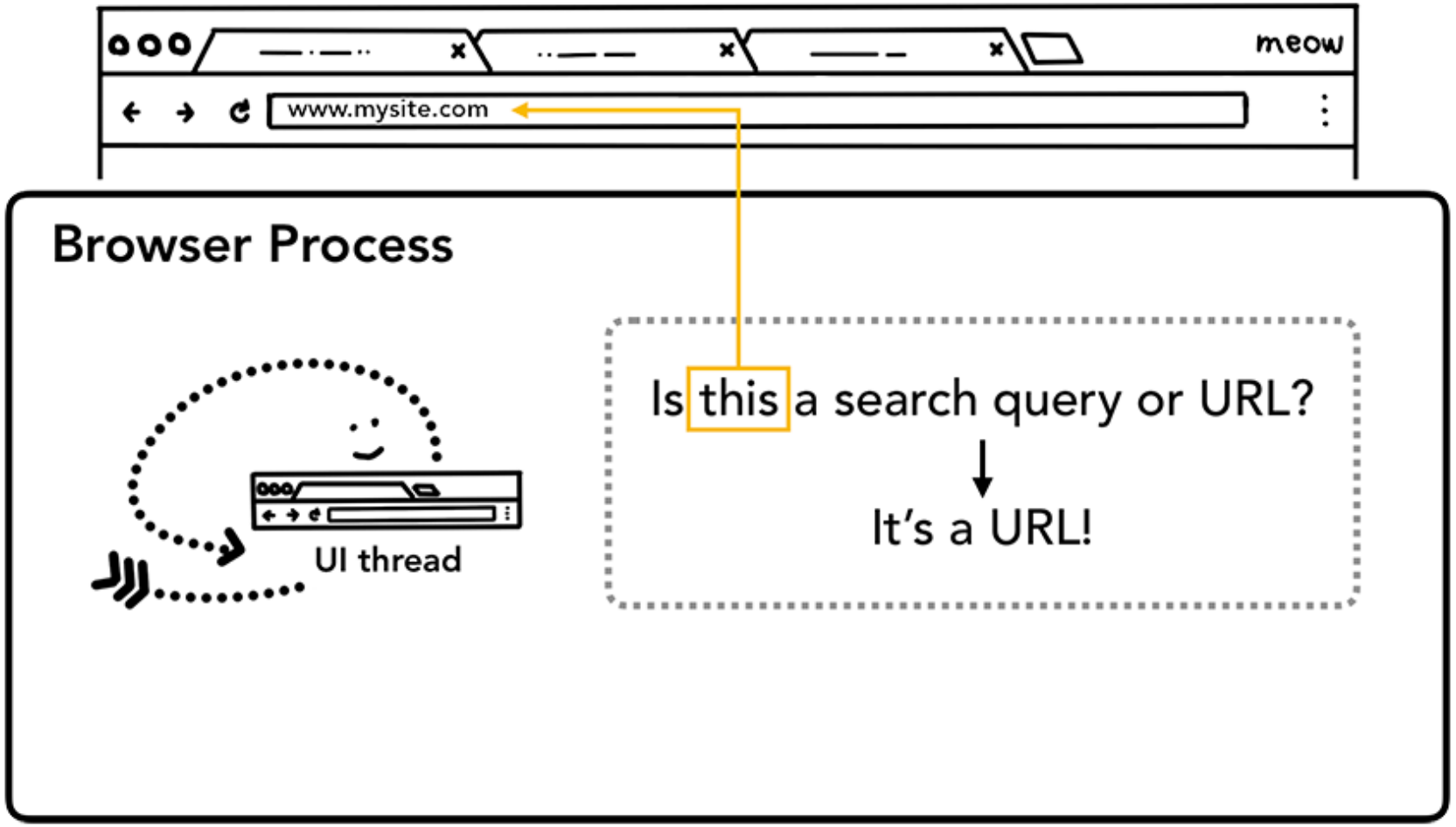
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Service-based browser architecture



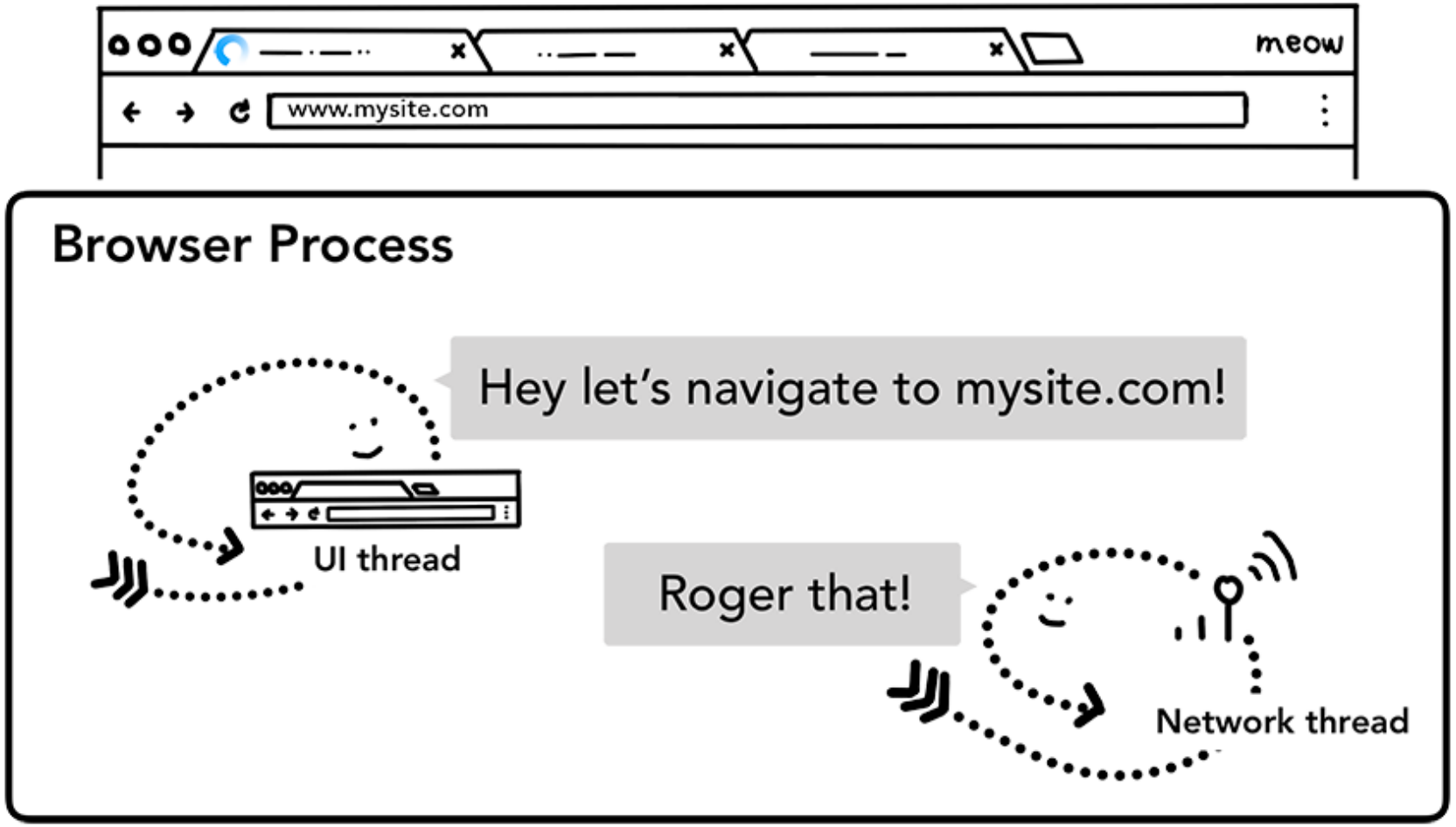
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



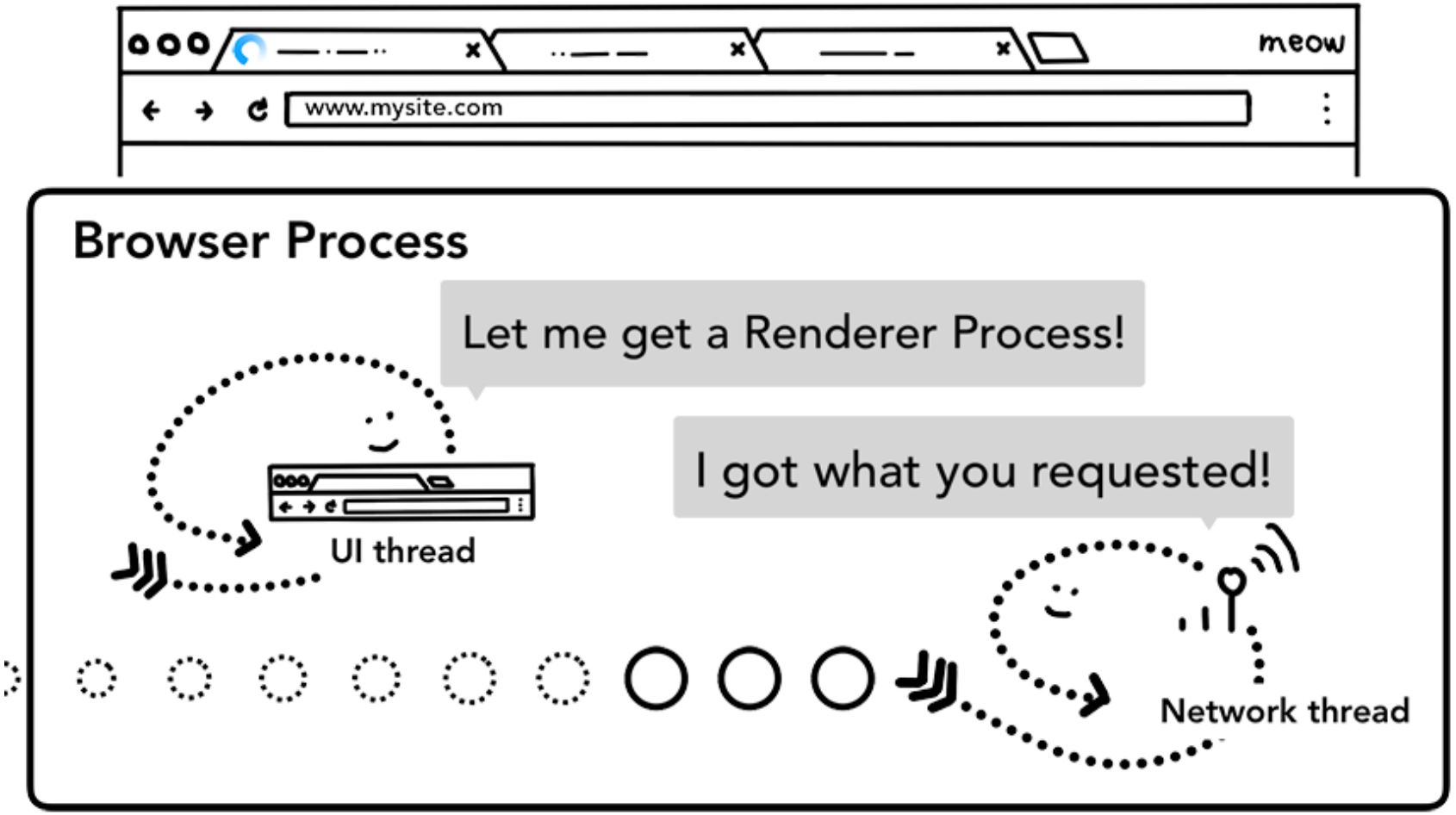
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



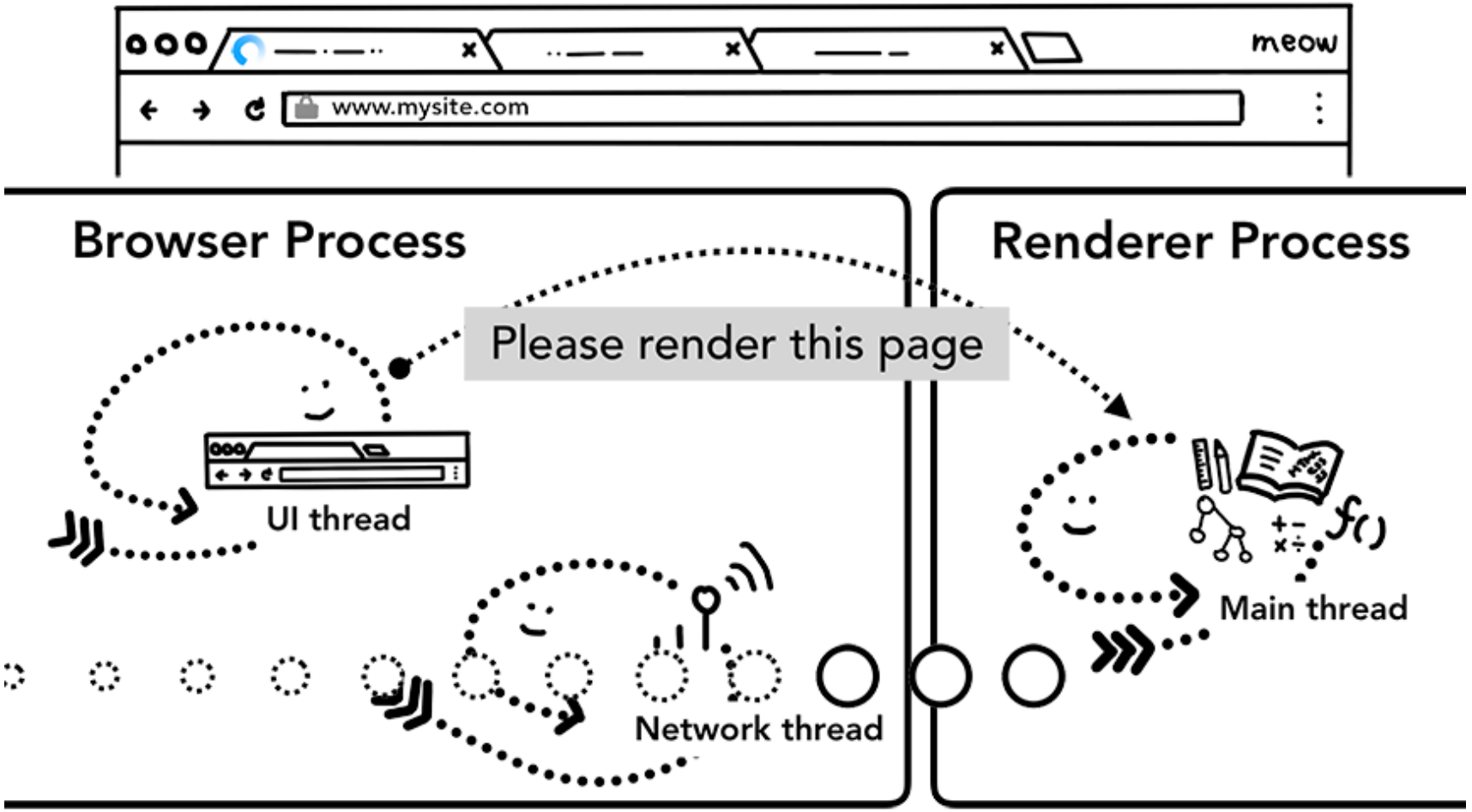
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



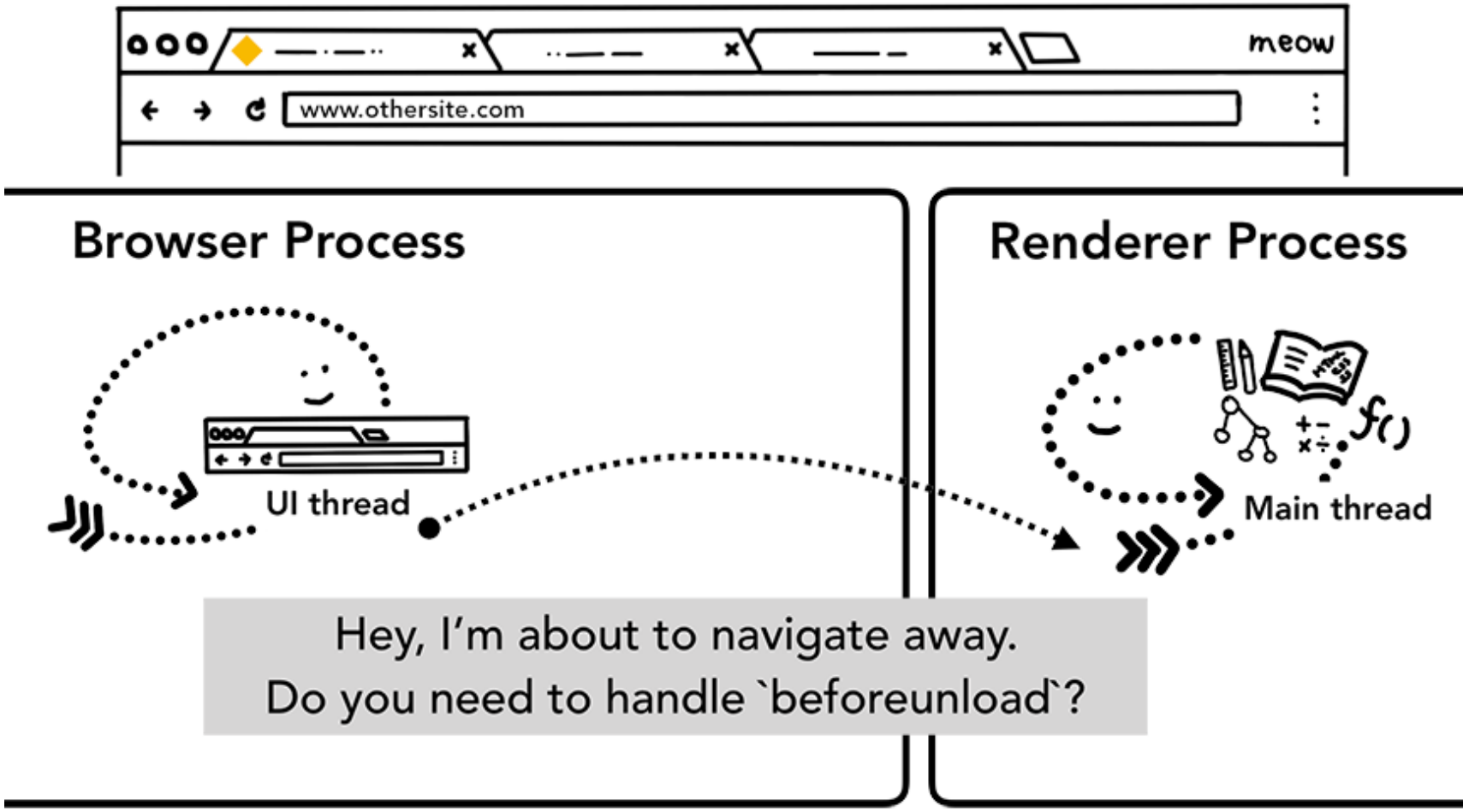
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



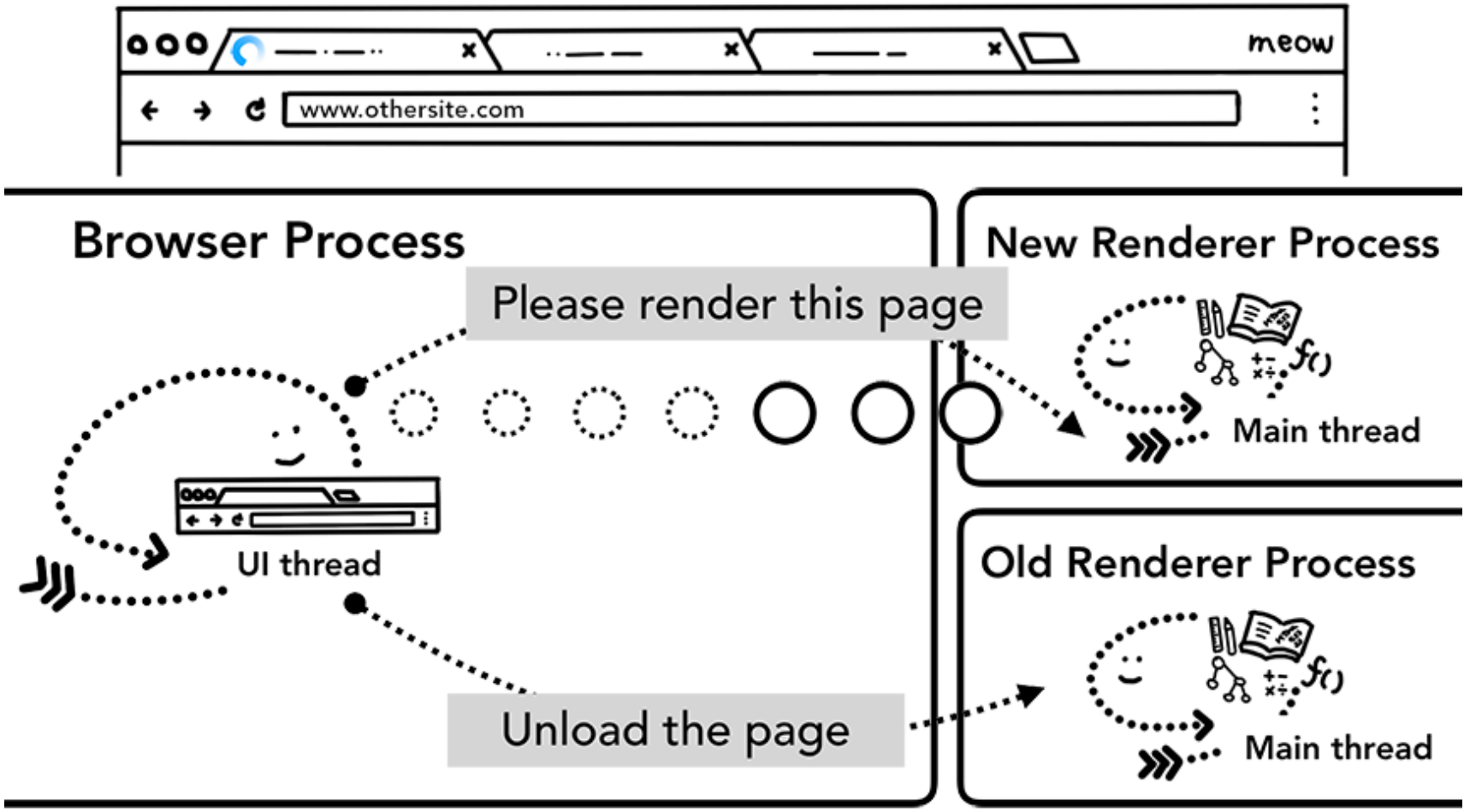
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Navigating to a web site uses service requests



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

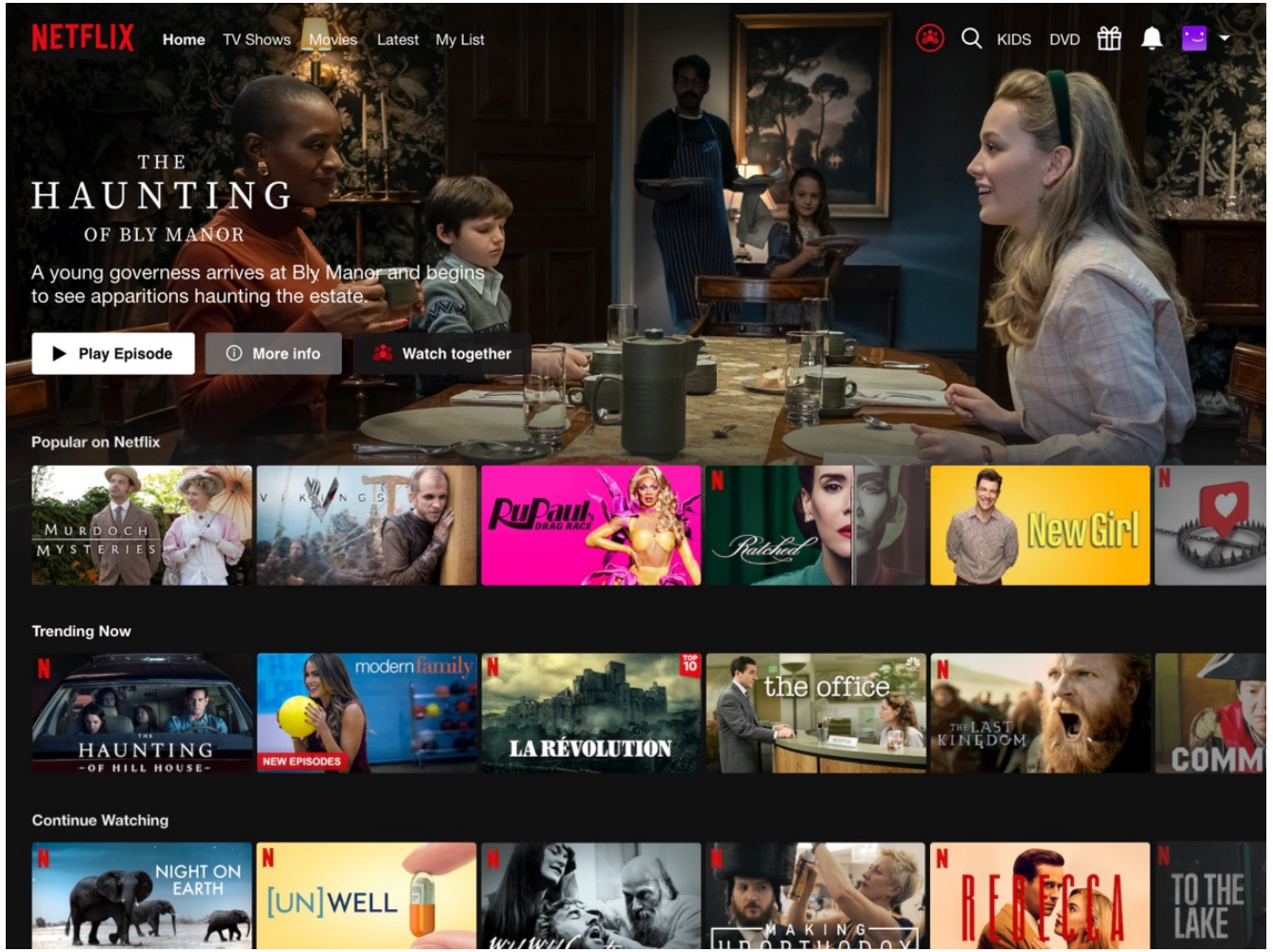
Navigating to a web site uses service requests



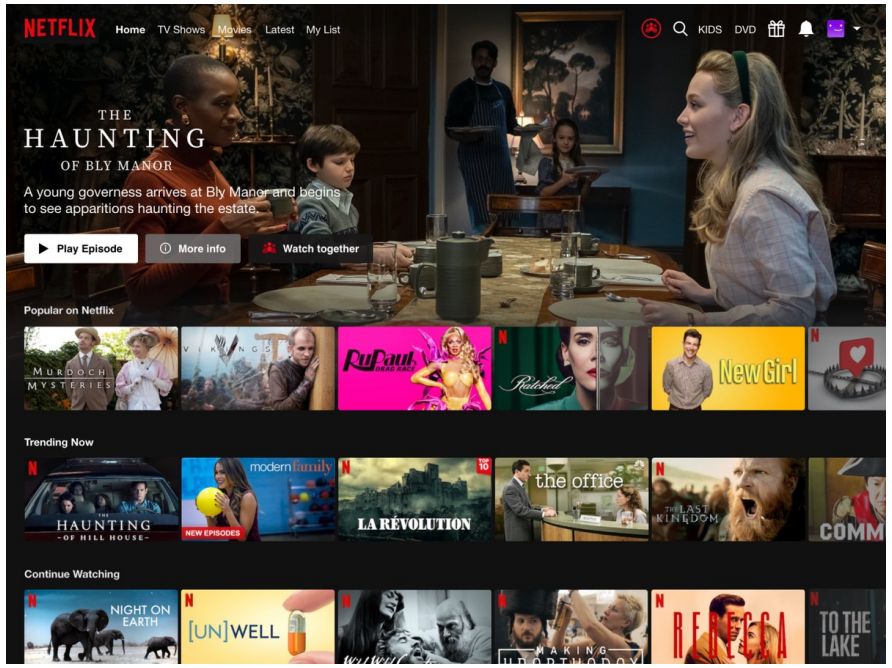
Source: https://developers.google.com/web/updates/2018/09/inside-browser-part1 (CC BY 4.0)

Microservice architecture – Netflix

Netflix


NETFLIX
Home
TV Shows
Movies
Latest
My List
KIDS
DVD
THE HAUNTING OF BLY MANOR
A young governess arrives at Bly Manor and begins to see apparitions haunting the estate.
Play Episode
More info
Watch together
Popular on Netflix
Trending Now
Continue Watching
NEW EPISODES
LA RÉVOLUTION
the office
NIGHT ON EARTH
[UN]WELL
TO THE LAKE

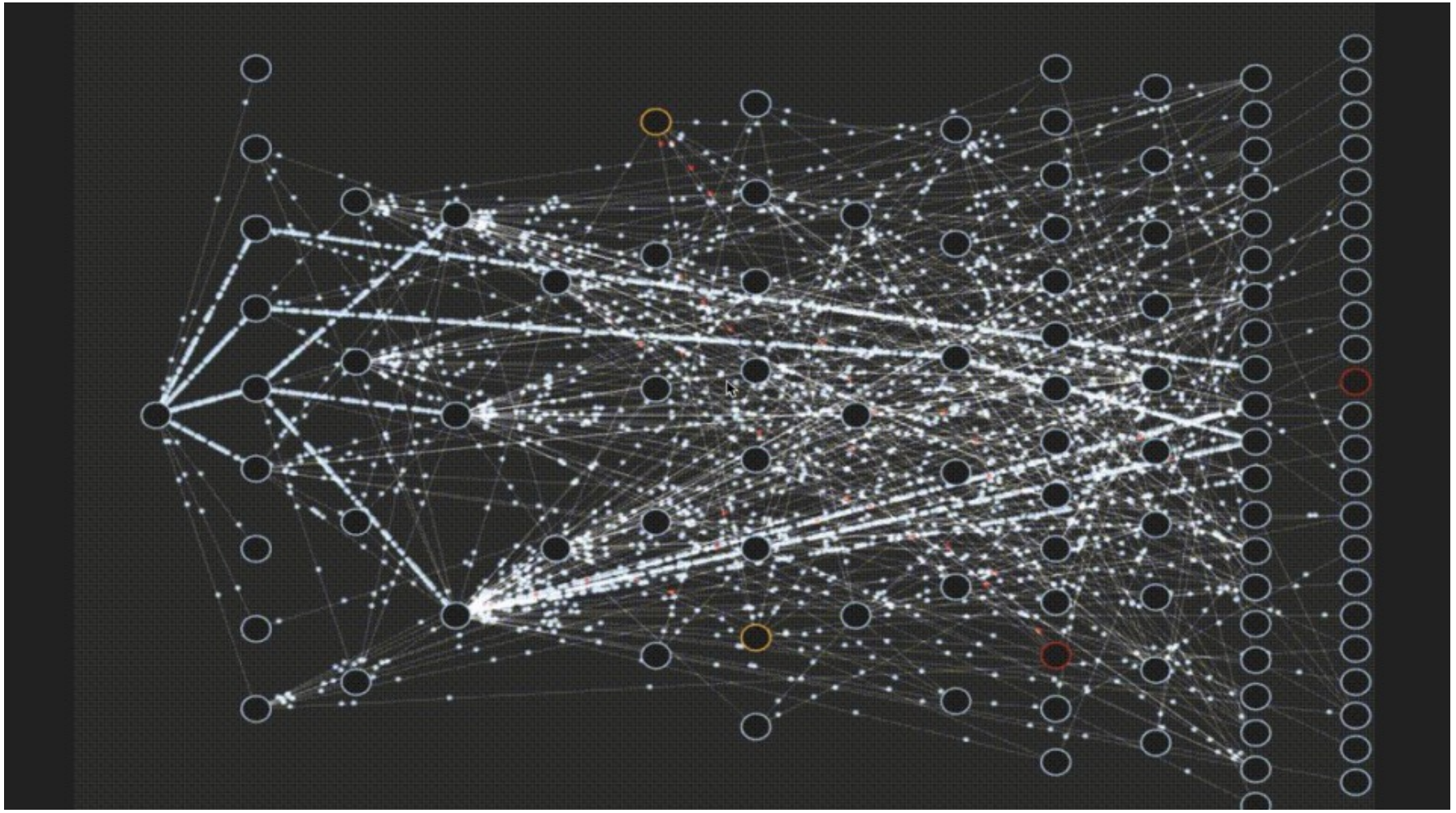
Netflix Microservices – App Boot



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

(as of 2016)

Netflix Microservices – One Request

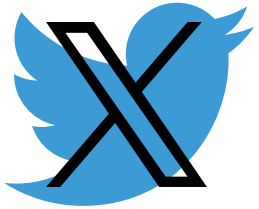


(as of 2016)

https://www.youtube.com/watch?v=CZ3wIuvmHeM

Who uses Microservices?

COMCAST



NETFLIX

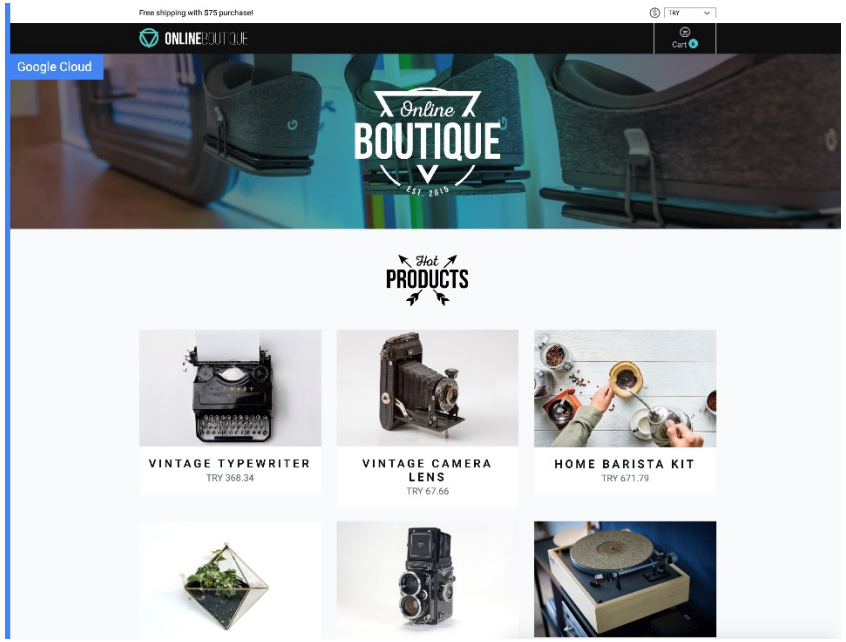
ebay

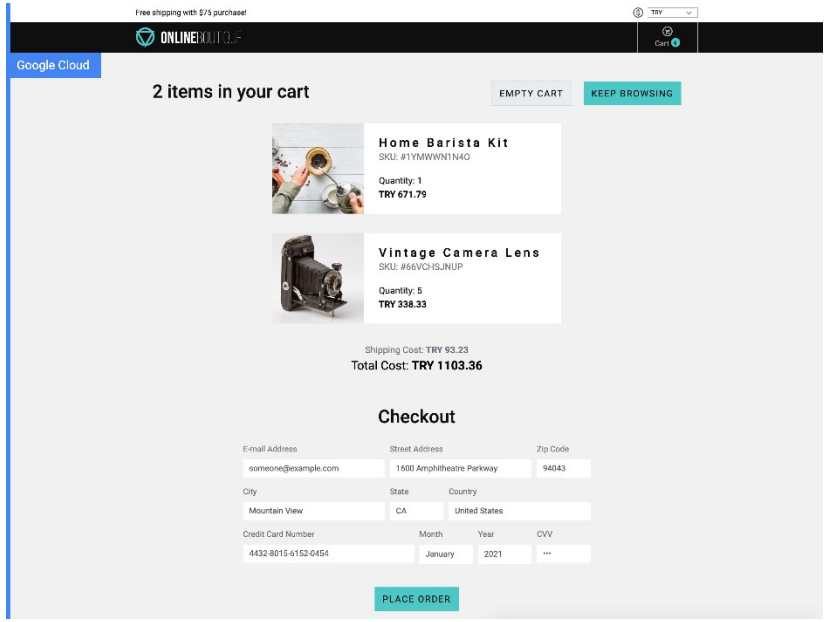
UBER

GROUPON®

Microservices –
The Hipster Shop Example

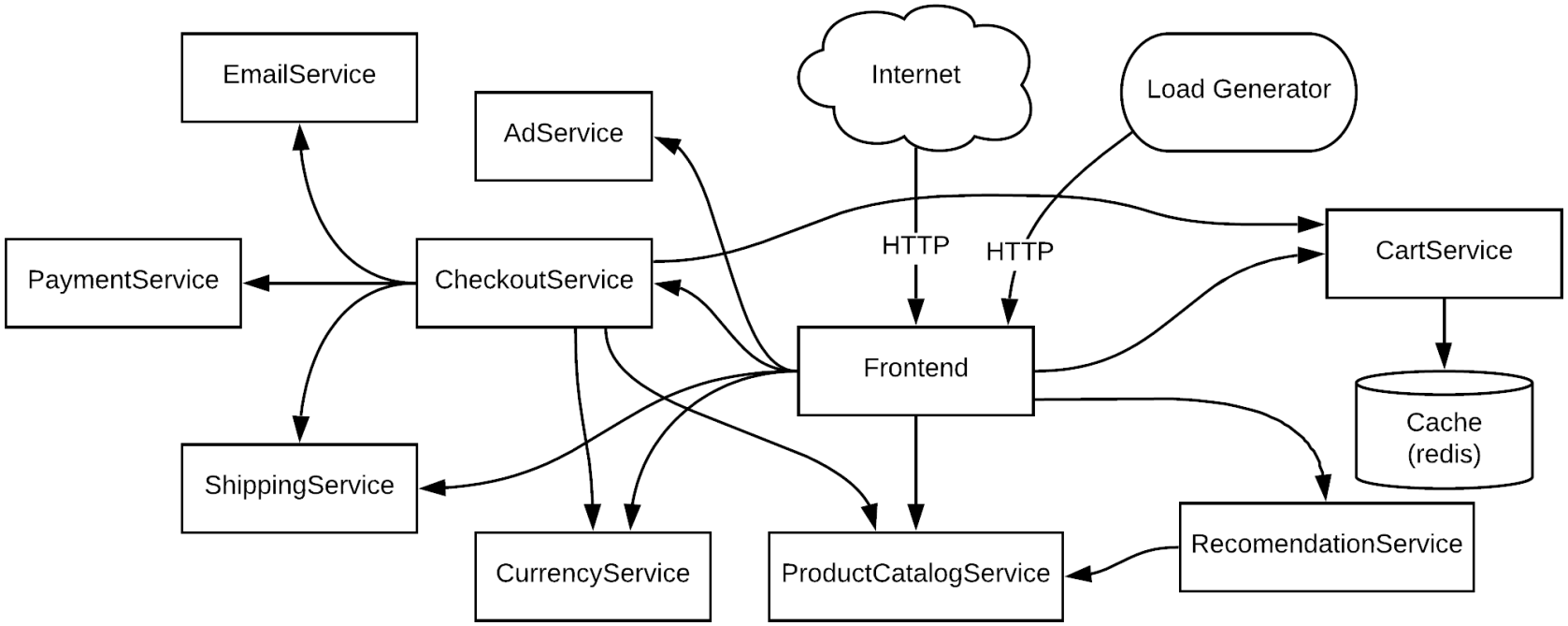
Hipster Shop: Guess some microservices





https://onlineboutique.dev

Hipster Shop Microservice Architecture



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

Microservices

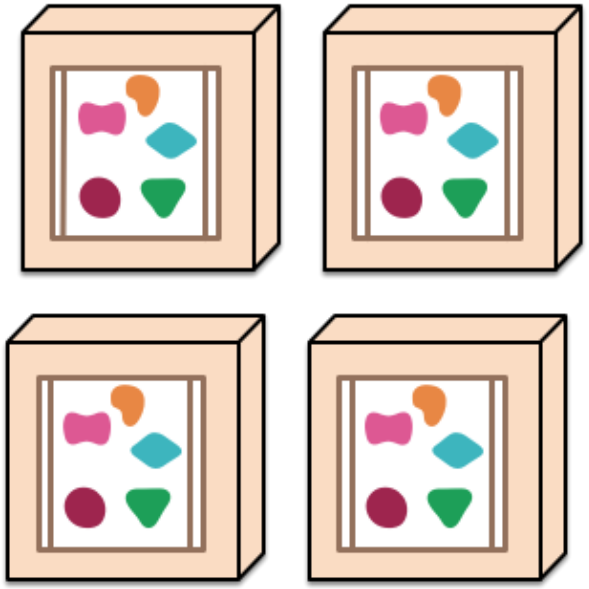
What are the consequences of this architecture? On:

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

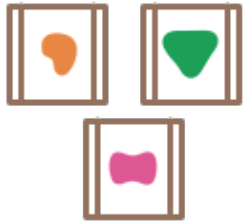
Scalability

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

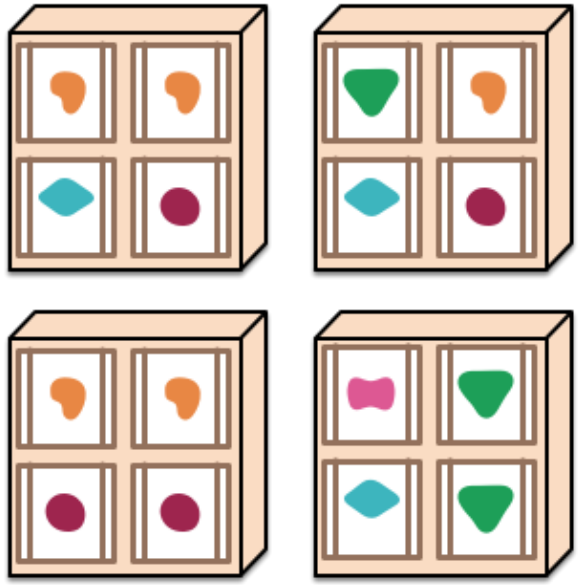
... and scales by replicating the monolith on multiple servers



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

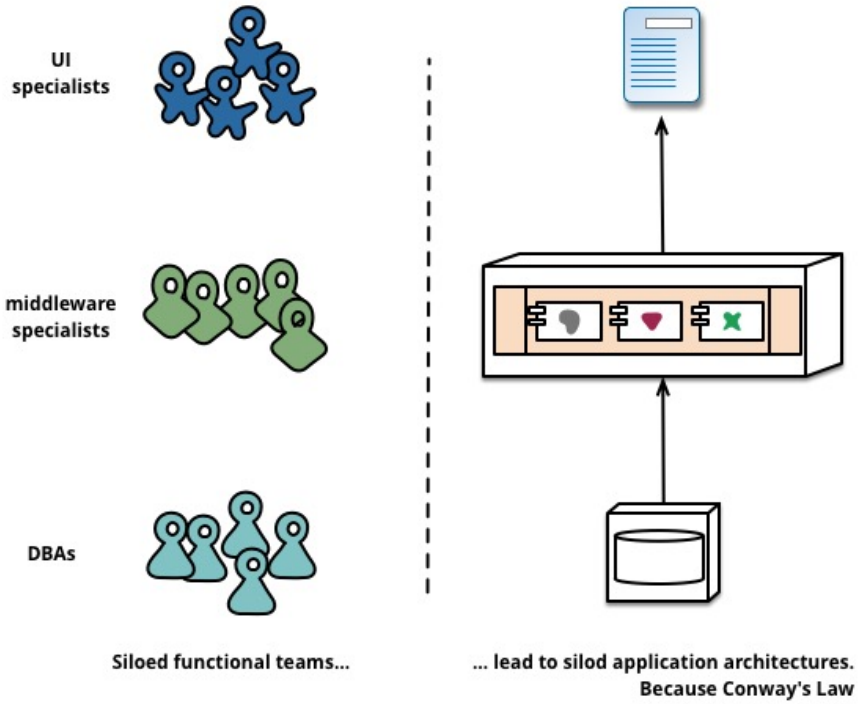


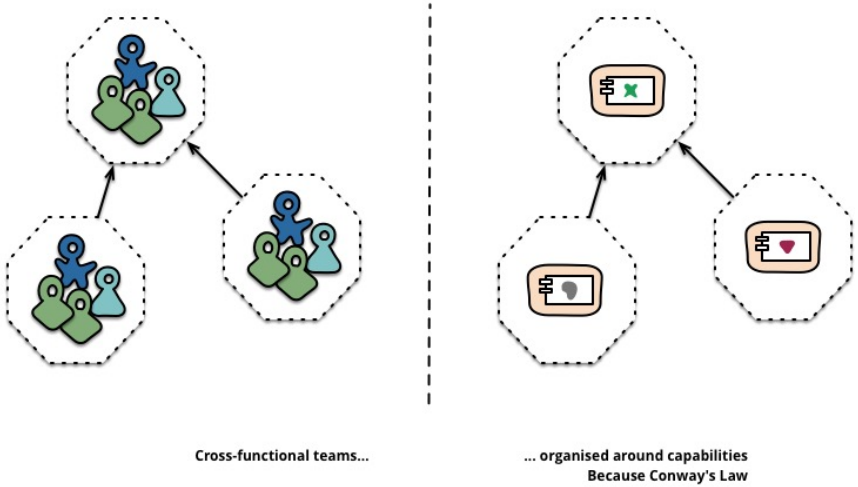
... and scales by distributing these services across servers, replicating as needed.



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

Team Organization (Conway's Law)

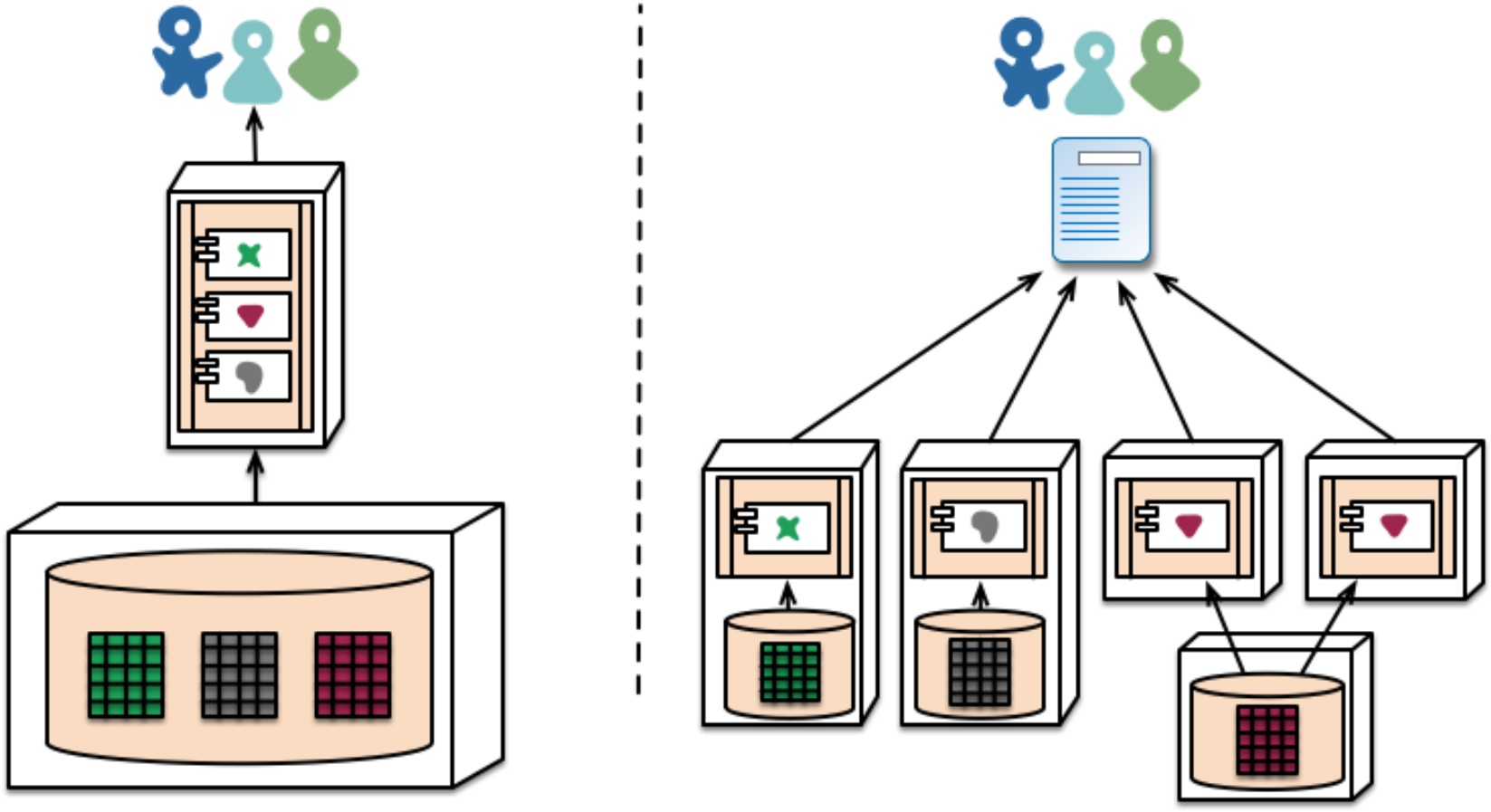




"Products" not "Projects"

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

Data Management and Consistency

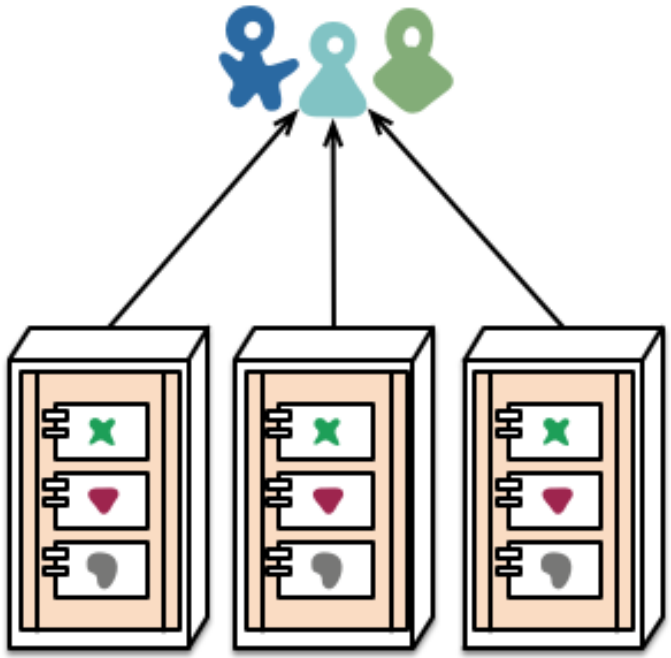


monolith - single database

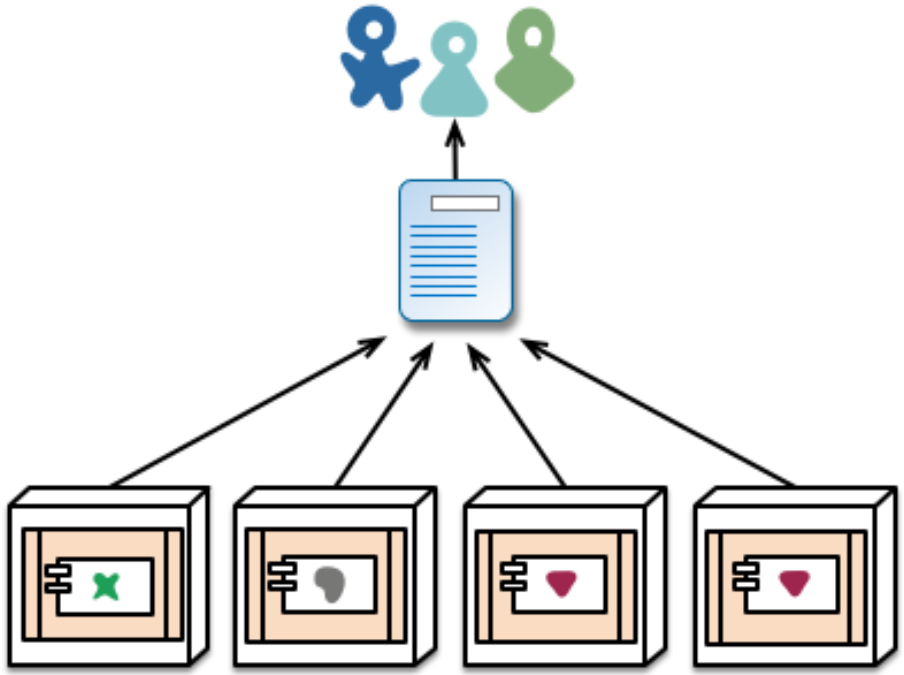
microservices - application databases

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

Deployment and Evolution



monolith - multiple modules in the same process



microservices - modules running in different processes

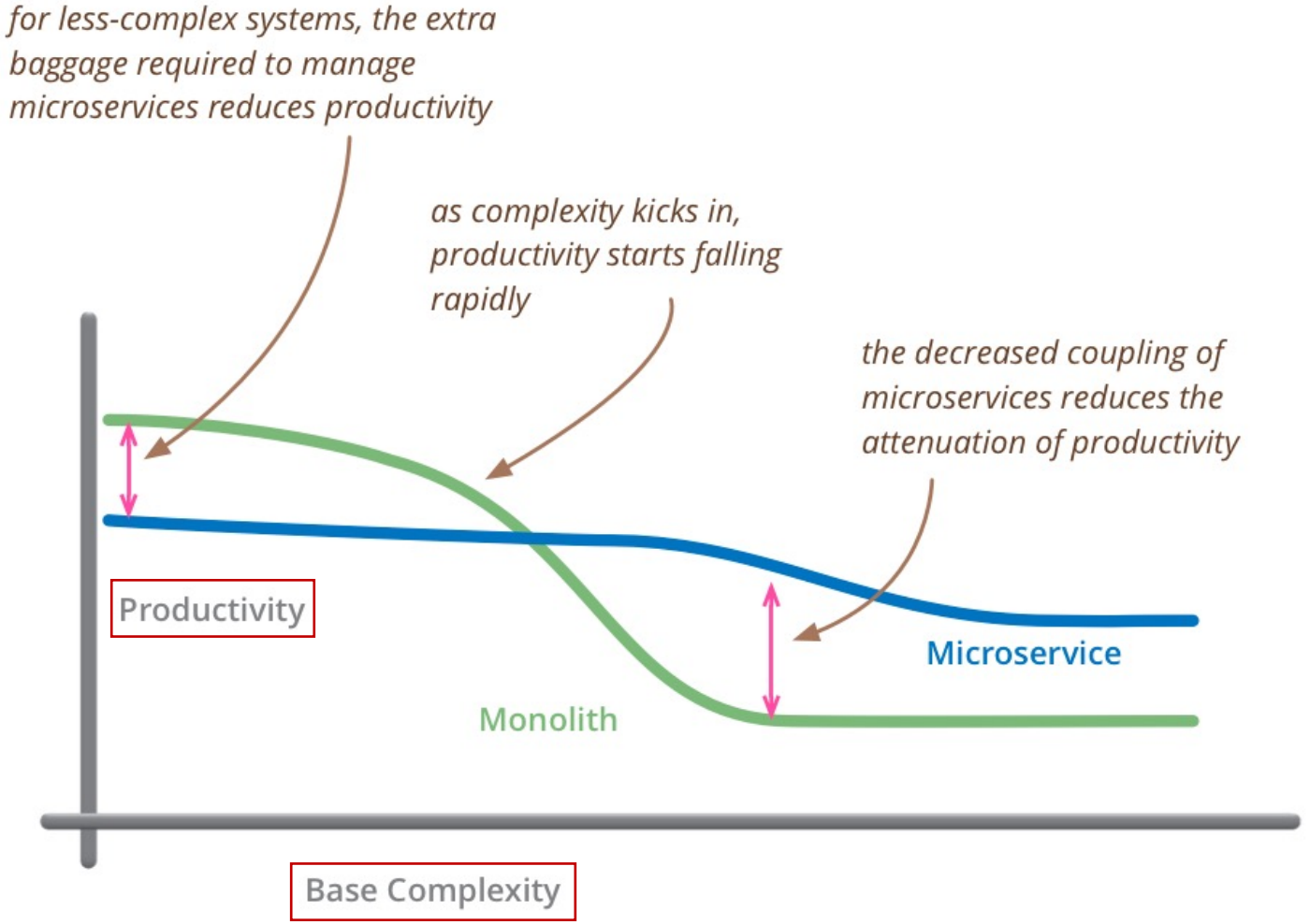
Source: http://martinfowler.com/articles/microservices.html

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



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